UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Human-Autonomy Teaming with a Supportive Situation Awareness Model

by Rinta Kridalukmana

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Haiyan (Helen) Lu and Mohsen Naderpour

University of Technology Sydney
Faculty of Engineering and Information Technology

March 2021

Certificate of Original Authorship

I, Rinta Kridalukmana, declare that this thesis is submitted in fulfillment of the requirements for the award of *Doctor of Philosophy*, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution,

This research is supported by the Indonesia Endowment Fund for Education (LPDP) and the Australian Government Research Training Program.

Signature of student: Production Note:
Signature removed prior to publication.

Date: 12 July 2021

Acknowledgement

I would like to express my appreciation to Associate Professor Haiyan Lu who supervised this Ph.D. research and offered her knowledgeable comments, valuable support, helpful guidance during difficult times and beneficial suggestions along the way. I am also grateful to Dr Mohsen Naderpour, my co-supervisor, for his knowledgeable suggestions and valuable advice throughout my Ph. D. study.

Looking back over my Ph. D. journey, I would like to express my appreciation to my wife (Lily Lamina) and my children (Anastasya Arinta Kridamaranatha and Mikael Berinta Kridasaktika) who supported me at all times. Also, I would like to deeply thank my parents (Professor Joetata Hadihardaja (R.I.P) and R.A Enny Keatin Diponegoro) who always encouraged me to pursue this Ph. D degree. My gratitude also goes to my big family, especially to my eldest brother (Professor Iwan Kridasantausa (R.I.P)) who was always by my side when I experienced hard times during this candidature.

In particular, thank you to the Indonesia Endowment Fund for Education (LPDP) for giving me a scholarship to support this study, the Faculty of Engineering and Information Technology and the Australian Artificial Intelligence Institute (AAII) which provided support for peer-reviewed journal publications, and all my colleagues in the Department of Computer Engineering, Diponegoro University who gave me ongoing support during this study.

Last, but not least, a special thank goes to Michele Mooney and Noniusar Theisa Omar for helping me to identify and correct the grammar and syntax problems in my dissertation and publications.

ABSTRACT

Coordination abilities are required for artificial agents with a high level of autonomy (also called autonomy agents) to perform collaborative work with humans in human-autonomy teaming (HAT). This study focuses on the development of the autonomy agent's coordination abilities for a HAT context in which team members have superior subordinate relations using collaborative driving as a study case. The key challenge in such a HAT context is that the agent as the subordinate lacks the cognitive ability to coordinate with its superior. To meet this challenge, this study addresses the following critical issues: (i) how to model the autonomy agent's situation awareness (SA), (ii) how to enable the agent to self-explain its behaviours, and (iii) how to enable the agent to effectively help its superior to maintain their SA. This study makes four novel proposals: (i) a situation awareness modelling framework for teaming situations, (ii) a new self-explanation method based on the autonomy agent's artificial situation awareness states (ASAS), (iii) a transfer learning approach with a directive offline augmentation technique for the autonomy agent to monitor its human superior's distraction using on-board camera-captured data, and (iv) a time-constraint-driven transparency model of the agent's SA in collaborative driving. The evaluation of these proposals was carried out using two typical scenarios in collaborative driving, namely passing traffic lights and overtaking vehicles. The evaluation results show that (i) the autonomy agent's situation awareness modelling framework for teaming situations can guide the modelling and identification of situation awareness requirements, (ii) the self-explanation method can overcome the existing process-based method in term of reducing the search space in generating explanations, (iii) the new transfer learning approach can effectively cope with the geometrical relation constraints in classification of human driver's distraction, and (iv) the time-constraint-driven transparency model of agent's SA can control the visibility of the agent's behaviours and their explanation. These proposals are significant to enhance the coordination performance in HAT and to calibrate the human member's trust in their autonomy agent partner and to avoid human driver distraction.

Table of Contents

List of Figures

List of Tables

Chapter 1 : Introduction

1.1. Background

The advancement of technologies enables intelligent non-human agents to have a high level of autonomy. These agents can use their prior knowledge and observable variables sensed from their surroundings to make decisions in response to given situations (Demir, McNeese & Cooke 2017; Johannsdottir & Herdman 2010; McAree & Chen 2013). These agents are able to work collaboratively with humans as teammates, in a situation which is referred to as human-autonomy teaming (HAT), and not merely as automation tools (McNeese et al. 2018). Therefore, like humans, such non-human agents (referred to as autonomy agents in this study) can also have some degree of understanding about what is happening which can be described as situation awareness (SA) (Stanton et al. 2006, 2017).

To be involved in HAT, an autonomy agent should have the ability to perform its assigned tasks and to coordinate with its human teammate. Good coordination in HAT is a critical challenge because it requires mutual understanding between the human and the autonomy agent. The way to establish coordination between a human and an autonomy agent depends on the relationship between them in a given teaming context. From the perspective of SA theory, such relations can be represented by a SA model, which describes the relationship between team members' goals, the goal-related environment, and SA requirements. SA model is critical as different SA models will result in different team situations, and thus, autonomy agents will require differing abilities to support the coordination with the human agents.

To develop the autonomy agent's coordination ability, existing studies use three teaming situations, namely a normal teaming situation, a mutual teaming situation, and a shared teaming situation (Endsley 1995; Shu & Furuta 2005). A normal teaming situation is one in which each member only has independent goals, however mutual and shared teaming situations are ones in which team members have shared goals as well as their independent

goals. For a normal teaming situation, the autonomy agent's ability, such as being able to send push notifications, is suggested to be part of a coordination support system to enhance the coordination performance (Demir et al. 2019; Demir, McNeese & Cooke 2017). For the other two teaming situations, the ability to share information/knowledge is needed (Endsley 1995). Based on the characteristics of the relationship between human and autonomy agents, existing teaming situations are only suitable for a teaming where the team members have non-shared goals. We refer to the coordination in such a teaming as horizontal coordination. Horizontal coordination is used in organization theories, in which horizontal coordination can be described as inter-departmental coordination where each department, which can be considered to be a team member, has its own responsibilities (non-shared goals) (Sting & Loch 2016). For example, the research and development team in an organization/enterprise have different goals and responsibilities to the marketing team. In a HAT context, an example of horizontal coordination can be seen in a military case which uses a robot squad to send supplies to human troops on a battle field (Selkowitz, Lakhmani & Chen 2017). In a teaming context, the robot squad can be considered to be team members of the supply chain unit which has different responsibilities from the human troops who are team members of a combat unit.

Vertical coordination refers to the coordination between someone in authority and their subordinates (Sting & Loch 2016). Therefore, in a teaming scenario with vertical coordination, team members have one of two roles, namely superordinate (or supervisor) who has superior authority and subordinate who does not have its own independent goals. For example, in an organization, a manager of a research and development department can be considered as the superordinate while the employees in this department are the subordinates. As a different example, the relation between a driver with an on-board advanced driving assistance system (ADAS) which has autopilot and collision avoidance features is an example of vertical coordination in a HAT context (Endsley 2017).

Due to the different characteristics between horizontal and vertical coordination, the coordination support systems designed for horizontal coordination are not applicable for vertical coordination. When the human member plays a role as an autonomy agent's superordinate, the critical problems are that it is difficult to comprehend the autonomy agent's behaviour and there is an over-reliance on the autonomy agent (Endsley 2017). Without a mechanism to help the human member understand the autonomy agent, it

becomes difficult to perform supervision tasks as a form of coordination within the teaming and to react properly when the autonomy agent experiences SA failures (Beavers & Hexmoor 2003; Shively et al. 2017). Transparency is a promising candidate to provide such a mechanism (Wright, Chen & Lakhmani 2019).

Some researchers suggested that transparency can be delivered through the autonomy agent's self-explanation abilities, and transparency can also help the human members properly calibrate their trust in this agent (Chen et al. 2018; Le 2002). Overwhelmingly, self-explanation abilities rely on two main methods, namely the component-based method and the process-based method (Hunt, Lee & Price 1993). However, these two methods encounter several complex issues. For example, they become unreliable when the number of logics and functions driving the autonomy agent's behaviours significantly increase (Harbers, Van Den Bosch & Meyer 2010). Furthermore, a self-explanation ability using a process-based method will have difficulty when an autonomy agent receives regular algorithm updates to obtain better prediction or behaviour performance. Therefore, it is necessary to overcome these issues.

In addition to having the ability to make its behaviour understandable, as a subordinate, an autonomy agent needs to have the ability to contribute to maintaining the SA of other team members, including its supervisors. For example, when a team member is distracted or misses some information, the teammates can refocus that team member's attention. Based on the aforementioned issues and challenges in HAT with vertical coordination, it is critical to embed an autonomy agent with abilities that make its behaviour understandable by its human teammate and to participate in maintaining the human teammate's SA. It is believed that these abilities can enhance coordination in a HAT context.

1.2. Research Questions

This section summarizes the main issues in this study and presents the research questions. The main issues are as follows:

Issue 1. The design of an autonomy agent should take into consideration its SA development as a team member. In this regard, the SA model is one way to model a situation-awareness-based team characteristic which relates to goals, the goal-related

environment, and SA requirements. As there are many team formations and each team can have multiple teaming situations, it is essential to develop a SA modelling framework for teaming situations. This framework can help in gaining a better understanding of the relationship complexities among team members in SA development.

Issue 2. Vertical coordination in teaming requires the supervision of subordinate team members. With an autonomy agent as a subordinate, it is difficult for a human to perform supervision tasks without a mechanism to make the autonomy agent's behaviours understandable and explainable (Endsley 2017; Langley et al. 2017). Also, when the autonomy agent is delegated to undertake certain tasks within the teaming, the human tends to overly trust the autonomy agent (Hoffman et al. 2013). Therefore, developing such a mechanism in a HAT context is significant to support the human's SA development and helps the human to calibrate their trust in the autonomy agent (Endsley 2017).

Issue 3. In a human teaming, each member can participate to maintain/develop the situation-awareness of other members including the supervisor (Salas et al. 1995; Seppänen et al. 2013). Applying this perspective to a HAT context, developing the non-human member's ability for such participation is a very critical part of an autonomy agent's design.

Issue 4. Making an autonomy agent transparent to support the human comprehension of an agent's behaviour is critical as an effort to enhance coordination in HAT. However, what level of transparency information that should be presented to the human teammate becomes another issue, particularly when the agent has to deal with a situation with a short time span. In such a case, the human teammate also has limited time to understand the agent's behaviour through the transparency information. Therefore, it is critical to develop a transparency model that can cope with the time constraint.

Three research questions (RQ) are raised based on the aforementioned issues (Issue 1 – Issue 4):

RQ1: How to synthesize the characteristics of SA models in different HAT contexts?

RQ2: How to develop coordination ability of an autonomy agent for a HAT context with vertical coordination?

RQ3: How to develop transparency model to deal with time constraint of situation?

RQ1 addresses Issue 1, RQ2 addresses Issues 2 and 3 and RQ3 addresses Issue 4.

1.3. Research Objectives

To answer the research questions, this research sets the following five research objectives:

Objective 1. Develop a framework to synthesize the characteristics of SA models in different HAT contexts including horizontal or vertical coordination or both. This framework will help determining the situation awareness requirements in different teaming context.

Objective 2. Develop a SA development process to implement SA models in a teaming with vertical coordination. This is a critical part for the autonomy agent to form its SA.

Objective 3. Develop a new method to improve an autonomy agent's self-explanation abilities to make its behaviours understandable. This method is based on the observation of the artificial situation awareness states of the autonomy agent and will outperform the two existing methods in terms of coping with complexity in generating behaviour explanations.

Objective 4. Develop an approach for the autonomy agent to observe its human teammate's behaviours in a collaborative driving situation. This approach enables an autonomy agent to detect its human teammate's distraction so that it can provide a support to maintain/develop their SA.

Objective 5. Develop a time-constraint based transparency model to control the visibility of an autonomy agent's behaviour for a given situation. This model will take into account the time-length of a situation as a human teammate has limited capability to absorb information in a short period of time.

1.4. Research Contributions

According to the research objectives, the research contributions of this study and their association with the research objectives are summarized as follows:

1) This study introduces a SA modelling framework for teaming situations (Objective 1)

2) This study develops a framework for situation awareness development process (Objective 2)

3) This study proposes a method to generate an autonomy agent's behaviour explanations based on its artificial situation awareness states (*ASAS*) (Objective 3). The evaluation of this proposed method shows that this method can reduce the complexities in generating behaviour explanations compared to the baseline method.

4) This study introduces a transfer learning method with a directive offline augmentation approach to generate the image classification model (Objective 4). The evaluation of this proposed approach indicates that the generated image classifier has better accuracy than the baseline methods.

5) This study introduces a time-constraint-driven transparency (TCDT) model as an approach to present transparency to a human teammate (Objective 5). This model is evaluated through a prototype developed using the CARLA autonomous driving simulator.

Figure 1-1 illustrates a block diagram of a HAT system that uses the framework, models, and methods developed in this study to highlight the contributions of this study. A HAT system can be described as a system unifying the functions in which a human teammate interacts with the autonomy agent in a teaming. The block "SA requirements" will be determined by going through the SA modelling framework for teaming situations (Contribution 1). Through this framework, this study becomes the first initiator to characterize teaming characteristics using SA model. Furthermore, the block "SA development process" implements the framework of SA development process (Contribution 2) which is useful to develop the autonomy agent's behaviour based on SA oriented design. The blocks "Artificial Situation Awareness States" and "Self-explanation ability" are developed based on Contribution 3 which generates explanation on the

autonomy agent's behaviour. The novelty of this contribution is that in generating behaviour explanations, this study focuses on observing the artificial situation awareness states of the autonomy agent while other studies focus on logics and hardware states.

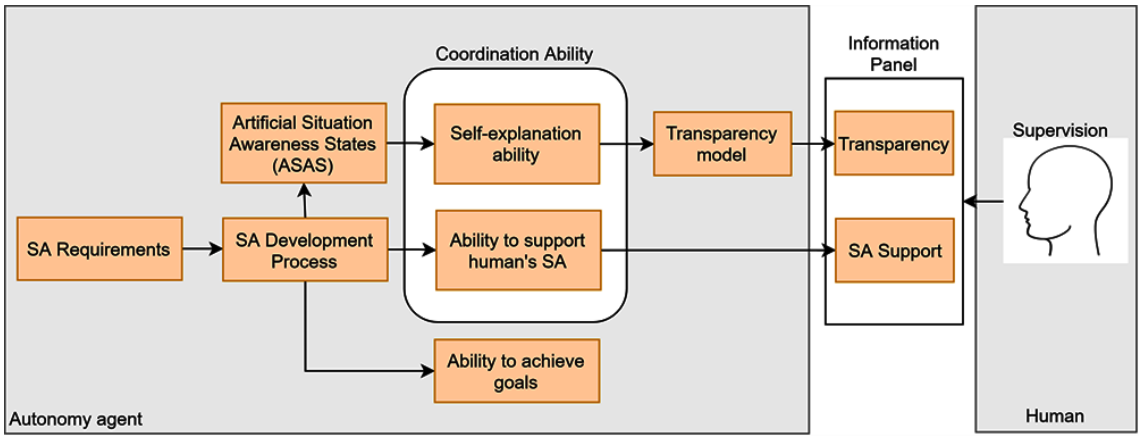


Figure 1-1. The big picture of research contributions

The block "Ability to support human's SA" is developed based on Contribution 4, which can recognize human's activities using images captured by a camera. Such recognition is critical to support human's SA. The novelty of this contribution is that the recognition model is trained under a transfer learning setting with a new transfer learning approach that can cope with the geometrical relation constraints using directive offline augmentation. Lastly, the block "Transparency model" is developed based on TCDT model (Contribution 5) which enables a human teammate to comprehend autonomy agent's behaviour through transparency. Its novelty is to take into account the time-constraints of situations to make transparency feasible in various situations while other transparency models only focus on kinds of information for transparency.

1.5. Research Methodology

Several research methodologies, such as case studies, field studies and experimentation, laboratory experimentation, and action research have been proposed in the domain of information systems. In this research, the design science research methodology (Peffers et al. 2007) is adopted. Using this methodology, problem-centred initiation becomes the entry point of this research. The design science research methodology has six stages, as illustrated in Figure 1-2.

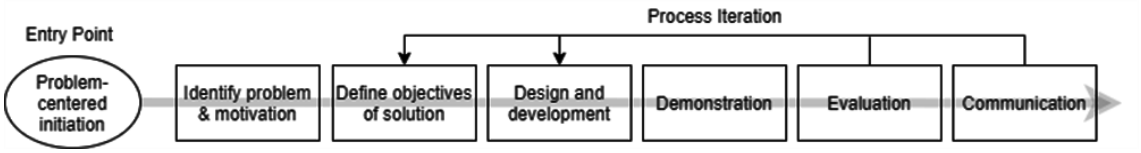


Figure 1-2. Design Science Research Methodology

1) **Problem identification and motivation.** The topic of this research was selected based on a systematic literature review and the author's personal interest and observation. A literature review of the previous research in the topic area is a critical component of the research process in this stage to define the state-of-the-art of the study. The relevant literature on the topic was collected and critically reviewed. The findings of the systematic literature review helped to define the specific research questions for this study.

2) **Define objectives of solution.** Several research objectives are defined for this study based on the author's knowledge at to what is feasible and possible to answer the research questions. A set of tasks was designed to achieve these research objectives. The strengths and weaknesses of the existing methods are identified to direct the development of new techniques and frameworks to complete the tasks.

3) **Design and development.** In this stage, the solution design and development are determined.

   Regarding developing a SA modelling framework, this study starts by retrieving the literature on team situations in various domains such as aviation, robotics, military, process industry, maritime, and other related domains. After this literature search stage, this study synthesizes the characteristics of existing team situations by extracting their common and special features. The common and special features are critical to show their similarities and differences. Based on the identified features, a general framework is developed to generate a SA model for any possible team formation including ones that involve vertical and horizontal coordination.

   Next, a vertical coordination support system for an autonomy agent is developed based on the characteristics of the new supportive situation awareness (SSA) model. This coordination support system consists of two critical components to ensure the autonomy agent's behaviour is understandable by its human teammate (self-

explanation ability) and to maintain the human teammate's SA by classifying their behaviour (human behaviour classification ability).

The solution design includes selecting the study cases, defining the dataset plans, choosing the relevant sensory tools, and selecting the programming language and mathematical techniques. The design phase emphasizes the novelty of the proposed solution and the design of the artefacts. Based on the dataset plans, the data collection is done by developing the software to connect the sensory tools and acquire the data. This research uses several sensors to collect the data, namely GPS sensor, OBD-2, LW-20 sensor, and a camera. Additionally, regarding the human behaviour classification ability, this study uses an existing dataset from Kaggle as the source domain. Another dataset that is used in the target domain is a self-collected dataset which has a similar classification to the source domain. The sample images in each dataset have 10 classes representing the classifications of human behaviours while performing their tasks as observed by the non-human agent. Based on the problems identified in existing self-explanation methods and human behaviour classification methods, new methods are proposed.

This stage produces a prototype as the research artefact to verify the proposed methods.

4) **Demonstration**. The developed artefact is tested to solve example problems from selected study cases.

5) **Evaluation**. To evaluate the proposed artefact, this study selects methods from existing studies as the benchmarking methods. A set of criteria is developed to determine whether the proposed artefact is better than the benchmarking methods. Regarding the self-explanation methods, the proposed method is evaluated in terms of how it is able to deal with complex issues in the process-based method. This study evaluates the human behaviour classification methods based on its accuracy performance. When the result is not good enough, this study returns to the design and development stage to make the necessary improvements to the proposed artefact. This process continues until the artefact is suitable.

6) **Communication**. The aim of this stage is to obtain a wide audience to communicate the novelties of the artefact, its utilities, and its effectiveness. A prestigious conference and high-quality peer-reviewed journals are selected as the outlets through which to communicate the research results. Furthermore, demo programs are provided to illustrate how the newly developed techniques are used in certain situations. This study provides three videos showing the implementation of autonomy agent's self-explanation ability and the TCDT model using a simulator. The first video[1] is about overtaking case using the process-based method. The second[2] and the third[3] videos are about overtaking case using the *ASAS*-based method and traffic light case using the *ASAS*-based method, respectively. Regarding human distraction recognition, the demo program can be downloaded from https://bit.ly/2QDAUaZ.

1.6. Thesis Structure

This thesis consists of nine chapters as presented in Table 1-1. Chapter 1 presents the research background, research problems, objectives, contributions, methodology, and the thesis structure.

Chapter 2 presents the literature review that focuses on three categories of information. One category is the literature covering the definition of automation as the basis of an autonomous agent, an autonomy agent, and a human-autonomy teaming. These definitions are important to establish a distinction between automation, which becomes the basis of autonomous agents, and autonomy agents. The second category comprises three parts: (i) a systematic review of SA theory in individual, autonomy agent, and teaming contexts; (ii) a systematic review of transparency and the existing me[4]thods in generating behaviour explanations highlighting the strengths and weaknesses of these methods; and (iii) a systematic review of the latest developments in relation to transfer learning techniques for image classification problems. The last category presents the basic

[1] https://youtu.be/_ssYcEJQ6nE
[2] https://youtu.be/9_vz_jHd890
[3] https://youtu.be/uLbMpliFHS0

techniques used in the proposed methods and frameworks of this study, including Bayesian networks, fuzzy theory, Manhattan distance, and neural networks.

Table 1-1. Thesis structure

Thesis Chapter	Chapter Objectives
Chapter 1: Introduction	Present the research background, research problems, objectives, contributions, methodology, and the thesis structure
Chapter 2: Literature review	Provide a systematic review on literatures to identify the gap of existing studies
Chapter 3: Situation awareness modelling framework for teaming situations	Present the development and the outcome of synthesizing the characteristics of SA models in different HAT context
Chapter 4: Development of an autonomy agent's situation awareness	Develop a framework for situation awareness development process and its implementation
Chapter 5: Coordination ability of an autonomy agent	Provide examples of coordination in the selected SSA-based HAT case and the proposal of autonomy agent's self-explanation abilities to support coordination
Chapter 6: Autonomy agent's behaviour explanation using ASAS-based method	Present the implementation of the proposed method for self-explanation ability
Chapter 7: Driver distraction recognition	Present the proposed approach to develop a model to recognize distracted human's behaviour as part of autonomy agent's coordination ability
Chapter 8: Time-constraint-driven transparency model	Introduce time-constraint-driver transparency model as an approach to present transparency information to human teammate
Chapter 9: Conclusion and future works	Draw a conclusion of this study and possible future works

Chapter 3 presents the development and outcomes from achieving Objective 1. To synthesize the teaming characteristics, this study starts from analysing team goals, team members, and the goal provision among team members. Based on goal provision, SA model is defined by identifying goal relationship using SA related factors that include goal-related environment of team members, SA elements, and SA requirements to form SA.

Chapter 4 focuses on Objective 2 to develop the autonomy agent's SA. In this chapter, the situation awareness development process (SADP) for an autonomy agent is proposed. SADP consists of three phases, namely perception phase, comprehension phase, and projection phase.

Chapter 5 focuses on the coordination ability of an autonomy agent involved in SSA-based HAT. This chapter provides examples of coordination in the selected SSA-based

HAT case, which is the collaborative driving context. Based on this context, the proposals for coordination abilities covered in Objective 3 and Objective 4 are developed. This chapter also introduces the proposed method to generate behaviour explanations to achieve Objective 3.

Chapter 6 presents the implementation of the proposed method to generate behaviour explanation in a collaborative driving context. The implementation includes two study cases which are under traffic light and overtaking situations. The evaluation of the proposed method is provided at the end of the chapter to show how the proposed method can address the issues in the existing methods.

Chapter 7 presents how an autonomy agent can recognize and classify a human's driving behaviour as normal behaviour and distracted behaviour from image data obtained from an on-board camera to achieve Objective 4. It presents in detail the transfer-learning based driving distraction detection approach. The validity of the newly proposed approach is evidenced by a thorough comparison of the classification performance of the new approach with the results of three baseline transfer learning methods using images containing ten types of driving distractions.

Chapter 8 introduces the TCDT model as the outcome of Objective 5. This framework divides transparency themes into four groups. In the first group, transparency information about behaviour outcomes and prediction is presented. Then, the information on agent intention is provided in the second group. The third group includes the transparency of plans, decisions, and coordination. Finally, the last group covers the agent's behaviour explanation and activities.

Finally, Chapter 9 draws conclusions from this study and makes suggestions for future work.

Chapter 2 : Literature Review

2.1. Introduction

This chapter presents a literature review which provides a discussion of the state-of-the-art related existing works in the thesis topic areas, including autonomy agents, transparency of autonomy agents in human-autonomy teaming (HAT), situation awareness (SA) models and applications in HAT, and transfer learning techniques for cross-domain image-based behaviour detection. Based on the findings of the literature review, the research gaps are identified to support the selection of the research questions. Table 2-1 presents the overview of the literature reviewed and associated sections in this chapter.

Table 2-1. Overview of the literature reviewed

Literature theme	Chapter Section
Provides a review of autonomy agent definitions in HAT to determine which side of the argument this study takes	Section 2.2
Introduces SA theory and presents a review of the existing SA models	Section 2.3
Presents a review of the transparency of autonomy agents in HAT and the existing methods used to generate self-explanations	Section 2.4
Presents a review of the transfer learning techniques for cross-domain image-based human behaviour detection	Section 2.5
Summarizes the findings of the literature review and lists the identified research gaps	Section 2.6

2.2. Working with Intelligent Agents: From Automation to Autonomy

This section presents the definitions of automation and autonomy agents, which are important for drawing a line between them. As autonomy is added, researchers have different perspectives as to whether human involvement is needed to override the agent's

decisions and actions. Finally, this section draws a conclusion about which perspective is used in this study regarding the definition of an autonomy agent.

2.2.1. Automation

Automation can be defined as the execution by an artificial agent (usually a computer, also referred to as an autonomous agent) of a function that was previously performed by a human (Parasuraman & Riley 1997). Once this function is completely transferred to a machine, automation will be simply a machine operation, therefore, today's automation could be tomorrow's machine operation. Automation can be immensely helpful to reduce the time-consuming and intensive activities performed by humans. This includes the automation of cognitive functions such as decision-making, creative thinking, and planning. In many systems, automation often deals with complexities which lead to different forms of automation considering numerous factors in its implementation.

Parasuraman (2000) and Endsley & Jones (2011) state that there are four functions, namely F1 to F4, of a system that can be applied for automation, namely:

1) information collection (F1)
2) information analysis (F2)
3) selecting decision and action (F3)
4) the implementation of action (F4)

Automation in collecting information may simply use a set of sensors to obtain low-level data or raw data for further observation and analysis. This includes the function to organize, highlight, or even filter the data to attract the operator's attention. The analysis function involves a higher level of cognitive function which allows automation to calculate predictions, extrapolation over time, or information integration. In decision and action selection, automation involves generating a hypothesis and choosing an action from the decision alternatives. In this regard, a certain degree of human-decision making is replaced by machine decision-making. The selected decision can be further processed by the next function, action implementation. The automation of action implementation can be defined by switching manual activities to automatic activities in order to execute

the responses, with or without operator consent (Endsley 1999; Endsley & Jones 2011a). Stanton et al. (2006) argued that SA can be held by either human or non-human agents. Table 2-2 presents how these four functions are implemented in the various systems.

Table 2-2. Level of Automation in Various Systems (Endsley 2017)

Level of Automation	Description	F1	F2	F3	F4
Manual control	Human controls all functions	H	H	H	H
Information cueing	Computer highlights key information	HC	H	H	H
Situation awareness support	Computer highlights and integrates information	HC	H	H	H
Action support/tele-operation	Computer helps in performing instructed actions	HC	H	H	HC
Batch processing	Computer executes tasks instructed by human	HC	H	H	C
Shared control	Human and computer make decision options; human selects which decisions and action to be taken; and human performs actions while computer provides support	HC	HC	H	HC
Decision support	Computer generates options, human decides, computer executes	HC	HC	H	C
Blended decision making	Computer generates options and recommends the best, human selects (can overrides), system executes	HC	HC	HC	C
Automated decision	Computer generates and selects options, then executes	HC	HC	C	C
Supervisory control	Same as automated decision, but human can intervene	HC	C	C	C
Full automation	Computer controls all functions	C	C	C	C

Notes:
H = Human, HC = Human and computer, C = Computer
F1 = Information Collection, F2 = Information Analysis, F3 = Decision and action selection, F4 = action implementation

Table 2-2 shows that every system has numerous variations in relation to how automation is implemented. In a fully automated system, all functions (F1 to F4) are controlled by a computer. In manual systems, it is a human who performs all the functions. In between is a system which is a mix of manual and automation functions, i.e., each function can be executed by either a human or a computer or both. When this system is intended to produce information, decision options, and action recommendations, human-system interaction has two main characteristics called parallel interaction and serial interaction (Endsley & Jones 2011a). Serial interaction refers to a situation in which the human (a

system user) only obtains information provided by the system and reacts accordingly. However, in parallel interaction, the human (a system user) can perform a double check on the information or alerts sent by the system using their own method, which is usually different from the algorithms used by the system (Inagaki 2003; Madhavan & Wiegmann 2005). Figure 2-1 shows how parallel interaction differs from serial interaction.

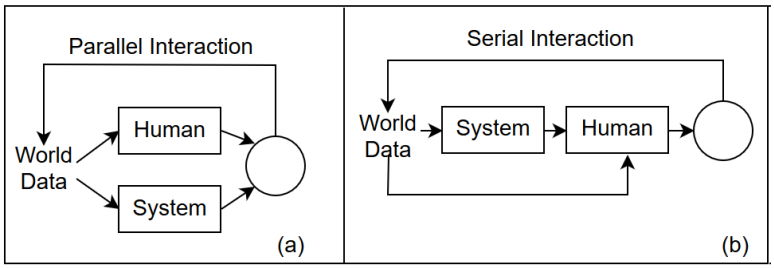


Figure 2-1. Human-system interaction: a) parallel interaction, b) serial interaction

Theoretically, it is believed that parallel interaction has better reliability than serial interaction. Let R denote reliability which can be determined by the following formula:

$$R = \begin{cases} 1-(1-HR)(1-MR), & parallel\ system \\ (HR)(MR), & serial\ system \end{cases}$$

where HR and MR are the percentage of human reliability and machine reliability, respectively. For example, if HR = 90% and MR = 85%, then the reliability of a parallel interaction is 98% whereas the serial interaction's reliability is only 77%.

2.2.2. Autonomy Agent

Even though it is claimed that a fully automated system does not require human intervention, some researchers argue that humans should be kept in the control loop as the final authority of automation (Inagaki 2003). Based on this perspective, McNeese et al. (2018) defined that when no human intervention is required in relation to an artificial agent's decisions and actions in automation, this agent is called an autonomy agent. An autonomy agent comprises a set of intelligence-based capabilities that enables this agent to react to situations that were not programmed or anticipated in its design. However, another perspective says that the main difference between automation and autonomy is that automation tends to be designed to work in a very strictly regulated environment (Smithers 1997). However, autonomy can work in a less regulated environment but human intervention is still required when given circumstances are out of the task

boundary of the autonomy agent (Bainbridge 1983; Cox 2013). In this context, such autonomy is often called adjustable autonomy (Hexmoor 2001). Adjustable autonomy is also a preferred definition used in a HAT context because a human teammate can have the authority to intervene in relation to the autonomy agent's behaviour.

Endsley et al. (2003) and Stanton et al. (2006) stated that goals are useful to define static knowledge such as rules, procedures, and general system knowledge. In other words, goals can be the driving factors of an autonomy agent's behaviour. Cox (2013) divides a goal-based autonomy into two groups, namely goal-following autonomy and goal-driving autonomy. The first group, goal-following autonomy, is defined as an autonomy agent that can accept a goal from another party (i.e. human teammate) and can automatically accomplish the goal by conducting a sequence of actions within its goal-related environment (Barber & Martin 1999). For example, an autonomy agent receives a goal from a driver to go to an intended destination, and then, it can drive the vehicle automatically to the destination. If the environment does not cooperate, complex tasks that cannot be specified will rely on human communication intervention (Castelfranchi & Falcone 2003b; Cox 2013). In this regard, it is implied that autonomy should balance its own states strategically in association with what is happening within the environment and what it is trying to achieve.

Goal-driving autonomy concerns the higher-level management of goals, knowledge, plans, and activities, not simply goal accomplishment. In this type of goal-based autonomy, an autonomy agent has the ability to self-manage its own goals, and it can learn from its failures to improve its performance (Vattam et al. 2013). There are three main reasons for an autonomy agent to self-manage goals. The first reason is to have a *rational anomaly response*, so that an autonomy agent will have a better response to situations against autonomy agent's interests. The second reason is called *graceful degradation* which makes an autonomy agent degrade its goals as the current goals are no longer accomplishable. Lastly, the third reason is for better *future performance*. In this regard, an autonomy agent needs to avoid the states that have a possibility of resulting in a failure to achieve its goals. In addition to these three reasons, self-managing goals can also come from the human expectations which an autonomy agent has to fulfill (Beavers & Hexmoor 2003).

2.2.3. Human-Autonomy Teaming

With more sophisticated technologies, an autonomy agent is allowed to perform collaborative work with a human as a teammate and not merely as a supporting tool. McNeese et al. (2018) described such collaborative work as Human-Autonomy Teaming (HAT). Minimally, HAT involves one human and one autonomy agent working interdependently over time to complete a task (Demir et al. 2019). Some researchers opined that to some extent, an autonomy agent can replace and is even equivalent to a human counterpart as a team member. Consequently, it is necessary to redefine a team in the HAT context from an actor-agent community perspective which views the autonomy agent as merely a tool which has a role in a team to support daily decision-making processes and managerial activities (Wijngaards et al. 2006). By far, this community still views that such autonomy agents are still facing a large obstacle in becoming an equivalent team member replacing human (Klien et al. 2004).

2.2.4. Summary of Human-Autonomy Teaming

In the current literature, the definition of an autonomy agent is still debated, mostly focusing on whether human intervention is needed or not in an agent's decisions and actions. In a strictly regulated environment, it is easy for an autonomy agent to perform its tasks as designed, particularly because all possible obstacles in an environment can be identified and minimized so that the agent can work properly. However, when the environment is not specifically designed for the autonomy agent, adjustable autonomy is preferred where the human teammates (particularly ones with supervisory roles) can direct their artificial agent teammates if the situation development is unexpected. Therefore, the definition of an autonomy agent held by this study is in line with the perspective of adjustable autonomy where all team members are supposed to be directable.

2.3. Theory of Situation Awareness

This section reviews the theories of SA that define SA, individual SA, the autonomy agent's SA, and SA in a teaming context. These theories are significant to provide a common understanding of how an autonomy agent can play a role that is equivalent to a human counterpart in a teaming context.

2.3.1. Concept of Situation Awareness

Since World War I, SA has been regarded as a critical component in the military to support the safe operation of aircraft (Endsley 1995). To describe SA, researchers have many perspectives regarding theoretical underpinning. According to Stanton et al. (2017), there are four main views of SA: 1) as a three-level representation, 2) as a perceptual cycle, 3) as attention and activity, and 4) as a holistic framework.

Three-level representation was firstly introduced by Endsley (1995) who described SA as "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future". In other words, SA can be represented by a hierarchy of three levels of awareness. SA level 1 (SA-Level 1), referred to as the perception level, is the interpretation of an environment's elements which requires the individual's knowledge about their surroundings and own conditions. SA level 2 (SA-Level 2), referred to as the comprehension level, is the development of a higher-level understanding of a situation by combining information about the status of all environment elements. SA level 3 (SA-Level 3), referred to as the projection level, concerns the ability to make projections for immediate future actions based on the individual's situation understanding formed in previous SA levels. Endsley (1995) also refers to SA as a situation model. Therefore, the perception level can also be considered as the collection of situation attributes within the environment while the comprehension level is the identified situation based on the status of collected attributes. The projection level is for situation-based action selection. One example usage of this three-level representation is demonstrated by (Feng, Teng & Tan 2009) in context-aware decision support.

Perceptual cycle theory was introduced by Niesser (1976) to identify the relation between perception and mental imaginary. Several researchers such as Adams, Tenney & Pew (1995) and Smith & Hancock (1995) tried to relate it to SA. In this regard, perceptual cycle theory views that every individual has a cyclical perception process of consciousness which always inspects their internal and external conditions of the surrounding environment. The part of adaptive consciousness that is externally directed is referred to as SA. SA relies on internal conditions as it holds an individual's knowledge, capacities, and beliefs.

Furthermore, Bedny & Meister (1999) relate attention and activities to SA and believe that SA is about consciousness and dynamic reflection on situations. In this regard, reflection provides an insight from the past, current, and future to seize opportunities and potential features from situations. Consequently, SA can also be referred to as the state of knowledge.

From the above three main views of SA, it can be inferred that SA is a state within an individual which updates their situational knowledge given the situations within an environment. The three-level representation of SA includes the projection of actions that will be taken in response to situations while the others only focus on updating the individual's knowledge (SA-Level 2). While there are many arguments as to whether SA is the state of consciousness, knowledge, or mind, Lundberg (2015) introduces SA as a holistic framework which uses the term SA state independently even though it might be a part of the state of consciousness, knowledge, or mind. The term SA is used by Lundberg (2015) to referring to the process to form SA state. A SA state can be described by three components: frames, implications, and objects on an event horizon during the dynamic development of plans. In sum, the SA state is an interpretation of implications which may be caused by objects in a certain frame of time within an event horizon.

Of the many definitions of SA, some researchers prefer to simplify the definition of SA. For example, Johannsdottir & Herdman (2010) describe SA as the people's interpretation and understanding about surrounding situations. In this study, we adopt the three-level representation of SA and refer to Lunberg's SA as the SA development process (SADP). We also view this SADP as a procedure that starts by capturing raw data from the surrounding elements in the environment to projecting immediate future actions. The result of this process is SA as a three-level hierarchy of awareness. For example, during driving, a driver continuously evaluates the surrounding situation by collecting information about the status of a leading vehicle if one exists, the status of road lines, and the status of road signs. Understanding the meaning of the status of each element can be considered as SA development at the perception level (SA-Level 1). When the leading vehicle is too slow, the road line type is dashed, and there are no hazardous situations detected, the driver starts to develop their understanding at the comprehension level (SA-Level 2) that this kind of situation is a safe driving situation to overtake the leading

vehicle. Based on this situation understanding, the driver's projection is developed at the projection level (SA-Level 3) by considering whether to overtake or to maintain a tailing manoeuvre.

Since an autonomy agent can be considered to hold its own SA to a certain degree (Stanton et al. 2006, 2017), the concept of SA can also be applied to autonomy agents. McAree & Chen (2013) use the term artificial SA for such an autonomy agent, and describe this term using Endsley's three-level hierarchy of SA, as illustrated in Figure 2-2. In this figure, the perception level is the raw data obtained from the sensory tools such as the global position of the autonomy agent. By combining the data collected from the environment with embedded prior knowledge, the autonomy agent develops its higher-level situation understanding which becomes the basis of its near future actions.

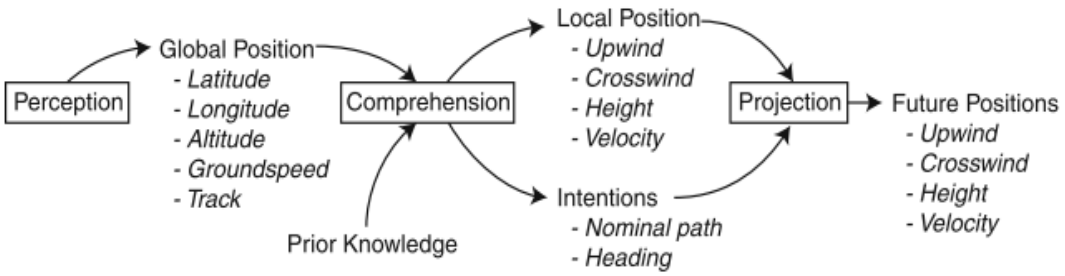


Figure 2-2. Artificial Situation Awareness

2.3.2. Factors Involved in the Situation Awareness Development Process (SADP)

According to Endsley (1995), SA is "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future". The implicit part in this description is an entity that perceives, comprehends and projects. We infer this entity as the owner of SA. SADP is affected by the internal factor of the owner of SA. An experiment performed by Saner et al. (2009) indicates that different individuals can have different SA for the same situations. Having different knowledge due to situations, different mental models (i.e., ignorant, inattentive), and experience become the examples of the internal factor of the owners for having such different situation understanding.

The goal and goal-related environment are another two essential factors in SADP. In many complex environments, goals become the basis of the decision-making process (Endsley et al. 2003). Also, goals are critical to set the boundary of the environment within the space of the universe. The environment within this boundary is called the goal-related environment. In this regard, in order to develop their SA, an individual will only focus on SA elements within the goal-related environment. SA elements refer to the things or objects that need to be perceived and understood by the owner to develop SA to achieve the goal. Any information required in association with the goal is referred to as SA requirements.

A goal can be possessed by one or more individuals. If a goal is possessed by an individual, i.e., this individual is the only player to achieve the goal, the SA is referred to as individual SA. In other words, individual SA can be defined as the developed situation understanding of an individual as a single player in their goal achievement process so that all decisions are made based on their situation comprehension only and will be executed by them. For example, according to Fastenmeier & Gstalter (2007), in a conventional driving situation, a driver has three main goals: navigating, controlling, and continuous monitoring. The driver is a single player to achieve the driving goals and is responsible for all these goals. The goals produce the driver's responsibilities in the goal-related environment. Also, all decisions to respond to situations within the driving environment are only made by the driver. In this regard, the driver's SA is referred to as an individual SA.

If the goal is shared by a team consisting of two or more individuals, each individual plays a specified role to achieve the team goal (Salas, Sims & Burke 2005). SA is referred to as team SA. More details about team SA are reviewed in the next sub-section.

2.3.3. Situation Awareness in a Teaming Situation

Teaming situation can be defined as *the involvement of multiple players in the achievement of team goals*. There are two kinds of SA in a team, namely team members' SA and team SA. A team member's SA is described as the degree to which a team member possesses SA of their own responsibility (Endsley & Robertson 2000). Salas et al. (1995) define team SA as a shared understanding among multi-individuals as the team members

at a certain point of time. The bridge to develop team SA from team members' SA is referred to as coordination (Stanton et al. 2017). During coordination, team members exchange information with each other, including explaining their actions within their responsibilities to their superior.

Even though Salas et al. (1995) view team SA as shared understanding among team members, there are arguments as to whether team SA should be possessed by all team members or not (Gorman, Cooke & Winner 2006). There is a case where a situation is only shared within a group of people within a team as only these people have authority in this situation. In such a case (where not all team members are involved), shared situation understanding can also be referred to as team SA, hence team SA is merely an updated individual team member's SA after obtaining new information about ongoing situations from their teammates.

Regarding SA in teaming situations, there are several examples of teaming situations in the existing studies. For example, Endsley (1995) identified two forms of teaming situations. In the first teaming situation, each team member has their own goals which are independent from each other, and therefore, they have to deal with completely different environments within their responsibilities. In this study, we refer to this teaming situation as a normal teaming situation. In such a situation, a team member may not need to observe other team members because each team member has a completely different environment and responsibilities. Thus, an autonomy agent involved in a normal teaming situation may not need to include its human teammates in its SA. In the second teaming situation, team members share an environment because their goals have some overlap and they have shared tasks to achieve the shared goals. We refer to this second teaming situation as a shared teaming situation in this study. In addition to the above two teaming situations, Shu & Furuta (2005) described another teaming situation in which the shared goals are described as a collaborative goal so that each member shares a common environment, but for these shared goals, each member has different independent responsibilities. We refer to this third teaming situation as the mutual teaming situation in this study.

2.3.4. Findings and Remarks

With the advancement of technologies, an autonomy agent with artificial SA is capable of being a team member to perform collaborative work with human. By having its own artificial SA, a teaming between a human agent and an autonomy agent can be established so that the autonomy agent becomes an equivalent teammate to a human counterpart. Since artificial SA is required not only to perform the autonomy agent's tasks, but also to perform coordination with its human teammate, the autonomy agent's design should be viewed from its SA development perspective by identifying its SA requirements (Endsley 2001). In teaming situations, the autonomy agent's design process can start from understanding the goals assigned to this agent. In this regard, goals are the specification of the autonomy agent's roles within the team that this agent is supposed to perform (Harbers, Van Den Bosch & Meyer 2010). After the goals are identified, the autonomy agent's designer can identify its SA requirements, which provide key information to maximize the artificial SA development process (Endsley & Jones 2011).

One outstanding challenge is how to develop artificial SA for an autonomy agent in a teaming context, especially when a teaming situation is complex regarding relationships between team members.

To promote the autonomy agent's design from a SA development perspective, it is highly desirable to develop a framework that can conceptualize SA models in teaming situations, particularly in a HAT context. We refer to a SA model as a description of the relations of factors involved in developing the SA of team members, such as team members' goals, team members' goal-related environments, and their SA requirements for a given team formation.

Using the SA models in a teaming context, SA requirements can be derived from the relation between team members' goals and the goal-related environment. From this relation, every element either physical or abstract within the goal-related environment can be identified. Then, the kind of information about such elements required to develop the autonomy agent's SA can be inferred. Designers of any system including an autonomy agent need to provide the system components and how these components fit together (Johnson, Bradshaw & Feltovich 2018). Once the autonomy agent's designer identifies

the information needed by an autonomy agent for its SA development, they can decide what kind of architecture or platform is required to collect such information from the environment and to plan the agent's behaviour accordingly.

2.4. Challenges in Human-Autonomy Teaming

Establishing HAT is not an easy task. This kind of teaming was firstly introduced in the late 1980s under the term human-electronic crew by Taylor (1988). In the 1990s, some researchers started using the term human-autonomous teaming (Sklar & Sarter 1999). In the early 2000s, the term human-autonomy teaming was introduced. To play the role of a teammate, some researchers view that an autonomy agent should be able to maintain the common-ground and direct each other's attention to critical cues, activities and situation changes (Sarter & Woods 2000). Moreover, an autonomy agent should be able to maintain the big picture about the current plan status, detect possible failures, evaluate the changes to plans, manage unexpected situations, agent re-tasking, and adjust the way to communicate with team members (Allen & Ferguson 2002). Klien et al. (2004) identified the requirements for an autonomy agent to be equivalent to the human counterpart in a teaming by: (a) having a common understanding of joint goals, (b) communicating the reasons for its decisions and behaviours, (c) being observable to each other, (d) being directable by a mature policy to make an autonomy agent predictable, (e) being able to understand, share and interpret the status and intentions of each other, (f) being able to adapt to strategic changes, such as goal, plan, re-tasking, (g) being able to inform when the given tasks cannot be done, (h) managing teammates' attention, and (i) managing the cost of coordination.

From the aforementioned requirements, it can be seen that the main problem of an autonomy agent being equivalent to a human counterpart is to perform coordination like a human teammate. Researchers have expended huge efforts to achieve these requirements. In this achievement process, the human still underestimates the ability of the intelligence inside an autonomy agent to confidently predict its human teammates. Ironically, giving the artificial agent the ability to be more adaptable to the environment will make it less predictable to humans (Zhong et al. 2004). Furthermore, a lesson from automation is that humans tend to wonder about what the autonomy agent is currently doing, and what is next (Theodorou, Wortham & Bryson 2016). However, providing an

autonomy agent's past and immediate future behaviour explanations is not an easy task (Chen et al. 2018).

Even though the HAT challenges were posed a long time ago, they still attract the attention of many researchers. For example, to design the behaviour of the autonomy agent in the problem domain, Radecký, Gajdoš & Fasuga (2010) introduced the behaviour diagram. Cox (2013) introduced a goal-driven approach to help the autonomy agent's designer to provide the agent's problem domain, focus, coordination, and direction. This includes how to develop a policy to regulate the agent's behaviour. Endsley (2017) proposed a human-autonomy system oversight model to depict the key system design features to assist the human cognitive process, such as information presentation and salience, predictability and transparency. Brandt et al. (2017) used the flight-following task to demonstrate transparency in terms of perception states, bi-directional communications to maintain the common ground, and conveying tasks that are unable to be performed by a human teammate. Chen et al. (2018) simulated the transparency effectiveness of an autonomous robot team. Even though humans can send signals to an autonomy agent to establish coordination activities, it is not sufficient because the autonomy agent should also have the ability to interpret the human's status and attention. In this regard, it is difficult for this autonomy agent to recognize the human's stance, knowledge, mental model or goals given the evolution of the plan in progress (Christoffersen & Woods 2002; Shively et al. 2017).

As coordination is a critical key in a teaming, Harbers et al. (2011) and Demir et al. (2019) stated that there are two main functions of the autonomy agent's coordination ability. In addition to explaining the internal states of an autonomy agent, a coordination ability includes the observation of its human teammate to recognize their states and behaviour. Explaining the internal state includes communicating strategic information and activities such as a change of plans, possible failures, and receiving input from a human teammate for re-tasking and direction. Therefore, developing the two main functions of an autonomy agent's coordination ability becomes the representation to answer the HAT challenges.

Developing a coordination ability requires at least four main tasks. First, the autonomy agent's designer needs to understand the teaming characteristics that describe the relation

between the autonomy agent and the human teammate. The extensive literature review of teaming characteristics based on the SA perspective were provided in Section 2.3. Second, it is necessary to define what needs to be explained to make an autonomy agent observable and understandable to a human teammate. The third task is how to generate explanations on the autonomy agent's behaviour. The behaviour explanations can also cover strategical information such as a change of plans. The literature review for the second and third tasks is covered in Sections 2.5 and 2.6. The fourth task involves an autonomy agent understanding a human teammate, using camera-captured images which are a popular approach to classify a human teammate's behaviour/activities. To develop an image classifier, one of the many approaches is transfer learning. The literature review on transfer-learning-based image classification is presented in Section 2.6. The result of the human teammate's activities/behaviour classification is also useful to provide an autonomy agent with the ability to participate in maintaining/managing attention. For example, if an autonomy agent detects that a human is exhibiting distracted behaviour, the autonomy agent can send an alert to draw the human's attention back to their task.

To address the HAT challenges, developing the autonomy agent's coordination ability is the main concern in this study. Overwhelmingly, the existing studies have proposed that autonomy agents need coordination abilities. For example, Demir et al. (2019) develop such an ability for a teaming with the team model, and Endsley (1995) and Endsley & Jones (2011b) for a teaming with a shared model. Push notification and sharing information become the main modules in the coordination ability for a team model and shared model, respectively. As the proposed coordination ability in the existing literature only focuses on a teaming in which all team members have a non-shared goal, how to develop a coordination ability in a teaming with a superior-subordinate relationship becomes the main research question in this study.

2.5. Self-explanation Abilities of Non-Human Agents

The self-explanation abilities of an autonomy agent refer to its capability to explain its behaviours including actions, recommendations, and reason behind their choices so that they become observable and understandable by humans (Anjomshoae et al. 2019; Fähndrich, Ahrndt & Albayrak 2013; Langley et al. 2017). The importance of such

abilities is empirically proven as it can increase the confidence and trust of the human teammate in the HAT context (Biran & Cotton 2017).

Extensive studies on an agent's self-explanation abilities have been conducted in many domains, such as robotics, games, human factors, and human-computer interaction (HCI), particularly to clarify the agent's behaviours so that it is understandable to humans. These studies can be divided into two groups. The first group focusses on what can be explained by the agent for increased transparency to humans. The second group focuses on behaviour explanation generation. The behaviour of an autonomy agent can be viewed from two perspectives, namely its component (physical) level and process level (program) (Castelfranchi & Falcone 2003a; Hunt, Lee & Price 1993). Therefore, this study uses the terms component-based method and process-based method to refer to the explanation generation techniques that exploit the component level and process level, respectively.

This section starts by describing transparency, autonomy agent types, and then presents self-explanation generation using component-based and process-based methods. At the end of this section, a summary is given to overview the strength and weaknesses of the existing behaviour explanation methods.

2.5.1. Transparency

In the HAT context, the fundamental problem is whether the human teammate is able to trust and rely on an autonomy agent. A human teammate's experiences in working with such an agent produces a default trust and negative trust (Hoffman et al. 2013). Default trust can be achieved when the autonomy agent performs its task under ideal conditions and circumstances. However, when the autonomy agent's sensor is miscalibrated or there is a biased interference algorithm, these problems can generate negative trust from the human teammate. Therefore, making the autonomy agent's behaviour explainable and understandable would help the human teammate to enhance their trust and achieve their goals in a situation in which vulnerability and uncertainties exist (Lee & See 2004).

Transparency can be described as the extent to which the autonomy agent's behaviour is understandable and predictable to a human teammate to calibrate trust in the autonomy agent (Endsley 2017; Janssen et al. 2019; Neuhaus et al. 2019; Ososky et al. 2014; Theodorou, Wortham & Bryson 2016). Through transparency, the human teammate can

enhance their situational awareness on this agent and receive strong cues to map his or her expectations of this agent to his or her mental model states so that his or her trust and reliance on this agent can be adjusted or calibrated. A mental model can be described as the knowledge about how things work and the awareness to adequately monitor the autonomy agent (Lundberg 2015). Hence, it becomes crucial to form an appropriate level of trust, particularly when this agent reaches the limit of its capabilities (Göritzlehner et al. 2014; Körber, Baseler & Bengler 2018). Without holding a good mental model, the human teammate can experience a poorer quality of reaction and a longer time to react when the autonomy agent is facing failure caused by its limited capabilities. As it is crucial to form trust and a good mental model, transparency is a critical requirement in autonomy agent development (Abdul et al. 2018; de Vries, Midden & Bouwhuis 2003; Wright, Chen & Lakhmani 2019). Moreover, Bhaskara et al. (2021) stated that transparency can help human teammate making faster decision to respond unexpected behaviour of the autonomy agent.

What is to be explained by an autonomy agent to a human teammate is an essential part of transparency. Bencomo et al. (2012) and Zhu et al. (2018) viewed that explaining agent is about answering *why* on its behaviour, characterized as $why = \{what, how, history\}$. Not merely about *why*, researchers defined two types of explanations, namely a backward chaining explanation (BCE) and a forward chaining explanation (FCE). In general, BCE is about explaining the reason why an action was taken while FCE is about what the agent will do in the immediate future based on given data. The former is also known as a goal-driven explanation while the latter is data-driven (Al-Ajlan 2015; Anjomshoae et al. 2019). Following is a simple illustration of BCE and FCE. Suppose that an agent is instructed by four rules written in a program as follows:

1) If $x < 50$ than x is unsafe distance
2) If $x \geq 50$ than x is safe distance
3) If x is an unsafe distance then stop
4) If x is a safe distance then keep going

and the given value for x is 40. FCE drives the inference engines to conclude x is an unsafe distance according to rule 1, and the agent will stop according to rule 3. In contrast, when the inference engine investigates why the agent has stopped, BCE will track down

the antecedent which is 'x is unsafe' given by rule 3 and ends up at $x < 50$ given by rule 1.

In a HAT context, transparency requirements are defined as information which needs to be revealed to the human teammate to understand the autonomy agent's behaviour (Chen et al. 2014). For this purpose, Chen et al. (2018) proposed a SA-based agent transparency (SAT) model (see Figure 2-3). There are three levels of transparency in this model. The first level is associated with the autonomy agent's goals which cover its current status, action, and plans. The related information at this level includes goal selection, progress, performance, and environment/teammate. At the next level, transparency information will be related to the autonomy agent's reasoning process and possible environment constraints. Finally, level 3, involves the autonomy agent's projection of future outcomes including the likelihood of success or failure and performance history (Chen et al. 2014). Even though the SAT framework is a good information structure to explain the autonomy agent's states and behaviour, it does not consider the time-length of the situation. In a situation with a small timespan, it is difficult to absorb a large amount of information to respond to the situation. Therefore, it is necessary to develop a new transparency requirement framework that considers such time constraints.

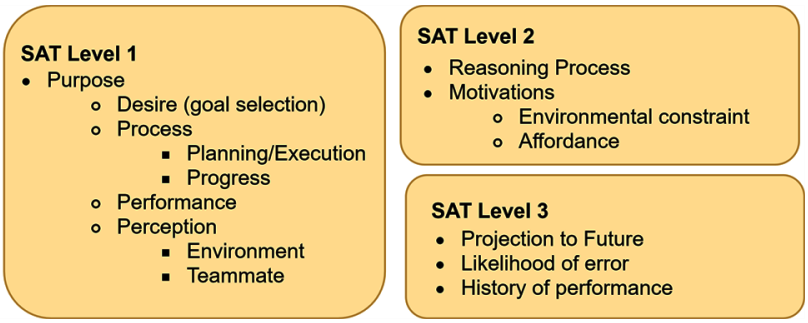


Figure 2-3. Situation Awareness-based Agent Transparency

Furthermore, other researchers opined that transparency involves reporting reliability, unexpected/abnormal behaviour, and to expose decision making and the reasons for the decision (Langley et al. 2017; Wortham & Theodorou 2017). In addition, transparency is also important to reveal the rules for behaviour trade-off in the autonomy agent design. For example, in the task of following a traffic light state, by design, an agent is instructed to keep going when it fails to recognize the traffic light state even though the actual state is red. In this regard, there is a trade-off between safety and other road users' convenience

which should be conveyed to human teammates (Kridalukmana, Lu & Naderpour 2020). Bhaskara, Skinner & Loft (2020) views that there are various levels and ways to present transparency. The lowest level only involves an icon to depict agent's reasoning. Higher level of transparency can contain textual or graphical explanations the provide many type of information.

2.5.2. Component-based Method

To produce autonomy agent's behaviour explanations, the component-based method observes the signal processing among the electronic components of this agent or the component states. For example, by observing the signal exchanges among the agent's components, Lucas (2001) developed a technique that differs from normal/abnormal behaviours. Le (2002) proposed a module within a robot called a self-inspection system (SI-system) which extracts behaviour explanations from abstract causal circuits. In this causal circuit abstraction, the currently active component actuating the agent is referred to as a symptom node, and a component state sending a signal to the actuator component is called an active cause node. A tag which is associated with the agent's action will be assigned to each component in the causal circuit. For example, a tag "turning" should correspond to a particular actuator node, and a tag "trying to get unstuck" is given to a certain node state representing the unwedged motor behaviour. The self-inspection mechanism is intended to look up the tags to generate an explanation. To generate the BCE of the current action stored in the symptom node, this approach looks at the active cause node where the rationale state is stored. For example, when the symptom node and active cause node have a tag "turning" and "trying to get unstuck" respectively, it will generate the explanation that the agent is turning because it is trying to get unstuck. However, by only comparing the symptom and active cause nodes, this approach only can generate a one-level explanation.

2.5.3. Process-based Method

The process-based method (also called algorithmic transparency (Diakopoulos 2016)), is developed to explain phenomena that cannot be characterized using the component structural approach by providing higher-level concepts within the agent's systems. For example, the component-based method cannot be used to reveal information such as the

traffic light state is red. Even though agent types may vary, all agents have the same process skeleton: input, input manipulation, and output. In this regard, a program is the implementation of a process which can consist of algorithms, functions, and instructions. Thus, the process is the representation of the agent's behaviour.

The technique to generate a behaviour explanation from the process level varies because an autonomy agent can have different types of processing systems. There are at least four groups of agent processing systems in the existing studies: 1) neural-network-based systems (Panwai & Dia 2007), 2) Markov-decision-process-based systems (MDP system) and its variants such as partially-observable-Markov-decision-process-based agent (POMDP system) (Liu et al. 2015), 3) reinforcement-learning-based agents (Fukuchi et al. 2017), 4) rule/logic-based systems (Mostafa et al. 2018). Each processing system type requires different techniques to extract a behaviour explanation.

For example, Elizalde et al. (2008) proposed an approach to generate an explanation of a simple-reflex MDP-based agent by defining relevant explanations for every state in the factored MDP representation. This approach is intended to provide a future action recommendation for the operator, and thus, it can be classified as FCE. Furthermore, Wang, Pynadath & Hill (2016) proposed PsychSim as an algorithm to produce a behaviour explanation on a goal-following POMDP-based agent. In this approach, a natural-language template is developed to translate the content in the POMDP model into human-friendly sentences. The explanation is generated in a BCE manner. Furthermore, Fukuchi et al. (2017) proposed an instruction-based technique to generate a behaviour explanation on a goal-following reinforcement-learning-based agent. This technique reuses the expression of instructions for generating a BCE.

This study focusses on a goal-following logic-based agent, particularly for BCEs. In this regard, the agent's functions and logics can be viewed as the goal model representation which become the execution plans of the agent when the agent is running (Shen, Gay & Miao 2002). As illustrated in Figure 2-4, the agent's logics can be modeled as a hierarchy and each node in the hierarchy can be considered as a decision logic. In this figure, there is one main system *M* and two sub-systems *S* and *D*. The logic paths executed by the agent are indicated using blue arrows and the unexecuted logic paths are indicated using

black arrows. Generating BCE means that the behaviour explanation generator should trace back the executed paths as indicated by the blue arrows.

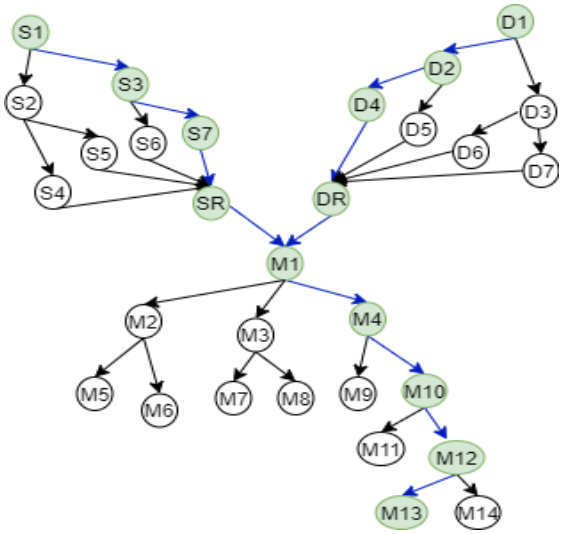


Figure 2-4. BCE in a Process-Based Method

There are many techniques to trace back such a hierarchy, such as a uniform search or an informed search (Al-Ajlan 2015). In a uniform search, the most common techniques are the breadth-first search (BFS), depth-first search (DFS), and uniform-search first (USF). The best-first search (BeFS), and A* search can be included in the informed search. The main problem with these techniques is that during the tracing back process, all the nodes connected by the executed and non-executed paths will be visited. Therefore, when a huge number of logics is involved, these techniques become inefficient as the search space will also significantly increase. Furthermore, an action can be caused by different executed paths of the logic hierarchy. Performing the whole trace in the hierarchy to find which logics are responsible for an outcome, particularly in a big logic hierarchy, often does not generate a useful explanation (Harbers, Van Den Bosch & Meyer 2010). Thus, an explanation generation algorithm should not only select all the possible logics that are causing an action, but they should also select from all the possible logics to find the ones that are truly causing an action. To resolve this problem, Al-Ajlan (2015) implements a working memory which stores the path discovered during a session. Thus, the search space cost will be determined by the depth level of the logic hierarchy.

The meaning of each logic might be difficult to understand as it contains decision rules using operators such as greater than, less than, equal to, and so on. Therefore, Harbers,

Van Den Bosch & Meyer (2010) argue that an agent must have meaningful and explicit representations to reflect its behaviour. For this purpose, they propose a belief-desire-intention (BDI) hierarchy as the goal-based behaviour representation. To trace the reason behind an action from this hierarchy, Kaptein et al. (2017) combine two algorithms: a belief-based and goal-based explanation algorithm which specifically trace the belief path to find the triggering conditions of an action and specifically examine the parent goal directly above the action, respectively. However, developing a goal-beliefs hierarchy is a very subjective task so that in non-organizational cases, it may be difficult to develop such a hierarchy. Therefore, Reynolds (2019) developed a provenance graph as the behaviour representation which is produced at the runtime to connect the agent's data and logics to provide explanations. This approach is referred to as a model-driven self-explanation. However, Reynolds (2019) explained that the existence of sub-processes or sub-functions within the autonomy agent's system might impact the difficulty in producing explanations with the model-driven approach. The use of a logic hierarchy may address the subjectively developed goal hierarchy problem, but the second problem still exists, particularly when complex procedural rules are involved and the logic in the algorithms determining the agent's behaviour is regularly updated. After the algorithms are updated, the logic hierarchy for the old algorithms will be obsolete so that a new hierarchy should be developed.

2.5.4. Summary of Self-explanation Ability Methods

This section shows that researchers suggested transparency which makes the autonomy agent's behaviour observable and understandable by humans in a teaming context. One way to deliver transparency is by embedding a self-explanation ability inside an autonomy agent. To generate explanations, current studies exploit the agent's component level and process level which are referred to in this study as component-based and process-based method, respectively. To summarize the characteristics of these two methods, Table 2-3 presents their strengths and weaknesses. As this study focuses on the goal following logic-based agent, only the strength and the weaknesses of the process-based method for the logic-based agent are presented. Based on these strengths and weaknesses, it is necessary to overcome the impact of the complexities of the autonomy

agent's system. Complexity in this context refers to the existence of a huge number of logics in algorithms with many sub-systems involved.

Table 2-3. Strengths and weaknesses of the existing methods to generate behaviour explanations

Method	Strengths	Weaknesses
Component-based	- Suitable for an autonomy agent with simple logics - Provide two-node based tracing back approach (symptom node and active node) - Focus on abstraction level of commands in assembly codes - Generate simple explanations	- The generated explanation is more about the components' problems rather than external/surrounding situations - Limited detail in explanations - Difficult to apply in a complex system
Process-based in logic-based agent	- Able to generate detailed explanations - Rely on the actual sources (algorithms) which drive an autonomy agent's behaviours - Able to provide every aspect of the autonomy agent's states - Suitable choice to extract the autonomy agent's current activities - Minimum impact on hardware changes	- The tracing back mechanism to generate explanations can be a burden when the autonomy agent's system becomes complex - Difficult to synthesize the meaning of every path when complexity increases - Hard to maintain when the software (algorithms) periodically receives an update which changes the structure of logics

2.6. Image Classification using Transfer Learning

2.6.1. Transfer Learning

Transfer learning refers to the use of a model learned from the camera-captured images in one domain, namely the source domain, to solve object classification tasks in a target domain (Sanodiya et al. 2019). In the case of image classification, transfer learning uses a model trained using the labeled dataset in the source domain to develop a model to classify images in the target domain. There are three categories of transfer learning techniques based on the training data in the target domain. Unsupervised transfer learning is the term used when there are no labeled data in both the source domain and target domain (Arnold, Nallapati & Cohen 2007; Lu et al. 2015; Pan & Yang 2009). Semi-supervised transfer learning (SSTL) is the term used when some data are labeled in the target domain (Deshmukh & Laftchiev 2018; Pereira & da Silva Torres 2018; Zuo et al.

2019). Supervised transfer learning is the term used when the data in both domains are labeled (Dai et al. 2009; Pan & Yang 2009; Shi et al. 2009). In a deep learning context, performing transfer learning means taking the benefits of the pre-trained network weight in the source domain to support the learning process in the target domain.

2.6.2. Transfer Learning with Unlabeled Data in the Target Domain

In traditional semi-supervised transfer learning called pseudo-label (Lee 2013), a pre-trained model from the source domain is used to predict the labels of the unlabeled data in target domain's training dataset and this model is retrained using a mixed dataset from the source domain and the target domain. In this regard, class probabilities in conditional entropy regularization are used to measure the overlap among classes. This traditional method was improved by Matasci et al. (2015) using transfer component analysis (TCA) to obtain a better prediction for the unlabeled dataset in the target domain before retraining the new model. As another way to make unlabeled data useful, Chapelle, Weston and Schölkopf (2003) noted that formulated assumptions should be defined either explicitly or implicitly. This assumption is called a cluster assumption which says that two points might have the same label if a path connecting them exists in a cluster with high density only. This concept is similar to the one proposed by Kumar, Saha and Daume (2010) which is called a linear hypothesis. Deshmukh and Laftchiev (2018) viewed that adding unlabeled data means that in addition to using a marginal distribution for transferring knowledge from the source to the target domain, data structure knowledge should also be added which can be represented in the form of a graph Laplacian.

In an effort to minimize domain discrepancy, Song et al. (2019) and Wang et al. (2019) minimized the domain discrepancy during the retraining of the pre-trained model using subspace alignment and a deep adaptation network (DAN), respectively. In this regard, DAN uses Maximum Means Dependencies (MMD) to measure the domain discrepancy, which is one of the preferred techniques in domain adaptation (Kang et al. 2019). Then, MMD with multiple kernels is formulized by Kumar et al. (2010) to jointly maximize the results in classifier testing and minimize the error probability when accepting the labels in a test dataset. Another effort to minimize domain discrepancy has been conducted by Long et al. (2017), and their method is called join adaptation network (JAN). This method is based on Joint Maximum Mean Discrepancy (JMMD) technique. JMMD follows the

virtue of maximum mean discrepancy (MMD) which uses Hilbert space embeddings for joint distributions to calculate the discrepancy of two joint distributions. JMDD is used to easily recognize any shift in the joint distributions.

Furthermore, the distance measurement between two objects can also be done using metric learning as proposed by Zhang & Yeung (2012). The main objective of metric learning is to establish object similarities by directly using a distance metric such as Mahalanobis which is based on the Euclidean distance function (Kaya & Bilge 2019). As Euclidean-based metric learning fails to optimally capture object similarities, Sanodiya and Mathew (2019) observed the instance weights to reduce distributions between two domains both geometrically and statistically. Then, Sanodiya et al. (2019) combined the instance weights with relative distance metrics to reduce domain shifts.

In addition to the aforementioned techniques, another critical factor affecting the training process is the image pre-processing preparation. To improve object classification, a technique called online image augmentation is often used to upsize the image samples with various object positions by generating an additional batch during the dataset learning process (see Figure 2-5). In this way, feature extraction information can be enriched (Perez & Wang 2017). Image augmentation itself is referred to as a function to transform an image, such as image flipping (horizontally and/or vertically), cropping, and shearing (Wang et al. 2019). Instead, there are many more transformation techniques that can be applied such as shifting, zooming, and shearing. When image augmentation is activated, it generates batches with real-time image augmentation that will be looped over the training process. The term online refers to generating a real-time additional batch during the training process.

Figure 2-5. Image is augmented but still in the same class, bicycle

However, when sample images in the source domain have geometrical relation constraints with the sample images in the target domain, such online image augmentations are not reliable to support classification tasks, and significantly degrade the classification

performance. Geometrical relation constraints in this context refer to a circumstance where some images from the target domain fall into a different class in the source domain when they are augmented. For example, the sample images from the source domain which are classified as 'talking on the phone using the right hand' in a left-steering vehicle can have a geometrical relation with sample images from the target domain which are labeled as 'talking on the phone using the left hand' in a right-steering vehicle. In this regard, when the image in the target domain is augmented (i.e., horizontally flipped), this image will be labeled as 'talking on the phone using the right hand in the source domain' (see Figure 2-6).

Talking on the phone – right hand, left-steering vehicle (a1)

Talking on the phone – left hand, right-steering vehicle (a2)

Geometrically related between (a1) and (a2)

Figure 2-6. An example of geometrical relations between image samples in two domains

2.6.3. Summary of Transfer-learning-based Image Classification

Producing an image classifier using the transfer learning technique has been widely used in many studies. To perform transfer learning, in addition to focusing on minimizing the discrepancy between two datasets called the source domain and the target domain, some studies also take advantage of the online image augmentation technique which has been proven to have a significant effect on enhancing image classification. But when geometrical relation constraints exist in the source domain and target domain, the accuracy of the image classifier generated during the transfer learning process can significantly drop even though this process applies online image augmentation. Proposing a new transfer learning approach to address geometrical relation constraints is critical to maintain image classification accuracy.

2.7. Summary

Based on the systematic literature review in the previous sections, there are three main findings that can be identified. First, in the HAT context, most of the existing studies

focus on enhancing the collaboration performance in a teaming with horizontal coordination such as a team model and a shared model. Only a few studies focus on a teaming with a vertical coordination hierarchy in the HAT context, particularly as to how this type of coordination is modeled using SAM. Therefore, to fill this gap, this study expands SAM for vertical coordination.

Second, the key ability for an autonomy agent to improve vertical coordination performance with a human teammate is the ability to explain its behaviour. However, with the significant increase in an autonomy agent's logics and functions, the current methods (component-based and process-based methods) are not reliable. Hence, finding a new way to generate a behaviour explanation which deals with an autonomy agent's logic and function complexities can be considered as one research gap which is bridged in this study.

Third, in addition to a self-explanation ability, it is also significant for an autonomy agent to have an ability to participate in maintaining a human teammate's SA, particularly for HAT in a vertical coordination context. Observing and classifying human behaviour is one way for such participation, and when only a few image data are available to model human behaviour, the transfer learning technique becomes a preferred choice to develop the human behaviour classifier. However, when geometrical relation constraints exist during the transfer learning process, it requires an appropriate transfer learning approach to avoid the degradation of the classification performance. Geometrical relation constraints have not been considered in existing transfer learning methods, and to fill this research gap, this study considers transfer learning that can handle geometrical relation constraints.

Chapter 3 : Situation Awareness Modelling Framework for Teaming Situations

3.1. Introduction

This chapter addresses the first research question in this study which is how to model team members' SA to synthesize the characteristics of a team. To answer this question, this chapter presents a framework of SA modelling for teaming situations. This framework conceptualizes the SA models for possible teaming situations, namely normal, shared, mutual, and superior-subordinate teaming situations. While previous studies (i.e., Endsley (1995), Shu & Furuta (2005)) focused on the individual goals of team members to model SA in a team, our proposed framework relates SA with SA-related factors including goal, goal owner, goal-related environment, SA requirements, and situation awareness development process. The definition and explanations of each factor will be presented in the following sections.

The developed framework can guide the formation of a SA model for a given teaming situation to gain a better understanding of relation complexities among SA factors in SA development.

This chapter is organized as follows. Section 3.2 defines the related terms. Section 3.3 presents the framework of SA modelling for teaming situations and specific SA models for various teaming situations. Section 3.4 details the application of the SA modelling framework. Section 3.5 summarizes the chapter.

3.2. Definitions of Related Terms

In order to better describe SA modelling, this section presents the definitions of the terms related to SA including elements, environment, goals, goal-related environment, situation, SA, SA elements, SA requirements, and situation awareness development process (SADP).

Definition 1. Element (ς) is a physical or abstract object. Each element has a status (v).

Definition 2. An environment (ϑ) can be defined as everything outside an individual (or an autonomy agent in the HAT context) (Lundberg 2015). It is viewed as a set of elements so that $\vartheta = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_n\}$, where n is the number of elements within an environment.

An environment has a very wide scope and it can be complex. The widest scope of an environment is the universe. In most cases, an environment is only a part of the universe. Boundaries can be set to limit the environment of concern. Limiting the environment is not necessarily in terms of size. It can also be in terms of the selected elements from a global environment.

Definition 3. A goal (ϱ) is defined as what is to be achieved. A goal can be achieved through a set of tasks so we can express a goal as $\varrho = \{\tau_1, \tau_2, \ldots \tau_m\}$, where τ_j denotes a task $j = 1, \ldots, m$, and m is the number of tasks.

The goals are the essential driving factors to set the boundary of an environment of concern.

Definition 4. Goal-related environment **(ϑ_G)** is an environment which is limited by a given goal. It is a subset of a global environment of concern, i.e., $\vartheta_G \subset \vartheta$**.**

Definition 5. Situation ($\mathfrak{S}$) is defined as what needs to be aware of the goal-related environment in a timeframe. It can be represented by a pair whose first component is a set of element's statuses in a goal-related environment (ϑ_G) and the second component is a timeframe (t), as denoted by $\mathfrak{S} \rightarrow \big((v_{\varsigma_1}, v_{\varsigma_2}, \ldots, v_{\varsigma_n}), \boldsymbol{t}\big)$. The status of these elements give the meaning of a situation while the timeframe indicates the time range in which the given situation is valid.

Definition 6. Situation awareness (SA) is a state of situational knowledge ($\varkappa^{situation}$) at time t and can be denoted as $SA = \varkappa_t^{situation}$.

Definition 7. Situation awareness elements (ς^{SA}) is a set of elements which needs to be considered for SA within a goal-related environment (ϑ_G) and therefore, $\vartheta_G = \{\varsigma_1^{SA}, \varsigma_2^{SA}, \ldots, \varsigma_n^{SA}\}$, where n is the number of elements needed for SA.

Definition 8. Situation awareness requirement ($\mathcal{R}^{SA}$) is defined as a collection of the required statuses of SA elements ($v_{\varsigma^{SA}}$), $\mathcal{R}^{SA} = \{v_{\varsigma_1^{SA}}, v_{\varsigma_2^{SA}}, \ldots, v_{\varsigma_n^{SA}}\}$ where n is the number of elements needed for SA. Each status $v_{\varsigma^{SA}}$ can be represented by a set of attributes and each attribute is represented by a key-value pair. Therefore, the status of SA element j can be represented by $v_{\varsigma_j^{SA}} = \{A_1^j: val_{\mathbf{1}}^j, A_2^j: val_{\mathbf{2}}^j, \ldots, A_{p_j}^j: val_{p_j}^j\}$ where p_j is the number of attributes for element j.

Definition 9. Situation awareness development process $\wp^{SA}$ is defined as the process to form SA. It is represented as a three-level hierarchical process and is denoted as $\wp^{SA} \rightarrow (\mathcal{L}_1^{SA}, \mathcal{L}_2^{SA}, \mathcal{L}_3^{SA})$, where $\mathcal{L}_1^{SA}, \mathcal{L}_2^{SA}, \mathcal{L}_3^{SA}$ represent level 1 to level 3 SA in the hierarchy, respectively. Let $\varkappa$ denote knowledge and $\mho$ denote a set of rules. We further define the level 1 process as a process using $\varkappa$ and $\mho$ to generate the understanding of the status of every SA element (v^{SA}) given their SA requirements ($\mathcal{R}^{SA}$), as denoted by $\mathcal{L}_1^{SA} = ((\varkappa, \mho) \rightarrow \forall(\varsigma^{SA} \rightarrow v^{SA})|\mathcal{R}^{SA})$. Similarly, we define the level 2 process as a process using $\varkappa$ and $\mho$ to understand situation ($\mathfrak{S}$) given the combination of the status of SA elements produced by $\mathcal{L}_1^{SA}$, as denoted by $\mathcal{L}_2^{SA} = ((\varkappa, \mho) \rightarrow \mathfrak{S}|\mathcal{L}_1^{SA})$ and the level 3 process as a process using $\varkappa$ and $\mho$ to execute action based on $\mathfrak{S}$, as denoted by $\mathcal{L}_3^{SA} = ((\varkappa, \mho) \rightarrow \text{Ⅎ}|\mathfrak{S})$, where Ⅎ denotes action.

3.3. Situation Awareness Modelling Framework for Teaming Situations

This section presents a framework of SA modelling for teaming situations. This framework conceptualizes SA models for teaming situations. It guides the formation of SA models for various kinds of teaming situations.

3.3.1. Generic Situation Awareness Model

A SA model ($\mathcal{M}^{SA}$) is defined as a representation of SA, SA-related factors, and the relations among them. The SA-related factors include the goal (ϱ), goal owner (ow^{ϱ}), goal-related environment ($\vartheta_{\boldsymbol{G}}$), SA requirements ($\mathcal{R}^{SA}$) and SADP ($\wp^{SA}$). It can be represented as a six-element tuple ($\varrho, ow^{\varrho}, \vartheta_{\boldsymbol{G}}, \mathcal{R}^{SA}, \wp^{SA}, SA$).

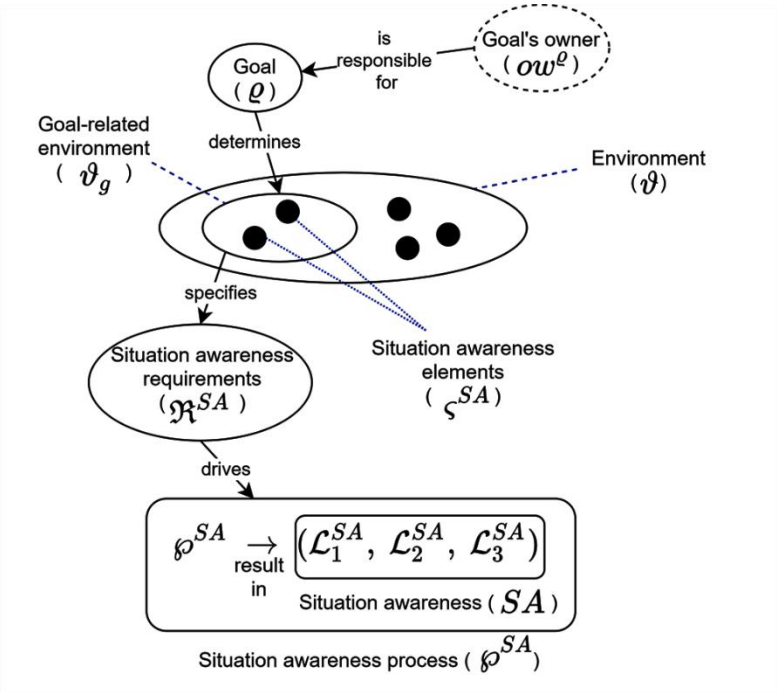


Figure 3-1. Generic situation awareness model

Figure 3-1 is an illustration of a generic SA model ($\mathcal{M}^{SA}$) to show the relationship of SA and its related factors, i.e., $\varrho,\ ow^{\varrho}, \vartheta_{\boldsymbol{G}}, \mathcal{R}^{SA}, \wp^{SA}, SA$. The goal ($\varrho$) has an owner and an owner **is responsible for** achieving the goal. The goal owner (ow^{ϱ}) can be an individual, an artificial agent or a team. The goal ϱ **determines** the boundary of the goal-related environment ($\vartheta_{\boldsymbol{G}}$)**.** $\vartheta_{\boldsymbol{G}}$ **specifies** the SA requirements ($\mathcal{R}^{SA}$). $\mathcal{R}^{SA}$ **drives** the SADP ($\wp^{SA}$). $\wp^{SA}$ **results in** SA**.**

This generic $\mathcal{M}^{SA}$ can be illustrated using an example of the SA model for a single autonomy agent (a) which is an autopilot feature in assistive driving.

The autonomy agent's goal, ϱ_a , is to replace the human driver by taking on the driving tasks to arrive to an intended destination safely. The agent drives a vehicle within an environment (ϑ) so that $ow^{\varrho} = a$. The goal-related environment ($\vartheta_{\boldsymbol{G}}$) is the driving environment which includes the surroundings within a certain distance away from the vehicle. The SA elements (ς^{SA}) are the objects in $\vartheta_{\boldsymbol{G}}$, whose statuses can affect the safety of driving, such as pedestrians, traffic signs, and other road users. Objects outside this environment, such as a house with a car park in the garage, will be excluded as they do not affect the safety of driving. Thus, they are not related to achieving the goal. Furthermore $\mathcal{R}^{SA}$ is the attributes of SA elements that describe the status of ς^{SA}, i.e.,

traffic light (red, green, yellow lights) and car in front (speed, distance). Agent a carries out a SADP ($\wp^{SA}$) to form its SA.

3.3.2. Situation Awareness Modelling Framework for Teaming Situations

Figure 3-2 illustrates the SA modelling framework for teaming situations. This framework consists of four blocks: (i) team specification block, (ii) goal provision block, (iii) teaming formation block, and (iv) teaming SA specification block.

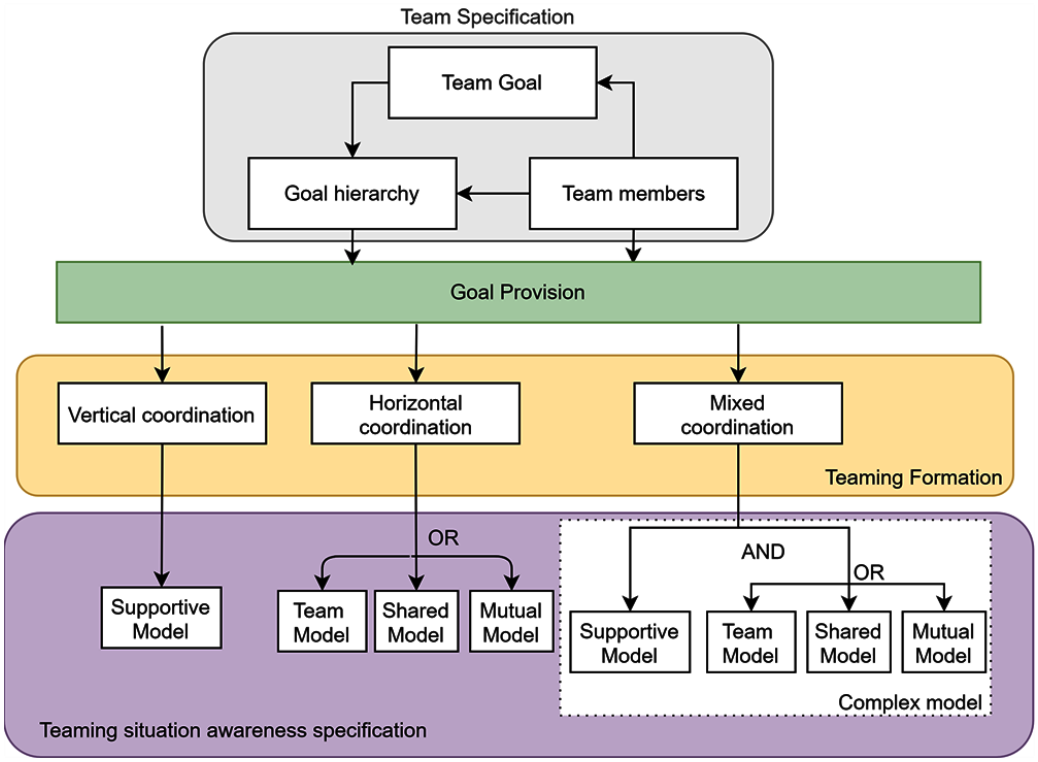


Figure 3-2. A situation awareness modelling framework for a teaming

3.3.2.1. Team Specification Block

The team specification block takes the pre-requisites to establish a team, which are team goal and team members as inputs and produces a goal hierarchy. A team goal is something to be achieved by the team. Assume team members are denoted by $\boldsymbol{m}$, $\boldsymbol{m} = \{m_1, m_2, \ldots, m_k\}$, where k indicates the number of team members. One or more team members can own the team goal. The team goal owner is responsible for achieving the team goal. If the owner is a single member, they will be in charge of determining a set of tasks to achieve the team goal by consulting with other team members. If the owner is the

entire team, the team will determine a set of tasks to achieve the team goal. In both cases, the team goal will be represented by a set of tasks, $\varrho = \{\tau_1, \tau_1, \ldots \tau_g\}$, and the team goal can be achieved by completing these tasks. According to their dependencies, these tasks can be represented using a hierarchical structure. We represent this hierarchical structure as a tree structure, which includes nodes and branches. The nodes include a root node, some inner nodes and some leaf nodes. The root node will be the team goal. In a case where the team goal owner is an individual member, a branch represents a "depend on" relation between nodes. In a case where the team goal owner is the entire team, a branch represents a "is a part of" relation between nodes. Based on this task hierarchy, a goal hierarchy can be developed by referring to a hierarchical task analysis method from Stanton (2006). The main principle of this hierarchical analysis method is to extract goals from tasks in such a way that each task corresponds to a goal. Therefore, we consider the goal hierarchy, denoted by $h(\varrho)$, is equivalent to the hierarchy of tasks ($h(\tau)$).

$$h(\tau) \approx h(\varrho) = \bigcup_{i=0}^{n} \varrho_{q_i}^{i} \tag{3-1}$$

where i denotes the hierarchy level. The larger i is, the lower the level is. q_i denotes the number of tasks at level i. Figure 3-3 shows an example of a task hierarchy.

The root node represents the team goal of "Conquer the enemy in City B quietly". It is not associated with any task. This goal can be achieved through two tasks at level 1, namely "determine a route to the destination" (Node 1.1) and "command troop" (Node 1.2). The task at Node 1.1 is an independent task as this node is a leaf node. The task at Node 1.2 is a dependent task, which depends on two tasks at level 2, which are 'decide if attacking" (Node 2.1) and 'decide if escaping' (Node 2.2). The task of 'decide if attacking' at Node 2.1 relies on two other tasks, namely, 'monitor situation in the city' (Node 3.1) and 'perform attack' (Node 3.2), which are independent tasks because both nodes are leaf nodes. Apart from the root node, other nodes represent specific tasks. The resultant goal hierarchy of the example task hierarchy is shown in Figure 3-3.

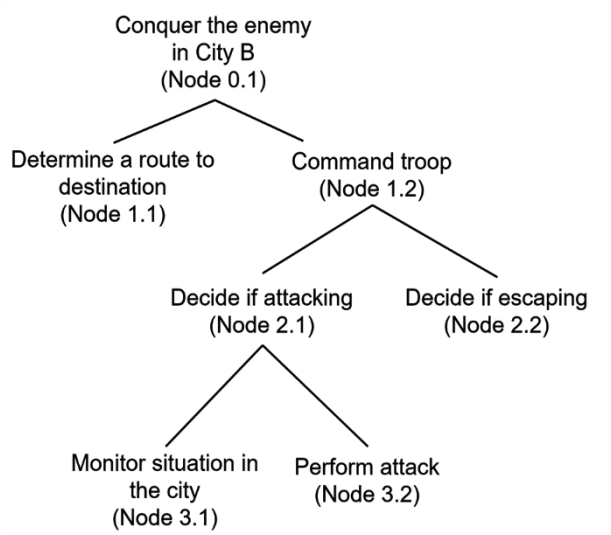


Figure 3-3. Example of a goal hierarchy

3.3.2.2. Goal Provision Block

The goal provision block is the second block in the framework. Using the goal hierarchy and team members from the previous block as the input, this block outputs a goal hierarchy with team member allocation according to team formation types.

Here we consider two possible cases of team formations based on the goal hierarchy or goal tree. The first case is that the root node is the team goal and branches represent "depend on" relations. In such a case, a team must have a superior, say m_1, and this member will be assigned to the team goal at Level 0 by default. This team goal is also the m_1 individual goal, denoted as $\varrho^0_{1,[\,][m_1]}$, which means that m_1 is responsible for achieving the team goal. Due to the "depend on" relations in the goal tree, m_1 is also responsible for achieving all the sub-goals in the hierarchy (i.e., all the goals except for the root node). The [ ] indicates that m_1 has no superior. m_1 will distribute some goals to some team members. In doing so, m_1 becomes the top superior and the members who are assigned some goals by m_1 become m_1's direct subordinates. For example, referring to the goal hierarchy Figure 3-3, m_1 assigns the goal at Node (1.1), denoted as ϱ^1_1 to member m_2, and we can represent this assignment as $\varrho^1_{1,[\,m_1]\,[m_2]}$, where $[m_2]$ denotes the member who is assigned to the target goal (ϱ^1_1), and [m_1] denotes the superior of m_2. When m_2 is assigned a goal (ϱ^1_1), this goal becomes m_2's individual goal, and m_2 will be responsible for achieving this goal. m_1 also assigns the goal at Node (1.2), denoted as ϱ^1_2 to member

m_3, and we can represent this assignment as $\varrho^1_{2,[\,m_1]\,[m_3]}$. With this assignment, ϱ^1_2 becomes m_3's individual goal and m_3 is responsible for achieving the goal ϱ^1_2 along with all the goals under this goal in the hierarchy. Since m_3 is the owner of a dependent goal ϱ^1_2, m_3 can assign the sub-goals to other members. For example, if m_3 assigns the goal at Node (2.1), denoted as ϱ^2_1 to member m_4, we can represent this assignment as $\varrho^2_{1,[m_1,m_3]\,[m_4]}$, where $[m_4]$ denotes the member who is assigned to the target goal (ϱ^2_1) and $[m_{1,}m_{3,}]$ denotes the superiors of m_4, with m_3 being the direct superior.

By generalising the representation method of the aforementioned goal assignments, we introduce the following notation for a goal distribution or goal provision:

$$\Phi_{h(\varrho)} = \bigcup_{i\,=\,0}^{n} \bigcup_{j=1}^{q_i} \varrho^i_{j,[m_{sup}][m_{dist}]} \tag{3-2}$$

where $\Phi_{h(\varrho)}$ denotes a goal provision, i represents the hierarchy level. q_i denotes the number of goals (ϱ) at level i, $i = 0,\ldots,n$, and n is the number of levels in the goal hierarchy or the depth of the goal tree. m_{dist} denotes one or more team members who are assigned to the j −th goal at level i (ϱ^i_j). m_{sup} denotes the superior(s) of m_{dist}, which is a list of members representing a chain of superiors from the top superior to their direct superior. The rationale for keeping a superior chain in m_{sup} is twofold: one is to indicate the coverage of supervision responsibilities of superiors and the other is to show the authority level (high to low) through the order of superiors in the list.

In this first case of team formation, there are two kinds of relationships: one is the relationship between a superior and their direct subordinate, which is referred to as a superior-subordinate relationship, e.g., the relationship between m_1 and m_2; the other one is the relationship between two team members who have the same superior chain, and their individual goals are at the same level in a goal hierarchy, which is referred to as a teammate relationship, e.g., the relationship between m_2 and m_3. We refer to such a team formation as a team with a superior.

One example of goal provision for the example goal hierarchy in a team of four members is illustrated in Figure 3-4.

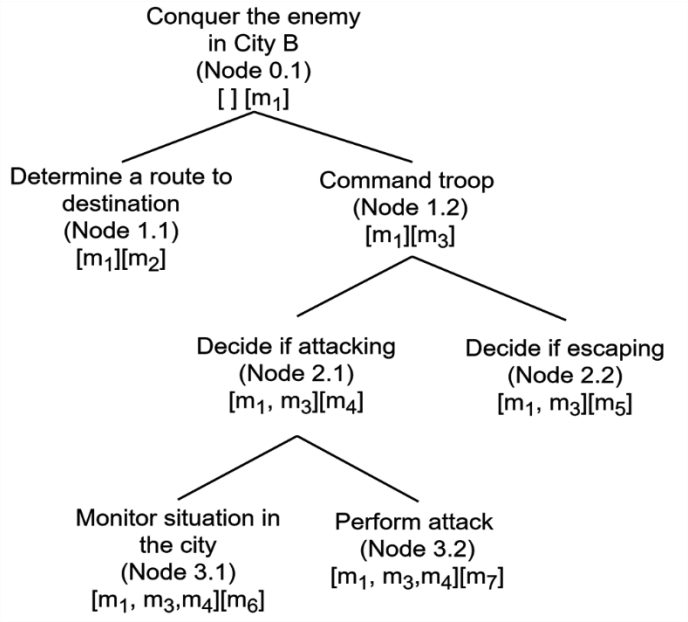


Figure 3-4. Example of goal provision

The second case of team formation is that all team members own the team goal equally. In such a case, the goal hierarchy is a two-level goal tree in which the root node is the overall team goal, and the leaf nodes are specific goals.

Goal provision in such a case can be represented using a similar notation to that in equation (3-2) as

$$\Phi_{h(\varrho)} = \bigcup_{i=0}^{1} \bigcup_{j=1}^{q_i} \varrho^{i}_{j,[][m_{dist}]}$$

where $m_{sup} = \{\}$ indicates all members do not have any superior. The overall team goal (ϱ) at level 0 (i=0, $q_0 = 1$) is assigned to all the members, i.e., $\varrho^{0}_{1,[\,][\boldsymbol{m}]}$ and q_1 denotes the number of specific goals that constitute the overall team goal at level 1. The relation between a specific goal and the overall team goal is "a part of" relation. This is different from the ones in the first case which represent "depends on" relations.

For example, assume a goal hierarchy/tree as shown in Figure 3-5(a), in which the root node represents the overall goal (ϱ) and the four-leaf nodes at level 1 represent four

specific goals, i.e., q_1=4. This overall goal is owned by the entire team and the specific goals are distributed among three members, i.e., k= 3. One goal distribution can be: member m_1 for goal ϱ_1^1, m_2 for goal ϱ_2^1, m_3 for ϱ_3^1, and m_2 and m_3 for goal ϱ_4^1, using the following notation: $\varrho_{1,[\,][m_1]}^1$, $\varrho_{2,[\,][m_2]}^1$, $\varrho_{3,[\,][m_3]}^1$ and $\varrho_{4,[\,][m_2,m_3]}^1$, respectively. This goal provision is shown in Figure 3-5(b)

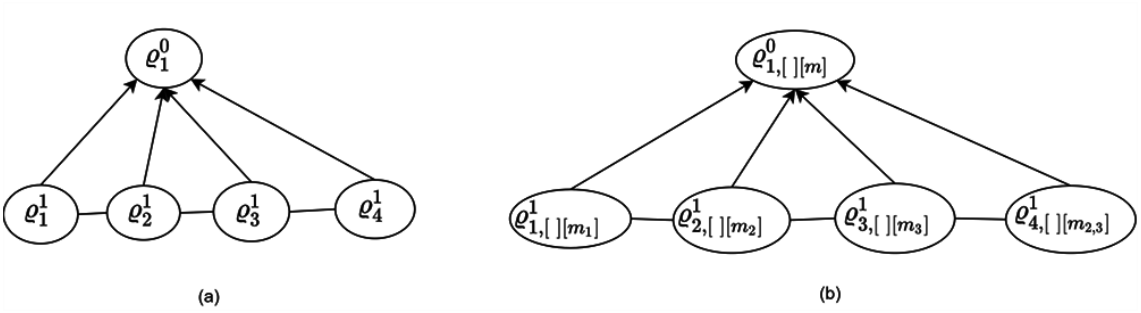


Figure 3-5. Example of goal hierarchy (a) and its provision in a team without a superior (b)

In this team formation, there is only one kind of relationship, which is the teammate relationship representing the relationships between members at level 1. We refer to such a team formation as a team without a superior.

3.3.2.3. Teaming Formation Block

The teaming formation block is the third block of the framework. It presents possible teaming formation types based on the output of the goal provision block.

A teaming formation refers to a subset of team members who work together to achieve some goals and need coordination in developing their SA. Coordination refers to the communication flow among team members.

There are three types of teaming formations depending on the relationship types among team members. The first one is a teaming in which team members have superior-subordinate relationships. Such a teaming is referred to as teaming with vertical coordination. The second one is a teaming in which team members have teammate relationships. Such a teaming is referred to as teaming with horizontal coordination. The third one is a teaming in which some team members are involved in both superior-subordinate and teammate relationships. Such a teaming is referred to as a teaming with mixed coordination.

A. Teaming with vertical coordination

A teaming with vertical coordination must have at least two team members ($k = 2$) and at least two members have a superior-subordinate relationship. For example, let a teaming consist of three members (namely m_1, m_2, and m_3) and the goal provision of these members, $\Phi_{h(\varrho)}$, is illustrated in the goal hierarchy, $h(\varrho)$, as shown in Figure 3-6.

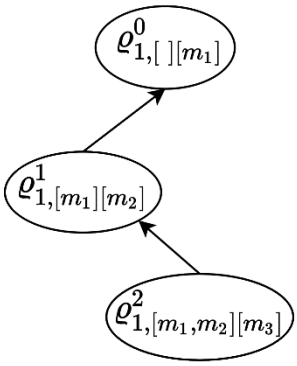


Figure 3-6. A teaming with vertical coordination

From the goal assignment, $\varrho^0_{1,[\,][m_1]}$, we know that member m_1 owns the team goal (also their individual goal), ϱ^0_1, and is the top superior with the responsibility of achieving goal ϱ^0_1 as well as the goals under ϱ^0_1 in $h(\varrho)$, i.e., ϱ^1_1 and ϱ^2_1. From the goal assignment, $\varrho^1_{1,[m_1][\,m_2]}$, we know that m_2 is assigned the goal ϱ^1_1 by m_1 and m_2 is responsible for achieving this goal as well as the goal under it in $h(\varrho)$. Based on this assignment, m_2's direct superior is m_1 and m_2 is the direct subordinate of m_1. From the goal assignment, $\varrho^1_{2,[m_1,m_2][m_3]}$, we know that m_3 is assigned the goal ϱ^2_1 by m_2 and m_3 and is responsible for achieving the goal ϱ^2_1. We also know that m_3 has two superiors, m_1 and m_2, and m_2 is m_3's direct superior.

The key characteristics of this type of teaming are the subordinates' individual goals which are a subset of the individual goals of their superiors. In other words, the subordinates have no individual goals outside their superiors' ones.

B. Teaming with horizontal coordination

A teaming with horizontal coordination must have at least two team members ($k = 2$) and all members are associated with teammate relationships.

Such a teaming can occur in both team formations which are a team with a superior and a team without a superior. In the former case, members in a teaming own goals at the same level in a goal hierarchy and they have the same superior chain (m_{sup}). One example of a teaming with three members is shown in Figure 3-7(a). In the latter case, members are the owner of the team overall goal and are assigned some specific goals. One example of a teaming with four members is shown in Figure 3-7(b).

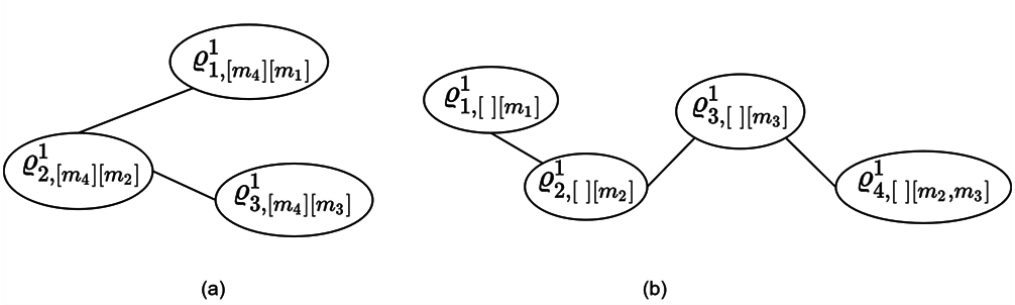


Figure 3-7. Horizontal coordination in teammate relationships with: (a) different goal, (b) shared goal

A member can have one or multiple goals and one goal can be assigned to one or multiple members. Depending on the relations between the individual goals of members in a teaming, there are two kinds of teaming situations. One situation is that all members have totally different individual goals. One example is shown in Figure 3-7(a), where members m_1, m_2 and m_3 have been assigned totally different goals by their superior m_4, ϱ^1_1 ϱ^1_2 and ϱ^1_3, respectively. We refer to this teaming situation as a normal teaming situation. Another situation is that the individual goals of some members overlap, i.e., some members have overlapping goals. One example of this situation is shown in Figure 3-7(b), in which all three members own the overall goal at level 0 and have their individual goals at level 1. Two members , m_2 and m_3 also have a shared goal ϱ^1_4.

The key characteristics of this type of teaming are the members are at the same level in a goal hierarchy and each member has their own goal although some members may have shared goals.

C. Teaming with mixed coordination

A teaming with mixed coordination must have at least three team members ($k = 3$) and some members are involved in both vertical coordination (due to superior-subordinate relationships) and horizontal coordination (due to teammate relationships). To explain this formation, let us consider a team with four members (namely m_1, m_2, m_3 and m_4) and two levels of goal hierarchy. The goals are distributed among four members, as shown in Figure 3-8.

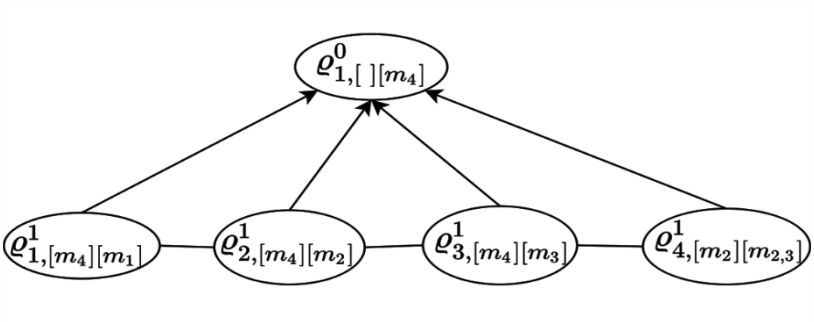


Figure 3-8. Mixed coordination

In this figure, the relationship between m_4 and the other three team members is a superior-subordinate relationship, respectively. In these relationships, m_4 is the superior while the other team members are subordinates. The relationships among m_1 to m_3 at level 1 are teammate relationships.

The key characteristics of this type of teaming are that there are two kinds of relationships among members and some members are involved in both a superior-subordinate relationship and teammate relationship at the same time.

3.3.2.4. Teaming Situation Awareness Specification Block

The teaming SA specification block is the fourth block in the framework. It specifies SA models for four kinds of simple teaming formations, namely, the supportive model, team model, shared model and mutual model, all of which contain only one kind of relationship among team members. We refer to these SA models as atomic teaming SA models. This block also suggests a complex SA model for a complex teaming formation with mixed coordination. The rest of this section explains each atomic SA model in detail with examples.

A. Supportive model

The supportive model (also referred to as the supportive situation awareness (SSA) model in this study) is used for a teaming with vertical coordination only. It specifies the relations among the SA-related factors for each individual member. In this study, we only consider the HAT context, in which a human member and an autonomy agent may have different SA requirements to achieve the same goal, e.g., to keep a safe distance from the vehicle in front in the same path, a human member and an autonomy agent will have different SA requirements. A human member needs to cognitively estimate the speed of the vehicle in front and the distance that they feel is the safe distance, while an autonomy agent needs to read the distance value from the sensory tools, calculate the vehicle in front's speed given the distance variable and its own vehicle's speed. Then, it reads the rule specified to define the safe margin. At this point, the human and autonomy agent have the same goal of keeping a safe distance, but they may have different sub-goals as the goal-hierarchy is extracted from the task-hierarchy.

Definition 10. A supportive model specifies the relations among SA and SA-related factors in a teaming with vertical coordination and contains the following features:

- Goal: A superior's individual goal subsumes their subordinate's individual goals. In other words, a subordinate's individual goal is a subset of the individual goals of its superiors.
- Owner: The top superior
- Goal-related environment: The goal-related environment for the teaming is the union of individual member's goal-related environments.
- SA elements: The SA elements for the teaming are the union of individual member's SA elements.
- SA requirements: The SA requirements for the teaming are the union of individual member's SA requirements and the individual member's SA requirements may be different for the same goal.

- SADP: Individual members have different situation awareness development processes.

- SA: The top superior's SA subsumes its subordinates' SA to represent that subordinates are under the supervision of their superiors.

Let's consider one example teaming with two members, $m = \{m_1, m_2\}$, where m_1 is the superior and m_2 is the subordinate who is an autonomy agent. Figure 3-9 shows the goal hierarchy with member assignments.

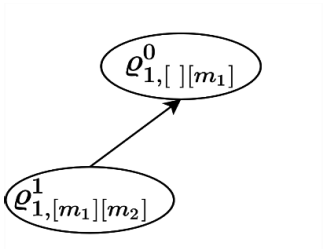


Figure 3-9. Supportive model case

The supportive model for this example teaming is depicted in Table 3-1 and is also illustrated in Figure 3-10.

Table 3-1. Situation awareness-related factors in the supportive model case

Situation awareness-related factors	Situation awareness model for m_1	Situation awareness model for m_2	Team situation awareness model
Goal	$\varrho^0_{1,[\,][m_1]}$	$\varrho^1_{1,[m_1][m_2]}$	$\varrho^0_{1,[\,][m_1]} \cap \varrho^1_{1,[m_1][m_2]} = \varrho^1_{1,[m_1][m_2]}$ $\varrho_{team} = \varrho^0_{1,[\,][m_1]}$
Owner	m_1	m_2	m_1
Goal-related environment	$\vartheta_G^{m_1}$	$\vartheta_G^{m_2}$	$\vartheta_G^{team} = \vartheta_G^{m_1} \cup \vartheta_G^{m_2}$, $\vartheta_G^{m_1} \cap \vartheta_G^{m_2} = \vartheta_G^{m_2}$
Situation awareness elements	$\varsigma_{m_1}^{SA}$	$\varsigma_{m_2}^{SA}$	$\varsigma_{team}^{SA} = \varsigma_{m_1}^{SA} \cup \varsigma_{m_2}^{SA}$, $\varsigma_{m_2}^{SA} \subset \varsigma_{m_1}^{SA}$
Situation awareness requirements	$\mathcal{R}_{m_1}^{SA}$	$\mathcal{R}_{m_2}^{SA}$	$\mathcal{R}_{team}^{SA} = \mathcal{R}_{m_1}^{SA} \cup \mathcal{R}_{m_2}^{SA}$, $\mathcal{R}_{m_1}^{SA} \cap \mathcal{R}_{m_2}^{SA} \neq \emptyset$ (on shared goal)
Situation awareness development process	$\wp_{m_1}^{SA} \rightarrow SA_{m_1}$	$\wp_{m_2}^{SA} \rightarrow SA_{m_2}$	$\wp_{m_1}^{SA} \neq \wp_{m_2}^{SA}$

Situation awareness-related factors	Situation awareness model for m_1	Situation awareness model for m_2	Team situation awareness model
Situation awareness	SA_{m_1}	SA_{m_2}	$SA_{m_1} \cup SA_{m_2}$, $SA_{m_2} \subset SA_{m_1}$

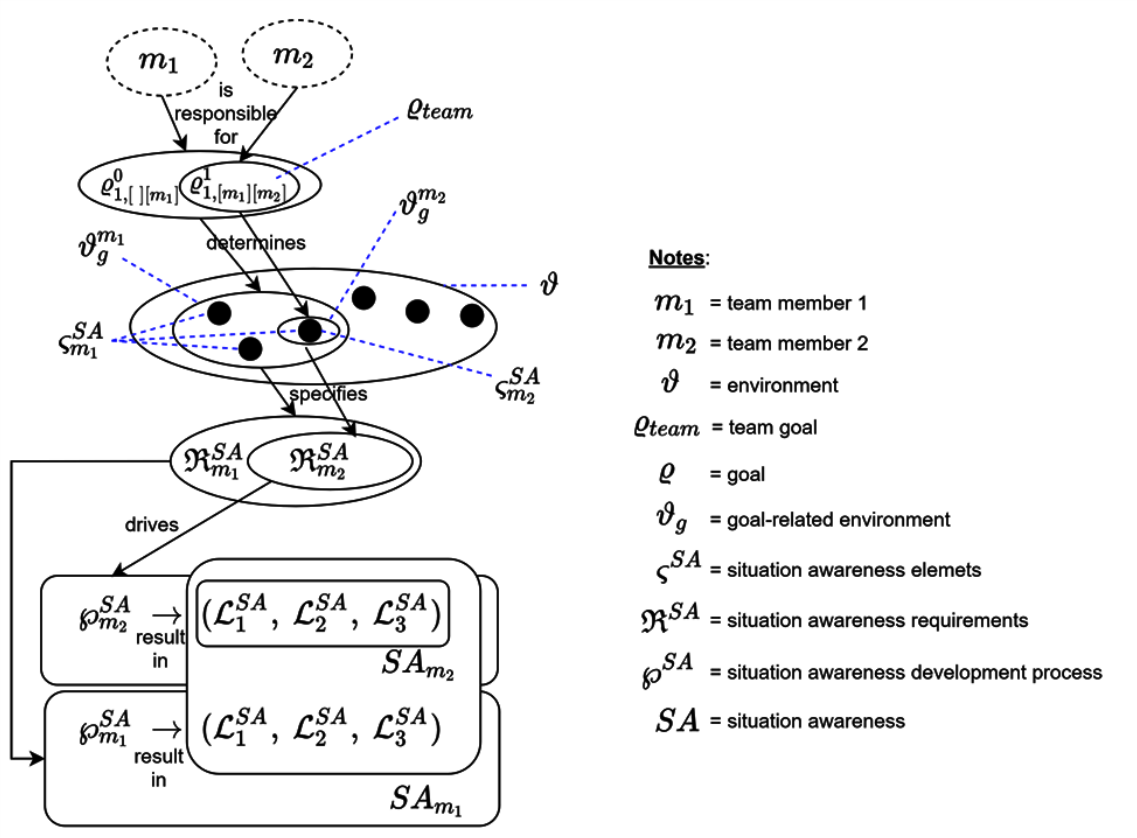


Figure 3-10. Supportive model in a two-case team member scenario

B. Team model

Definition 11. A team SA model specifies the relations among SA and SA-related factors in a teaming with horizontal coordination only.

This model contains the following features:

- Goal: The individual goals of members are different, and their intersection produces an empty set.
- Owner: All members are involved in the teaming

- Goal-related environment: The goal-related environment for the teaming is the union of individual member's goal-related environments. However, the team members' goal-related environment is independent of each other.

- SA elements: The SA elements for the teaming are the union of an individual member's SA elements.

- SA requirements: The SA requirements for the teaming are the union of the individual member's SA requirements and the individual member's SA requirements for the shared goal is an empty set as they have completely different goals.

- SADP: Individual members have different situation awareness development processes.

- SA: SA for the teaming is the unification of the individual member's SA.

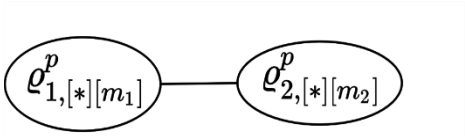


Figure 3-11. Team model case

Figure 3-11 illustrates the goal distribution in a teaming with two members, $m = \{m_1, m_2\}$, where all members of m are at the same level p. There are two possible cases depending on the team formation. One case is in a team without a superior, in which $p = 0$ and $*$ represents an empty list ($\{\}$) (i.e., no superior for these team members). The other case is in a team with a superior, in which $p \neq 0$ and $*$ represents the superior chain $\{m_1, \dots, m_{p-1}\}$. In both cases, the members in a teaming have teammate relationships and they have horizontal coordination only in developing their SA.

The team SA model for this example teaming is depicted in Table 3-2 and illustrated in Figure 3-12.

Table 3-2. Situation awareness-related factors in the team model case

Situation awareness-related factors	Situation awareness model for m_1	Situation awareness model for m_2	Team situation awareness model
Goal	$\varrho^{p}_{1,[*][m_1]}$	$\varrho^{p}_{1,[*][m_2]}$	$\varrho^{p}_{1,[*][m_1]} \cap \varrho^{p}_{1,[*][m_1]} = \emptyset$ $\varrho_{team} = \varrho^{p}_{1,[*][m_1]} \cup \varrho^{p}_{1,[*][m_1]}$
Owner	m_1	m_2	m_1, m_2
Goal-related environment	$\vartheta_{G}^{m_1}$	$\vartheta_{G}^{m_2}$	$\vartheta_{G}^{team} = \vartheta_{G}^{m_1} \cup \vartheta_{G}^{m_2}$, $\vartheta_{G}^{m_1} \cap \vartheta_{G}^{m_2} = \emptyset$
Situation awareness elements	$\varsigma^{SA}_{m_1}$	$\varsigma^{SA}_{m_2}$	$\varsigma^{SA}_{team} = \varsigma^{SA}_{m_2} \cup \varsigma^{SA}_{m_1}$
Situation awareness requirements	$\mathcal{R}^{SA}_{m_1}$	$\mathcal{R}^{SA}_{m_2}$	$\mathcal{R}^{SA}_{team} = \mathcal{R}^{SA}_{m_1} \cup \mathcal{R}^{SA}_{m_2}$, $\emptyset$ (on shared goal)
Situation awareness development process	$\wp^{SA}_{m_1} \rightarrow SA_{m_1}$	$\wp^{SA}_{m_2} \rightarrow SA_{m_2}$	$\wp^{SA}_{m_1} \neq \wp^{SA}_{m_2}$
Situation awareness	SA_{m_1}	SA_{m_2}	$SA_{m_1} \cup SA_{m_2}$

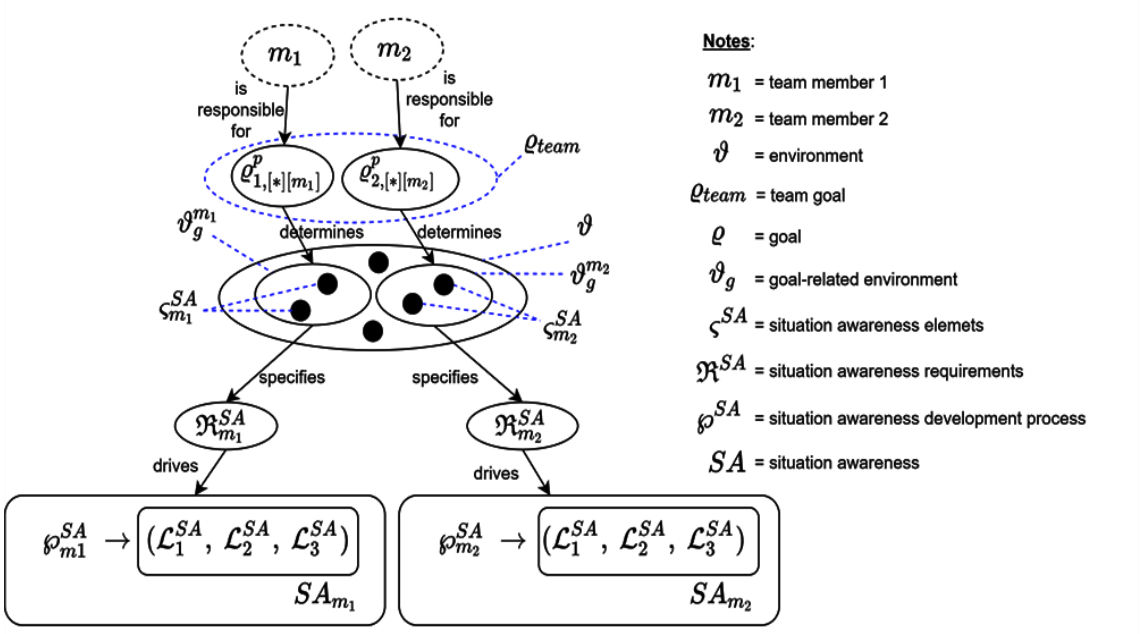


Figure 3-12. Team model in a two-team member scenario

C. Shared model

Definition 12. A shared SA model specifies the relations among SA and SA-related factors in a teaming with horizontal coordination, where each member possesses a shared goal and some non-shared goals, and the goal-related environment and the SA requirements of all team members for the shared goal are the same.

This model contains the following features:

- Goal: The individual goals of some or all team members overlap, and each team member possesses a non-shared goal.
- Owner: All members are involved in the teaming.
- Goal-related environment: The goal-related environment for the teaming is the union of an individual member's goal-related environments. However, team members' goal-related environments may have an overlapping part that corresponds to the shared goal.
- SA elements: The SA elements for the teaming are the union of an individual member's SA elements.
- SA requirements: The SA requirements for the teaming are the union of an individual member's SA requirements and the individual member's SA requirements for the shared goal are similar but completely different for the non-shared goal.
- SADP: Individual members have different situation awareness development processes.
- SA: SA for the teaming is the unification of an individual member's SA.

Figure 3-13 illustrates one example case of teaming with two members, $m = \{m_1, m_2\}$ where all members of m are at the same level p.

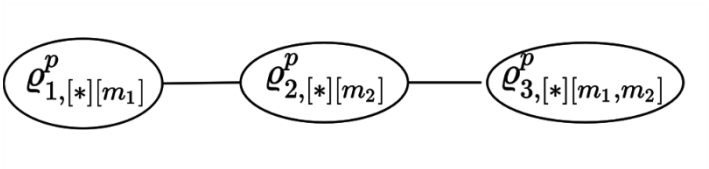


Figure 3-13. Shared model case

The shared SA model for this example teaming is depicted in Table 3-3 and Figure 3-14.

Table 3-3. Situation awareness-related factors in shared model case

Situation awareness-related factors	Situation awareness model for m_1	Situation awareness model for m_2	Team situation awareness model
Goal	$\varrho^{p}_{1,[*][m_1]}$	$\varrho^{p}_{1,[*][m_2]}$	$\varrho^{p}_{1,[*][m_1]} \cap \varrho^{p}_{1,[*][m_2]} \neq \emptyset$ AND $\varrho^{p}_{1,[*][m_1]} \cap \varrho^{p}_{1,[*][m_2]} \neq \varrho^{p}_{1,[*][m_1]}$ AND $\varrho^{p}_{1,[*][m_1]} \cap \varrho^{p}_{1,[*][m_2]} \neq \varrho^{p}_{1,[*][m_2]}$, $\varrho_{team} = \varrho^{p}_{1,[*][m_1]} \cup \varrho^{p}_{1,[*][m_2]}$
Owner	m_1	m_2	m_1, m_2
Goal-related environment	$\vartheta_{\boldsymbol{G}}^{m_1}$	$\vartheta_{\boldsymbol{G}}^{m_2}$	$\vartheta_{\boldsymbol{G}}^{team} = \vartheta_{\boldsymbol{G}}^{m_1} \cup \vartheta_{\boldsymbol{G}}^{m_2}$, $\vartheta_{\boldsymbol{G}}^{m_1} \cap \vartheta_{\boldsymbol{G}}^{m_2} \neq \emptyset$ AND $\vartheta_{\boldsymbol{G}}^{m_1} \cap \vartheta_{\boldsymbol{G}}^{m_2} \neq \vartheta_{\boldsymbol{G}}^{m_1}$ AND $\vartheta_{\boldsymbol{G}}^{m_1} \cap \vartheta_{\boldsymbol{G}}^{m_2} \neq \vartheta_{\boldsymbol{G}}^{m_2}$
Situation awareness elements	$\varsigma_{m_1}^{SA}$	$\varsigma_{m_2}^{SA}$	$\varsigma_{team}^{SA} = \varsigma_{m_1}^{SA} \cup \varsigma_{m_2}^{SA}$, $\varsigma_{m_1}^{SA} \cap \varsigma_{m_2}^{SA} \neq \emptyset$ AND $\varsigma_{m_1}^{SA} \cap \varsigma_{m_2}^{SA} \neq \varsigma_{m_1}^{SA}$ AND $\varsigma_{m_1}^{SA} \cap \varsigma_{m_2}^{SA} \neq \varsigma_{m_2}^{SA}$
Situation awareness requirements	$\mathcal{R}_{m_1}^{SA}$	$\mathcal{R}_{m_2}^{SA}$	$\mathcal{R}_{team}^{SA} = \mathcal{R}_{m_1}^{SA} \cup \mathcal{R}_{m_2}^{SA}$, $\mathcal{R}_{m_1}^{SA} = \mathcal{R}_{m_2}^{SA}$ (on shared goal)
Situation awareness development process	$\wp_{m_1}^{SA} \rightarrow SA_{m_1}$	$\wp_{m_2}^{SA} \rightarrow SA_{m_2}$	$\wp_{m_1}^{SA} \neq \wp_{m_2}^{SA}$
Situation awareness	SA_{m_1}	SA_{m_2}	$SA_{m_1} \cup SA_{m_2}$

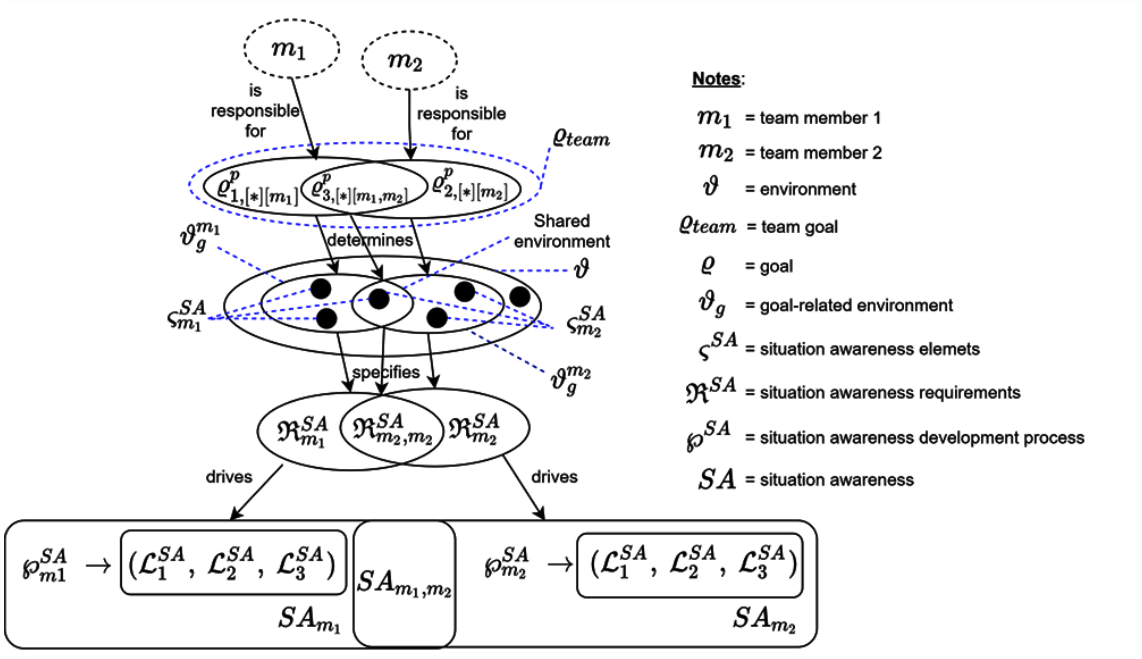


Figure 3-14. Shared model in a two-team member scenario

D. Mutual model

Definition 13. A mutual SA model specifies the relations among SA and SA-related factors in a teaming with horizontal coordination, where each member possesses a shared goal and some non-shared goals, and the SA requirements of all team members are completely different from each other.

This model contains the following features:

- Goal: The individual goals of some or all team members have an overlap (i.e., the shared goal), and each team member possesses some non-shared goals.
- Owner: All members are involved in teaming.
- Goal-related environment: The goal-related environment for the teaming is the union of an individual member's goal-related environments. However, team members' goal-related environment may have partially the same goal-related environment if the goals intersect.

- SA elements: The SA elements for the teaming are the union of an individual member's SA elements.
- SA requirements: The SA requirements for the teaming are the union of an individual member's SA requirements and the individual member's SA requirements for the shared goal and non-shared goal are completely different.
- SADP: Individual members have different situation awareness development processes.
- SA: SA for the teaming is the unification of an individual member's SA.

To explain this model, we use Figure 3-13 to illustrate the goal hierarchy distribution among team $m = \{m_1, m_2\}$. The mutual SA model for this example teaming is depicted in Table 3-4 and illustrated in Figure 3-15.

Table 3-4. Situation awareness-related factors in the mutual model case

Situation awareness-related factors	Situation awareness model for m_1	Situation awareness model for m_2	Team situation awareness model
Goal	$\varrho^{p}_{1,[*][m_1]}$	$\varrho^{p}_{1,[*][m_2]}$	$\varrho^{p}_{1,[*][m_1]} \cap \varrho^{p}_{1,[*][m_2]} \neq \emptyset$ AND $\varrho^{p}_{1,[*][m_1]} \cap \varrho^{p}_{1,[*][m_2]} \neq \varrho^{p}_{1,[*][m_1]}$ AND $\varrho^{p}_{1,[*][m_1]} \cap \varrho^{p}_{1,[*][m_2]} \neq \varrho^{p}_{1,[*][m_2]}$, $\varrho_{team} = \varrho^{p}_{1,[*][m_1]} \cup \varrho^{p}_{1,[*][m_2]}$
Owner	m_1	m_2	m_1, m_2
Goal-related environment	$\vartheta_{\boldsymbol{G}}^{m_1}$	$\vartheta_{\boldsymbol{G}}^{m_2}$	$\vartheta_{\boldsymbol{G}}^{team} = \vartheta_{\boldsymbol{G}}^{m_1} \cup \vartheta_{\boldsymbol{G}}^{m_2}$, $\vartheta_{\boldsymbol{G}}^{m_1} \cap \vartheta_{\boldsymbol{G}}^{m_2} \neq \emptyset$ AND $\vartheta_{\boldsymbol{G}}^{m_1} \cap \vartheta_{\boldsymbol{G}}^{m_2} \neq \vartheta_{\boldsymbol{G}}^{m_1}$ AND $\vartheta_{\boldsymbol{G}}^{m_1} \cap \vartheta_{\boldsymbol{G}}^{m_2} \neq \vartheta_{\boldsymbol{G}}^{m_2}$
Situation awareness elements	$\varsigma^{SA}_{m_1}$	$\varsigma^{SA}_{m_2}$	$\varsigma^{SA}_{team} = \varsigma^{SA}_{m_1} \cup \varsigma^{SA}_{m_2}$, $\varsigma^{SA}_{m_1} \cap \varsigma^{SA}_{m_2} \neq \emptyset$ AND $\varsigma^{SA}_{m_1} \cap \varsigma^{SA}_{m_2} \neq \varsigma^{SA}_{m_1}$ AND $\varsigma^{SA}_{m_1} \cap \varsigma^{SA}_{m_2} \neq \varsigma^{SA}_{m_2}$

Situation awareness-related factors	Situation awareness model for m_1	Situation awareness model for m_2	Team situation awareness model
Situation awareness requirements	$\mathcal{R}_{m_1}^{SA}$	$\mathcal{R}_{m_2}^{SA}$	$\mathcal{R}_{team}^{SA} = \mathcal{R}_{m_1}^{SA} \cup \mathcal{R}_{m_2}^{SA}$, $\mathcal{R}_{m_1}^{SA} \cap \mathcal{R}_{m_2}^{SA} = \emptyset$ (on shared goal)
Situation awareness development process	$\wp_{m_1}^{SA} \rightarrow SA_{m_1}$	$\wp_{m_2}^{SA} \rightarrow SA_{m_2}$	$\wp_{m_1}^{SA} \neq \wp_{m_2}^{SA}$
Situation awareness	SA_{m_1}	SA_{m_2}	$SA_{m_1} \cup SA_{m_2}$

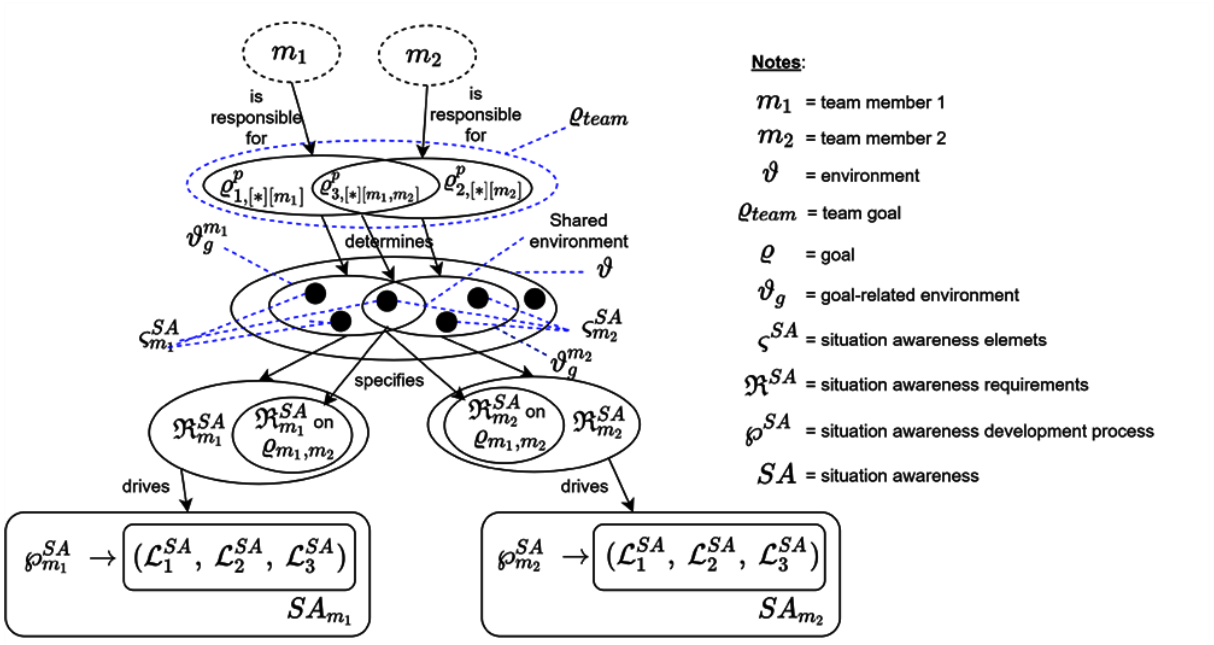


Figure 3-15. Mutual model in a two-team members scenario

E. Mixed model

The mixed model is for complex teaming in which there are members who are involved in both vertical and horizontal coordination.

Definition 14. A mixed SA model is a combination of atomic teaming SA models which include one supportive model and one of the following three atomic models: normal, shared, or mutual models.

To illustrate this complex teaming SA model, let's consider a teaming with three members, $m = \{m_1, m_2, m_3\}$ and two goal provision cases, as shown in Figure 3-16

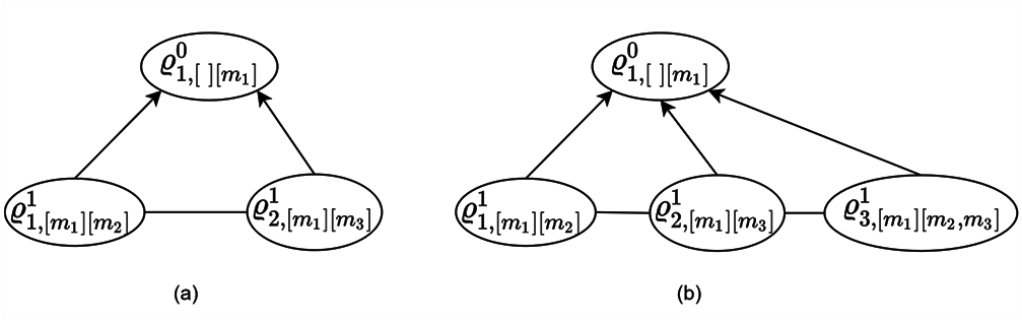


Figure 3-16. Mixed model case

In the first case (see Figure 3-16(a)), the goal provision shows a complex teaming which contains two kinds of simple teaming situations, one is teaming with vertical coordination and the other is teaming with horizontal coordination (normal teaming situation). The SA model for this kind of teaming is illustrated in Figure 3-17.

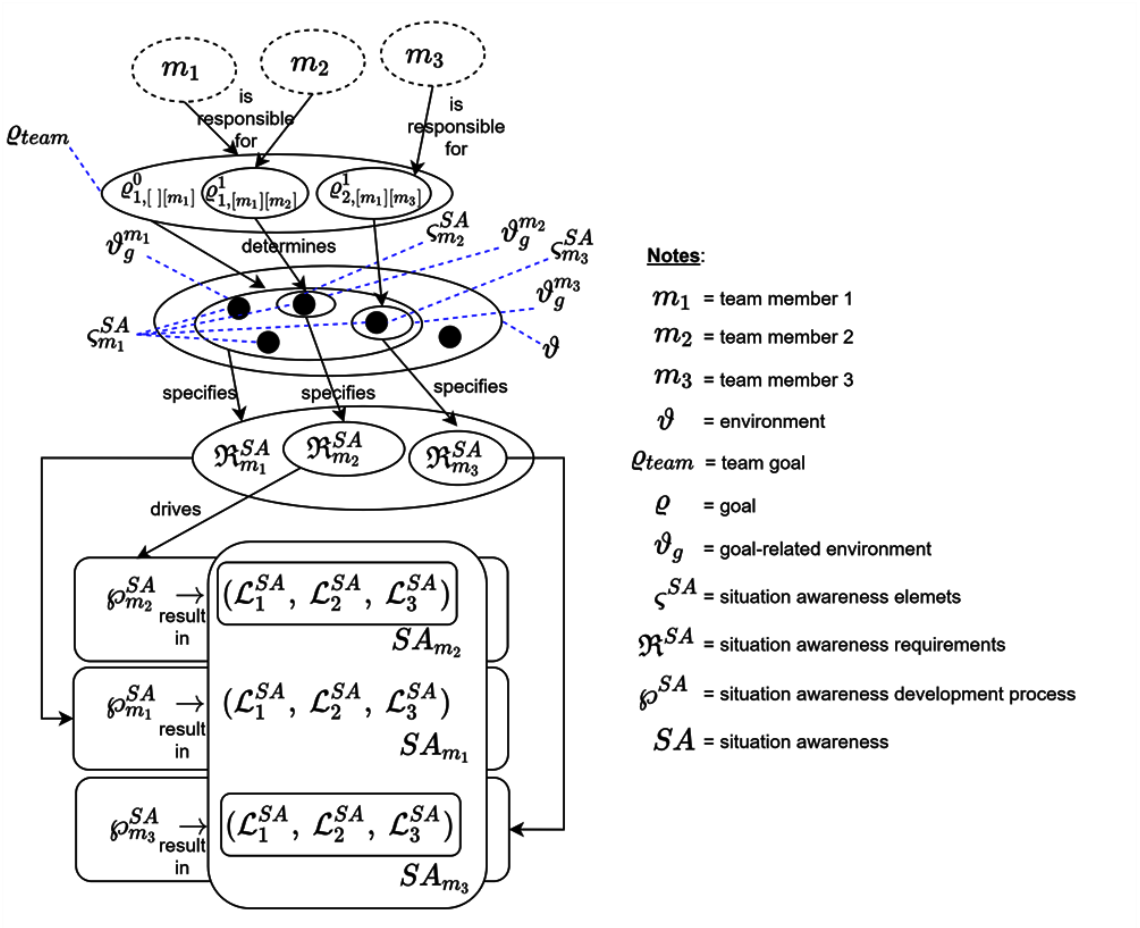


Figure 3-17. Mixed model: supportive and team model

In the second case (see Figure 3-16(b)), the goal provision shows a complex teaming which contains two kinds of simple teaming situations, one is teaming with vertical

coordination and the other is teaming with horizontal coordination. The goal provision illustrated in Figure 3-16(b) can be broken down into two possible SA models: the shared model or mutual model. The mixed SA model for a complex teaming with the shared model and mutual model is illustrated in Figure 3-18 and Figure 3-19, respectively.

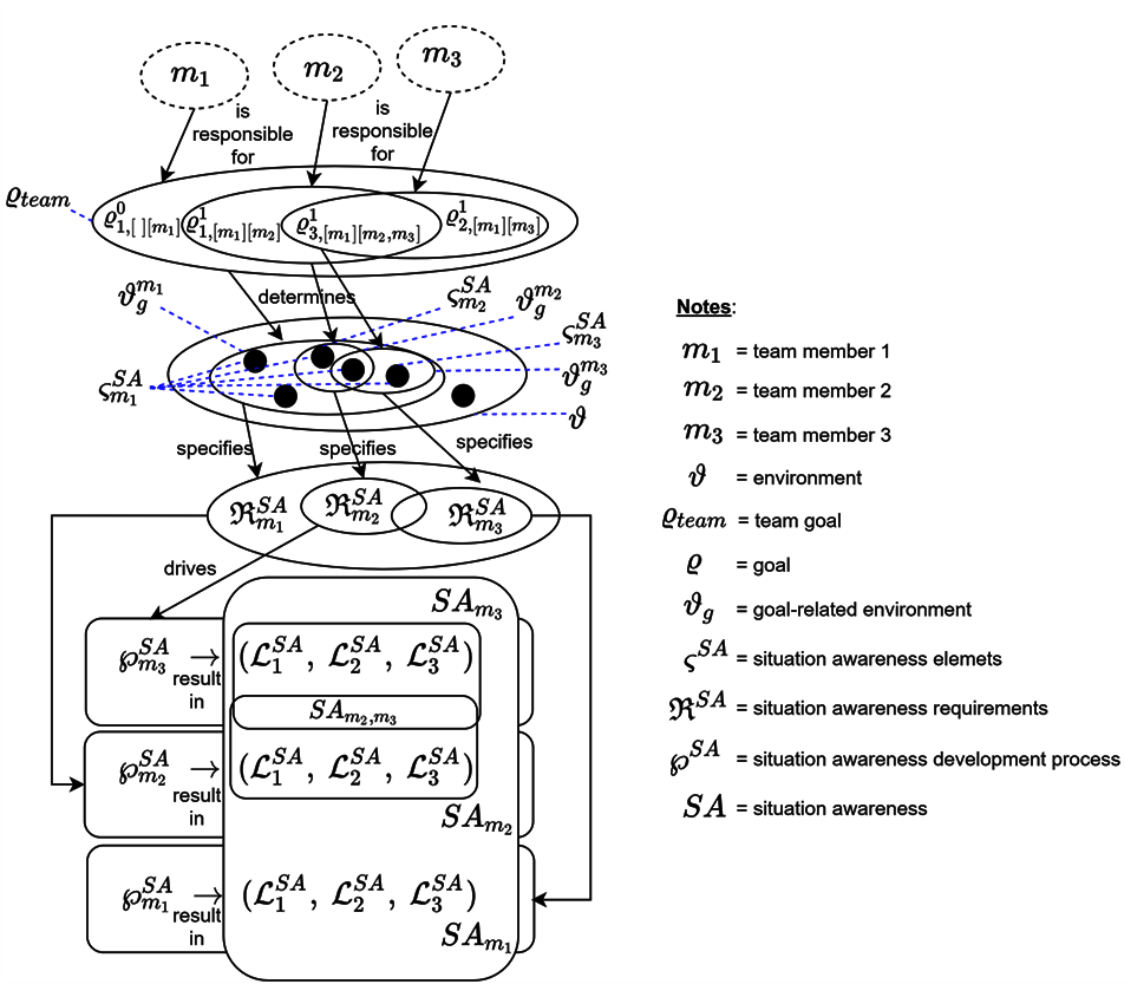


Figure 3-18. Mixed model: supportive and shared model

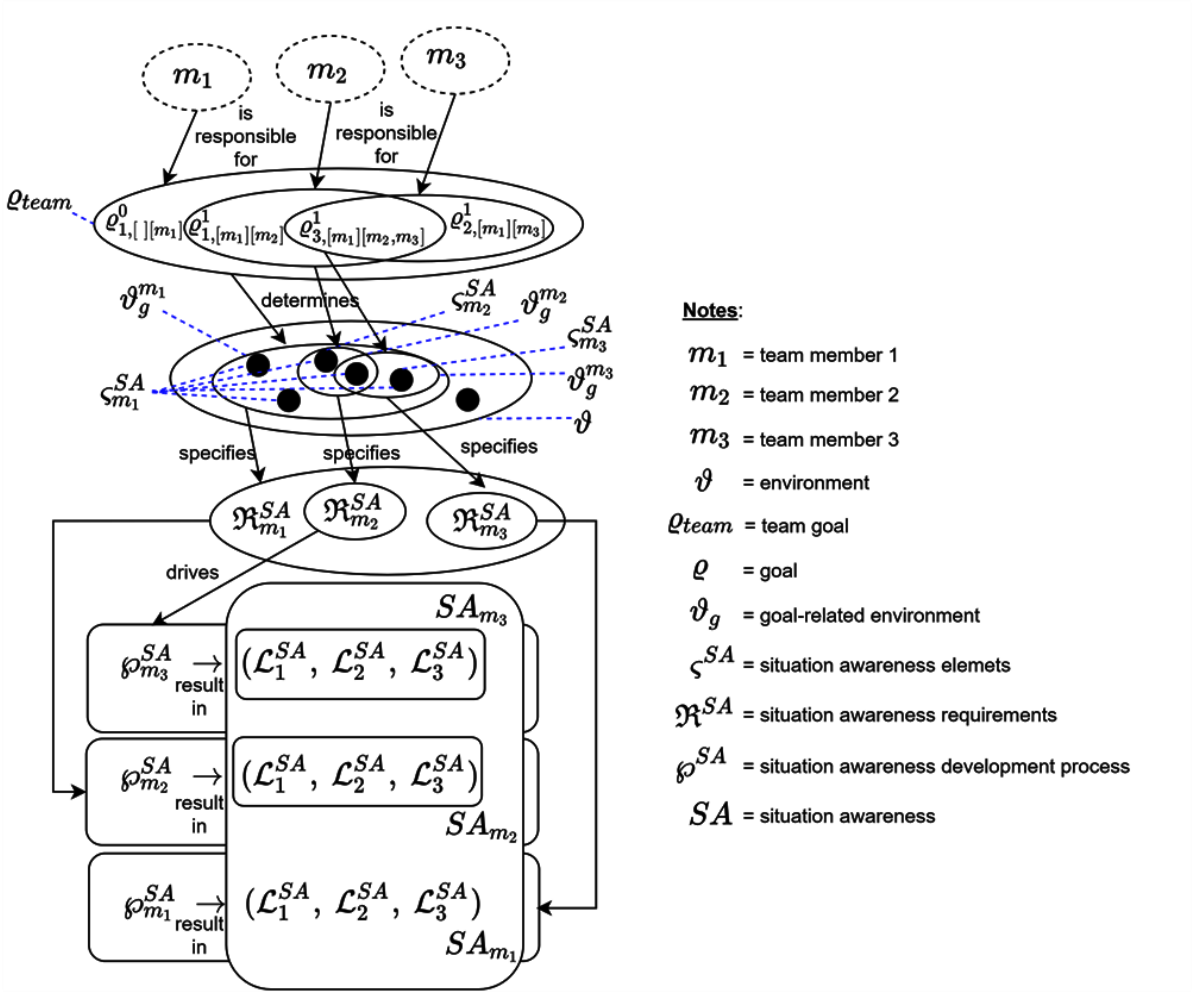


Figure 3-19. Mixed model: supportive and mutual model

3.4. An Illustrative Example of a Dry-Run

To validate the proposed framework conceptually, this section presents an example case of running through this framework to determine the SA models for teaming situations in the team.

3.4.1. Team Specification Block

In this block, we assume a five-member team $m = \{m_1, m_2, m_3, m_4, m_5\}$ with the team goal of "conquer the enemy in city B quietly". m_1, m_2, m_3 and m_5 are human team members and m_4 is an unmanned aerial vehicle, which can be considered as an autonomy agent.

By applying the hierarchical task analysis method, a task hierarchy is developed as shown in Figure 3-20.

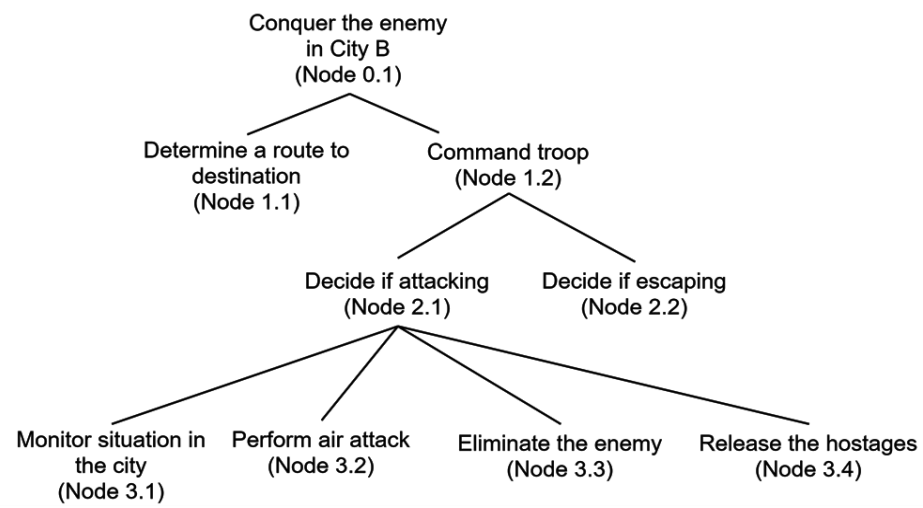


Figure 3-20. Task hierarchy for a team with mixed coordination in the example case

3.4.2. Goal Provision Block

In this block, a team with a superior is formed, in which m_5 is a general who owns the team goal ($\varrho^0_{1,[\,][m_5]}$). m_5 appoints m_1 as the field commander and assigns goals at Node 1.1 and Node 1.2 to m_1. m_1, as the field commander, assigns the goals at Node 3.3 and Node 3.1 to m_2, the goals at Node 3.4 to m_3 and the goals at Node 3.1 and Node 3.2 to m_4. Consequently, Node 3.1 becomes the shared goal between m_2 and m_4. A goal provision showing the above goal distributions and relations among members' individual goals are shown in Figure 3-21 and Figure 3-22, respectively.

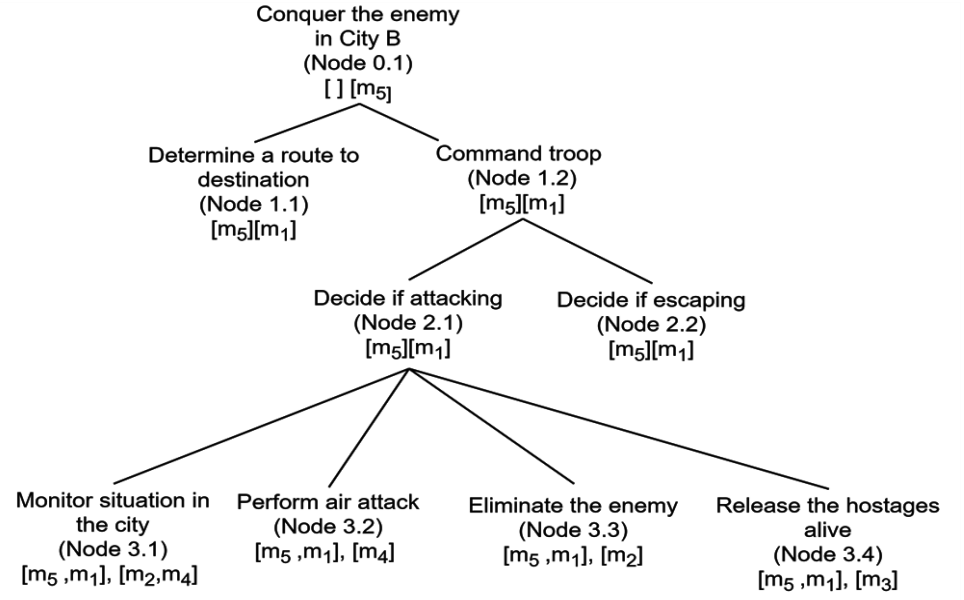


Figure 3-21. Goal provision for the example case

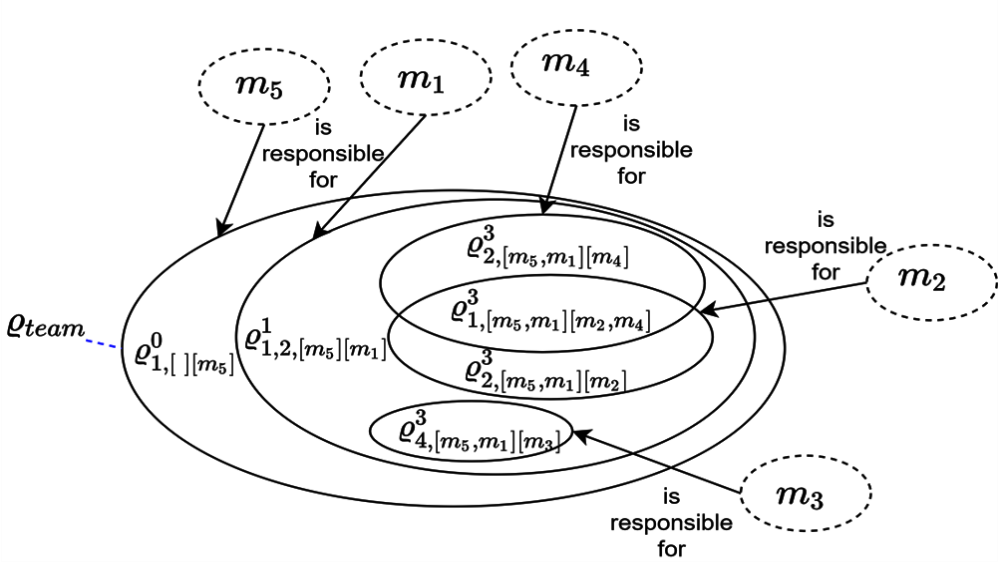


Figure 3-22. Goal relations for the example case

It can be seen that in this team, m_5 is the top superior with subordinates m_1 to m_5, and m_1 is the level-1 superior with subordinates m_2 to m_4.

3.4.3. Teaming Formation Block

In this block, two kinds of teaming are formed. One is teaming with vertical coordination and there are four instances of this kind:

(1) Teaming of m_5 and m_1,

(2) Teaming of m_5, m_1 and m_2,

(3) Teaming of m_5, m_1 and m_3,

(4) Teaming of m_5, m_1 and m_4.

The other is teaming with horizontal coordination and there is one instance of this kind, which is teaming of m_2 to m_4.

Consequently, this team has one complex teaming situation consisting of four simple teamings with vertical coordination and one simple teaming with horizontal coordination.

3.4.4. Teaming Situation Awareness Specification Block

In this block, based on the teaming formations in the previous block, SA-related factors are determined by defining the environment, goal-related environment, SA elements, and SA requirements for each member to form the teaming SA model. The following subsections illustrate the determination of relations among the individual SA-related factors of team members, respectively.

A. Goal

The team goal and members' individual goals are shown in the goal provision (Figure 3-21) and the relations of goals (Figure 3-22).

Owner: the top superior m_5

B. Environment

Based on the assignment $\varrho^0_{1,[\,][m_5]}$, we can infer that the environment ϑ = "City B and the region under m_5's authority" which means that in order to achieve the team goal, m_5 can arrange whatever is necessary within the region under his authority as a part of the strategy in conquering the enemy in City B. Hence, m_5's region and City B become the boundary of the environment.

C. Goal-related environment

To achieve the team goal, m_5 focuses only on his basecamp, a specific part of his region and City B such as the location to escape outside City B and the enemy's basecamp and its surrounding environment. Therefore, we define this scope of environment as m_5's goal-related environment ($\vartheta_G^{m_5}$) and the team's goal-related environment because m_5 is the team goal's owner. As m_5 appointed m_1 as the commander in the field, the subset of $\vartheta_G^{m_5}$ is shared with m_1 which becomes $\vartheta_G^{m_1}$. In this regard, $\vartheta_G^{m_1}$ covers the location to escape from City B and the enemy's basecamp and its surrounding environment. m_5's basecamp is not included in $\vartheta_G^{m_1}$.

With the assignments at level 3 set by field commander m_1, team members m_2 to m_4 acquire their individual goals which are a subset of $m_1's$ individual goals. Thus, they have a goal-related environment with a more narrowed scope than their superiors, which is the enemy's basecamp and its surroundings.

A more specific determination of their goal-related environment is as follows: From the assignment $\varrho^3_{1,[m_5,m_1][m_2,m_4]}$, the goal-related environments for m_2 and m_4 $\vartheta_G^{m_2}$ and $\vartheta_G^{m_4}$ include the enemy's basecamp from an outdoor perspective. From the assignment $\varrho^3_{3,[m_5,m_1][m_2]}$, $\vartheta_G^{m_2}$ also includes details of the human enemy. From assignment $\varrho^3_{2,[m_5,m_1][m_4]}$, $\vartheta_G^{m_4}$ also covers the airspace situation. From the assignment $\varrho^3_{4,[m_5,m_1][m_3]}$, the goal-related environment for m_3 $\vartheta_G^{m_3}$ covers the enemy's basecamp from an indoor perspective.

The relations among the five members' goal-related environments are presented in Table 3-5 and illustrated in Figure 3-23.

Table 3-5. Goal-related environment for the example case

Teaming formation instances	Goal-related environment coverage	Goal-related environment relation among team members
$\varrho^{0}_{1,[\,][m_5]}$	$\vartheta_G^{m_5}$= his basecamp, a specific part of his region, City B, a certain location outside City B	$\vartheta_G^{team} = \vartheta_G^{m_5}$
1. Teaming of m_5, m_1: $\varrho^{1}_{1,[m_5\,][m_1]}$, $\varrho^{1}_{2,[m_5\,][m_1]}$	$\vartheta_G^{m_1}$ = the escape location outside City B, City B, and the enemy's basecamp and its surrounding environment	$\vartheta_G^{m_1} \subset \vartheta_G^{m_5}$
2. Teaming of m_5, m_1, and m_2: $\varrho^{3}_{3,[\,m_5,m_1][m_2]}$	$\vartheta_G^{m_2}$ = human enemy	$\vartheta_G^{m_2} \subset \vartheta_G^{m_1} \subset \vartheta_G^{m_5}$
3. Teaming of m_5, m_1, and m_2: $\varrho^{3}_{4,[\,m_5,m_1][m_3]}$	$\vartheta_G^{m_3}$ = airspace	$\vartheta_G^{m_3} \subset \vartheta_G^{m_1} \subset \vartheta_G^{m_5}$
4. Teaming of m_5, m_1, and m_4: $\varrho^{3}_{2,[\,m_5,m_1][m_4]}$	$\vartheta_G^{m_4}$ = enemy's basecamp (indoors), hostages	$\vartheta_G^{m_4} \subset \vartheta_G^{m_1} \subset \vartheta_G^{m_5}$
5. Teaming of m_2 *to* m_4: $\varrho^{3}_{1,[\,m_5,m_1][m_2,m_4]}$	$\vartheta_G^{m_2}$, $\vartheta_G^{m_3}$ = enemy's basecamp (outdoors)	$\vartheta_G^{m_2} \cap \vartheta_G^{m_4} \neq \emptyset$, $\vartheta_G^{m_3} \cap \vartheta_G^{m_2} = \emptyset$, $\vartheta_G^{m_3} \cap \vartheta_G^{m_4} = \emptyset$

It can be seen from Figure 3-22 and Figure 3-23 that even though members m_2 to m_4 sound like they have the same environment, according to Definition 2, which states that an environment consists of a set of elements, they have different foci as their own goal-related environments.

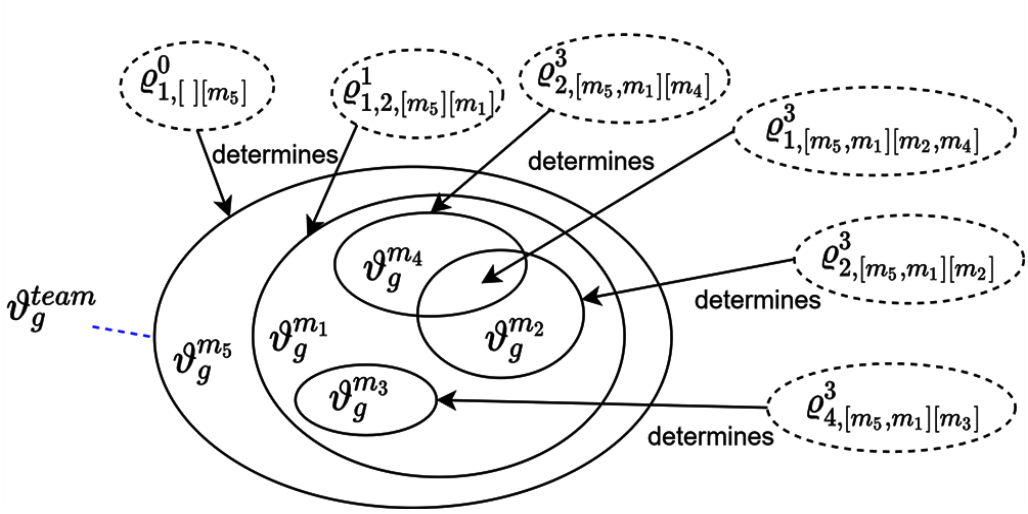


Figure 3-23. Goal-related environment relations for the example case

D. Situation awareness elements

The SA elements for each member ($\varsigma^{SA}{}_{m_j}$) are the relevant elements inside the corresponding members' goal-related environment $\vartheta_{\boldsymbol{G}}^{m_j}, j = 1, 2, \ldots, 5$. The identified SA elements for each member are presented in Table 3-6.

Table 3-6. Situation awareness elements for the example case

Team members' goal	Situation awareness elements ς^{SA}
$\varrho^{3}_{1,[\,m_5,m_1][m_2,m_4]}$	$\varsigma^{SA}_{m_2,m_4}$ = enemy's basecamp building, tower, ammunition storage, vehicles, activity's schedule
$\varrho^{3}_{2,[\,m_5,m_1][m_4]}$	$\varsigma^{SA}_{m_4}$ = enemy's airspace situation and squad
$\varrho^{3}_{3,[\,m_5,m_1][m_2]}$	$\varsigma^{SA}_{m_2}$ = enemies and people associated with them
$\varrho^{3}_{4,[\,m_5,m_1][m_3]}$	$\varsigma^{SA}_{m_3}$ = hostages, enemy's building block/rooms
$\varrho^{1}_{1,[m_5\,][m_1]}$, $\varrho^{1}_{2,[m_5\,][m_1]}$	$\varsigma^{SA}_{m_1}$ = $\varsigma^{SA}_{m_2} \cup \varsigma^{SA}_{m_3} \cup \varsigma^{SA}_{m_4} \cup$ route, building, possible transportation from enemy's basecamp to escape spot outside City B
$\varrho^{0}_{1,[\,][m_5]}$	$\vartheta_{\boldsymbol{G}}^{m_5}$ = $\varsigma^{SA}_{m_1} \cup \varsigma^{SA}_{m_2} \cup \varsigma^{SA}_{m_3} \cup \varsigma^{SA}_{m_4} \cup$ other elements related to his other task at his basecamp

E. Situation awareness requirements

After the SA elements (ς^{SA}) are identified, SA requirements ($\mathcal{R}^{SA}$) can be determined. A specific determination of SA requirements for individual members is explained as follows: From $\varsigma^{SA}_{m_2,m_4}$, the $\mathcal{R}^{SA}$ for these two team members m_2 and m_4 in association with goal assignment $\varrho^{3}_{1,[\,m_5,m_1][m_2,m_4]}$ include a map of enemy buildings, activity schedule, and vehicle identification numbers, times when the gate will be locked or opened, and so on. In addition, $\mathcal{R}^{SA}_{m_4}$ also includes the number of the enemy's airspace squad and the target spots to attack which will have the maximum effect on the enemy's destruction. $\mathcal{R}^{SA}_{m_2}$ also includes the enemy's identity and the people associated with them, their habits, and the regular places to visit. Based on the $\varsigma^{SA}_{m_3}$, $\mathcal{R}^{SA}_{m_3}$ contains the identity of hostages, the way in and way out of the enemy's building, the enemy's building layout, and so on. Finally, the $\mathcal{R}^{SA}_{m_1}$ covers all $\mathcal{R}^{SA}$ from his subordinates (m_2 to m_4). The relations among $\mathcal{R}^{SA}$ of individual members are illustrated in Figure 3-24.

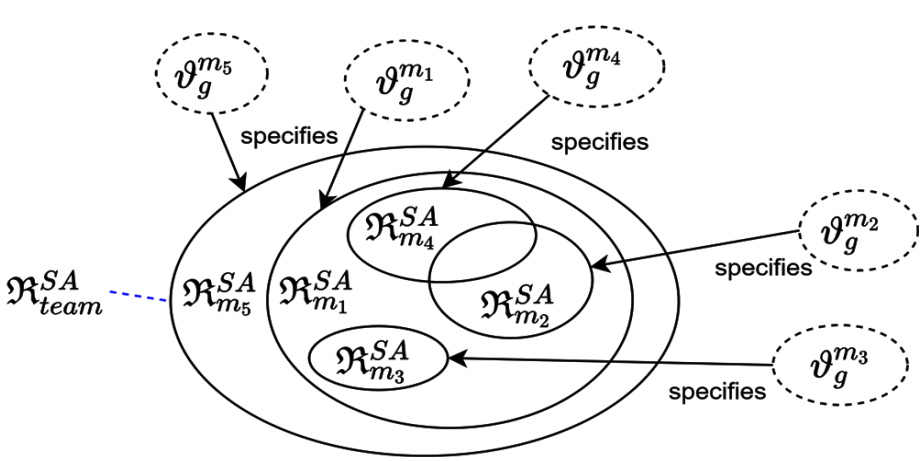


Figure 3-24. Situation awareness requirement relations for the example case

F. Situation awareness development process

Based on the SA requirements, the SA of each individual will be developed through their own SADP ($\wp^{SA}$). During these processes, they coordinate with relevant members according to the teaming in which they are involved. Table 3-7 lists four instances of teaming with vertical coordination and one instance of teaming with horizontal coordination along with the teaming SA models used, the roles of members in coordination and communication flows in coordination.

Table 3-7. Coordination among team members for the situation awareness development processes in the example case

Teaming situation	Members involved with their roles	Communication flow	Teaming situation awareness model
Teaming with vertical coordination	m_1 and m_2 m_1 superior m_2 subordinate	m_1 receives report from m_2 regarding goals in node 3.1 and node 3.3, which includes enemy's basecamp situation and enemy. m_1 issues attack order to m_2	Supportive model
	m_1 and m_3 m_1 superior m_3 subordinate	m_1 receives report from m_3 regarding goals in node 3.4 which includes hostage condition and enemy's building situations. m_1 approves the plans to release the hostages	Supportive model

	m_1 and m_4 m_1 superior m_4 subordinate	m_1 receives report from m_4 regarding the goals in node 3.1 and issue a command to attack associated with the goal in node 3.2	Supportive model
	m_5 and m_1 m_5 superior m_1 subordinate	m_1 reports to m_5 about the dynamic situation when conquering the enemy in City B	Supportive model
Teaming with horizontal coordination	m_2 and m_4	m_2 and m_4 work together to achieve the goals in node 3.1.	Shared model
	m_3 and both m_2 and m_4	m_3 has no share goal with the other two members	Team model

G. Situation awareness

Each member has an individual SA. For example, with SA_{m_2}, m_2 can be aware of the plans to eliminate the enemy. With SA_{m_3}, m_3 can decide when and how to get in to release the hostage and how to get out. With SA_{m_4}, m_4 can provide an effective airspace movement to destroy the vital objects in the enemy's basecamp. With SA_{m_5}, m_5 can orchestrate his subordinates to conquer the enemy, such as arranging a time for each subordinate to attack and the timing for his subordinates to go to an assembly point after attacking the enemy. SA_{m_1} can provide additional support just in case undesired results occur during the team operation. The relation among all the team members' SA is illustrated in Figure 3-25.

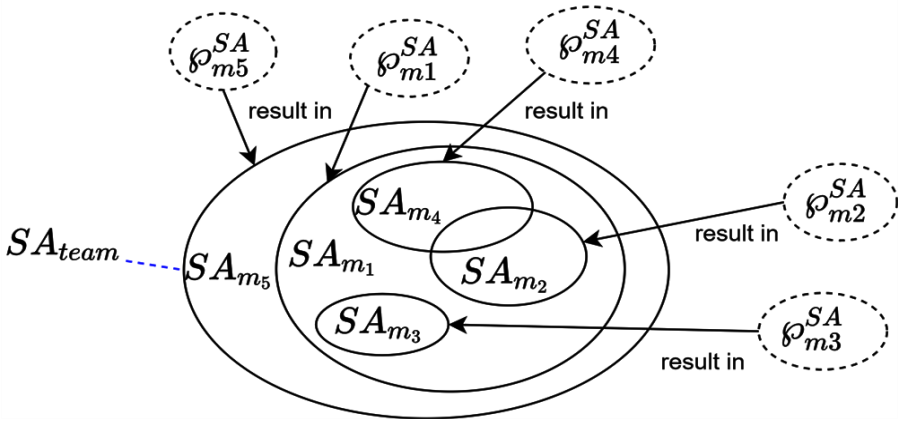


Figure 3-25. Situation awareness-relations for the example case

3.5. Summary

In this chapter, a set of notions related to SA is given first to provide a basic common ground to understand the SA modelling framework for teaming situations. A SA modelling framework for teaming situations is then proposed. With the explanation of this proposed framework, a number of terms including team formation, relationships among team members, teaming formation and coordination in the SA development process, a generic SA model and a teaming SA model are clearly defined. This SA modelling framework can conceptualize SA models for a given teaming situation based on team specifications, goal provision and teaming formation. Based on this framework, four atomic SA models are explained, namely the supportive model, team model, shared model, and mutual model. While the supportive model is generated from a simple teaming with vertical coordination, the other three models (team, shared, and mutual models) are designed for a simple teaming with horizontal coordination. The SA model for a complex teaming can be obtained by combining the atomic SA models. During the SA modelling framework development, the main issue is on how we define a goal. Among many perspectives of the goal definition, we viewed a goal to be achieved as a set of tasks with two considerations: 1) to make easier to convert team goals into individual goals of team members, and 2) to support the design of autonomy agent. An illustrative example of a dry run through this framework is presented to conceptually validate this framework. This example emphasizes that identifying team goals and individual goals of team members are critical point and not always an easy task. In an organizational case, team goals might have been written. However, in HAT case, such written goals might not exist (i.e., collaborative driving case).

Based on this framework, a HAT context with vertical coordination in which this study focused on, can be considered as “HAT with a supportive model”.

Chapter 4 : Development of an Autonomy Agent's Situation Awareness

4.1. Introduction

To address the second objective of this study, which is how to develop the autonomy agent's SA to be part of a HAT with a superior-subordinate relationship, we selected a collaborative driving context as a HAT case. In this HAT case, the team has two members, one is the human driver, and the other is an autonomy agent that is an on-board advanced driving assistance system (ADAS) used in the autopilot mode in a vehicle. Suppose this autonomy agent has its own SA so it can be considered as the teammate of the human driver in this team. This chapter presents the development of the SA of an autonomy agent in a HAT based on the SA modelling framework presented in Chapter 3.

The rest of this chapter is structured as follows. The team specification and goal provision in collaborative driving is presented in Section 4.2 and Section 4.3, respectively. Teaming formation and teaming SA specification is described in Section 4.4. A situation awareness development process (SADP) in the collaborative driving context using traffic lights and overtaking scenarios is explained in Section 4.5. In Section 4.6, we proposed a SADP framework as a design of SADP and the implementation of this SADP framework in two goal achieving cases. In Section 4.7, the autonomy agent's SA in the two goal achieving cases is presented along with the simulation in CARLA, which is an open-source simulator for autonomous driving. Section 4.8 summarises and concludes the chapter.

4.2. Team Specification in Collaborative Driving

As driving situations can be overly complex, this study narrows the driving situations to collaborative driving situations to showcase how the teaming SA modelling framework in Chapter 3 can guide the development of individual SA in a teaming.

In collaborative driving, let $m = \{m_1, m_2\}$ denote the two team members where m_1 represents the human driver and m_2 represents the on-board ADAS, which can be

considered as the autonomy agent. The human driver m_1 is the superior and the ADAS agent m_2 is the subordinate. This is a team with a superior. In this collaborative driving team, the team goal is to drive from one place to a destination. Through the hierarchical task analysis, a goal hierarchy is developed as illustrated in Figure 4-1.

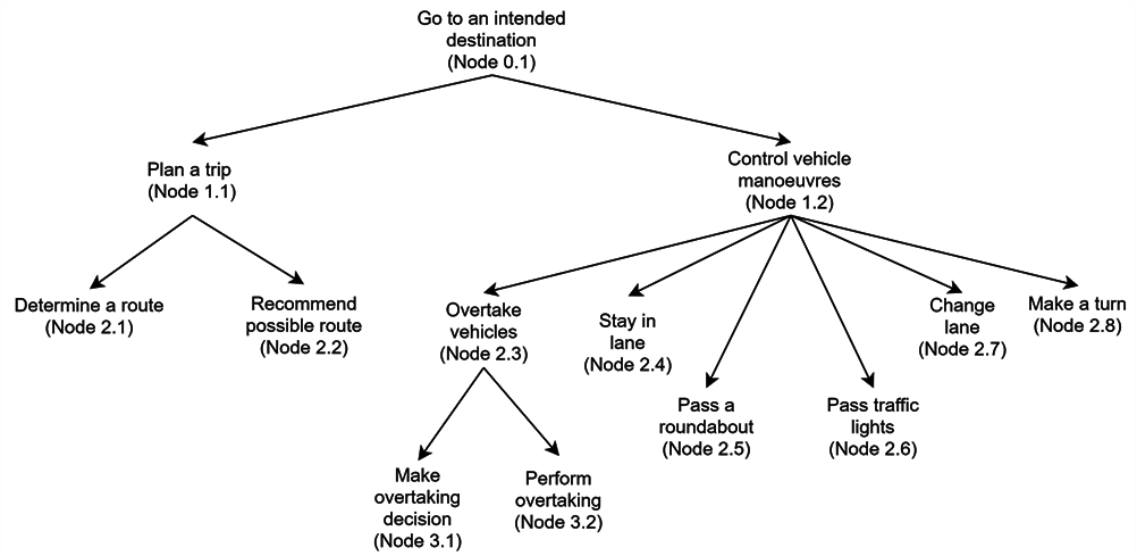


Figure 4-1. Goal hierarchy for collaborative driving

In this goal hierarchy, the team goal *Go to an intended destination* is split into two sub-goals, namely *Plan a trip* and *Control vehicle manoeuvres*. The goal *Plan a trip* has two sub-goals, *Determine a route* and *Recommend possible route*. The goal *Control vehicle manoeuvres* has several sub-goals, namely *Overtake vehicles*, *Stay in lane*, *Pass a roundabout*, *Pass traffic lights*, *Change lane*, and *Make a turn*. Furthermore, the goal *Overtake vehicles* is broken down into two sub-goals, namely *Make overtaking decision* and *Perform overtaking*.

4.3. Goal Provision in Collaborative Driving

4.3.1. Individual Goals for the Autonomy Agent

Based on the goal hierarchy for collaborative driving in Figure 4-1, goal provision is illustrated in Figure 4-2. In this figure, it can be seen that the human driver m_1 owns the team goal ($\varrho^0_{1,[\,][m_1]}$). Recalling Equation 3-2, it should be noted that in the goal provision, when a goal is assigned to a team member, its sub-goals also belong to this team member. Therefore, m_1 is responsible for the team goal along with all the sub-goals. In the

collaborative driving, the superior m_1 assigns some of his or her individual goals to m_2. Then, m_2 is responsible for these assigned goals and their sub-goals while the human driver also takes responsibility for achieving them.

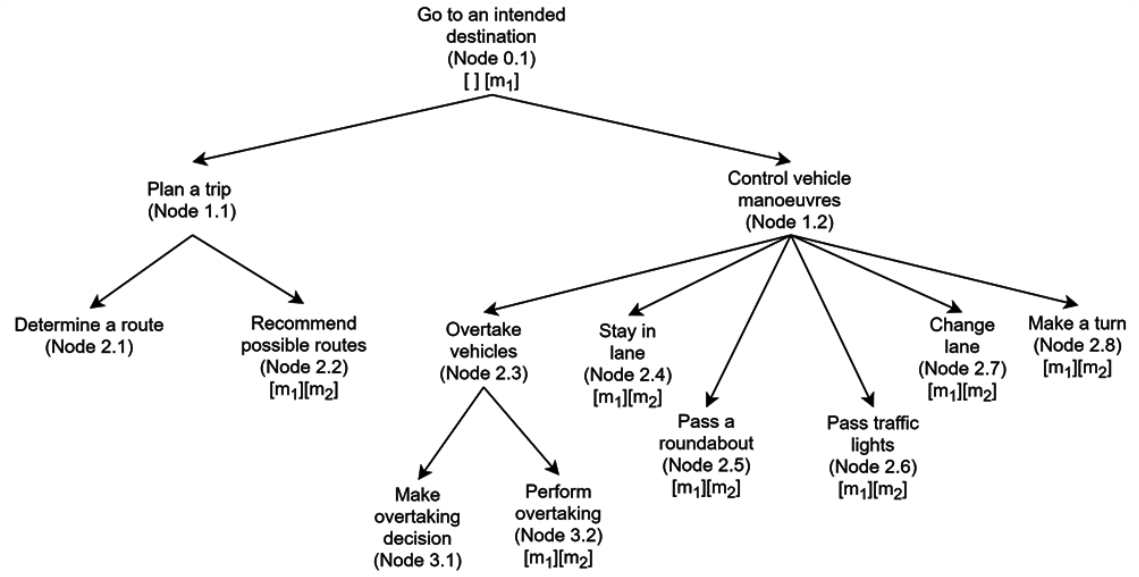


Figure 4-2. Collaborative driving's goal hierarchy and goal provision

For example, under node 2.3, m_2 is assigned to the goal in node 3.2. The reason why node 3.1 is excluded is to ensure that m_2 is only responsible for performing overtaking but not responsible for the goal *Make overtaking decision*. In other words, m_2 requires m_1's consent to perform overtaking. This example also indicates that collaborative driving is different from either manual driving or fully autopilot driving, particularly when m_2 is in charge. In manual driving, all goals under node 2.3 are performed by the human driver. In fully autopilot driving, all goals under node 2.3 are performed by the autopilot agent. Thus, collaborative driving sits between these two kinds of driving situations.

Similar to the overtaking case, m_2 is not responsible for the goal *Determine a route* in collaborative driving. In this regard, m_2 only recommends possible routes to the human driver, but the authority to select the desired route to the intended destination rests with m_1.

4.3.2. Implementation of Autonomy Agent's Goals

Since we focus on the coordination ability of an autonomy agent, from this sub-section onwards in this chapter, we only consider agent m_2 in this collaborative driving HAT.

In the autonomy agent development, a set of functions is developed for this agent to achieve the assigned goals. A function itself contains a set of processes. In order to model the relation among functions, we borrow one technique, use case modelling (i.e., use case diagram) (see Figure 4-3), which was introduced by Jacobson (1993) in the software development area. Due to the hierarchical structure of the goals, it is critical to indicate which function will be executed continuously and simultaneously and which one will be executed based on an event or a trigger by the autonomy agent. With this use case modelling technique, a use case (denoted by ellipse symbol) that represents a function can have three types of relationships: a generalized relationship, a normal relationship and an extended relationship (Chanda et al. 2009).

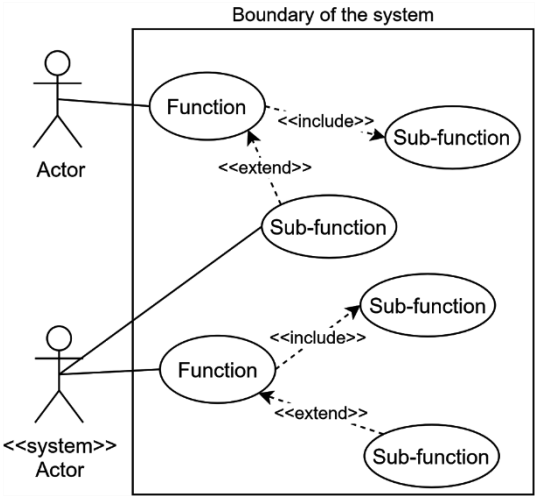


Figure 4-3. A Use case diagram

A generalized relationship is the relation between a function and its sub-functions (denoted by the dashed-arrow line) where a function is the generalization of sub-functions. For example, the functions to monitor a rear situation and a front situation can be considered as sub-functions while the generalization of these two functions is a monitoring situation. Therefore, the relation between a monitoring situation and its two sub-functions can be described as a generalized relationship. A normal relationship (denoted by the <<include>> symbol over the dashed line with the arrowhead pointing to the child functions) is referred to as the relation between a parent function to its child functions, in which the child functions are continuously and simultaneously executed by the parent function (Shen & Liu 2003). When a child function is executed in the case of a special event by a parent function, it can be described as an extended relationship

(denoted by the <<extend>> symbol over the dashed line with the arrowhead pointing to the parent function). Furthermore, an agent performing functions or sub-functions in a use case diagram is referred to as an actor, which can be either a human or non-human agent (autonomy agent). However, in this section, we consider only one actor who is the agent. A solid line is used to connect an actor to functions or sub-functions. When an actor is connected to a function, this means that this actor is also assigned to its child functions.

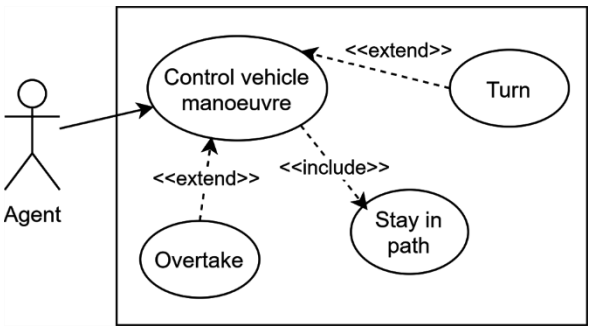


Figure 4-4. An example of use case diagram

A simple example of using the use case modelling technique is illustrated in Figure 4-4. A goal *Control vehicle manoeuvre* is served by a function also called "Control vehicle manoeuvre". This function can be broken down into three sub-functions, namely "Overtake", "Turn", and "Stay in path". Then, an actor (i.e., agent) is connected to these three sub-functions. "Overtake" and Turn" can be considered as functions that have an extended relationship with the function "Control vehicle manoeuvre" as they are only executed when a special event occurs. The "stay in path" sub-function can be considered as having a normal relationship with the function "Control vehicle manoeuvre" as it is continuously executed.

Using the use case modelling technique, a use case diagram is developed to model the functions used to achieve the m_2's goals in the collaborative driving context (see Figure 4-5) as the implementation of m_2's goals. In this diagram, m_2 is the actor which is connected to two main functions called "Calculate the efficient route to destinations" to achieve the goal in node 2.2 and "Control manoeuvres" to achieve the rest of the assigned goals. The function "Control manoeuvres" can be broken down into various functions such as "Brake, "Steer", "Adjust speed", and "Adjust gear" to achieve the assigned goals under the node 1.2 (*Control vehicle manoeuvres*). These four functions are critical as they

regulate the vehicle movements, particularly during "Overtake", "Turn" and "Stay in lane". For example, marking a turn (goal *make a turn)* involves braking (function "Brake"), speed adjustment (function "Adjust speed") and steering (function "Steer").

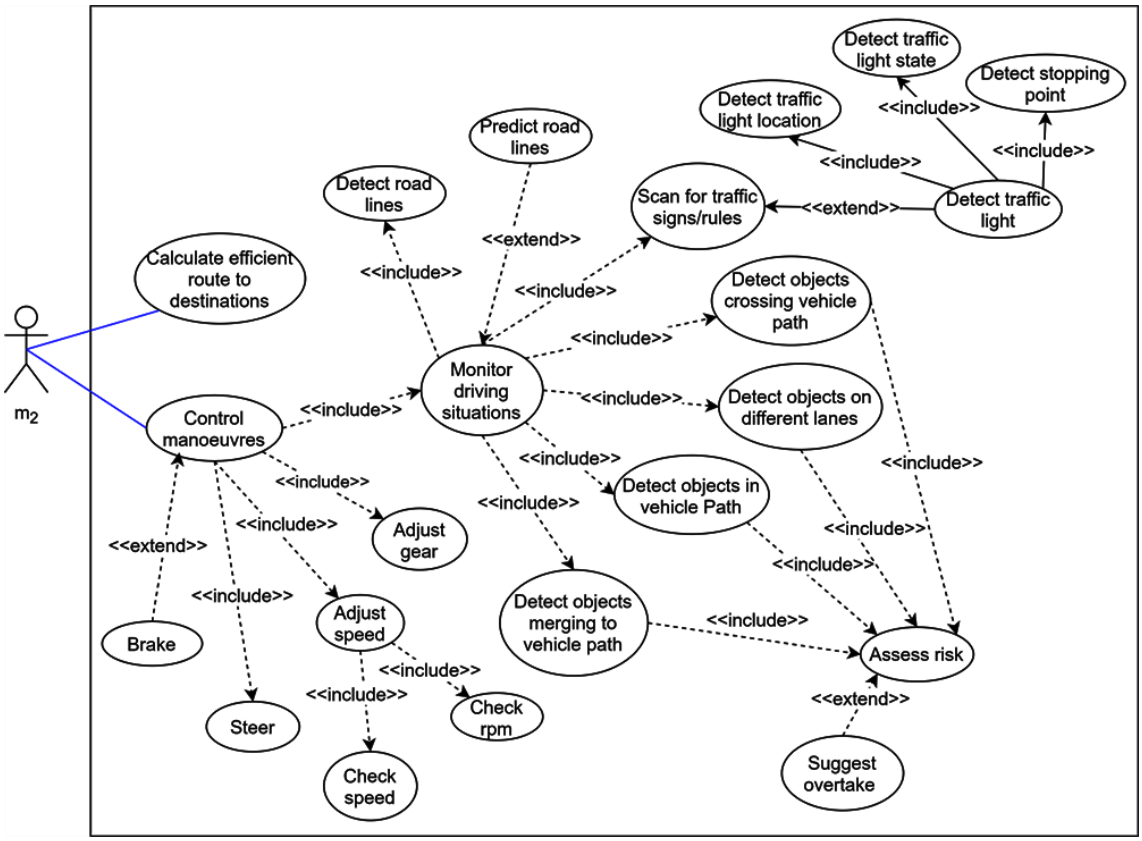


Figure 4-5. A use case model showing the relation among functions to achieve m2's goals

The vehicle's movements can also be considered as the response to the dynamic changes of driving situations observed using the function "Monitor driving situations". This function includes various functions to detect surrounding objects and scan traffic signs/rules including the road lines, pedestrians, and stop sign. For example, to achieve the goal *Stay in lane*, two supporting functions are developed under "Monitor driving situations", namely "Detect road lane" and "Predict road lane". Road lanes should be recognized at all times to ensure that the vehicle is travelling on the correct path. Therefore, the function to detect road lane will be executed at all times. However, it is a common that some road segments do not have road lines such as on a newly developed road segment or on a road segment where the road line paint is not clear anymore. In such a case, the function to predict road lines is summoned by making virtual road lines. During the scanning of the surrounding objects, m_2 also has a function called "Assess risk" which

is useful for this agent to avoid collision with those objects. Moreover, based on the result of "Assess risk", a function called "Suggest overtake" is executed.

4.4. Teaming Formation and Teaming Situation Awareness Specification in Collaborative Driving

In this collaborative driving team, there are two teaming instances. The first instance is when the autopilot mode is on. In this instance m_1 as the supervisor monitors the subordinate and also takes the assigned goals back from m_2 if it fails to achieve its assigned goals. This instance is referred to as teaming with supervision. The other instance is when the autopilot mode is off. In this instance, the subordinate m_2 acts as an assistant to m_1. This instance is referred to as teaming without supervision. Both teaming instances are of type of teaming with vertical coordination.

According to the teaming SA modeling framework in Chapter 3, the teaming SA model for the collaborative driving teaming is the support SA model which is illustrated in Figure 3-10. This section only covers the first teaming instance while the second instance will be covered in Chapter 7.

The SA-related factors in the first instance are elaborated in the rest of this subsection.

4.4.1. Goal

The human driver is the owner of the team goal and is responsible for all the goals in the goal hierarchy. As the superior, m_1 assigns some goals to m_2 and these assigned goals become m_2's individual goals. Thus, m_2's individual goals are a subset of m_1's individual goals, i.e., m_2 has no other goals outside m_1's individual goals.

4.4.2. Environment and Goal-Related Environment

The environment for a collaborative driving context (also called the driving environment) includes the surroundings between the departure location and the destination. The goal-related environment (ϑ_G) includes the surroundings within a certain distance away from the vehicle which can affect driving safety. Therefore, surrounding objects, such as pedestrians, traffic signs, and other road users are considered as part of the driving environment. Other parts of the environment, such as a house with a car parked in the

garage, will be excluded as they do not affect driving safety, thus they are not related to achieving the goal.

Based on the goal provision hierarchy defined in the previous sub-section, we can see that m_2 has no other goals outside m_1's individual goals, that is the teaming in collaborative driving is teaming with vertical coordination. Therefore, the goal-related environment for m_2 ($\vartheta_G^{m_2}$) is a subset of m_1's goal-related environment ($\vartheta_G^{m_1}$), as shown in Figure 3-10 and presented in Table 4-1.

Table 4-1. Goal-related environment relations for collaborative driving

Teaming formation instances	Goal-related environment coverage	Goal-related environment relation among team members
$\varrho^0_{1,[\,][m_1]}$	$\vartheta_G^{m_1}$= all driving environments	$\vartheta_G^{team} = \vartheta_G^{m_1}$
6. Teaming of m_1, m_2:	$\vartheta_G^{m_2}$ = driving environments with task limitation	$\vartheta_G^{m_2} \subset \vartheta_G^{m_1}$

For example, to safely achieve the assigned goals under node 1.2 in Figure 4-2, which are the goals under node 2.4 to node 2.8, and node 3.2, one prerequisite for m_2 is the ability to monitor driving situations and assess the risks. However, such an ability of m_2 depends on what objects have been defined for m_2 to monitor. These objects include pedestrians and other road users, but does not include unspecified objects, such as falling trees. Since a human driver can be intuitively aware of falling trees when they occur, m_2 does not have the cognitive ability to be aware of them. Therefore, the goal-related environment for m_2 ($\vartheta_G^{m_2}$) is a subset of m_1's goal-related environment ($\vartheta_G^{m_1}$).

Another example to emphasize that $\vartheta_G^{m_2}$ is a subset of $\vartheta_G^{m_1}$ can be seen in the goal *Plan a trip* (node 1.1). This node is divided into two sub-goals *Determine a route* and *Recommend possible routes* where the latter is assigned to m_2. To go to an intended destination, obtaining route information is critical. m_1 can use many sources such as his/her own knowledge, map, radio, mobile phone GPS map, or a built-in navigation system provided by m_2. However, m_2 can only use its built-in navigation system to recommend possible routes. Therefore, we can see that $\vartheta_G^{m_2}$ is a subset of $\vartheta_G^{m_1}$.

4.4.3. Situation Awareness Elements and Situation Awareness Requirements

This sub-section presents how to identify SA elements (ς^{SA}) and SA requirements ($\mathcal{R}^{SA}$) for the m_2 agent. From Figure 3-10, it can be seen that the SA elements for both members include two parts: one is the physical objects in the goal-related environment as illustrated by the dots and the other is the teammate, which is indicated by the superior-subordinate relationship. Regarding each goal, the SA elements for m_1 include all the physical objects in the goal-related environment while the SA elements for m_2 include only the specified objects in the goal-related environment.

From Figure 4-2 it can be seen that achieving all the sub-goals of *Control vehicle manoeuvres* is rather complex. To showcase how to develop SA elements and SA requirements for the agent in the collaborative driving teaming, we consider only two specific sub-goals as a case study, which are *Pass traffic light* and *Overtake vehicles*.

In order to describe the SA elements and SA requirements for the agent to achieve the above two goals, let's set up two separate scenarios.

4.4.3.1. Traffic Light Scenario

General speaking, a road section with traffic lights can be divided into two segments with reference to the location point of a traffic light (TL), as illustrated in Figure 4-6.

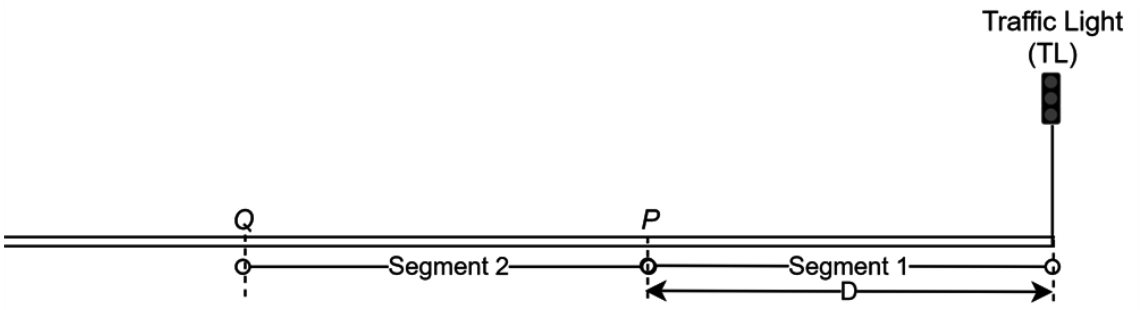


Figure 4-6. Traffic light segmentation

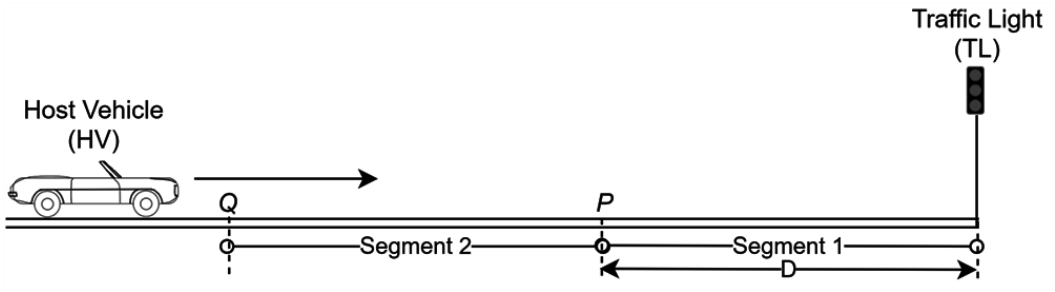
Segment 1 is the part between a pre-defined point (*P*) and traffic light location. This point is defined by a pre-defined distance (*D*) away from the traffic light. Segment 2 is the part between another pre-defined location point (*Q*) and the location point *P*. This point *Q* is

defined as the location of the host vehicle (*HV*, our vehicle, also called ego vehicle) when it begins to enter a traffic light situation.

The distance D can be determined based on the distance over which a driver makes necessary action changes when he or she drives towards a TL, e.g., shifting from a normal speed to starting to stop (if the TL state is red). The process of dividing the road into segments starts when the navigation system recognizes the existence of a TL ahead.

Recognizing a TL location using the information from the vehicle's navigation system is much more reliable than the camera-based recognition. When using camera detection, there is a possibility a TL location will not be recognized due to factors such as fog and sun light. Also, there are cases where the traffic light states cannot be well-recognized by recent autopilot technologies due to their placement which might be hidden from camera detection. These cases show that road infrastructures may not provide a friendly environment for autopilot technology because they existed before the technology. In contrast, the navigation system provides GPS coordinates representing the latitude and longitude of the TL, which tells the autonomy agent about the TL location. This geographic information can be obtained from the GPS database which is specifically developed to store traffic information. Based on these coordinates, the two-location points P and Q can be calculated. In this study, the information from the GPS navigation system will be always available for the agent (m_2) as this is the main requirement to turn on the autopilot mode, i.e., without this system, agent m_2 cannot be assigned to drive the vehicle.

Based on the road segment division, this study considers four TL situations, as depicted in Figure 4-7.



(a)

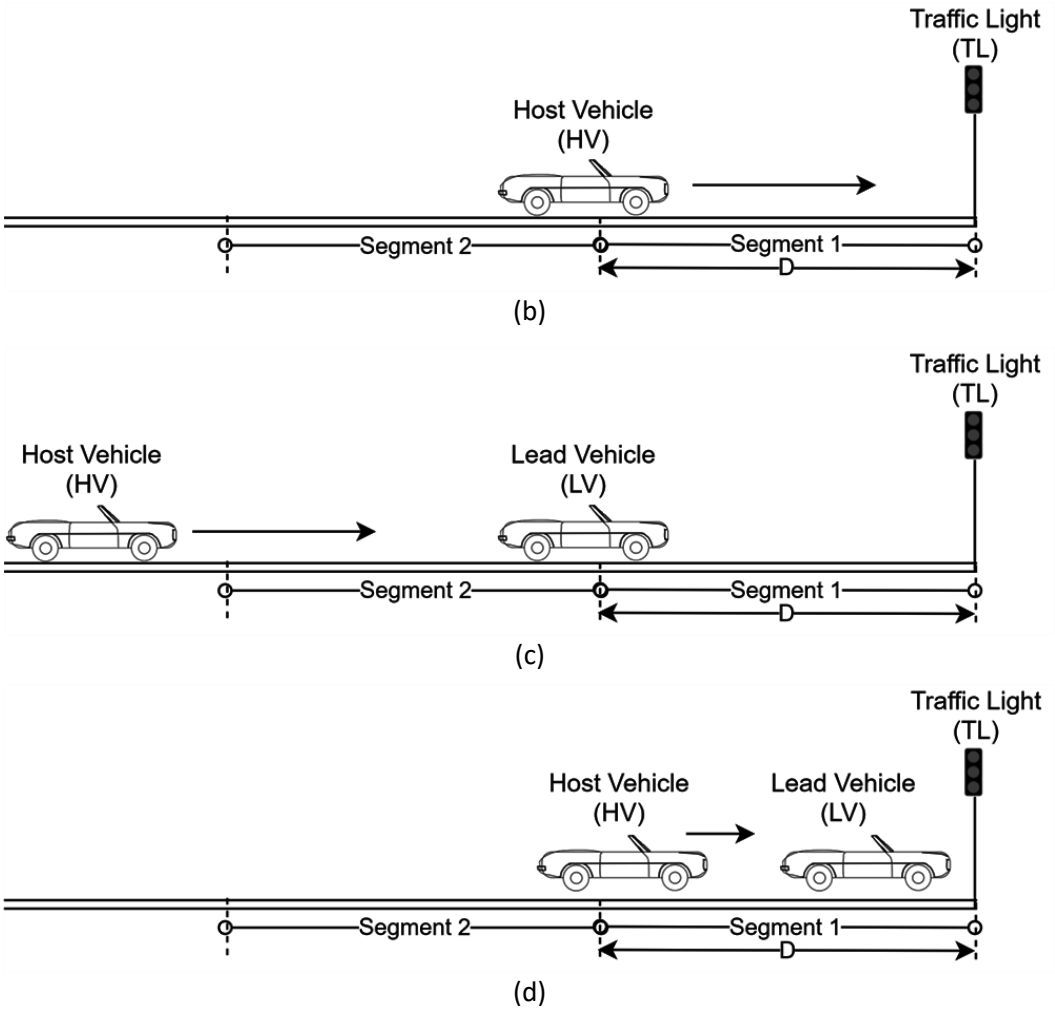


Figure 4-7. Types of traffic light situations

The first type (see Figure 4-7a) is a situation where the *HV* enters segment 2 without a leading vehicle (*LV*) in front (a free ride situation). In this situation, m_2 will maintain the vehicle state (i.e., speed), which is also called *keep going manoeuvres*, no matter what the TL state is. When the host vehicle enters segment 2, a process to recognize the TL state is initiated. This process usually relies on the capability of the vehicle's built-in camera to confidently obtain the optimum image quality to recognize the TL state from a certain distance. According to Mu et al. (2015), to recognize traffic light states, the minimum distance required by the camera sensor is about 120 meters. However, the confident level would be 40 meters. Hence, the TL state recognition might start when the vehicle is approaching point *P*.

Even though the TL state detection can recognize red, yellow, and green lights, when responding to TL states, there are two main behaviours: to stop (when the TL state is red)

and to keep going (for the other TL states). In addition to the three TL states, this study also uses an unknown state. The 'unknown' state is used in three circumstances: 1) when the navigation system acknowledges the existence of a TL in upcoming traffic, but the detection has not been activated yet, 2) when the agent cannot recognize the TL state, 3) as an interchangeable term for yellow states. The yellow state is labeled as an 'unknown' state because it is ambiguous for the agent to know whether to stop or keep going, just like when m_2 cannot recognize the TL state.

Figure 4-7b illustrates the TL situations where the vehicle is in a free ride situation and enters Segment 1. In this segment, the autonomy agent will plan to perform the necessary actions based on the traffic light state. If it is red, this agent will gradually stop the vehicle behind the determined stopping point near the TL location. For other TL states (green, yellow, unknown), the keep going manoeuvres will be executed for any free ride situation. This type of TL situation is critical as this situation could result in a high possibility of a fatal accident. For example, if the actual TL state is red, but it is not recognized by the autonomy agent, based on the rule, the keep going manoeuvre is selected. This can lead to a worst-case situation in which the vehicle will drive through the red light and an unwanted accident might happen.

While Figure 4-7a and Figure 4-7b present TL situations that involve free riding, the other two types of TL situations involve a tailing situation in segment 2 and segment 1 as illustrated in Figure 4-7c and Figure 4-7d, respectively. When a tailing situation is involved, m_2 will select a manoeuvre based on the movement of the *LV* only, which can be slowing down, slightly increasing the speed, or stopping the vehicle. In this regard, the TL state and road segment will not influence the manoeuvre selection process. For example, even though currently the vehicle is in segment 1 and a green light is detected, m_2 will stop the vehicle if the *LV* stops. Stopping based on *LV* movement is also referred to as an *LV*-based stop.

This scenario supports the achievement of the goal in node 2.6 (*Pass traffic lights*) which means that m_2 can react properly based on the TL state.

4.4.3.2. Overtaking Scenario

Depending on the road type (several road lanes in a one way or two-way street), the number of overtaken vehicles, the existence of *LVO* (the vehicle in front in an adjacent lane), the existence of *RLV* (the vehicle in rear position in an adjacent lane), and the existence of *LVH* (the vehicle located in front of the vehicle that will be overtaken in the same lane with the *HV*, which is the ego lane), there are a number of overtaking situations. When an overtaking manoeuvre is executed, the adjacent lane is also referred to as the overtaking lane and the ego lane will be called the departure lane. The simplest overtaking situation is illustrated in Figure 4-8 where there is only one *LV* that will be overtaken but *LVO*, *RLV*, and *LVH* are not present.

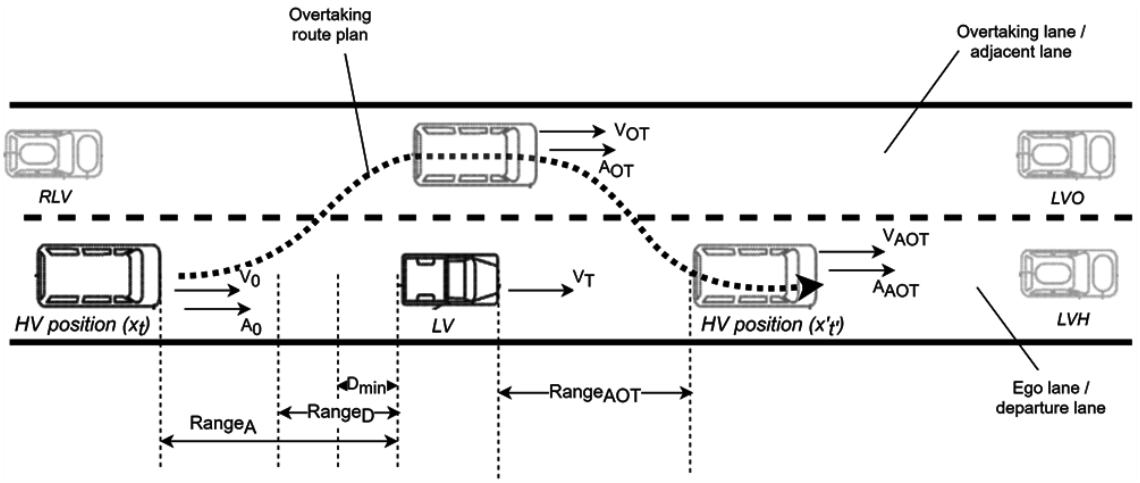


Figure 4-8. Simple overtaking scenario

An overtaking situation can only be possible when the *HV* is in a tailing situation. After approaching *LV*, the *HV* will be at the state where it maintains the desired range to the *LV* ($Range_D$). During this state, the speed of the *HV* may be considerably under the road speed limit and the driver's desired speed. After the state in which the speed is under the limit for a certain time-length, m_2 infers this circumstance as the beginning of an overtaking situation and a notification suggesting the overtaking manoeuvre will be sent to m_1 to obtain approval. In addition to the speed, the agent also considers the rear and front situations in the adjacent lane before sending an overtaking suggestion. In this regard, the *RLV* and *LVO* should not in a position to endanger the *HV* when the overtaking manoeuvre is executed. In this study, the timeframe from the moment when m_2 infers a

safe situation to perform overtaking until the moment when the overtaking suggestion is sent to m_1 will be referred to as a pre-overtaking situation.

After m_1's consent is sent to m_2, the overtaking situation starts and a set of manoeuvres is performed. First, the autonomy agent generates an overtaking route plan and calculates the acceleration needed (A_{OT}) to obtain the predicted overtaking speed (V_{OT}). Once the route plan is ready, the first lane changing manoeuvre is initiated. After the first lane changing, the *HV* will be in the overtaking lane to pass the vehicle to be overtaken. Then, m_2 examines the desired range to go back to the departure lane ($Range_{AOT}$). When a safe range is detected, the autonomy agent performs the second lane changing manoeuvres which puts the *HV* back in its departure lane.

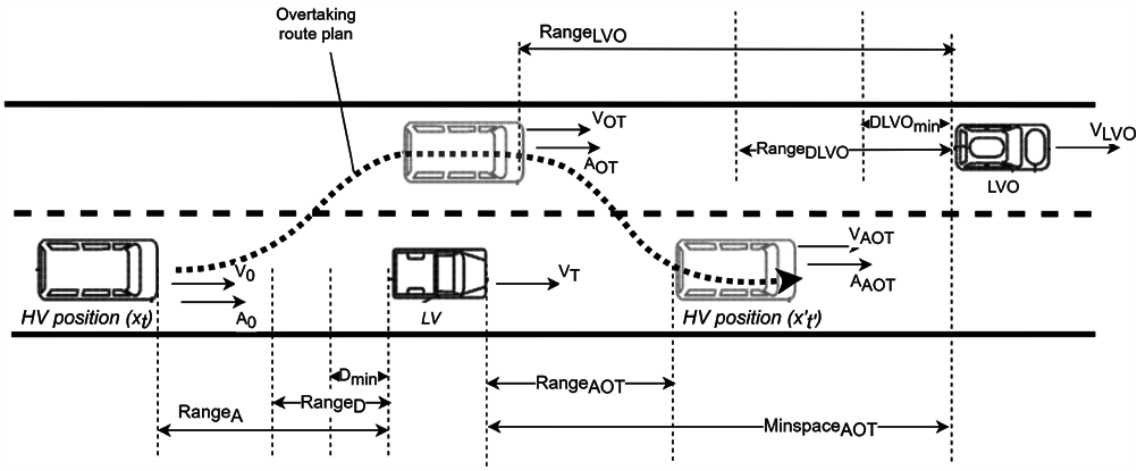


Figure 4-9. Simple overtaking with a car in front in overtaking lane

The existence of the *LVO* during the overtaking manoeuvre increases the situation complexities, particularly in a two-way street. The situation illustrated in Figure 4-9 is considered as an overtaking situation which takes place on a one-way street and involves an *LVO*. Even though the *LVO* is going in the same direction as the LV, it is also necessary to measure the collision risk with the *LVO* as this risk can call the cancelation of overtaking manoeuvre. For example, after the *HV* is in the overtaking lane, the *LVO* suddenly decreases its speed so that it is not possible for m_2 to continue its manoeuvre because there is insufficient space between the *LV* and *LVO* ($Minspace_{AOT}$) to finalize overtaking. There are two possible actions for m_2 when this situation occurs. First, m_2 will go back to its original position in the departure lane which is behind the *LV*. Second,

m_2 will keep the vehicle in the overtaking lane and adjust its speed to obtain a safe margin with the *LVO*.

Another type of overtaking situation is multi-vehicle overtaking as illustrated in Figure 4-10. In this setting, m_2 examines the possibility of overtaking two or more vehicles at once, particularly when the speed of the vehicle in front of the *LV* (indicated by *LV2*) is also under the recommended speed limit and the driver's desired speed. Therefore, $Range_{A2}$ is calculated to acquire the relative speed of *LV*2. In addition to *LV*2's speed, m_2 also checks the space between *LV* and *LV*2 ($Range_{E1}$). When this space is too large, multiple overtaking is not considered and *LV*2 becomes *LVH*. Thus, after passing the *LV*, the *HV* will go behind the *LVH*.

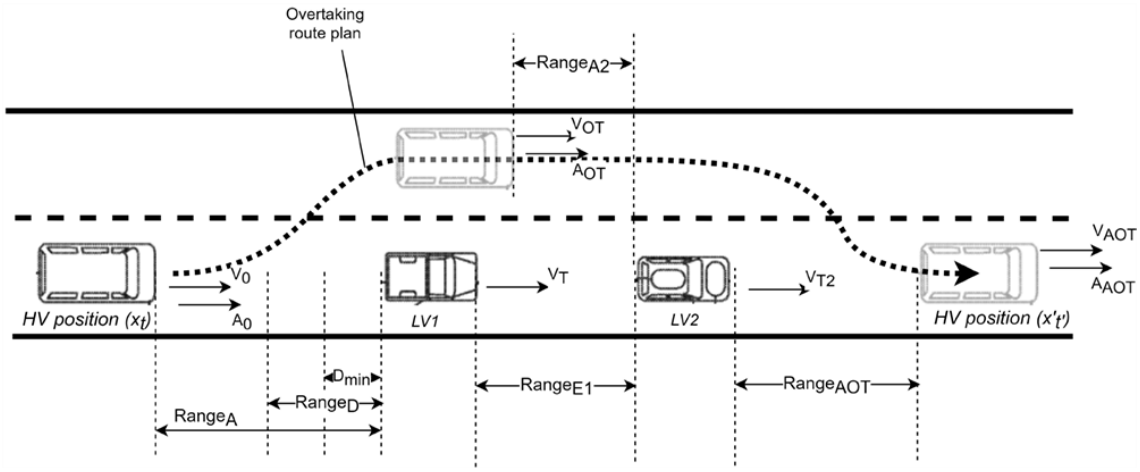


Figure 4-10. Overtaking multiple vehicles

In overtaking situations, the highlighted goals for m_2 referring to the goal hierarchy are the goals under node 3.2 (*perform overtaking*) and node 2.7 (*change lane*).

After discussing these two scenarios, in the rest of this sub-section, we present how to determine the SA elements and the SA requirements for ADAS agent m_2 in both scenarios.

4.4.3.3. Determination of Situation Awareness Elements and Situation Awareness Requirements

We adopt a technique, namely goal-directed task analysis (GDTA), which was firstly introduced by Endsley (2001), to identify the SA elements and SA requirements.

Figure 4-11 presents the GDTA format. Through this GDTA process, the SA elements can be identified for each level of the SA hierarchy. First, GDTA identifies the required information which the goal owner needs to be aware of during the goal achievement process as the required data, even though it is difficult (or there are no supporting tools yet) to collect such data. This required information/data can be used for perception at SA Level 1 and will be the SA elements. At the SA level 2 (SA Level 2), GDTA identifies the information to support the higher-level understanding of the goal owner in performing the tasks to achieve the goals by integrating the status of the SA elements at SA Level 1. This information will be the SA elements at SA Level 2. At SA level 3 (SA Level 3), GDTA identifies the decisions that need to be taken by the goal owner in response to the developed situations. These decisions will be the SA elements at SA Level 3.

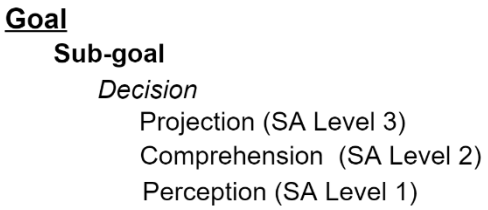


Figure 4-11. Goal-directed task analysis format

Using GDTA, the SA elements and SA requirements to achieve the goal *Pass traffic lights* in the traffic light scenario are presented in Table 4-2. The SA elements to achieve the goal *Pass traffic lights* (see SA-Level 1 column) consists of road lines, GPS coordinates of the TL location, traffic lights, and the LV. According to Definition 8 in Chapter 3, the SA requirements are presented in two levels, SA-Level 1 and SA-Level 2. In SA-Level 1, SA requirements are the status of each SA element. For example, the status of road lines (dashed or solid lines or none) can be obtained through a road line recognition system. When the status is none, i.e., the road lines are not available, recent auto-steering can predict the path of the *HV* from a relative position between the *HV* and *LV*. Additionally, road lines can also be predicted from the relative position of the vehicle path with the support of the GPS when no lead vehicle is in front.

Another example is the status of the traffic lights. On-board sensors or the GPS system can help recognise the state of the traffic lights as red or green or unknown (including yellow). When the red state of a TL is not recognized by m_2, the existence of the *LV* can assist the agent to make a stop manoeuvre.

Furthermore, the combination of the status of each SA element at SA-Level 1 forms the higher-level understanding which can be considered as situation status at SA-Level 2 (this can be seen in the SA-Level 2 column). Finally, SA-Level 3 describes the designed action for m_2 in response to the recognized situation in SA-Level 2.

Table 4-2. Situation awareness elements for m_2 (ADAS) in achieving goal pass traffic lights

Goal	SA-Level 1	SA-Level 2	SA-Level 3
Pass traffic lights	Road lines (Status: dashed/solid lines, existence), GPS coordinates of traffic light location (Statuses: current road segment (segment 1 or segment 2), stopping point), traffic light (Statuses: red/yellow/green/unknown), LV (Statuses: the existence, distance)	Vehicle's position between lines (between the lines or across the line), the relative distance between HV and traffic light location, HV is entering segment 2, HV is entering segment 1, the meaning of traffic light state, aware that currently under tailing or free ride situations, safe margin level with the vehicle ahead, and the position of LV (after/before TL location)	Centering vehicle position in the middle of the lane, the decision to keep going or stop based on the traffic light state, adjusting speed (including stop) based on lead vehicle

Regarding the goal *Overtake vehicle*, its SA elements and their status as SA requirements can be seen in the SA-Level 1 column of Table 4-3. The elements consist of road lines, junction, LV, RLV, LVO, LVH, HV, HV's speed, road speed, driver's set speed, and human consent. The higher-understanding of situations and designed action are presented in the SA-Level 2 and SA-Level 3 columns, respectively.

Table 4-3. Situation awareness elements for m_2 (ADAS) in achieving goal overtake vehicles

Goal	SA-Level 1	SA-Level 2	SA-Level 3
Perform overtake	Road lines (Statuses: dash/solid lines), Junction (Statuses: is junction?), LV, RLV, LVO (Statuses for LV, RLV, LVO: existence, distance), HV (Statuses: speed, road speed, driver set speed), human consent (statuses: yes/no)	Whether currently in tailing situations, current speed is less than road speed and driver's speed and its time factor, rear situation is safe enough for overtaking, predicted overtaking speed is less than road speed, collision risk	Overtake decision (m_2), overtake suggestions, proceed overtaking, cancel overtaking

Goal	SA-Level 1	SA-Level 2	SA-Level 3
		with LVO, overtaken vehicle increases its speed so that overtaking speed is greater than road speed, lane change risk	

4.5. Autonomy Agent's Situation Awareness Development Process

This section focuses on the next SA-related factor which is the SADP. Conceptually speaking, a situation awareness development process (SADP) is from the SA requirements to form situation awareness from the goals. For an autonomy agent, SADP can be viewed as the process to obtain the status of each SA element defined in GDTA (SA-Level 1), the higher-level understanding (SA-Level 2), and to select an action based on the situation understanding (SA-Level 3).

For example, for the goal *Pass traffic lights*, the process to develop SA is presented in the column "Process to obtain status" in Table 4-4. In GDTA for this goal, the identified SA elements are road lines, GPS coordinates of TL location, TL, and LV. Therefore, the SADP at the perception level (SA-Level 1) means the process inside m_2 to obtain the status of each element. Regarding road lines, m_2 uses its camera and an image recognition model to classify the type of road lines. Regarding traffic light location, m_2 can use the data from its navigation system and perform some calculations to infer segment 1, segment 2, and the stopping point. Similar to road lines, the traffic light state and LV are recognized using image recognition models. Moreover, the distance sensor is used to obtain the distance status of LV.

Table 4-4. Situation awareness development process for achieving the goal Pass traffic lights

Goal	Awareness Level	Situation Awareness Requirements	Process to obtain status
Pass traffic lights	Perception (SA-Level 1)	Road lines (Status: dashed/solid lines, existence)	m_2 uses an image recognition model to identify the type of road lines and vehicle's position
		GPS coordinate of traffic light location (Statuses: current road segment (segment 1 or segment 2), stopping point)	m_2 calculates the distance of current vehicle location and traffic light location obtained from navigation system to obtain segment 1, segment 2, and stopping point
		Traffic light (Statuses: red/yellow/green/unknown)	m_2 uses an image recognition model to identify the traffic light state
		LV (Statuses: the existence, distance)	m_2 uses an object recognition model and distance sensor to identify the existence of LV and its distance
	Comprehension (SA-Level 2)	Vehicle's position between lines (between the lines or across the line), the relative distance between HV and traffic light location, HV is entering segment 2, HV is entering segment 1, the meaning of traffic light state, aware that currently under tailing or free ride situations, safe margin level with the vehicle ahead, and the position of LV (after/before TL location)	m_2 uses rules and logics to infer the higher-level understanding of situations. For example, if the vehicle 40 meters away from TL location with LV exists and red light recognized, this means that the vehicle is in segment 1 under tailing and red-light situations. If the vehicle is 80 meters away from TL location and no LV exists, this means that the vehicle enters TL scenario under a free ride situation.
	Projection (SA-Level 3)	Centering vehicle position in the middle of the lane, the decision to keep going or stop based on the traffic light state, adjusting speed (including stop) based on lead vehicle	m_2 uses rules and logics to select actions (i.e., if the vehicle is just entering TL scenario under a free ride situation, then the agent will make keep going manoeuvres)

Regarding SADP at the comprehension and projection levels for the goal *Pass traffic lights*, rules and logics are involved to combine the status of each SA element at the perception level to obtain higher-level understanding (SA-Level 2) and to select

appropriate action (SA-Level 3). For example, if the vehicle is 40 meters away from the TL location with an LV ahead and the red light recognized, m_2 will infer this situation as "the vehicle is in segment 1 under tailing and red-light situations". Thus, if such a situation is detected, keep going manoeuvres will be selected.

For the goal *Perform overtake*, the process to develop SA is presented in Table 4-5. Similar to the traffic light scenario, several recognition models are involved in SADP at SA-Level 1 to infer the status of the LV, RLV, LVO, LVH and road lines. Furthermore, the status of other SA elements such as junction and road speed limit are obtained from the navigation system readings. Regarding m_1's consent, m_2 will recognize when m_1 turns on the turning signal as the representation of his/her consent.

Table 4-5. Situation awareness development process in overtaking scenario

Goal	Awareness Level	Situation Awareness Requirements	Process to obtain statuses
Perform overtake	Perception (SA-Level 1)	Road lines (Status: dashed/solid lines, existence)	m_2 uses an image recognition model to identify the type of road lines and vehicle's position
		Junction (Statuses: is junction?)	m_2 gets the information from navigation system which provide the statuses of road segments (junction, intersection, etc.)
		LV, RLV, LVO, LVH (Statuses for LV, RLV, LVO, LVH: existence, distance)	m_2 uses an object recognition model and distance sensor to identify the existence of LV, RLV, LVO, LVH and their distances
		HV (Statuses: speed, road speed, driver set speed)	m_2 gets the statuses from speed sensors, navigation systems, and driver setting
		human consent (statuses: yes/no)	m_2 reads turning signal sent by m_1 as a representation of his/her consent

Goal	Awareness Level	Situation Awareness Requirements	Process to obtain statuses
	Comprehension (SA-Level 2)	Whether currently in tailing situations, current speed is less than road speed and driver's speed and its time factor, rear situation is safe enough for overtaking, predicted overtaking speed less than road speed, collision risk with LVO, overtaken vehicle increases its speed so that overtaking speed is greater that road speed, lane change risk	m_2 uses rules and logics to infer the higher-level understanding of situations. For example, if LV's speed under driver's set speed and road speed limit and no risks to RLV and LVO, it means that it is safe to overtake LV.
	Projection (SA-Level 3)	Overtake decision, overtake suggestions, proceed overtaking, cancel overtaking	m_2 uses rules and logics to select actions (i.e., if safe to overtake, then the agent will propose overtaking manoeuvre)

Regarding SADP at the comprehension and projection level for the goal *Perform overtake*, rules and logics are also involved to combine the status of each SA element at the perception level to obtain a higher-level understanding (SA-Level 2) and to select an appropriate action (SA-Level 3). For example, if the RLV or LVO does not exist, this means that the overtaking risk is very low. When it is combined with other logics, such as, if the LV's speed is under the driver's set speed and the road speed limit, m_2 will recognize this situation as safe to overtake the LV. When such a situation occurs, m_2 will propose the overtaking manoeuvre.

4.6. Design of SADP

This section presents a design of SADP, referred to as the SADP framework, which can guide the autonomy agent to form its SA based on the SA requirements.

4.6.1. SADP Framework

In this section, we propose a framework to describe the SADP for autonomy agent m_2. The block diagram of this framework is presented in Figure 4-12.

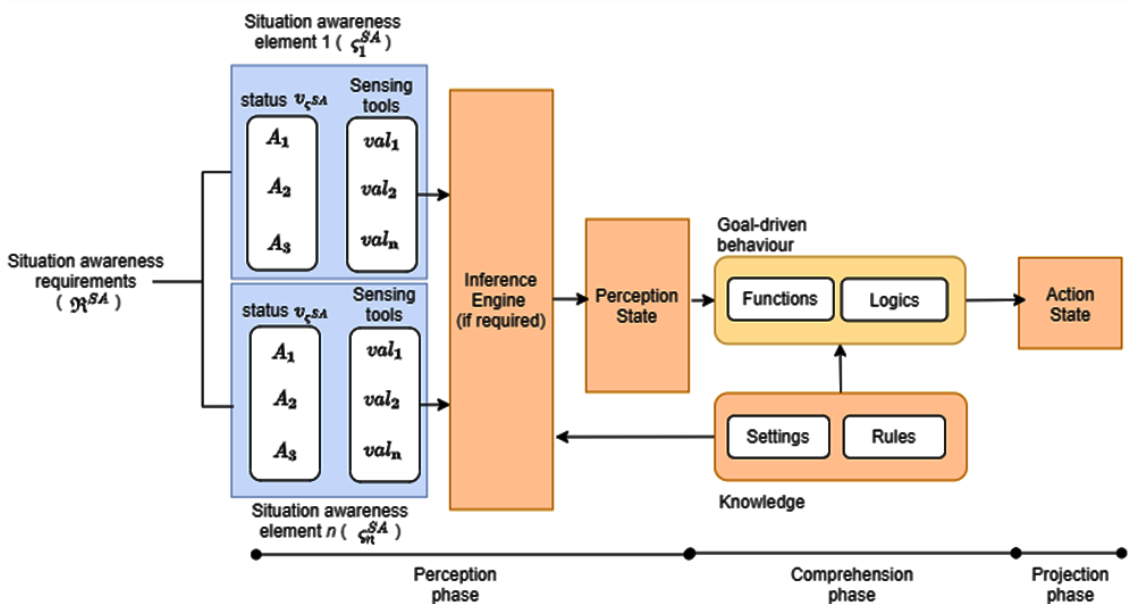


Figure 4-12. Situation awareness development process of an autonomy agent

This framework illustrates how SA requirements are related to the autonomy agent's system and how its SA is formed. There are seven modules in this framework: 1) input module which feeds in the SA requirements; 2) inference module which is an inference engine; 3) perception state module; 4) knowledge module; 5) goal-driven behaviour module, and 6) action module which selects or determines the actions. Adopting the three-level hierarchy of SA, the development of an autonomy agent's SA can also include three phases corresponding to the three levels in the hierarchy: perception, comprehension and projection phases. In the SADP framework, Modules 1 to 3 correspond to the perception phase. Modules 4 and 5 correspond to the comprehension phase, and Module 6 corresponds to the projection phases, respectively.

4.6.1.1. Perception Phase

In this phase, there are three modules, which are the input module (also called the sensing module), inference module and perception state module. The sensing module takes in the status of SA elements (ς^{SA}) that need to be sensed by an autonomy agent, specified in the SA requirements ($\mathcal{R}^{SA}$). The status of a SA element is inferred from its attributes (A). To obtain the values (val) of these attributes, an autonomy agent uses a set of sensory tools. For example, assume that the car in front is a SA element that is a part of the autonomy agent's SA requirements. The attributes to describe this car element are i.e., its distance, its speed, its braking light, and its turning light. The value $val^{\mathcal{R}}$ of each attribute is

collected by sensing tools. In another example, assume that the autonomy agent is an assistant to the human teammate in monitoring the driving situation in a HAT context, then the human teammate becomes the autonomy agent's SA element and his/her fatigue, concentration and activities are examples of the human teammate's status.

The data collected by sensing tools can be of various types, such as numeric or images. Some data are received as they are by the autonomy agent, for example, the distance value from a sensor is an example of data that can directly perceived by the agent. However, some of the data need to be pre-processed to be meaningful to the agent. For example, the camera-captured data needs to be inferred with a recognition technique to translate the image to a specific status of a SA element. Therefore, the inference module is introduced in the SADP framework to provide prediction data for the perception state of an autonomy agent. The inference module consists of prediction functions and the embedded knowledge of the autonomy agent. For example, a classifier in the inference module is a prediction function that can recognize the traffic light state and is used to transform camera-captured data into a specific status of traffic light.

The perception state module is a collection of the status of SA elements which is represented by the value of their attributes, based on current data and prediction data.

The output of the perception phase is the status of SA elements.

4.6.1.2. Comprehension Phase

Similar to the perception phase, in the comprehension phase, the autonomy agent also relies on the embedded knowledge. Knowledge can be stored as settings or in the form of rules. For example, to know the human teammate's desire regarding to keep a convenient distance between his/her vehicle and the car in front when the autopilot is activated, the autonomy agent can read from the user input stored in the settings. Moreover, the knowledge can also contain rules such as the minimum time to collision value required for the autonomy agent to overtake vehicle control from the human driver when his/her vehicle is dangerously approaching a car in front without any sign of braking execution.

The combination of input from the perception state and the autonomy agent's knowledge is then used to drive the autonomy agent's behaviour. As this study designs the autonomy

agent as a goal-based autonomy agent and the execution of goals and rule constraints is reflected in its functions and logics, the agent's behaviour is driven by functions and logics. There are other types of autonomy agents such as a machine-learning-based agent which uses a machine learning model to process the input and performs the selected action. For example, an autonomy agent receives a human voice as its input and processes the input using a voice recognition model to perform the instructed action. In this regard, the selected action is the result of the recognition model, not the autonomy's agent logics. This study selects goal-based agents because these are widely used in the HAT context as they can receive individual goals as part of the team goal.

4.6.1.3. Projection Phase

In the projection phase, the behaviour of the autonomy agent is determined by the process in the goal-driven behaviour module which consists of logics and functions. These logics and functions are intended to select an autonomy agent reaction given the status of the SA elements. This reaction is referred to as the action state.

4.6.2. SADP Framework Implementation

In this section, we present an implementation of the SADP framework through a simulation in CARLA, which is an open-source platform specifically developed for simulating autonomous driving and provides various features such as the autonomous driving sensor suite, traffic scenario simulation, and virtual worlds. The simulation implements the SA requirements (see Table 4-4 and Table 4-5) with sensory tools used by the autonomy agent to sense the surrounding objects. The sensor suites include GPS, LIDARs, GNSS sensor, collision sensor, lane invasion sensor, LIDAR, and cameras.

The GNSS sensor is used to obtain the vehicle's geo position by referring to the geo locations defined within the OpenDRIVE map. This sensor is similar to a GPS, where a position in a certain location is denoted by a coordinate with three positioning values, latitude, longitude, and altitude. However, the GNSS sensor has a UE (Unreal Engine) coordinate system with three axes called *x, y, z* for forward, right, and up, respectively. In CARLA, a coordinate is referred to as a waypoint. When a vehicle goes from a certain location to another, its route consists of waypoints provided by the navigation system.

The GNSS sensor helps the autonomy agent to drive the vehicle by following waypoints to a destination.

A collision sensor is used to obtain information about an object (i.e., fence, pole, pedestrian, other road users) hit by the HV in the driving environment. The lane invasion sensor sends information when the HV is invading another lane by considering the space between the wheels. To support distance measurement, this simulation uses LIDAR. A LIDAR measurement contains the points generated during a certain interval. During this interval, all points reflect the "same static picture" of the driving scene.

Regarding the camera, CARLA has a module called camera manager which manages three kinds of camera sensors, namely the RGB camera, depth camera, and semantic segmentation camera. An RGB camera is a regular camera to capture images from the driving scene. A depth camera supplies the raw data on the driving scene to calculate the distance of each pixel to the camera. Finally, a semantic camera is used to classify the surrounding SA elements.

Furthermore, the CARLA's virtual world becomes the driving environment and its vehicle's sensor suites are used to sense the SA elements or objects from this driving environment. However, in this SADP implementation, we use two selected goals *Pass traffic lights* and *Overtake vehicle* as study cases.

4.6.2.1. SADP for the Goal *Pass Traffic Lights*

Refer to column 3 in Table 4-4 which shows the agent's SA requirements, specifically at the perception level (SA-Level 1) to see the identified SA elements and requirements for the goal *Pass traffic lights*. Then, Table 4-6 lists the perception state of the autonomy agent prototype and the available sensors in CARLA for different SA elements. This represents the perception phase in the SADP framework which produces the perception state.

According to the use case model in Figure 4-5, m_2 is set to continuously perform the "Monitor driving situations" function as the prerequisite of the "Control manoeuvres" function by scanning all the SA elements. Hence, the processes to form the perception

states are used in most sub-functions under "Monitor driving situations" associated with detecting surrounding objects and recognizing traffic signs/rules.

Table 4-6. Forming perception state in the goal Pass traffic lights

Situation awareness elements	Sensory Tools	Inference Engine	Perception state
HV	Lane invasion sensor, vehicle's speed sensor	Carla built-in model	Position between the lane, speed
Traffic light	RGB camera, semantic camera, navigation system	Carla built-in model	Traffic light location, traffic light state
LV	Semantic and depth cameras, LIDAR	Carla built-in model	The existence, distance
Other road users (pedestrians, bikes, etc.)	Semantic and depth cameras, LIDAR	Carla built-in model	The existence, distance, classification, direction

As a goal-following agent, m_2 receives the goal from the goal owner. At the beginning, the autonomy agent gets the destination point. Then the navigation system will supply the waypoints to get there, including every traffic light location that will be passed on the route. The traffic light state recognition will be activated when the vehicle is within a certain distance of the target traffic light location.

B) Comprehension phase

Once the perception state is obtained, the comprehension phase starts. In the use case model, this phase is represented by the function "Assess risk". In achieving the goal *Pass traffic lights*, this function is used to identify the types of TL situations as illustrated in Figure 4-6 and to calculate the risk of collision with the surrounding objects, particularly the LV. Table 4-7 lists some logics which are applied in this function.

Table 4-7. Logics in comprehension phase of the goal Pass traffic light to form higher-level understanding

Logics	Higher-level understanding
If LV or other road users do not exist and HV just reached segment 2 distance	TL scenario 1 (Figure 4.7a)
If LV or other road users do not exist and current HV's location is in segment 1 range and a particular TL state is detected	TL scenario 2 (Figure 4.7b) under red/green/unknown light situations
If LV or other road users exist and its location between HV and TL location and HV just reached segment 2	TL scenario 3 (Figure 4.7c) with safe/unsafe risk to LV/other road users

Logics	Higher-level understanding
If LV or other road users exist and its location between HV and TL location and current HV's location is in segment 1 range and a particular TL state is detected	TL scenario 4 (Figure 4.7d) under red/green/unknown light situations with safe/unsafe risk to LV/other road users

C) Projection phase

After higher-level situation understanding is formed, the next phase is the projection phase which will select the action taken by agent m_2. For the goal *Pass traffic lights*, the corresponding action based on situation understanding in Table 4-7 is presented in Table 4-8.

Table 4-8. Logics in projection phase of the goal Pass traffic light to select action

Logics	Designed Actions
If TL scenario 1	Keep going
If TL scenario 2 and green/yellow/unknown lights situation	Keep going
If TL scenario 2 and red light situation	Stop behind the line at TL location
If TL scenario 3 with safe/unsafe risk to LV/other road users	Adjusting speed based on LV (LV-based manoeuvre)
If TL scenario 4 under red/green/unknown light situations with safe/unsafe risk to LV/other road users	Adjusting speed based on LV (LV-based manoeuvre)

4.6.2.2. SADP for the Goal *Overtake Vehicle*

There are many types of overtaking scenarios but for this SADP framework implementation, we select a simple overtaking scenario as illustrated in Figure 4-8 (LVO, RLV, LVH are not present). It is assumed that the scenario is a one-way road with several road lanes. Every phase in the SADP framework for the goal *Overtake vehicles* is explained in detail in this section.

A) Perception phase

To achieve the goal *Overtake vehicles*, Table 4-9 presents the sensory tools used and models in the inference engine to recognize the status of the SA elements associated with this goal. The perception states for this goal include road line type (dashed/solid), HV's speed, road speed limit, road segment type (i.e., junction), the existence and distance of RLV, RVO, LVO, LVH, and other road users.

Table 4-9. Forming perception state in the goal Overtake vehicles

Situation awareness elements	Sensory Tools	Inference Engine	Perception state
Road lines	Lane invasion sensor, semantic camera	Carla built-in model	Line type: dashed/solid lines
HV	Lane invasion sensor, vehicle's speed sensor	Carla built-in model	Position between the lane, speed
LV	Semantic and depth cameras, LIDAR	Carla built-in model	The existence, distance
RLV	Semantic and depth cameras, LIDAR	Carla built-in model	The existence, distance
LVO	Semantic and depth cameras, LIDAR	Carla built-in model	The existence, distance
LVH	Semantic and depth cameras, LIDAR	Carla built-in model	The existence, distance
Traffic rules	RGB camera, semantic camera	Carla built-in model	Road speed limit
Other road users (pedestrians, bikes, etc.)	Semantic and depth cameras, LIDAR	Carla built-in model	The existence, distance, classification, direction
Road	Navigation system	-	Road type, segment type, waypoints

B) Comprehension phase

Similar to the TL case, the function in the use case model that represents this phase is the function "Assess risk". In the process of this function, some logics are applied as presented in Table 4-10. These logics result in the higher-level understanding of m_2 for the goal *Overtake vehicles* such as m_2 knows that it is safe to overtake, m_1 agrees to perform overtake, the risk in the middle of overtaking is high as LV's speed is increasing.

Table 4-10. Logics in comprehension phase of the goal Overtake vehicles to form higher-level understanding

Logics	Higher-level understanding
If LV exists and LVO, RLV, LVH are not present and road segment is not junction and one of adjacent lanes is same direction and LV's speed under driver's set speed and LV's speed under road speed limit and calculated overtaking speed under road speed limit and HV is behind LV for a *t*-time and HV is still in ego lane	Safe to overtake under simple overtaking scenario (Figure 4.8)
If LV exists and LVO, RLV, LVH are not present and road segment is not junction and one of adjacent lanes is same direction and LV's speed under driver's set speed and LV's speed under road speed limit and calculated overtaking speed under road speed limit and HV is behind LV for a *t*-time and m_1 decides to overtake and HV is still in ego lane	m_1 agrees to perform overtake under simple overtaking scenario
If HV is in overtaking lane and LVO, RLV, LVH are not present and HV position is not after LV and LV's speed is not increasing	Simple overtaking scenario is in progress
If HV is in overtaking lane and LVO, RLV, LVH are not present and HV position is after LV with safe margin and LV's speed is not increasing	Simple overtaking scenario needs to be finalized
If HV is in overtaking lane and LVO, RLV, LVH are not present and HV position is not after LV and LV's speed is increasing	Risk in the middle of simple overtaking scenario is high as LV's speed increasing
If HV is in overtaking lane and LVO, RLV, LVH are not present and HV position is after LV with a safe margin and LV's speed is increasing	Risk in the middle of simple overtaking scenario is increasing as LV's speed increasing

C) Projection phase

In this projection phase, the actions designed for every situation understanding is formed (see Table 4-11). For example, if it is safe to overtake, m_2 sends an overtaking suggestion to m_1. Other actions designed in achieving the goal *Overtake vehicles* include making the first lane change to overtake the LV, keep processing overtaking, make the second lane change to go to the departure lane, cancel overtaking and stay in the overtaking lane as the new ego lane.

Table 4-11. Logics in comprehension phase of the goal Overtake vehicles to form higher-level understanding

Logics	Designed Actions
If safe to overtake under simple overtaking scenario	Send overtaking suggestion to m_1
If m_1 agrees to perform overtake under simple overtaking scenario	Make the first lane change to overtake LV
If simple overtaking scenario is in progress	Keep processing overtaking
If simple overtaking scenario needs to be finalized	Make the second lane change to go back to departure lane
If risk in the middle of simple overtaking scenario is high as LV's speed increasing	Cancel overtaking and stay in overtaking lane

4.7. Autonomy Agent's Situation Awareness

Principally, every phase in the SADP framework is used to form SA at every level (perception, comprehension, and projection levels). Table 4-12 summarizes m_2's SA at every level in association with the goal *Pass traffic light* and *Overtake vehicles*.

Table 4-12. Three-level autonomy agent's situation awareness for the goal Pass traffic light and Overtake vehicles

Goal	Level 1 awareness	Level 2 awareness	Level 3 awareness
Pass traffic light	Aware of traffic light location and state, road lines, the existence of LV	Entering TL situation (segment 2) under free ride situation	Decision to keep going
		Entering TL situation (segment 2) or Segment 1 under tailing situation	Decision to perform LV-based manoeuvre
		Entering segment 1 under free ride situation	Decision to perform action based on TL-state (stop for red light, keep going for other)
		Aware that the vehicle is crossing the lines	Decision to send lane violation warning to m_1, centering the vehicle

Goal	Level 1 awareness	Level 2 awareness	Level 3 awareness
Overtake vehicles	Aware of road segment status (i.e., junction), the existence of surrounding vehicles (LV, RLV, LVO), road speed limit, driver's set speed limit	Inferring safe to overtake situation	Decision to send overtake suggestion to m_1
		Knowing all risks are low during overtaking	Decision to continue to proceed overtaking
		Knowing that overtaken vehicle increases the speed	Cancelling overtaking and stay to overtaking lane (or come back to departure lane)
		Knowing that status to LVO is changing to high risk during overtaking	Cancelling overtaking and stay to overtaking lane (or come back to departure lane)

Regarding the goal *Pass traffic lights*, m_2's SA at the perception phase is obtained with the help of an inference engine which recognizes the status of related SA elements. After all the statuses are received, several calculations are made to form m_2's SA at the comprehension level, such as whether the *LV* is located between *HV* and TL. With this knowledge, m_2 can develop its SA at the projection level by selecting whether to react based on the TL state or based on the *LV*. The simulation of how m_2 reacts based on the identified situation in TL scenarios is developed in a CARLA simulator and the results are illustrated in Figure 4-13.

Figure 4-13. Stopping point without/with other objects ahead in vehicle's path

In the CARLA simulator, let assume the function *get_vehicle()* is used to obtain the existence of the LV, and the function *get_traffic_light()* is used to obtain the TL location and the stop point. A function called *get_tl_id()* is used to acquire the TL state from the inference engine. Furthermore, to get the HV's current location from the location sensor (i.e., GPS), we use the function called *agent.get_location()*. These four functions can be considered as a mechanism of the autonomy agent to obtain its perception states from the related sensors. Now, the values returned by these functions are stored in several variables as follows:

```
vehicle = self.get_vehicle()
tl_loc, tl_stop_point = self.get_traffic_light()
tl_state = self.get_tl_id()
my_loc = self.agent.get_location
```

For the comprehension phase, we use additional functions, for example, a function called *compute_distance()* which is used to calculate the distance from the HV to the TL location and to the LV. A function *get_tl_segment()* to obtain the HV's current segment (segment 1 or segment 2) in TL situations. The basic logic behind get_tl_segment() is "if the computed distance between HV location (stored in variable *my_loc*) and TL location (stored in variable *tl_loc*) is greater than 40 metres then it is Segment 2, otherwise Segment 1". Now, let assume the value in the variable *vehicle* is false (represents no LV), then the HV speed will be set to zero (0) at the *tl_stop_point* when the value of variable *tl_state* is red. As illustrated in Figure 4-13a, the autonomy agent is designed to stop behind the line in the traffic light location as there are no surrounding road users and thus, it reacts based on the identified TL state. The basic logic in a CARLA simulator for the comprehension phase and projection phase for this scenario are as follows.

```
if get_tl_segment() = "Segment 1" and vehicle = false and tl_state = "red":
        agent.stop()
```

For the other case (see Figure 4-13b), the autonomy agent is designed to stop based on the lead vehicle following the safe distance rule set in the autonomy agent setting. To do

this, the autonomy agent examines the surrounding vehicles by executing a function called *collision_and_car_avoid_manager()* as follows:

```
def collision_and_car_avoid_manager(self, location, waypoint):
        vehicle_list = self._world.get_actors().filter("*vehicle")
        def dist(v): return v.get_location().distance(waypoint.transform.location)
        vehicle_list = [v for v in vehicle_list if dist(v) < 45 and v.id !=self.vehicle.id]
        vehicle_state, vehicle, distance = self._bh_is_vehicle_hazard(
                waypoint, location, vehicle_list, max(
                        self.behavior.min_proximity_threshold, self.speed_limit/3),
                        up_angle_th = 30)

        return vehicle_state, vehicle, distance
```

After being summoned, the *collision_and_car_avoid_manager()* function starts listing the surrounding vehicles and saves it in a variable called *vehicle_list*. After this, the distance to each surrounding vehicle is populated by calling another function called *_bh_is_vehicle_hazard()* as follows:

```
def _bh_is_vehicle_hazard(self, ego_wpt, ego_loc, vehicle_list, proximity_th,
                                up_angle_th, low_angle_th = 0, lane_offset=0):
  if ego_wpt.lane_id < 0 and lane_offset != 0:
    lane_offset *= -1

  for target_vehicle in vehicle_list:
    target_vehicle_loc = target_vehicle.get_location()
    target_wpt = self.map.get_waypoint(target_vehicle_loc)
    if target_wpt.road_id != ego_wpt.road_id or
                    target_wpt.lane_id != ego_wpt.lane_id + lane_offset:
      next_wpt = self._local_planner.get_incoming_wpt_and_direction(steps=5)[0]
      if target_wpt.road_id != next_wpt.road_id or target_wpt.lane_id !=
                    next_wpt.lane_id + lane_offset:
        continue
      if is_within_distance(target_vehicle_loc, ego_loc,
                    self._vehicle.get_transform().rotationyaw, proximity_th,
                    up_angle_th, low_angle_th):
        return(True, target_vehicle, compute_distance(target_vehicle, loc, ego_loc))
  return (False, None, -1)
```

Finally, the function *collision_and_car_avoid_manager()* returns three values containing three pieces of information. The first piece of information is stored in a Boolean variable called *vehicle_state*. When *vehicle_state* equals to *true*, this means that a hazard with surrounding vehicles (in this case the vehicle ahead (*LV*) in the same path) is detected. The second piece of information is about the identifier of a vehicle which has a hazard status with our vehicle, and this is stored in a variable called *vehicle*. The third piece of information is the distance with the vehicle stored in the variable *vehicle*, and the distance information is stored in a variable called *distance*. Furthermore, we can write instructions as follows:

```
vehicle_state, vehicle, distance = self.collision_and_car_avoid_manager(
        ego_vehicle_loc, ego_vehicle_wp)

if vehicle_state:
  distance = distance – max(vehicle.bounding_box.extent.y,
        vehicle.bounding_box.extent.x ) – max(self.vehicle.bounding_box.extent.y,
        self.vehicle.bounding_box.extent.x)
  if distance < self.behaviour.braking_distance:
    return self.emergency_stop()
  else:
    control = self.car_following_manager(vehicle, distance)
```

In the above instructions, if *vehicle_state* equals true, the variable *distance* will be examined. If the value of *distance* is less than the braking distance (or safe distance to the vehicle ahead), the function *emergency_stop*() will be executed. In this way, the vehicle controlled by the autonomy agent will stop based on the *LV*.

When the value in the variable *distance* is still greater than the braking distance, the autonomy agent will execute a function called *car_following_manager()*. The main role of this function is to set the target speed based on the *LV*'s speed, time to collision (TTC), and safety time factors. The TTC and safety time factors are determined in the autonomy agent's settings. The function *car_following_manager()* is written as follow:

```
def car_following_manager(self, vehicle, distance, debug=False):
  vehicle_speed = get_speed(vehicle)
  delta_v = max(1, (self.speed – vehicle_speed) / 3.6)
  ttc = distance / delta_v if delta_v != 0 else distance/np.nextafter(0., 1.)

  if self.behavior.safety_time > ttc > 0.0:
    control = self._local_planner.run_step(
      target_speed = min(positive(vehicle_speed – self.behaviour.speed_decrease),
        min(self.behavior.max_speed,
          self.speed_limit – self.behavior.speed_lim_dist)), debug = debug)
  else:
    control = self.local_planner.run_step(
      target_speed = min(self.behavior.max_speed,
        self.speed_limit – self.behavior.speed_lim_dist), debug = debug)
  return control
```

Regarding the goal *Overtake vehicles*, in the perception phase, m_2 perceives the LV and surrounding vehicles in the HV's lane and its adjacent lanes which include their existence, distance, and speed. Additionally, the SA in the perception phase also includes various types of road lines (i.e., dashed/solid) and road segments (i.e., junction). With this information, m_2 can infer whether a certain situation is safe for overtaking including understanding various types of road lines (i.e., dashed/solid) and road segments (i.e., junction) as a part of the factors in overtaking decisions. During overtaking, the SA in the perception phase is updated to measure the dynamic change of risk in overtaking situations. Based on this risk measurement, m_2 selects its action as its SA at the projection level, which can be proceed overtaking or cancel overtaking.

Let's recall the function *collision_and_car_avoid_manager()*. In this function, we add the logic for the autonomy agent to decide whether overtaking will be performed or not. The risk assessment function, *_bh_is_vehicle_hazard(),* can also be used to evaluate the surrounding vehicle status on the adjacent lanes before performing overtaking. The information from the risk assessment will be combined with the information from the road line detection (dashed/solid) and road segment type (around turning section or intersection). When there is another vehicle in the target overtaking lane at an unsafe distance, the road lines are solid, and road segment contains turning section or intersection, overtaking cannot be performed.

```
def collision_and_car_avoid_manager(self, location, waypoint):
        vehicle_list = self._world.get_actors().filter("*vehicle")
        def dist(v): return v.get_location().distance(waypoint.transform.location)
        vehicle_list = [v for v in vehicle_list if dist(v) < 45 and v.id !=self.vehicle.id]
        vehicle_state, vehicle, distance = self._bh_is_vehicle_hazard(
                waypoint, location, vehicle_list, max(
                        self.behavior.min_proximity_threshold, self.speed_limit/3),
                        up_angle_th = 30)

        if vehicle_state and self.direction == RoadOption.LANEFOLLOW and
                not waypoint.is_junction and self.speed > 10
                and self.behavior.overtake_counter == 0
                and self.speed > get_speed(vehicle):
          self._overtake(location, waypoint, vehicle_list)

        return vehicle_state, vehicle, distance
```

In addition to the risk assessment, the critical part in the overtaking situation is under what kind of situation will the autonomy agent decide/suggest overtaking. In this simulation, the suggestion is triggered when the LV's speed is less than the HV's target speed over a certain length of time and the risk assessment shows there are no critical events to avoid overtaking.

When all the requirements to perform overtaking are met, *_overtake()* function is executed. The autonomy agent checks the waypoints in the adjacent lanes to select the best overtaking route which can be overtaking from the left adjacent lane or right adjacent lane. When the best route is found, the new waypoints will be assigned to the autonomy agent and the overtaking manoeuvre is executed. The function *_overtake()* is written as follows:

```
def _overtake(self, location, waypoint, vehicle_list):
  left_turn = waypoint.left_lane_marking.lane_change
  right_turn = waypoint.right_lane_marking.lane_change

  left_wpt = waypoint.get_left_lane()
  right_wpt = waypoint.get_right_lane()

  if (left_turn == carla.LaneChange.Left or left_turn == carla.LaneChange.Both) and
      waypoint.lane_id * left_wpt.lane_id > 0 and
      left_wpt.lane_type == carla.LaneType.Driving:
    new_vehicle_state, _, _ = self._bh_is_vehicle_hazard(waypoint, location,
      vehicle_list, max(self.behavior.min_proximity_threshold, self.speed_limit/3),
      up_angle_th=180, lane_offset = 1)
    if not new_vehicle_state:
      self.behavior.overtake_counter = 200
      self.set_destination(left_wpt.transform.location,
          self.end_waypoint.transform.location, clean=True)

  if (right_turn == carla.LaneChange.Right and
      waypoint.lane_id * right_wpt.lane_id > 0 and
      right_wpt.lane_type == carla.LaneType.Driving:
    new_vehicle_state, _, _ = self._bh_is_vehicle_hazard(waypoint, location,
      vehicle_list, max(self.behavior.min_proximity_threshold, self.speed_limit/3),
      up_angle_th=180, lane_offset = 1)
    if not new_vehicle_state:
      self.behavior.overtake_counter = 200
      self.set_destination(right_wpt.transform.location,
          self.end_waypoint.transform.location, clean=True)
```

An illustration of how m_2 reacts in the CARLA simulator based on its SA in association with the goal *Overtake vehicles* can be seen in Figure 4-14. Figure 4-14a demonstrates the *safe to overtake* situation, and therefore, the overtaking suggestion will be sent. Figure 4-14b demonstrates how m_2 proceeds overtaking after m_1 agrees to overtake. Figure 4-14c illustrates how m_2 cancels the overtaking tasks when the overtaken vehicle increases its speed.

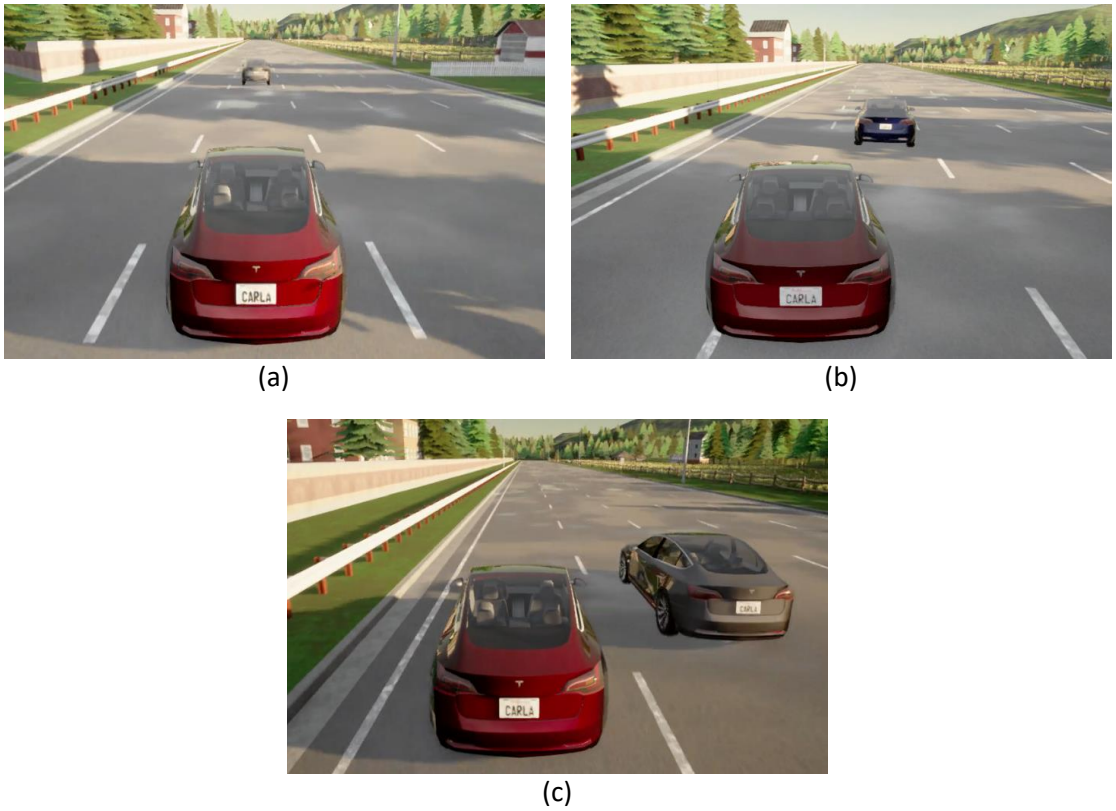


Figure 4-14. Overtaking: (a) Safe to overtake, (b) proceed overtaking, (c) overtaking is cancelled

4.8. Summary

This chapter described the development of the autonomy agent's SA in the collaborative driving context. The HAT in this context is modelled based on the teaming SA modelling framework in Chapter 3 to produce team specifications, goal provision, teaming formation and teaming SA specification for the autonomy agent and the autonomy agent's SADP. A use case modelling technique is introduced for implementing the agent's goals and the GDTA format is adopted for developing the SA elements and SA requirements. Furthermore, this chapter also described the development of a SADP design by introducing a SADP framework to link the SA requirements to the autonomy agent's SA. This study uses the CARLA simulator to simulate how the autonomy agent's reaction based on its SA at every level (perception, comprehension, and projection).

The autonomy agent's SA discussed in this chapter will become the base for developing the agent's coordination ability by reporting to the superior (i.e., the human teammate) based on its self-explanation ability in Chapter 5.

Chapter 5 : Coordination Ability of an Autonomy Agent

5.1. Introduction

Coordination ability is an essential part of the autonomy agent for it to be involved in SSA-based HAT. Taking collaborative driving as an example of HAT, we know that the team is a team with a supervisor as the team members have a superior-subordinate relationship. This team consists of two essential teaming instances called teaming with supervision and teaming without supervision.

In the first instance, the autopilot mode is on and the agent controls the vehicle. The human driver supervises the agent and takes control back if the agent fails to achieve its allocated goals. Supervising an autonomy agent becomes a critical challenge in HAT as it requires a mechanism for the human supervisor to understand the behaviour of an autonomy agent. One way to address this challenge is to equip the agent with an ability to coordinate with the human driver by making itself transparent to the human driver so the superior can monitor it.

In the literature, several researchers highlight transparency which can be described as the extent to which the autonomy agent's behaviour is understandable and predictable to the human teammate for him or her to calibrate trust in the autonomy agent (Endsley 2017; Janssen et al. 2019; Neuhaus et al. 2019; Ososky et al. 2014; Theodorou, Wortham & Bryson 2016). Through transparency, the human teammate can obtain strong cues on what the agent is currently doing so that their trust in and reliance on the autonomy agent can be adjusted. In order to make transparency possible, we need to equip the agent with a coordination ability.

In the second teaming instance, the autopilot mode is off and the driver controls the vehicle. The agent offers assistance to the human driver by helping him or her maintain and/or develop their SA. We consider the agent's ability to assist the human driver as another part of the agent's coordination ability in a SSA-based HAT.

These two parts of the agent's coordination ability in collaborative driving are summarized along with the communication flows in Table 5-1.

Table 5-1. Coordination in collaborative driving

Collaborative driving mode	Teaming Instances	Communication flow
Autopilot on (m_2 is in charge)	Teaming with supervision	m_1 receives report from m_2 regarding types of traffic light situations recognized by m_2 and its consequences m_1 receives report from m_2 regarding overtaking status and cancelation reports
Autopilot off (m_1 is in charge)	Teaming without supervision	m_2 assists m_1 in maintaining/developing his/her situation awareness

Regarding the first part of coordination ability, we will first extend GDTA by adding two components to support coordination to have an extended GDTA (e-GDTA). One component is situation awareness failure (SAF) and the other is SAF handler. We then present a new method for the collaborative driving agent to self-explain its behaviour so that the human driver can comprehend its behaviour. This method can explain the agent's behaviour based on its SA. Based on this method, the agent will have a self-explanation ability as a part of its coordination ability in a SSA-based HAT, which achieves the fourth objective of this study to propose a new method for self-explanation abilities based on artificial situation awareness states (*ASAS*), and the human driver can calibrate their trust in the autonomy agent. This chapter describes the design of this method while the detailed implementation of this method and its validation are presented in Chapter 6. There are two scenarios used for the implementation. The first scenario uses TL situations while the other uses overtaking situations. Transparency on agent's situation awareness in TL situations is critical, as the designed action for this agent is to keep going when it fails to recognize the TL state even though the actual state is red. In this regard, there is a trade-off in agent's design action between safety and other road users' convenience. Such a design increases the potential for a road incident, i.e., violating a red light and colliding with other vehicles. In overtaking situations, transparency is useful to calibrate trust, i.e., the human driver may wonder why the target vehicle stays in the overtaking lane without finishing the overtaking task.

Regarding the second part of coordination ability, in this chapter, we first list a number of types of assistance that the agent can offer to support its superior, and then introduce a solution as to how the agent can detect if the human driver is distracted. In Chapter 7, we give a detailed description of how the agent monitors the human driver by detecting their level of distraction when driving.

The rest of this chapter is structured as follow. In Section 5.2, we illustrate the first teaming instance (teaming with supervision) in a collaborative driving context and analyze the coordination problems using e-GDTA. In section 5.3, we present the design of a method for the autonomy agent to self-explain its behaviours, which is referred to as the *ASAS*-based method. In Section 5.4, we present six kinds of assistance that the autonomy agent can offer to the human driver as its superior in the second teaming instance (teaming without supervision) of collaborative driving. In Section 5.5, we briefly summarize the chapter.

5.2. Teaming with Supervision

Teaming with supervision in collaborative driving exists when the on-board ADAS as the autonomy agent (denoted by m_2) is in charge (autopilot mode on) of the vehicle and the human driver (denoted by m_1) supervises the agent. To clarify coordination in the SA context and how to design the autonomy agent to perform the coordination, we extend the goal-directed task analysis tool by adding two parameters, one called situation awareness failures (SAF) and the other called SAF handlers.

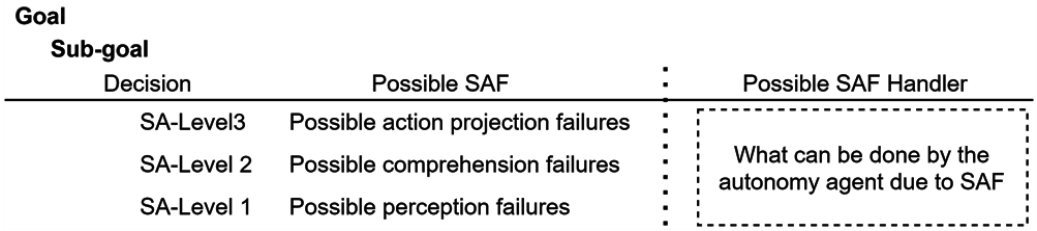


Figure 5-1. Extended goal-directed task analysis

SAF is used to identify the various kinds of coordination problems and SA development problems based on SA identified by GDTA. The SAF handler is used to identify the features that can be provided in the autonomy agent for SAF. We refer to the extended version of GDTA as e-GDTA. The e-GDTA contains three kinds of analyses, namely SA

level 1 (SA-Level 1), level 2 (SA-Level 2) and level 3 (SA-Level 3) analysis. The e-GDTA format is described in Figure 5-1.

and possible features to mitigate the effect of such failures. At SA-Level 2, the analysis includes the possible higher-level situation understanding obtained from the combination of element status at level 1 to form SA. Also, the level 2 analysis includes the possible SAF which makes the autonomy agent fail to develop higher-level situation understanding and the possible handler when such failures occur. At SA-Level 3, the analysis includes the possible failure to take appropriate action and the possible handler for such failures. For each level, the supervision difficulties experienced by the human teammate on an autonomy agent are also identified.

Table 5-2. e-GDTA on goal 'Pass traffic lights' for teaming with supervision

Goal	SA-Level 1	SA-Level 2	SA-Level 3
Pass traffic lights	**Possible SAF:** m_1: difficult to supervise m_2's traffic light recognition performance m_2: visibility/ recognition problems	**Possible SAF**: m_1: does not know how m_2 recognized the current situation m_2: generate wrong higher-level situation understanding due to recognition problems (i.e., red light situation is recognized as unknown situation)	**Possible SAF**: m_1: Difficult to supervise the autonomy agent's response m_2: inappropriate actions due to recognition problems (i.e., the vehicle should stop but it keeps moving)
	SAF Handlers: Present the perception state to human teammate (i.e., traffic light state represented by traffic light icon)	**SAF Handlers**: Present autonomy agent's situation understanding (i.e., red light with tailing or free ride situations)	**SAF Handlers**: Present the taken action to calibrate trust in autonomy agent's decisions with/without request to take over vehicle

By referring to GDTA for the goals *Pass traffic lights* and *Overtake vehicles* presented in Table 4-2 and Table 4-3, respectively, the e-GDTA for these two goals is developed. For the goal *Pass traffic lights* (see Table 5-2) , while agent m_2 develops its SA, such as recognizing the types of traffic light situations and projecting its reaction to the situation, it also needs to take responsibility for coordination with the human driver, such as being able to explain to the human driver about its SA, e.g., TL state recognised and about to execute a *LV*-based stop. Without such coordination ability of m_2 , the human driver m_1 will not receive any information about what TL situation that the agent has recognised,

nor what action the agent will take. When m_1 observes that the vehicle keeps going, they will consider that the agent can recognize the TL state correctly until the vehicle goes through the red light. In fact, based on a design rule in TL situations which says that the agent will utilize a *keep going* action when the TL state is not recognized, the agent can control the vehicle and keep it going even though the TL state is not recognised. Therefore, to help the human driver have a correct understanding of what the agent is doing to avoid unexpected results, the autonomy agent can generate related information as defined in SAF Handler from SA-Level to SA-Level 3. Such information can be considered as a form of coordination from the autonomy agent in response to a possible SAF at every level of SA.

Table 5-3. e-GDTA on goal 'Overtake vehicles' for teaming with supervision

Goal	SA-Level 1	SA-Level 2	SA-Level 3
Overtake vehicles	**Possible SAF:** m_1 and m_2: observation failure (i.e., road condition in overtaking lane is very poor, which cannot be recognized before overtaking has commenced)	**Possible SAF:** m_1: Difficult to supervise the autonomy agent's awareness, particularly when overtaking risks change during the overtaking manoeuvre (i.e., the autonomy agent's response when the overtaken vehicle changes its speed) m_2: Wrong risk assessment in pre-overtaking and overtaking situations, wrong situation understanding	**Possible SAF:** m_1: Difficult to supervise the autonomy agent's response (i.e., whether to cancel overtaking to respond to the change in the overtaken vehicle's speed, where the vehicle will be directed after overtaking cancellation i.e., stay in overtaking lane or back to departure lane) m_2: wrong action due to wrong situation understanding
	SAF Handlers: Sending inattentive driving alert	**SAF Handlers**: Request confirmation to perform overtake so that human teammate can re-assess pre-overtaking situations, presents autonomy agent's situation understanding during overtaking manoeuvres	**SAF Handlers**: Present the taken action (including update overtaking plans such as overtaking cancellation) to calibrate trust in autonomy agent's decisions with/without request to take over vehicle

For the goal *Overtake vehicles* (see Table 5-3), it is also useful for m_1 to receive an explanation regarding the overtaking status and/or overtaking cancelation from m_2. Without m_2's self-explanation ability, m_1 will have no such information, thus, m_1 may wonder what is going on when m_2 stays in the overtaking lane without finishing the overtaking task. For this goal, what should be presented through a self-explanation ability can be seen in SAF handler in Table 5-3. In addition to generating explanations, an inattentive driving alert feature is also required for the goal *Overtaking vehicles*, particularly to mitigate the possible SAF at SA-Level 1.

In summary, in order to make the autonomy agent's SA transparent, the agent must have the ability to self-explain its behaviour to the human driver. In the following part of this section, we will present the design of a method to give the agent a self-explanation ability. This self-explanation ability is intended to explain the autonomy agent's behaviour based on its artificial situation awareness states (*ASAS*) and the proposed method for developing this ability is referred to as the *ASAS*-based method.

5.2.1. ASAS-Based Method for Behaviour Explanation

5.2.1.1. Artificial Situation Awareness States of an Autonomy Agent

Recalling the literature review about self-explanation methods in Chapter 2, existing studies can be grouped into two main methods, component-based methods and process-based methods. A component-based method focuses on the physical components of the autonomy agent. Therefore, these methods infer an autonomy agent's behaviour from its physical parts such as sensing tools and other hardware. An example of behaviour explanation using a component-based method is "Camera is abnormal, recognition function will fail". By focusing on the physical parts of the autonomy agent, these methods can only generate limited explanations.

Process-based methods focus on an autonomy agent's functions and logics. These methods generate behaviour explanation by tracing back the execution path of logics and functions. However, with the huge number of logics and functions inside an autonomy agent, the search space required to generate behaviour explanations will significantly increase. Moreover, as the functions and logics contain operators, rules, and instruction loops, it becomes a difficult task to transform the execution path into a meaningful

explanation as this task should be done manually. However, a process-based method can be considered to be better than a component-based method because the process-based method can reflect the autonomy agent's logics which drive its behaviour.

To mitigate the difficulties resulting from process-based methods by reducing the search space, we propose a new method based on artificial situation awareness states (*ASAS*). *ASAS* can be described as the representation of an autonomy agent's three-level SA that has been developed through SADP. This representation is implemented in the form of a situation model. For example, an autonomy agent has logics as follows:

```
If LV = true and ttc < 10:
  agent.car_following_manager()
```

Such a logical rule indicates that if the autonomy agent detects the existence of LV and the time to collision (TTC) to LV is below 10, then the autonomy agent will execute a function called *car_following_manager()* which will adjust the vehicle's acceleration based on LV's speed to maintain *TTC* at the value of 10. The statement *LV = true* and TTC < *10* represent the autonomy agent's perception (SA-Level 1). The autonomy agent's higher-level understanding of this situation is that currently the vehicle in engaged in a tailing situation (SA-Level 2). Consequently, the SA-Level 3 will be an LV-based manoeuvre. As we only have rule expressions in the autonomy agent's logics, we use *ASAS* to represent the agent's behaviour driven by these rules and logics into artificial situation awareness states. For this purpose, this study uses a situation model, and the situation model for the above logics is illustrated in Figure 5-2.

Figure 5-2. A situation model example

The situation model has three nodes called 'LV', 'Tailing Situation', and 'LV-based manoeuvre'. Each node can have states, and in this example, it is assumed that each node has the state 'true' and 'false'. Thus, if the state of node 'LV' is 'true', this indicates that the autonomy agent recognizes the existence of LV and the state in node 'Tailing Situation' will be set to true to indicate the higher-level understanding of a tailing

situation. When these two nodes have the state 'true', the node 'LV-based manoeuvre' will be set to true which represents SA-Level 3. When the state of node 'LV' is 'false', the states of other nodes will be set to 'false' as well. Based on this illustration, the situation model has a causal relationship from the root node ('LV') through the intermediate node ('Tailing Situation') to the edge node ('LV-based manoeuvre') to represent the autonomy agent's three-level SA.

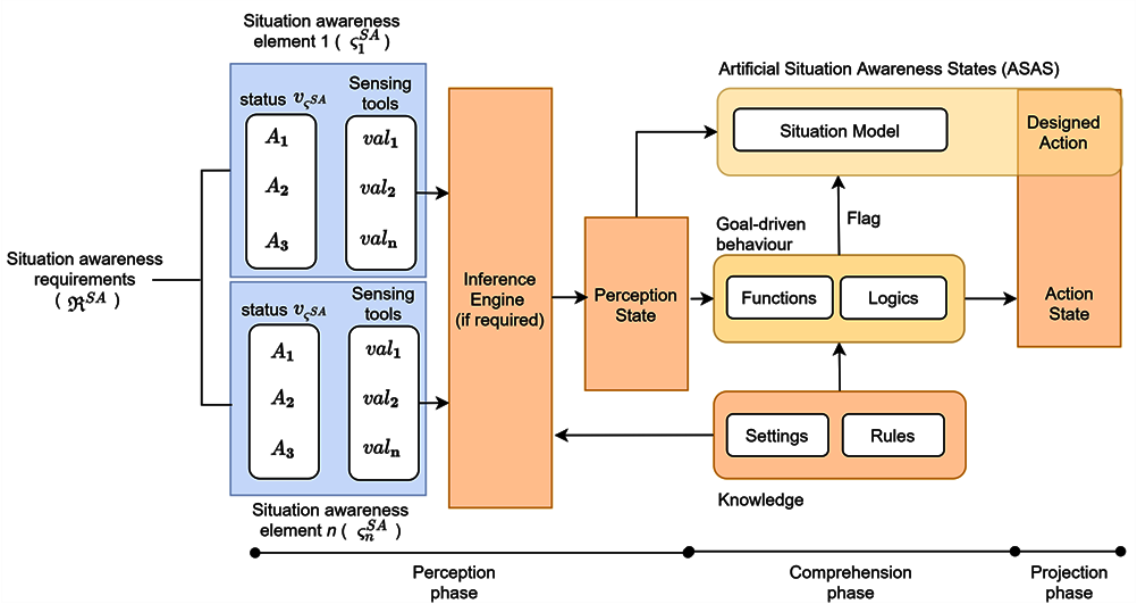


Figure 5-3. SADP framework with ASAS block

Figure 5-3 illustrates how *ASAS* is incorporated into SADP which was previously illustrated in Figure 4-12 (Chapter 4). In Figure 5-3, the *ASAS* block is added. The input of the situation model is obtained from the process level of functions and logics. For example, when the process level produces *LV* = *true*, it will update the state of node "LV" in the situation model to 'true'. Flag variables from the process can also be the input of the situation model. For example, when an autopilot successfully performs a lane changing manoeuvre to overtake the vehicle ahead, a flag (i.e., *isOvertakingPath* = true) can be sent to the SA model to represent that currently the target vehicle is already on an overtaking path.

The most important point is that the *ASAS* does not affect the autonomy agent's behaviour, but it plays a role as situational-based behaviour representation to support coordination. The functions and logics block can be considered as the actual behaviour of the autonomy

agent. As situational-based behaviour representation, this part also includes the designed action in the situation model. The designed action should be the same as the action state as the actual action. Furthermore, as the situation model is considered a causal network, this study selects a Bayesian network to implement the situation model for its capability to model the causal factor.

5.2.1.2. Design of ASAS-Based Self-explanation Method

This section presents the *ASAS*-based self-explanation method in detail. This method uses a new way to make observations on an autonomy agent's artificial situation awareness states (*ASAS*) to address system complexity while supporting transparency in HAT. It consists of three parts, (i) an *ASAS* model which draws a causal relationship between the autonomy agent's perception and higher-level understanding of situations to obtain agent's decision projection and the likelihood of task failure, (ii) an observer engine that captures the state of the *ASAS* model as an episodical memory, and (iii) a self-explanation module which has the capability to generate a backward chaining explanation (BCE) and presents the observation results in a human-friendly manner to support transparency in HAT. Its block diagram is shown in Figure 5-4.

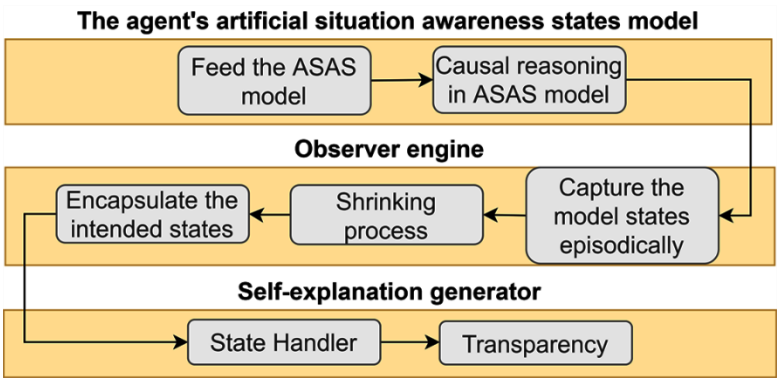


Figure 5-4. Schema of the ASAS-based method for generating self-explanation

A) The *ASAS* model

The *ASAS* model is a core component of this autonomy agent's self-explanation method. It is a Bayesian network-based model which consists of three categories of nodes: root nodes, intermediate nodes, and leaf nodes. The root nodes and intermediate nodes represent the autonomy agent's perception (SA-Level 1) and higher-level situation understanding (SA-Level 2), respectively. The leaf nodes are used to indicate the action

designed for an artificial agent in order to respond to given situations (SA-Level 3). For each node, relevant states should be specified to avoid an excessive number of states as this increases the complexity of determining a Conditional Probability Table (CPT) and degrades the computational performance of the causal reasoning process.

The prior probability for a root node is determined to quantify the degree of belief in uncertain occurrences. Such beliefs can be obtained from subjective expert assessment, normal distribution of experiments, or the principle of indifference which assigns equal probabilities to all possibilities. A normal distribution on the basis of experiments is a reasonable way to assign the prior probability distribution as it provides definite information about the observed variables. In contrast, a subjective assessment only has partial information about the variables, therefore, it can be categorized as a weakly informative prior. When information about variables is not available, prior probabilities can be determined using the principle of indifference to assign the same possibilities to all states in a node. To establish the causal relations among dependent nodes, conditional probabilities are set. There are several possible scenarios in defining conditional probabilities, such as with the help of domain experts, through the respective frequencies of random variables' joint realization within a dataset and using the joint realization of random variables within a dataset for intermediate and leaf nodes.

Once the causal relations in the *ASAS* model are established, the proposed autonomy agent's self-explanation method begins by feeding the model with inputs, also referred to as evidence, which can be obtained from many sources such as sensor readings, information stored in the system settings, or the agent's embedded knowledge. The input data may be of various data types, i.e., continuous variables or Boolean, so the data needs to be discretised using the following two steps: (i) mapping them into a fuzzy state by assigning the degree of membership value and (ii) applying an α-cut to discretize them. For example, the process to discretize continuous variables starts from generating random variables, random vectors, and random sets, which can be defined as fuzzy random variables (Puri & Ralescu 1986). Let (Ω, F, P) be a probability of space where Ω represents the universe of discourse, F and P denote the mapping function and probability respectively. Let $F(\mathbb{R})$ denote the continuous functions u: $\mathbb{R} \rightarrow [0,1]$, a fuzzy random variable (M) is a mapping function $F: \Omega \rightarrow F(\mathbb{R})$ if and only if for a given $w \in \Omega, M\alpha(w)$

is a random interval acting as an α-cut of the fuzzy set $M(w)$ for any α ϵ [0,1], in which $M(w)$ is nonempty and compact for each $0 < \alpha \leq 1$ (Naderpour, Lu & Zhang 2013; Puri & Ralescu 1986). Furthermore, let O = $O_1, O_2, \ldots, O_k, \ldots, O_n$, k = 1, 2, …, n, be the set of variables extracted from the sensors and system settings. Then O_k can be mapped into fuzzy states within corresponding fuzzy set $M_i(w)$. The fuzzy states are defined by crisp states $P_{M_i(w)} = \{p_{M_i(w)1}, p_{M_i(w)2}, \ldots, p_{M_i(w)j}\}$ and characterized by $M\alpha(w)$ which assigns each element O_k a degree of membership to:

$$M_{i_\propto}(w) \coloneqq O_k \colon \mu_{M_i}(O_k) > \propto \tag{5-1}$$

which cuts through the membership function of O_k at height $\propto$. When O_k is a Boolean, the membership function is characterized by two functions: $\mu_{M_n}(O)$ and $v_{M_n}(O)$, which is also referred to as L-Fuzzy set, so that:

$$\mu_{M_n}, v_{M_n} : U \rightarrow [0,1] \; with \; \forall \; O \in U : \; \mu_{M_n}(O) + \; v_{M_n}(O) = 1 \tag{5-2}$$

Now, suppose $p_{M_i(w)1}$ is a member of a fuzzy set at $M_i(w)$ representing a state of conditional probability $\left(P\left(X_i \middle| Pa(X_i)\right)\right)$ in a parent node of the Bayesian network, $P(M_i(w)|Pa(M_i(w))$ can be used to feed the root node of the Bayesian network using the following equation:

$$P(X_i|Pa(X_i)) \rightarrow P(M_i(w)|Pa(M_i(w))) \tag{5-3}$$

After the states of the root nodes are set by equation (5-3), causal reasoning of the Bayesian network is performed by calculating the joint probability distributions. By assuming $M = \{M_1(w), \ldots, M_n(w)\}$ is a set of nodes in the Bayesian network, the joint probability distribution function for this network is defined by:

$$P(M) = P \prod_{i=1}^{n} P(M_i(w)|Pa(M_i(w))) \tag{5-4}$$

where $P(Pa(M_i(w))$ and $P(M_i(w)|Pa(M_i(w))$ are the prior probability distribution and likelihood probability distribution respectively. These two probability distributions are what we have in order to get a posterior probability distribution that can be denoted as

$P(Pa(M_i(w)|M_i(w))$. Furthermore, the marginal probability in all nodes i.e. B $(P(M_i(w) = b))$ can be written as follows:

$$P(M_i(w) = b) = \sum_a P(M_i(w) = b \mid Pa(M_i(w) = a)P(Pa(M_i(w) = a))) \quad (5\text{-}5)$$

To ensure the integrity or validity of the developed $ASAS$ model, the dependencies among the involved nodes should be re-examined to obtain a confidence level that node relationships within the model reflect real-world scenarios. The quantitative values of the nodes' states should be validated by analyzing well-known scenarios to ensure that the given evidence produces the expected results from the model. In this regard, a sensitivity analysis should be performed to generate the derivative values that reflect which states of the parent nodes contribute to the dynamic change of the posterior probability values at the target nodes. To produce the derivative values, entropy reduction, I, becomes the expected function within the mutual information of Q given various states of parent nodes F that is denoted as follows:

$$I = H(Q) - H(Q|F) = \sum_q \sum_f \frac{P(q,f)log_2[P(q,f)]}{P(q)P(f)} \quad (5\text{-}6)$$

where $H(Q)$ and $H(Q|F)$ represent the entropy of Q before and after parent nodes affected by values from evidence. The sensitivity analysis produces a rank-order of evidence nodes allowing a quantitative comparison so that entropy (uncertainty) or variance can be reduced in a target node.

B) <u>The observer engine</u>

The observer engine performs the following four tasks: (1) to capture the state of the *ASAS* model, (ii) to produce the shrunk state of the *ASAS* model, (iii) to minimize the possibility of mutually exclusive and collectively exhaustive problems, and (iv) to provide the intended states for behaviour explanations. These tasks are performed by following three procedures:

1) **Capture the node states in the model episodically**. The intermediate nodes are a critical part of the *ASAS* model as they provide the rationale behind a certain action of an agent in a time frame t. We refer to t as an episode window that needs to be

stored to relate action and its rationale. In this regard, a node mirroring technique is used, not only as a mechanism to make intermediate nodes observable, but also to serve as an episodical memory that stores the states of the model within time frame t. Principally, this technique makes a duplication of every intended node.

2) **Produce shrunk states**. This procedure is intended to remove unnecessary node states. Node state removal is applied when a state can be reflected from another state. For example, a failure state can be inferred from a success state, therefore, the failure state will be eliminated. To generate the shrunk states, this study uses a fuzzy relation operation using two matrices. For example, Figure 5-5 illustrates a process to generate a shrunk state of a node k as one of the nodes in the ASAS model. Node k has three states ('success', 'failure', and 'N/A') which will be represented by a matrix denoted by nd_k. Each probability value of each state in the node becomes the element of matrix nd_k. Therefore, if node k has n number of states, the number of elements in matrix nd_k will equal n, and nd_k will have a size of $A \times B$ where A equals the number of states n and B equals 1. The element with the top position is indexed by zero (0), and for every position below, the index will be incremented by 1.

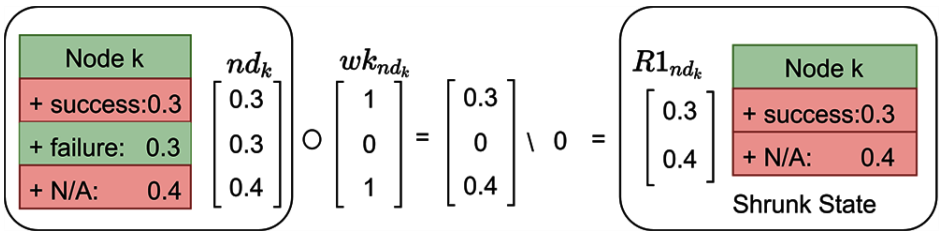


Figure 5-5. Shrinking state process

To perform the shrinking state process, another matrix called wk_{nd_k} is set up and this matrix has 1-to-1 relations with matrix nd_k. The matrix wk_{nd_k} has the same dimensions as nd_k. Elements in wk_{nd_k} only have two possible values, either 1 or 0. Each element in nd_k is in association with an element in wk_{nd_k} with the same index position. When an element in a particular index of wk_{nd_k} is set to zero (0), this means that the associated element in nd_k with the same index will be removed. In node wk_{nd_k}, the element which has a value 0 is at index 1 and this index is associated with the element at index 1 of matrix nd_k representing the 'failure' state of node k. Therefore, in the shrunk state of node k (denoted by $R1_{nd_k}$), the element representing

state 'failure' will not be included. After the shrunk state process, node k only has two states (from previously three states), namely 'success' and 'N/A'. To summarize, the shrunk state generation of a node k is denoted by following equation:

$$R1_{nd_k} = (\{\mu_{nd_k}(A,B) \circ \mu_{wk_{nd_k}}(A,B)\}) \setminus \{0\} \quad (5\text{-}7)$$

3) **Encapsulate the intended states of the model**. Depending on the results of the fuzzy relation operation denoted by equation (5-7), the rest states of nd_k are examined to determine which of the following two options will be taken: One is to select all of the rest states to be further processed and the other is to select a particular state only, i.e., the state with the maximum probability, to be further processed. If option 1 is taken, no further processing is required. If option 2 is taken, it is possible that more than two states with the same probability are identified as the maximum. If this is the case, then the mutually exclusive and collectively exhaustive conditions need to be satisfied. The mutually exclusive condition says that only one state can occur at time t, and the collective exhaustive condition requires at least one state to exist at time t. To be formal, assume that $E = \{e_1, e_1, \ldots, e_n\}$ is a set of states in a node nd_k, then the mutually exclusive condition can be defined as $IF\ e_k = true\ THEN\ \forall(E - e_k) = false$ and the collective exhaustive condition can be formally defined as $\bigvee_{k=1}^{j} e_k = true$. To select the state that has the maximum probability and satisfies both mutually exclusive and collective exhaustive conditions, another matrix, namely wp_{nd_k}, is used. This matrix has the same dimension as nd_k. Similar to the previous step, each element in this matrix also relates to each element in nd_k. The value of each element in wp_{nd_k}can be assigned a weight score of each element in nd_k. In other words, when more than one state has the same maximum value, the state with the highest weight score will be selected. Thus, the mutually exclusive condition is satisfied. The collective exhaustive condition is satisfied by avoiding all-zero values for elements in wp_{nd_k}. This process produces $R2_{nd_k}$ and will be implemented with the following operations:

$$R2_{nd_k} = max(\{\mu_{nd_k}(A,B) \circ \mu_{wp_{nd_k}}(A,B)\}) \quad (5\text{-}8)$$

This procedure is called the fuzzy relation operation layer 2. As illustrated in Figure 5-6, the example node k has relations with two matrices wk_{nd_k} and wp_{nd_k}.

Previously, Equation (5-7) produced the shrunk state of node *k* so that this node now has only two states, 'success' and 'N/A'. However, the shrunk states of node *k* have same value which is 0.4. As only one state with a maximum value is required, fuzzy relation operation layer 2 is proceeded by referring to matrix wp_{nd_k}. In this matrix, the weighted score of the state 'success' in node *k* is 1 while the state 'N/A' is 0. The multiplication of each state in node *k* after being shrunk and its weighted score makes the value of the state 'N/A' zero. Now, by applying *max* function, the state 'success' will be selected as the other state ('N/A') is weighted by 0. In other words, after Equation (5-8), the state 'success' in node *k* will be selected based on the illustration in Figure 5-6.

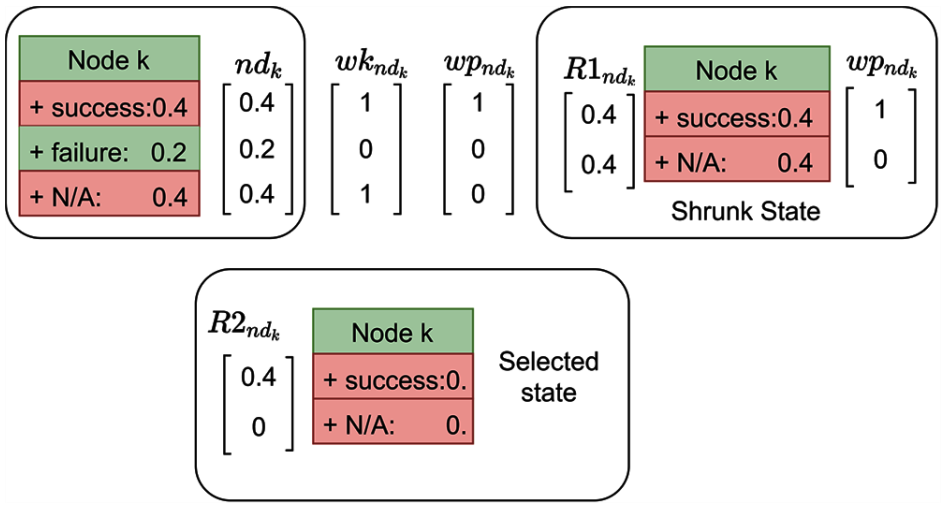


Figure 5-6. Fuzzy Relation Operation Level 2

After the procedures from Equation (5-7) and (5-8) are completed, there are a set of intended nodes (nds) including their states (val_{nds}) in the episodical memory. In the implementation, episodical memory is a variable to store a set of values (the states of intended nodes) generated in a certain period of time *t*. Then, let SN be the content of the episodical memory that comprises a set of key-value pairs of nds and val_{nds}, we can have $SN = \{nds_1 \rightarrow val_{nds_1}, nds_2 \rightarrow val_{nds_2}, \dots, nds_n \rightarrow val_{nds_n}\}$ where *n* is a finite number. In this regard, SN can also be viewed as situation attributes. For example, suppose in the ASAS model there are five nodes representing the existence of a car in front (node *a*), the existence of a car behind (node *b*), the distance to the car in front (node *c*), the distance to the car behind (node *d*), and a dangerous convoy situation (node *e*), respectively. Node *a*, node *b*, and node *e* have

two states: *exist* and *not-exist*, respectively. Node *c* and node *d* also has two states: *safe* and *unsafe*, respectively. In this regard, the states of node *a* to node *d* can be considered as the situation attributes of a dangerous convoy situation. The example of SN content for a dangerous convoy situation is $SN = \{a \rightarrow exist = 1,\ b \rightarrow exist = 1, c \rightarrow unsafe = 1, d \rightarrow unsafe = 1\}$.

C) <u>Self-explanation generator</u>

The critical part of the self-explanation generator is a module called the explanation provider (EXP). EXP can be considered as a situation library which consists of situation attributes for every situation of interest. Also, EXP provides the natural language expressions describing every situation of interest during the explanation generation. Conceptually, EXP is like SN. While SN represents a situation, EXP is a situation collection. EXP is also a set of key-value pairs. Let SC be a situation where $SC \in EXP$ and $SC = \{nds_1 \rightarrow val_{nds_1},\ nds_2 \rightarrow val_{nds_2}, \dots,\ nds_n \rightarrow val_{nds_n}\}$, then $EXP = \{exp_1 \rightarrow SC_1, exp_2 \rightarrow SC_2, \dots, exp_i \rightarrow SC_i\}$ where exp is the situation description key to access the natural language description of this situation, i and n are finite numbers. The basic idea of generating an explanation is implemented by comparing SN to each situation (SC) in EXP.

Furthermore, the explanation generator performs the intersection operation between SN and each element in EXP. From the operation result, a set of the same size will be picked up as the situation candidate and one of them will be selected after a similarity integrity examination is performed. This examination reconfirms that the explanation candidate is truly a match with the one existing in EXP. The Manhattan distance-based similarity function is used to perform the similarity check on SN and EXP. Let's assume these two tuples are two sets of attributes, $q_{i,y}$ and $q_{j,y}$ representing SN and SC, respectively. Using a similarity function $f(q_{i,y}, q_{j,y})$, $r_{i,j}$ is the similarity degree between elements q_i and q_j ($i, j \in \{1, \dots, y\}$). The value range of $r_{i,j}$ is between 0 (q_i and q_j are completely dissimilar) and 1 (q_i and q_j are identical). The similarity degree $r_{i,j}$ and similarity function $f(q_{i,y}, q_{j,y})$, then, are mathematically expressed as follows:

$$r_{i,j} = \frac{1}{m} \sum_{y \in \{1,\dots,m\}} f(q_{i,y}, q_{j,y}) \tag{5-9}$$

$$f(q_{i,y}, q_{j,y}) = 1 - \frac{|q_{i,y} - q_{j,y}|}{max\{q_{i,y} | \, i \in \{1,..,n\}\} - min\{q_{i,y} | \, i \in \{1,..,n\}\}} \quad (5\text{-}10)$$

The result of $f(q_{i,y}, q_{j,y})$ has two possible values as follows:

$$f(q_{i,y}, q_{j,y}) = \begin{cases} 1, & if \; q_{i,y} = q_{j,y} \\ 0, & otherwise \end{cases} \quad (5\text{-}11)$$

where i and j indicate the index of the attributes of elements in SN and SC to be compared, respectively, while y represents the size of the matrices of each attribute set.

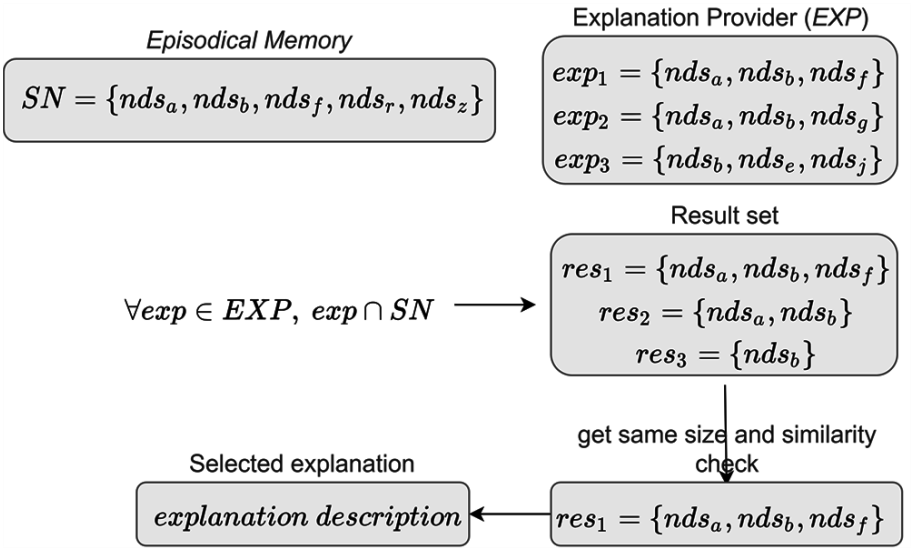


Figure 5-7. Retrieving backward chaining explanation process

When the two sets are equal, the result of similarity function $f(q_{i,y}, q_{j,y})$ will equal 1. Figure 5-7 illustrates the process of obtaining an explanation. Once a leaf node representing the autonomy agent's action is in a *true* state, this explanation generation process examines the content of episodical memory SN. After this, the states of the selected nodes will be compared in the explanation provider to obtain the closest situations which have similar situation attributes. This comparison process uses a similarity function which produces a value from 0 to 1 for each situation in the explanation provider. The selected situation will be the one with the value nearest to 1 (maximum value). At the end of the process, a situation description will be produced as generated explanations to support transparency.

The detailed implementation of this ASAS-based self-explanation method will be presented along with its evaluation in Chapter 6.

5.2.2. Principles of Designing ASAS-model

To optimally obtain the benefit of the *ASAS*-based method, this section provides 8 principles of designing the *ASAS* model as the core part of this method. These principles are as follows:

1. Principle 1: Avoid an excessive number of states within a node

   The number of node states in the *ASAS* model is critical as they affect the complexity of the inference process. Moreover, by maintaining a low number of states, it reduces the size of the conditional probability table in case an intermediate node has many contributory nodes. In this way, this table is easier to manage. From the study cases, splitting the state of traffic lights into different nodes is an example of the effort to maintain a low number of states. Ideally, keeping a node in a two-state mode, such as *true*/*false* or *success/failure*, is suggested with three states as the maximum number in a node.

2. Principle 2: Avoid using the Two-Timeslice technique in modelling *ASAS*

   As the *ASAS* model is based on the Bayesian network, it is possible to implement the Two-Timeslice technique (also called Dynamic Bayesian Network) to model a situation. This technique calculates the value of a node from the immediate prior value (time T-1). For example, to produce an overtaking suggestion, it requires a certain time period of observation to infer a safe state to overtake. Even though the Two-Timeslice technique can be used to model such an observation to infer the overtaking suggestion, it is not recommended that it be applied in the *ASAS* model to avoid the two-running processes computing a safe state for overtaking. It is enough for the *ASAS* model to use the results at the process level as this model focuses on providing behaviour representation.

3. Principle 3: Take advantage of the algorithm's flags or critical process variables to feed the root nodes

It should be highlighted that using the *ASAS* model does not mean using different a computation from the process level to feed the root nodes of the model. In fact, these nodes obtain their values from the process level by accessing its variables. However, it is possible as well that a root node requires a value from the combination of process variables. In this regard, additional variables called flags can be used to supply what the ASAS model needs.

4. Principle 4: Keep one-level-down nodes from the leaf to hold a unique knowledge/situation

   To avoid the tracing back mechanism going to an excessive depth level, try to keep the nodes in the one-level before the leaf nodes hold a unique knowledge/situation summary.

5. Principle 5: Keep the network simple, separate the *ASAS* model for every different task intention

   Keeping the network for the *ASAS* model simple is a part of making behaviour explanation generation efficient. One way to do this is by separating the *ASAS* model for a different task intention. The two cases in this study demonstrate two separate *ASAS* for two different situations, traffic light and overtaking situations. As the networks are different based on task intentions, each network can have a node representing the same perception. For example, the *ASAS* model for the two case studies requires a node, namely node *LV*, to represent the existence of the lead vehicle. Based on this principle, there will be two *LV* nodes, one for the traffic light case model and one for the overtaking case model.

6. Principle 6: Use a binary value for conditional probabilities

   A conditional probability is a means to regulate the model behaviour in inferring the states of the leaf nodes. This probability is managed in a conditional probability table. In the *ASAS* model, it is suggested that this probability is set using only two values either 1 or 0. This because when representing behaviour, strict boundaries must be given. Using only these two values for conditional probability helps the *ASAS* model to avoid inferring a circumstance where two action projections have the probability

to be executed at the same time, even though one action has a lower probability than the other. In this regard, the Bayesian network in the *ASAS* model is not intended to be used as a tool to infer action projection under uncertainties. The *ASAS* model merely takes advantage of the Bayesian network's capability to model the causal factor to reflect non-human agent behaviours from the process level.

7. Principle 7: Understand the process level carefully

   Understanding the process level is essential in designing the *ASAS* model as the actual behaviour of the non-human agent is driven at this level. When the process is extremely complex, identifying the possible agent action projection at the process level can be the best way to start designing the *ASAS* model. After this, the situation causing such actions and the situation attributes should be defined. In defining the situation attributes, it is good to refer to the available variables in the process level.

8. Principle 8: Describe higher-level situation understanding for intermediate nodes using simple words

   The challenge in clarifying a non-human agent's behaviour is to select how the explanations will be conveyed to the human teammate. There are many options such as the use of graphical icons, charts, or words. The point is that whatever options are selected, it should be understandable and absorbable by humans and the time available for humans to figure out the information presented should be considered. However, when graphical icons are used, it will require 'human reasoning' to interpret the relation among icons. Words are chosen when the clarification contains a lot of detail, but the words must be carefully and precisely selected.

5.3. Teaming without Supervision

This section presents coordination examples in the second teaming instance (teaming without supervision) in collaborative driving. In this teaming instance, m_1 is in charge of the vehicle, and m_2 helps maintain/develop m_1's SA as an assistant as part of the coordination in a team. To analyze the coordination problems in the teaming without supervision in a collaborative driving context, e-GDTA is conducted using two example goals, *Pass traffic lights* and *Overtake vehicles* as presented in Table 5-4.

It can be seen in Table 5-4 that one of the SAF for a human driver is due to inattentive driving. Inattentive driving can be caused by many factors, such as drowsiness or distraction. Hence, related features such as driver drowsiness or distraction detection can be attached to an autonomy agent. Furthermore, there is a possibility that m_1 does not realize that they are driving across a road line. When this situation occurs, it can endanger other road users in the adjacent lane. Monitoring whether the vehicle is always within the lane should be done by both m_1 or m_2. Even though m_1 is in charge, m_2 still monitors the vehicle position and reports its results to m_1, in particular, when the vehicle crosses the road lines. Therefore, when m_1 does not realize that they are driving across the road line, the report from m_2 will be very useful for them to regain their focus. This report can be sent in the form of a line invasion alert feature. Hence, this feature can help to maintain m_1's SA to stay in the lane and form the coordination activities.

Table 5-4. e-GDTA on goals 'Pass traffic lights' and 'Overtake vehicles' for teaming without supervision

Goal	SA-Level 1	SA-Level 2	SA-Level 3
Pass traffic lights	**Possible SAF:** m_1: fails to recognize the status of surrounding objects caused by inattentive driving	**Possible SAF**: m_1: does not realize that their vehicle has an unsafe margin to LV, does not realize that the vehicle is out of the lane	**Possible SAF**: m_1: fails to react based on the actual traffic light state due to the SA-Level 1 problem
	SAF Handlers: distracted driver recognition, drowsiness alert	**SAF Handlers**: Road line invasion alert, collision avoidance and alert	**SAF Handlers**: See SAF handlers at SA-Level 1
Overtake vehicles	**Possible SAF:** m_1: fails to recognize critical objects' statuses due to limited vision, cannot find road speed limit sign	**Possible SAF**: Violating road speed limit when overtaking vehicles, fails to be aware of a sudden change in overtaking situations	**Possible SAF**: fails to respond immediately when the vehicle overtaking (LVO) suddenly stops so the collision risk is high (see overtaking scenario illustrated in Figure 4.9)
	SAF Handlers: Traffic information assistance	**SAF Handlers**: Road speed limit alerts	**SAF Handlers**: collision avoidance and alert

Keeping a safe distance to the LV is also a success factor for the goal '*Pass traffic light'* and '*Overtake vehicles*'. When m_1 is in charge, m_2 also still monitors the LV. Similar to the unintentionally crossing the road line case above, inattentive driving can also be present and m_1 may not realize that the safe margin to LV has been violated. As m_2 also monitors this situation, m_2 can send a report to m_1 regarding such an unsafe margin. This report can be implemented using collision avoidance and the alert feature. With recent technology improvements, this feature is also capable of avoiding a collision by having the authority to execute the emergency brake if necessary. In addition to keeping a safe distance, maintaining vehicle speed under the road speed limit is also important. In this regard, m_2 can report to m_1 when m_1 tends to violate the speed limit rule.

The features that can be deployed to an autonomy agent based on e-GDTA in Table 5-4 are summarized in Table 5-5.

Table 5-5. Driving assistance feature

Assistance features	Description
Road line invasion alert	To help m_1 become aware when they unintentionally cross the road lines.
Collision avoidance and alert	To notify m_1 regarding collision risk with surrounding objects which can execute the emergency brake if necessary.
Speed limit alert	To alert m_1 when they are approaching the road speed limit.
Driver distraction detection	To alert m_1 to put their focus back on the road.
Driver drowsiness detection	To alert m_1 of their drowsiness.
Traffic information assistance	To help in providing traffic rules/information around vehicle location

In this study, we showcase the agent's coordination ability in teaming without supervision by developing a classifier to recognize types of driver distraction. For this purpose, sample images should be collected using the on-board camera. As the collected images from the target vehicle are often limited in number, it is suggested to take advantage of the existing classifier for the distracted driver from another study which has a larger size of sample images.

Since there are two possible steer sides for a vehicle, we may run into two situations in which the steer sides in the target vehicle and the vehicle from which the sample images were taken for training the existing classifier are either the same or different. In the first situation, the existing classifier can be directly used to classify the driver distraction types.

However, in the second situation, the performance of the existing classifier tends to be rather poor. Therefore, it is highly desirable to develop a new classifier.

In this study, we propose a new method to resolve the issue in this second situation by adopting the transfer learning approach in the deep learning environment to improve the accuracy of the classification of the driver's distraction types. As the collected images and existing sample images from other studies have different steer sides, the transfer learning process becomes more challenging. The detailed implementation of our proposed transfer learning approach for distracted driver recognition is presented in Chapter 7.

5.4. Summary

This chapter presents the autonomy agent's coordination ability in two teaming instances of the collaborative driving context as an example of SSA-based HAT. To analyze the coordination problems, this chapter introduces an extended version of the goal-driven task analysis method (e-GDTA). For teaming with supervision, the autonomy agent's self-explanation ability is required for coordination, particularly to make it easier for the human teammate to comprehend and supervise the autonomy agent. To develop a self-explanation ability, this chapter presents the design of an ASAS -based method to generate the behaviour explanation of an autonomy agent including a set of principles for designing the ASAS model. The implementation of this method is presented in Chapter 6.

Another aspect of the coordination ability is for teaming without supervision. In this regard, an autonomy agent should be equipped with the ability to assist its human superior to maintain/develop their SA. Six kinds of features in an autonomy agent are explained as examples of the coordination ability for teaming without supervision with a focus on the feature of monitoring the driver's distraction types. The detail of the implementation of this selected coordination feature is presented in Chapter 7.

.

Chapter 6 : Autonomy Agent's Behaviour Explanation using the ASAS-Based Method

6.1. Introduction

In Chapter 5, we identified two teaming instances in collaborative driving which is an example of SSA-based HAT as teaming with supervision and teaming without supervision. To enable coordination in the teaming with supervision, Chapter 5 presents the design of an *ASAS*-based method for the agent to self-explain its behaviours based on its *ASAS*.

This chapter presents the implementation of this *ASAS* -based method in two study cases to generate the autonomy agent's behaviour explanation. The first study case generates self-explanation when the agent achieves the goal *Pass traffic lights,* and the second study case generates self-explanation when the agent achieves the goal *Overtake vehicles*. This *ASAS* -based method is compared with the process-based method as its evaluation. The comparison results show that the *ASAS* -based method significantly reduces the search space when generating explanations.

The rest of this chapter is structured as follows. Section 6.2 presents how to use the *ASAS* -based method to generate the agent's behaviour explanation in case study 1. Section 6.3 presents how to use this new method to generate the agent's behaviour explanation in case study 2. Section 6.4 presents the evaluation of the *ASAS* -based self-explanation method through a comparison with the process-based method. Finally, Section 6.5 summarises the chapter.

6.2. Study Case 1: Behaviour Explanations in Achieving the goal *Pass Traffic Lights*

6.2.1. Introduction to Study Case 1

This first case study is to explain the autonomy agent's behaviour based on its three-level situation awareness as defined in Chapter 4 and presented in Table 4-12. Based on this

table, level 1 awareness for the goal *Pass traffic lights* includes the TL location and the existence of LV. Furthermore, level 2 awareness includes the position of HV (segment 1 or segment 2), free ride situations, tailing situations. Finally, level 3 awareness includes the autonomy agent's designed action such as keep going, LV-based manoeuvre, and TL-based manoeuvre. In the following sub-sections, the ASAS-based method is implemented based on situation awareness for the goal *Pass traffic lights.*

There are three scenarios of TL situations that are used in generating explanations. These scenarios are as follows:

1) **Scenario 1:** In the first scenario, it is assumed that the autonomy agent fails to recognize the TL state. No *LV* is detected. Without any information revealed about the TL state or action explanations, the human teammate will consider that the agent can recognize the TL state recognition correctly until the vehicle runs through a red light. It is believed that generating an appropriate explanation can help the human teammate to take the necessary action when a situation like this scenario occurs.

2) **Scenario 2:** In the second scenario, the autonomy agent fails to recognize the TL state, but the *LV* is detected. The purpose of providing behaviour in this scenario is to provide information for the human teammate so that trust can be calibrated. In this regard, there is still a possibility of stopping at a red light even though the TL state is not recognized by determining the manoeuvre based on *LV* movement.

3) **Scenario 3:** In the third scenario, the explanation is intended to differentiate which situation leads to a certain autonomy agent's behaviour as different situations may cause the same behaviour. For this purpose, this scenario assumes that this autonomy agent can recognize the TL state (green) and no *LV* is detected. In this scenario, the autonomy agent's behaviours will be the same as the behaviours under scenario 1.

For all these scenarios, it is assumed that the vehicle is entering/in segment 1.

6.2.2. Implementing the ASAS-based Method

6.2.2.1. Modelling the ASAS of the Autonomy Agent for the goal *Pass Traffic Lights*

The *ASAS*-based method to generate the autonomy agent's behaviour explanations can be initiated by identifying its perception, higher-situation understanding, and action projection, which are available from the agent's situation awareness obtained by the methods presented in Chapter 4. To model *ASAS,* this study uses a Bayesian network. In the ASAS model, the root nodes of the Bayesian network are used to represent the autonomy agent's perception (level 1 awareness). Intermediate nodes and leaf nodes are used for the autonomy agent's higher-level understanding (level 2 awareness) and designed action (level 3 awareness), respectively.

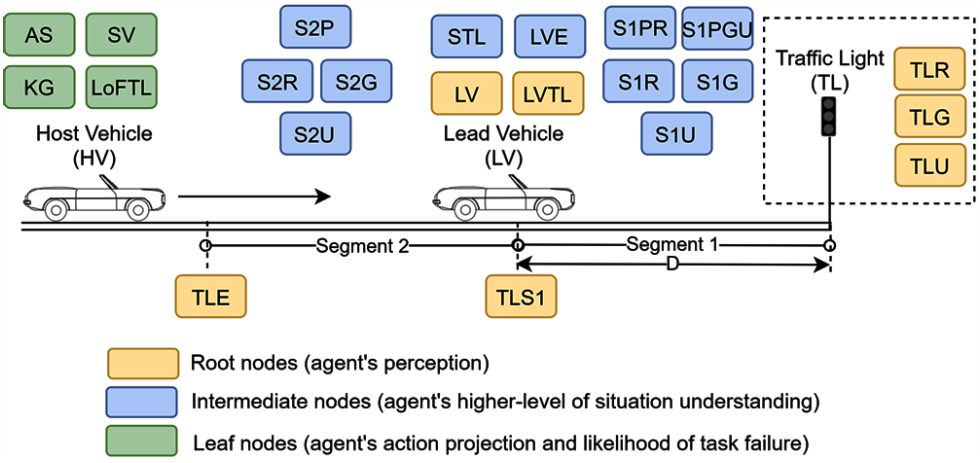


Figure 6-1. Scenario of passing Traffic light (TL) and NN nodes in the ASAS model

The defined nodes for the scenario of passing traffic light are illustrated in Figure 6-1, which include nodes for the physical objects, such as *HV*, *LV*, and TL and nodes for the abstract objects such as segment 1, segment 2 and distance D. Distance D can be determined based on the distance over which a driver makes the necessary action changes when they drive towards a TL, e.g., shifting from normal speed to starting to stop (if the TL state is red).

Once the nodes representing the autonomy agent's *ASAS* are identified, the *ASAS* model is developed, as shown in Figure 6-2, which consists of seven root nodes (shown in yellow in Figure 6-1), 11 intermediate nodes (shown in blue in Figure 6-1), and 4 leaf nodes (shown in green in Figure 6-1). These root nodes' descriptions are given in Table 6-1.

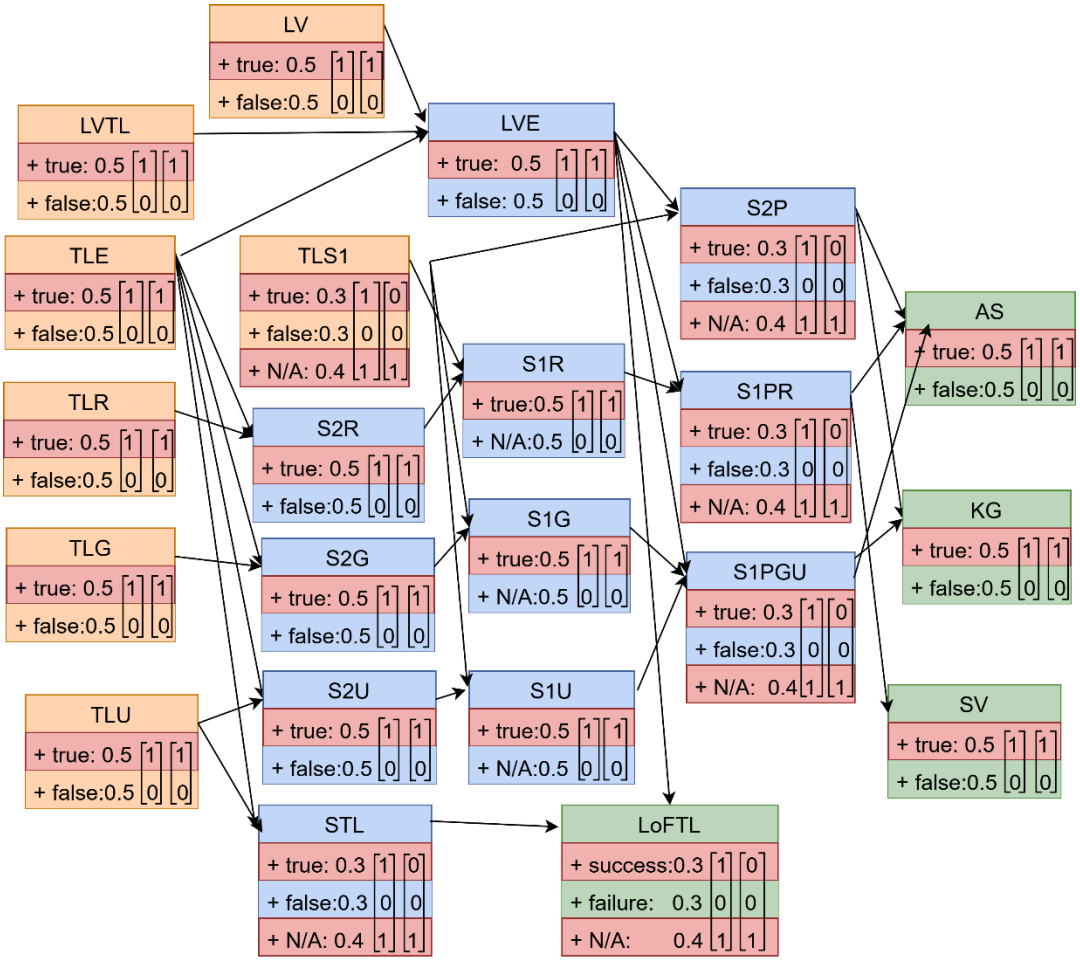


Figure 6-2. A Bayesian network model representing the development of the agent's artificial situation awareness states (ASAS) in TL situations

Table 6-1. The description of root nodes in TL situations

Node	State	Description	Node	State	Description
TLE	True	Entering TL situations	TLR	True	TL state is red
	False	TL situations does not exist		False	This situation does not exist
TLS1	True	Entering segment 1	TLG	True	TL state is green
	False	In segment 2		False	This situation does not exist
	N/A	TL situations does not exist	TLU	True	TL state is unknown
LV	True	Lead vehicle exists (tailing situations)		False	This situation does not exist
	False	Lead vehicle does not exist (free ride situations)	LVTL	True	Lead vehicle before TL
				False	Lead vehicle before TL does not exist

Furthermore, the intermediate nodes are divided into three groups representing higher-level situation understanding for segment 2, segment 1, and LV with TL state as their references. There are four nodes to represent the agent's action projections: node AS

(adjusting speed and distance based on *LV*, stop if necessary), *SV* (stopping the vehicle in red light situations) and *KG* (keep going). Figure 6-2 presents the posterior probability for each state along with the two matrices for each node, in which the left and right matrix is used to support the fuzzy relation operations at the first and second layer respectively.

In addition to determining the posterior probability for each state in a node, the most critical part in the *ASAS* model is to determine the agent's behaviour from causal relationships which are stated in the conditional probability table (CPT) under the assumption that the root nodes are the origin contributory nodes. This table shows how the states of the parent nodes can be a determinant factor of the state in their child nodes. In the rest of this section, we present the CPT of the intermediate nodes and leaf nodes in the *ASAS* model, and the description of the autonomy agent's situation understanding is obtained from the combination of the intermediate nodes' states as defined in the explanation provider.

A) CPT of intermediate nodes

Now, let's take a look at the node *LVE* which has three contributory nodes: *LV*, *LVTL*, and *LVE*. When an *LV* ahead (tailing situation) is detected ($P_{LV} = true$), the *LV* can be either after or before the TL location. *LV* located after TL means that it cannot be used as the basis of the stop manoeuvre when the red light is unrecognized. Therefore, the distance between *HV* and both the TL location and *LV* is calculated. When the distance from HV to *LV* is greater than to the TL location, this knowledge is stored in node *LVTL* and is denoted by $P_{LVTL} = false$. Otherwise, $P_{LVTL} = true$. Given that the states of the nodes *LV* and *LVTL* are set to true, the agent's knowledge about the *LV* exists before the TL location is captured in node *LVE* by assigning $P_{LVE} = true$. Other state combinations of node *LV* and *LVTL* will result in $P_{LVE} = false$ indicating the agent holds the knowledge that there is no *LV* (free ride situations) before the TL location. With such behaviours, the CPT of node *LVE* is shown in Table 6-2.

Table 6-2. CPT of node LVE

Combination of root node states			Intermediate node states	
LVTL	TLE	LV	$P_{LVE} = True$	$P_{LVE} = False$
True	True	True	1	0
Other combinations			0	1

When the autonomy agent knows that *HV* enters the segment 2 ($P_{TLE} = true$) and the recognized TL state is red ($P_{TLR} = true$), this combination of the agent's knowledge is stored in node *S2R* by assigning its state to *true*. After entering segment 1 (indicated by $P_{TLS1} = true$), if the TL state is still red, *S2R* will turn into the *false* state and *S1R* into the *true* state. $P_{S1R} = true$ means that the agent knows that the current segment is segment 1 and the TL state is red. The same flow is also applied when the TL state is green or unknown. While in segment 2, *S2G* or *S2U* will change into the *true* state. After entering segment 1, these two nodes will have the *false* state, but *S1G* and *S1U* change from the *false* state into the *true* state. The detail CPT for nodes *S2R*, *S2G*, *S2U*, *S1R*, *S1G*, and *S1U* is presented in Table 6-3.

Table 6-3. CPT of nodes S2R, S2G, S2U, S1R, S1G, and S1U

Combination of root node states		**Intermediate node states**	
TLE	TLR	$P_{S2R} = True$	$P_{S2R} = False$
True	True	1	0
Other combinations		0	1
S2R	TLS1	$P_{S1R} = True$	$P_{S1R} = False$
True	True	1	0
Other combinations		0	1
TLE	TLG	$P_{S2G} = True$	$P_{S2G} = False$
True	True	1	0
Other combinations		0	1
S2G	TLS1	$P_{S1G} = True$	$P_{S1G} = False$
True	True	1	0
Other combinations		0	1
TLE	TLU	$P_{S2U} = True$	$P_{S2U} = False$
True	True	1	0
Other combinations		0	1
S2U	TLS1	$P_{S1U} = True$	$P_{S1U} = False$
True	True	1	0
Other combinations		0	1

The developed *ASAS* model uses three nodes called *S2P*, *S1PR*, and *S1PGU* to capture the agent's knowledge after being affected by the state of node *LVE*. For example, when $S2P$ holds the true state, this means that currently the *HV* in segment 2 and *LV* before TL

is detected. Similarly, if the state of *S1PR* equals true, this means that currently, *HV* in segment 1 with *LV* before TL is detected. Furthermore, *S1PGU* becomes the node representing the situations under the green light, yellow light, and unknown TL state because this kind of situation lead to the same autonomy agent's action projection. When *S1PGU*'s state equals true, this means that currently *HV* in segment 1 and *LV* before TL is detected, but either green light, yellow light, or unknown TL state is recognized. The complete CPT table which shows how the nodes *S2P*, *S1PR*, and *S1PGU* can have the true or false state based on their contributory nodes, can be seen in Table 6-4.

Table 6-4. CPT of nodes S2P, S1PR, and S1PGU

Combination of root node states			**Intermediate node states**		
LVE	TLS1		$P_{S2P} = True$	$P_{S2P} = False$	$P_{S2P} = NA$
True	False		1	0	0
False	False		0	1	0
Other combinations			0	0	1
LVE	S1R		$P_{S1PR} = True$	$P_{S1PR} = False$	$P_{S1PR} = NA$
True	True		1	0	0
False	True		0	1	0
Other combinations			0	0	1
LVE	SIG	S1U	$P_{S1PGU} = True$	$P_{S1PGU} = False$	$P_{S1PGU} = NA$
True	True	NA	1	0	0
True	NA	True	1	0	0
False	True	NA	0	1	0
False	NA	True	0	1	0
Other combinations			0	0	1

B) CPT of leaf nodes

In the developed *ASAS* model, nodes S2P, S1PR, and S1PGU also have a direct connection to the ones representing the agent's action projection: node AS, SV, and KG. For example, knowing the existence of LV before TL ($P_{LVE} = true$) while the vehicle is currently in segment 2 ($P_{TLS1} = false$) is denoted by $P_{S2P} = true$, and as a consequence, the agent's action projection for this state is AS($P_{AS} = true$). Other state combinations of nodes LVE and TLS1 will turn the S2P state into false and the corresponding action

projection for this state is KG ($P_{KG} = true$). The CPT of node AS, KG, and SV can be seen in Table 6-5.

Table 6-5. CPT of node AS, KG, and SV

Combination of root node states			**Leaf node states**	
S1PR	S2P	S1PGU	$P_{AS} = True$	$P_{AS} = False$
True	NA	NA	1	0
False	True	NA	1	0
NA	NA	True	1	0
Other combinations			0	1
S2P	S1PGU		$P_{KG} = True$	$P_{KG} = False$
False	NA		1	0
NA	False		1	0
Other combinations			0	1
S1PR			$P_{SV} = True$	$P_{SV} = False$
False			1	0
Other Combinations			0	1

In addition to representing the situation abstraction that determines the agent's behaviours, the *ASAS* model also generates the likelihood of task failure for TL situations represented by the status of node *LoFTL*. The causal factors for this node are coming from nodes *STL* and *LVE*. In this regard, $P_{STL} = true$ refers to an autonomy agent's state where it knows the TL state. In a situation with $P_{STL} = true$ but $P_{LVE} = false$, the autonomy agent still has a high possibility of successfully performing its task in the TL situation. Yet, when $P_{STL} = false$ and $P_{LVE} = true$ are given, the possibility of success will decrease to around i.e. 70% because even though the TL state is unknown, the autonomy agent can still decide the vehicle manoeuvres based on the vehicle ahead. The task failure of the autonomy agent in the TL situation occurs when P_{STL} and P_{LVE} are equal to *false* which means that this agent does not know the TL state and an LV-based manoeuvre is not possible as no LV is detected. The detail of CPT for nodes *STL* and *LoFTL* can be seen in Table 6-6.

Table 6-6. CPT of nodes STL and LoFTL

Combination of root node states		**Intermediate node states**		
TLE	TLU	$P_{STL} = True$	$P_{STL} = False$	$P_{STL} = NA$
True	False	1	0	0
True	True	0	1	0
Other combinations		0	0	1

STL	LVE	$P_{LoFTL} = True$	$P_{LoFTL} = False$	$P_{LoFTL} = NA$
True	True	0.99	0.01	0
True	False	0.99	0.01	0
False	True	0.7	0.3	0
False	False	0.01	0.99	0
Other combinations		0	0	1

C) Description of the autonomy agent's situation understanding

The states of the perception nodes and leaf nodes in the *ASAS* model can be easily translated into a meaningful explanation to a human teammate. For example, when the root node $TLR = true$, it will be easily translated as the TL state is red. However, the critical part in generating a behaviour explanation is in the intermediate nodes as they represent the autonomy agent's higher-level situation understanding. This understanding is sometimes obtained from the combination of some intermediate nodes. In our proposed *ASAS*-based method, the meaning of the intermediate nodes' states combination is defined in the explanation provider. Table 6-7 presents the description of the higher-level situation represented by the combination of intermediate nodes' states.

Table 6-7. Snippet information in Explanation Provider for TL situation

Detected agent's action projection	The representation of possible contributory situations	The meaning of situation representation
AS = True	- S2P=true	Entering TL situations, keeping the safe distance with LV
	- S1PR=true	TL state is red, yet preparing to stop based on relative distance to LV
	- S1PGU=true, S1G=true, S1U=NA	TL state is green, currently keeping the safe distance to LV
	- S1PGU=true, S1G=NA, S1U=true	Unknown TL state, LV-based stop/go performed
KG = True	- S2P=false	Entering TL situations under free ride situations
	- S1PGU=false, S1G=true, S1U=NA	TL state is green and free riding

Detected agent's action projection	The representation of possible contributory situations	The meaning of situation representation
	- S1PGU=false, S1G=NA, S1U=true	TL state is unrecognized and no LV ahead
SV = True	- S1PR=false	TL state is red, preparing to stop

6.2.2.2. Self-explanation Generation on for the Goal *Pass Traffic Lights*

Referring to the simulation route (see Figure 6-3), TL is located on the GPS coordinates -7.73450859 (latitude), 110.39534569 (longitude). Based on this TL location, three simulated situations are established to determine the scenarios to generate the autonomy agent's behaviour explanation using the ASAS-based method, which uses leaf nodes as the basis of explanation generation. These leaf nodes, *KG*, *AS*, and *SV* represent the autonomy agent's action projection.

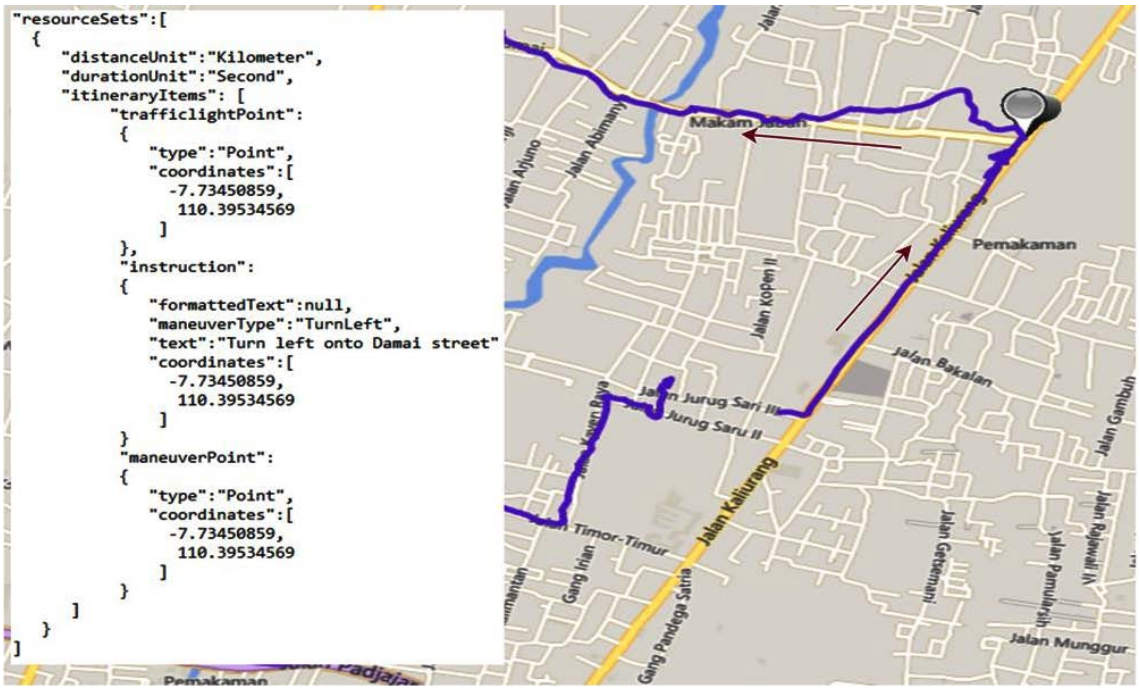


Figure 6-3. The route in simulation data

A. Generating Explanation: Scenario 1

The focus of this scenario is to explain why the vehicle keeps going (*KG*) by examining what kind of situations makes the agent perform such a manoeuvre. Once *KG* is detected, the self-explanation generator starts to retrieve the captured *ASAS* model state in the

episodical memory. However, based on the explanation provider (see Table 6-7), there are three different situations that cause *KG* to be executed.

As an example of the shrinking process, initially, the first layer matrix values for node *LoFTL* which has three states, *success*, *failure*, and *not available,* are [1, 0, 1]. The first layer fuzzy relation operation eliminates the *failure* state as the first layer matrix indicates zero value for this state. In this regard, the 'failure' state can be implicitly represented by the 'success' state. As a result, the remaining states in LoFTL are '*success'* and '*not available*'. In the second fuzzy relation operation, the probability of these two remaining states is examined. When the 'success' state in $LoFTL$ has a higher probability than the 'not available' state, this means that the 'success' state is selected. Once the compact state of the *ASAS* model is generated, it will be encapsulated and sent to the state handler.

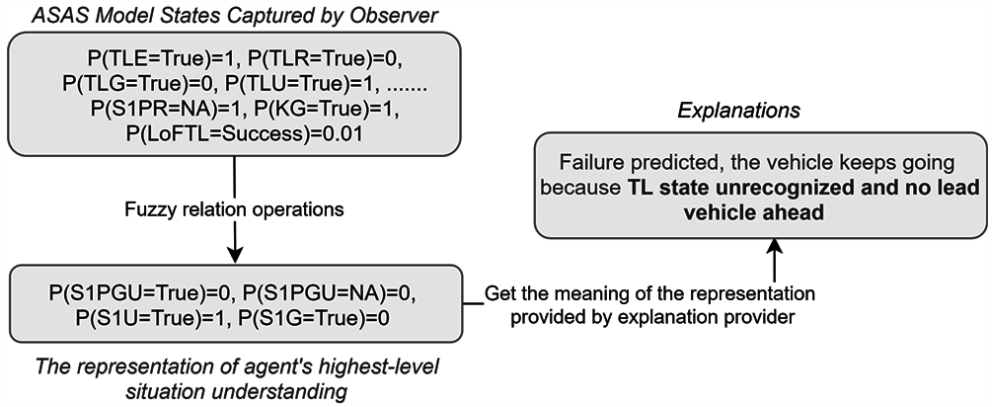


Figure 6-4. Illustration of generating agent's behaviours explanations for scenario 1 of TL

As illustrated in Figure 6-4, from the episodical memory, all *ASAS* model states are captured by the observer. Then, the fuzzy relation operations are performed for this episode, and these operations produce the shrunk states to indicate the representation of the agent's highest-level situation understanding which drives the agent's behaviours.

In generating the explanations, the state handler performs critical tasks to trace back the causal factors based on the *ASAS* model states obtained from the episodical memory. Since the *ASAS* model is a causal network, the tracing back process can be performed efficiently because it does not need to crawl all the states of the contributory nodes. Using the Manhattan distance similarities technique, when it is found that $P_{S1PGU} = false$, $P_{S1PGU} = NA$, $P_{S1U} = true$ and $P_{S1G} = false$ for this scenario, it is not necessary to

examine all of its 11 contributory nodes (*SIG*, *LVE*, *TLS1*, *S2U*, *S2G*, *TLU*, *TLE*, *TLG*, *LVE*, *LV*, and *LVTL*) because the states of nodes P_{S1PGU}, P_{S1PGU}, P_{S1U} and P_{S1G} already represent the highest-level understanding of the situation held by the agent given the contributory nodes' states. As a result, by comparing the states of nodes P_{S1PGU}, P_{S1PGU}, P_{S1U} and P_{S1G} with the explanation provider, the explanation for scenario 1 is generated which tells that 'the vehicle keeps going and most likely to fail to follow TL, because the TL state unrecognized and no lead vehicle ahead'.

B. Generating Explanation: Scenario 2

In this scenario, the simulation shows that the current agent's reaction is AS. Based on the *ASAS* model, this agent's action projection can be caused by four contributory situations (see Table 6-7). Similar to scenario 1, after the explanation generator performs the tracing back processes for this episode throughout the episodical memory, this indicates that the closest causal situation for this episode is represented by $P_{S1PGU} = true$, $P_{S1G} = NA$, and $P_{S1U} = true$. As a result, the corresponding description of the situation understanding is 'the vehicle reacts based on lead vehicle and may succeed to pass TL situations even though TL state unrecognized'. To generate the explanation for this scenario, the self-explanation generator successfully performs an efficient tracing-back on the *ASAS* model nodes by only visiting the contributory nodes that are two-levels down, which are nodes *S1PGU*, *SIG*, and *S1U*. The other contributory nodes that are at levels further down (*LVE*, *LV*, *LVTL*, *TLS1*, *TLE*, *TLU*, *TLG*, *S1U*, *S2U*, and *S2G*) are not required to be visited.

C. Generating Explanation: Scenario 3

In this scenario, *KG* reveals as the agent's action projection, and the self-explanation generator finds that the closest situation captured by the observer in episodical memory is represented by $P_{S1PGU} = false$, $P_{S1G} = true$, and $P_{S1U} = NA$. Therefore, the generated explanation is 'the vehicle keeps going and successfully passes TL situations as green light is detected with no lead vehicle ahead'. This explanation fits with the situations given by the simulation data. Moreover, the tracing back performance also shows that the explanation generator only needs to visit three nodes in the *ASAS* model and ignores the other 11 contributory nodes. Therefore, it can be concluded that the self-explanation generator works as expected.

6.2.2.3. Sensitivity Analysis

In this part, the sensitivity analysis is detailed to examine the degree of dependencies, particularly between the root nodes and leaf nodes used in the *ASAS* model. Figure 6-5 presents four graphs with bars illustrating the sensitivity analysis result in target node states: $P_{AS} = true$ (Figure 6-5a), $P_{KG} = true$ (Figure 6-5b), $P_{SV} = true$ (Figure 6-5c), and $P_{LoFTL} = true$ (Figure 6-5d). Bars with maximum length indicate that the corresponding node states have a strong effect on the target node states.

Also, the bars are divided into two parts from the middle, each part can have a red and green colour. When the state of a node shows green on the right side, this means this state positively contributes to target nodes. Yet, when the green part is on the left side, this means that the positive contribution comes from the reverse state of the node. For example, in Figure 6-5a, when the nodes *TLE*, *LVTL*, *LV*, and *TLS1* hold the *true* state, they will significantly contribute to the possibility of the agent selecting *AS* for its action projection ($P_{AS} = true$). However, when *TLS1* holds N/A or *false* state, it will decrease the possibility of an agent selecting *AS* for its action projection.

(a) Target state: P(AS) = true

(b) Target state: P(KG) = true

(c) Target state: P(SV) = true

(d) Target state: P(LoFTL) = Success

Figure 6-5. Sensitivity analysis to measure the strength of the dependencies between root nodes and leaf nodes

6.3. Study Case 2: Behaviour Explanations in Achieving the Goal *Overtake Vehicles*

6.3.1. Introduction to Study Case 2

This second case study explains the autonomy agent's behaviour based on its three-level situation awareness for the goal *Overtake vehicles* as defined in Chapter 4 and presented in Table 4-12. Overtaking can be viewed as a sophisticated feature in autopilot technology that requires a form of coordination between human and an autonomy agent, particularly in a collaborative driving context. When the autonomy agent controls the vehicle, it may recognize a situation where an overtaking manoeuvre can be made, but this manoeuvre cannot be executed before obtaining the human's consent. To acquire this approval, the autonomy agent sends a notification to the human teammate to re-examine the current situation. When agreed, the human conveys their consent by sending a turning signal to the agent as a confirmation to proceed with the overtaking manoeuvres. Without this confirmation, the autonomy agent keeps the vehicle in the tailing mode.

Overtaking involves several functional abilities including adaptive cruise control, blind spot warning function, closing vehicle detection function, lane change function, lane departure detection, and overtaking trajectory generator. However, as a driving situation is sometimes unpredictable, not all overtaking manoeuvres run smoothly without causing road incidents. Current studies show that mostly, road incidents involving this kind of manoeuvre are caused by sideswipe (Ghaffari et al. 2012). Sideswipe is a type of collision where the host vehicle (*HV*) hits another vehicle in another lane when it starts or finishes overtaking. The speed change of the overtaken vehicle or rear vehicle becomes the primary factor of such sideswipe incidents.

In addition to increasing collision risk, the speed change of the overtaken vehicle can also cause the cancelation of the overtaking manoeuvre as the adjusted overtaking speed may violate the road speed limit. Therefore, developing the human driver's awareness is essential during the overtaking manoeuvre performed by the autonomy agent, particularly when an overtaking cancelation is issued. There are several possible autonomy agent actions in response to such a cancelation such as *stay in overtaking lane* (in one-way street) or *return to departure lane*. Therefore, understanding the cause of overtaking cancelation and execution of *HV* positioning after cancellation are very useful to calibrate

trust and enhance the human situation awareness of the overall situation and enable them to take over the vehicle control from the autonomy agent if necessary.

In this study case, explanation generation clarifies the causal factors of overtaking cancelation and *HV*'s positioning after cancelation. For this purpose, this study provides three scenarios in association with the circumstances causing the cancelation. In all scenarios, it is assumed that overtaking suggestions have been received and approved by the driver and the first lane changing which takes *HV* to the overtaking lane is already executed. These three scenarios are described as follows:

1) **Scenario 1:** The first scenario is a one vehicle overtaking manoeuvre. There is a vehicle to be overtaken, but *LVO* does not present. However, after overtaking, the overtaken vehicle adds its speed which means the overtaking speed will exceed the maximum road speed limit. Therefore, overtaking is cancelled but *HV* will stay in the overtaking lane.

2) **Scenario 2:** In the second scenario, *HV* has just passed the overtaken vehicle (one vehicle only) but before going back to its departure lane, *LVO* reduces its speed. Consequently, the minimum space to perform the lane change to *HV*'s departure lane is not met. Therefore, overtaking is cancelled and *HV* will stay in the overtaking lane.

3) **Scenario 3:** In the third scenario, *HV* just started the overtaking manoeuvre and finished the first lane change to overtake *LV*. However, before passing *LV*, *LVO* slows down so that it is unable to continue overtaking under such circumstances. Therefore, *HV* will go back to its departure lane.

The implementation of the explanation generation methods is intended to produce clarification as described in the scenarios.

6.3.2. Implementing the ASAS-based Method

6.3.2.1. Modelling the ASAS of Autonomy Agent for the goal *Overtake Vehicles*

There are 11 Bayesian network nodes in total in the ASAS model of the autonomy agent in association with pre-overtaking. As seen in Figure 6-6, the pre-overtaking situation involves 7 perception nodes (shown in yellow). In this situation, the salient objects

affecting the autonomy agent's perception are the existence of a car behind *HV* in an adjacent lane, the existence of *LV*, and the existence of a car in front in an adjacent lane which are represented by nodes *RLV* and *LVO* respectively. Whether the *HV* has a safe margin to *RLV* and *LVO* in this overtaking context is represented by nodes *RLV-Range* and *LVO-Range*. Furthermore, the most critical perception node in the pre-overtaking situation is *TLIM*. This node represents a time limiter which stores how long *HV* runs under the target speed and road speed during a tailing situation so that it is necessary to decide to take the overtaking manoeuvre. However, before this decision is made, it needs another perception which comes from the prediction of overtaking speed. This perception is represented by *POTR-VOT*. Table 6-8 presents the description and states of the root nodes involved in the pre-overtaking situation.

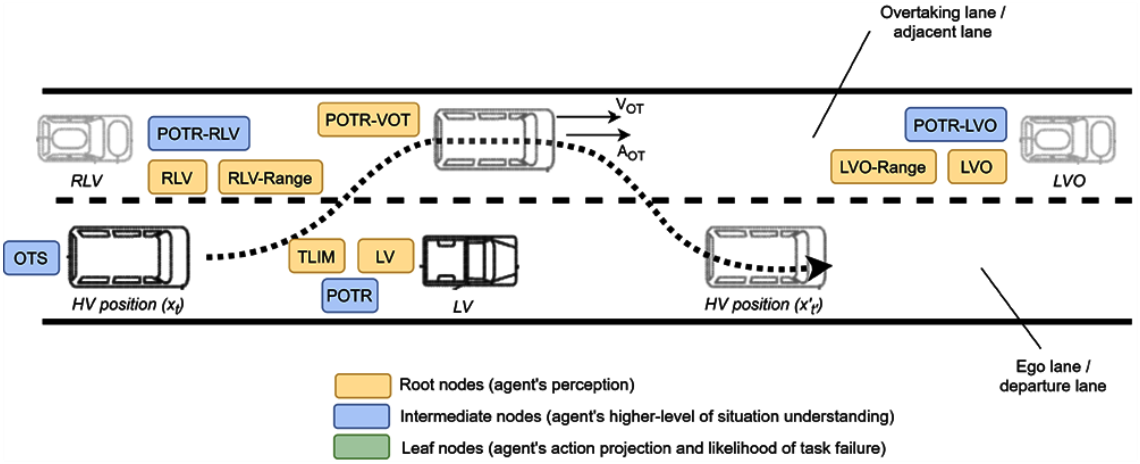


Figure 6-6. Corresponding nodes for ASAS model in pre-overtaking

Table 6-8. The description of root nodes in pre-overtaking situation

Node	State	Description	Node	State	Description
LV	True	LV exists	LVO-Range	Safe	Range to LVO is safe
	False	LV does not exist		Unsafe	Range to LVO is not safe
RLV	True	RLV exists	TLIM	True	Time factor to overtake decision is met
	False	LV does not exist		False	Time factor to overtake decision is not met
RLV-Range	Safe	Range to RLV is safe	POTR-VOT	Safe	Predicted overtaking speed below road speed limit
	Unsafe	Range to RLV is not safe		Unsafe	Predicted overtaking speed exceeds road speed limit
LVO	True	LVO exists			
	False	LVO does not exist			

Using the 7 perception nodes in the pre-overtaking situation, a higher-level situation understanding is developed and represented by nodes *POTR-RLV*, *POTR-LVO*, *POTR*, and *OTS, respectively*. *POTR-RLV* and *POTR-LVO* are the nodes to represent collision risk calculation to *RLV* and *LVO*, respectively. The values of these two nodes, then, contribute to determining the state of *POTR* which represents the overall risk for suggesting an overtaking manoeuvre. In addition to *POTR-RLV* and *POTR-LVO*, another contributory node to *POTR* is *POTR-VOT*. Once the state of *POTR* is set and combined with the state of nodes *TLIM* and *LV*, it can be used to infer an overtaking suggestion which is denoted by node *OTS*. The conditional probability table (CPT) of all intermediate nodes affecting the state of *OTS* is presented in Table 6-9.

Table 6-9. CPT of nodes POTR-RLV, POTR-LVO, POTR, OTS

Combination of root node states			**Intermediate node states**	
RLV	RLV-Range		$P_{POTR-RLV} = Safe$	$P_{POTR-RLV} = Unsafe$
True	Unsafe		0	1
Other combinations			1	0
LVO	LVO-Range		$P_{POTR-LVO} = Safe$	$P_{POTR-LVO} = Unsafe$
True	Unsafe		0	1
Other combinations			1	0
POTR-RLV	POTR-LVO	POTR-VOT	$P_{POTR} = Safe$	$P_{POTR} = Unsafe$
Safe	Safe	Safe	1	0
Other combinations			0	1
TLIM	POTR	LV	$P_{OTS} = True$	$P_{OTS} = False$
True	Safe	True	1	0
Other combinations			0	1

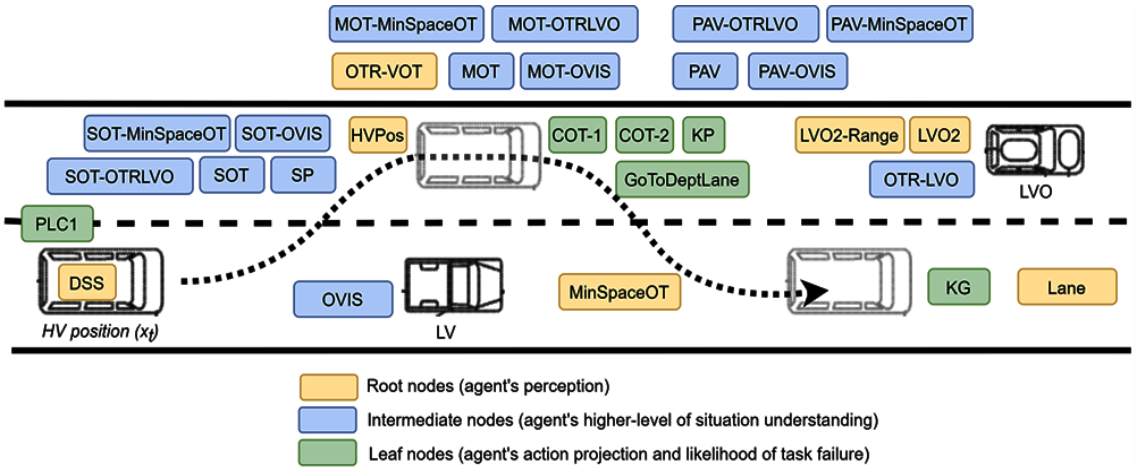


Figure 6-7. Corresponding nodes for ASAS model in overtaking situation

When $P_{OTS} = true$ and node *DSS* (which represents the driver's approval) indicates the 'Yes' state, the overtaking manoeuvre will proceed. Therefore, *DSS* node is considered as one of seven perception nodes in the overtaking situation. In Figure 6-7, all corresponding nodes for the overtaking situation are presented. In addition to *DSS*, other perception nodes for the overtaking situation are *Lane*, *HVPos*, *OTR-VOT*, *MinSpaceOT*, *LVO*2, and *LVO*2-*Range*. Node *Lane* represents *HV*'s current lane (which can be ego lane, overtaking lane, or departure lane) after an overtaking suggestion is approved by the human driver. When *HV* is in the overtaking lane, the overtaking speed is re-evaluated and captured in node *OTR-VOT*. Re-evaluation is also performed for collision risk with *LVO*, therefore, *LVO*2 and *LVO*2-*Range* are provided as the perception nodes in association with *LVO*. These two nodes have the same function as nodes *LVO* and *LVO-Range* in the pre-overtaking situation. Furthermore, nodes *HVPos* represent *HV*'s position, that is whether it is behind LV_1 (the first vehicle that will be overtaken), after LV_1, or after LV_n (the last overtaken vehicle). The last perception node in the overtaking situation, *MinSpaceOT*, captures the state of space needed to return to the departure lane after passing the last overtaken vehicle. The description and states of perception nodes in the overtaking situation are presented in Table 6-10.

Table 6-10. The description of root nodes in the overtaking situation

Node	State	Description	Node	State	Description
Lane	OTLane	HV is currently in overtaking lane	HVPos	BehindLV1	HV is currently behind LV1
	EgoLane	HV is currently in ego lane		AfterLV1	HV is currently after LV1
	DeptLane	HV is currently in departure lane		AfterLn	HV is currently after LVn
OTR-VOT	Safe	Predicted overtaking speed below road speed limit	LVO2	True	LVO exists
	Unsafe	Predicted overtaking speed exceeds road speed limit		False	LVO does not exist
MinSpaceOT	Safe	Overtaking range is safe	LVO2-Range	Safe	Range to LVO is safe
	Unsafe	Overtaking range is unsafe		Unsafe	Range to LVO is not safe
DSS	Yes	Driver agrees to overtake			
	No	Driver disagrees to overtake			

In the rest of this section, we present the CPT of the intermediate nodes and leaf nodes in the *ASAS* model, and the description of the autonomy agent's situation understanding.

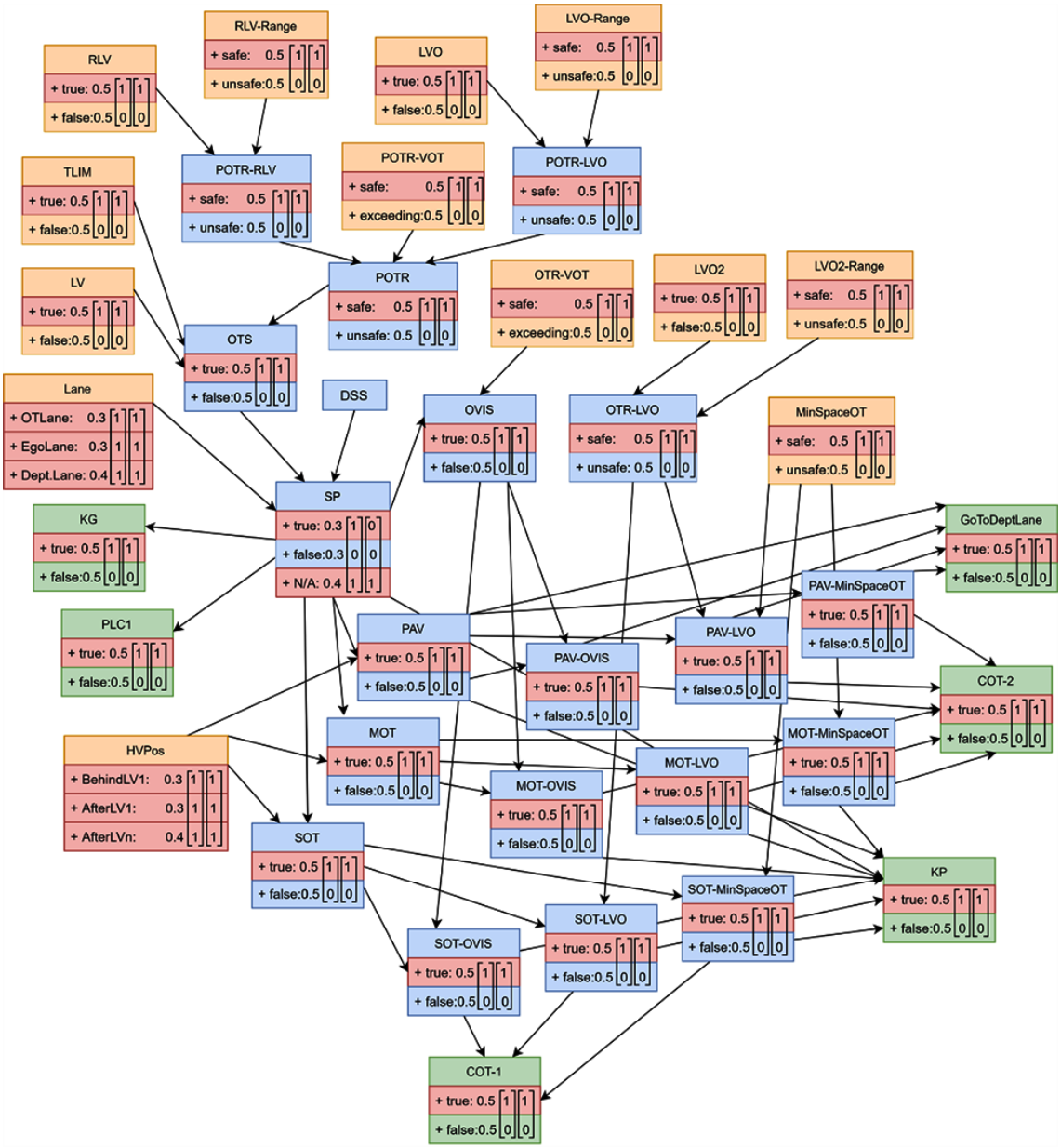


Figure 6-8. A Bayesian network model representing the development of the agent's artificial situation awareness states (ASAS) in the overtaking situation

A) CPT of intermediate nodes

There are 15 intermediate nodes representing a higher situation understanding and 6 leaf nodes representing the autonomy agent's action projection in the overtaking situation. The relation among these nodes with their contributory nodes, including nodes from the pre-overtaking situation, can be seen in Figure 6-8. In the overtaking situation, node *SP*

plays a significant role which captures whether *HV* is currently preparing to overtake ($P_{SP} = False$), currently passing the overtaken vehicles in the overtaking lane ($P_{SP} = True$), or done with overtaking ($P_{SP} = NA$). The state of *SP* is driven by three contributory nodes, *DSS*, *OTS*, and *Lane*. When $P_{SP} = False$, the action projection of the autonomy agent is to prepare the first lane change to start overtaking. This action is captured in a leaf node called *PLC*1. When $P_{SP} = NA$ (Not Available), the keep going manoeuvre which is represented by node *KG*, will be selected as the overtaking manoeuvre is accomplished and the vehicle is already back to the departure lane. The CPT of *SP*, *PLC*1, and *KG* can be seen in Table 6-11.

Table 6-11. CPT of node SP, PLC1, and KG

Combination of root node states			**Child node states**		
DSS	OTS	Lane	$P_{SP} = True$	$P_{SP} = False$	$P_{SP} = NA$
Yes	True	OTLane	1	0	0
Yes	True	EgoLane	0	1	0
Other Combinations			0	0	1
SP			$P_{KG} = True$	$P_{KG} = False$	
True			1	0	
False			0	1	
SP			$P_{PCL1} = True$	$P_{PCL1} = False$	
False			1	0	
Other Combinations			0	1	

However, when $P_{SP} = True$, the action projection will depend on the current *HV* position as stated in node *HVPos*. It should be noted that the HV position behind *LV* can either be in the ego lane or the overtaking lane. Therefore, to capture the actual position in the overtaking lane, the *ASAS* model uses three nodes, *SOT*, *MOT*, and *PAV*. The states of these three nodes are contributed by nodes *SP* and *HVPos*. When *HV* is only entering the overtaking lane and is still located behind *LV*, this situation is captured by node *SOT* ($P_{SOT} = True$). However, when *MOT* holds $P_{MOT} = True$, it represents a situation where *HV* is in about the same position as the overtaken vehicle (if there is only one overtaken vehicle) or somewhere between LV_1 and LV_n (if there are several overtaken vehicles). If *HV* has passed LV_n but is still in the overtaking lane, it is captured by node *PAV* with $P_{PAV} = True$. To drive the state of *SOT*, *MOT*, and *PAV*, the CPT table which links the relation of these three nodes with their contributory nodes (SV and HVPos) can be seen in Table 6-12.

Table 6-12. CPT of SOT, MOT, and PAV

Combination of root node states		Intermediate node states	
SP	HVPos	$P_{SOT} = True$	$P_{SOT} = False$
True	BehindLV	1	0
Other combinations		0	1
SP	HVPos	$P_{MOT} = True$	$P_{MOT} = False$
True	AfterLV1	1	0
Other combinations		0	1
SP	HVPos	$P_{PAV} = True$	$P_{PAV} = False$
True	AfterLVn	1	0
Other combinations		0	1

While the overtaking speed increases ($P_{OTR-VOT} = True$), it can be inferred that the overtaken vehicle's speed also increases its speed which is captured in node *OVIS* by $P_{OVIS} = True$. The increase of collision risk caused by the unsafe distance to *LVO* is captured in node *OTR-LVO*. The CPT which defines the causal factor of nodes *OVIS* and *OTR-LVO* can be seen in Table 6-13. These risks, then, will be used to evaluate each *HV* position stated in *SOT*, *MOT*, or *PAV* including the risk captured by node *MinSpaceOT* which reflects the risk of returning to the departure lane.

Table 6-13. CPT of OVIS and OTR-LVO

Combination of root node states		Intermediate node states	
OTR-VOT	SP	$P_{OVIS} = True$	$P_{OVIS} = False$
Unsafe	True	0	1
Other combinations		1	0
LVO2	LVO2-Range	$P_{OTR-LVO} = Safe$	$P_{OTR-LVO} = Unsafe$
True	Unsafe	0	1
Other combinations		1	0

Therefore, for every *SOT*, *MOT*, and *PAV*, there will be a node which captures the situation understanding from their combination with nodes *OVIS*, *OTR-LVO*, and *MinSpaceOT*. For example, nodes *SOT-OVIS* are used to hold the knowledge obtained from node *SOT* and node *OVIS*. Given $P_{SOT-OVIS} = True$, this means that when HV is just entering the overtaking lane and is still located behind *LV*, *LV* increases its speed so that the overtaking speed will exceed the road speed limit. Given $P_{SOT-OTRLVO} = True$, the autonomy agent holds the knowledge that when *HV* is just entering the overtaking lane and is still located behind *LV*, *LVO* decreases its speed so that the collision risk with *LVO* becomes high. Furthermore, when $P_{SOT-MinSpaceOT} = True$, this means that when

HV is just entering the overtaking lane and is still located behind LV, the possibility of finishing the overtaking manoeuvre is low as the minimum space to return to the departure lane is not met. The CPT of each combination of the nodes above can be seen in Table 6-14.

Table 6-14. CPT of SOT, MOT, PAV combined with OVIS, OTRLVO, MinSpaceOT

Combination of root node states		**Intermediate node states**	
SOT	OVIS	$P_{SOT-OVIS} = True$	$P_{SOT-OVIS} = False$
True	True	1	0
Other combinations		0	1
MOT	OVIS	$P_{MOT-OVIS} = True$	$P_{MOT-OVIS} = False$
True	True	1	0
Other combinations		0	1
PAT	OVIS	$P_{PAV-OVIS} = True$	$P_{PAV-OVIS} = False$
True	True	1	0
Other combinations		0	1
SOT	OTRLVO	$P_{SOT-OTRLVO} = True$	$P_{SOT-OTRLVO} = False$
True	Unsafe	1	0
Other combinations		0	1
MOT	OTRLVO	$P_{MOT-OTRLVO} = True$	$P_{MOT-OTRLVO} = False$
True	Unsafe	1	0
Other combinations		0	1
PAV	OTRLVO	$P_{PAV-OTRLVO} = True$	$P_{PAV-OTRLVO} = False$
True	Unsafe	1	0
Other combinations		0	1
SOT	MinSpaceOT	$P_{SOT-MinSpaceOT} = True$	$P_{SOT-MinSpaceOT} = False$
True	Unsafe	1	0
Other combinations		0	1
SOT	MinSpaceOT	$P_{MOT-MinSpaceOT} = True$	$P_{MOT-MinSpaceOT} = False$
True	Unsafe	1	0
Other combinations		0	1
SOT	MinSpaceOT	$P_{PAV-MinSpaceOT} = True$	$P_{PAV-MinSpaceOT} = False$
True	Unsafe	1	0
Other combinations		0	1

B) CPT of leaf nodes

Referring to the overtaking process from the previous section, when *HV* is still behind LV_1 in the overtaking lane but the collision risk to *LVO* becomes high, the overtaking cancelation will require *HV* to go back to its original position before LV_1. Such an action projection is represented by node *COT*-1 in the *ASAS* model. If all risks are considerably low and HV has already passed all the overtaken vehicles, the action projection to respond

to this situation is to finish overtaking by going back to the departure lane. This action projection is represented by node *GoToDeptLane* in the *ASAS* model. The CPT for these two action projection nodes is presented in Table 6-15.

Table 6-15. CPT of node COT-1 and GoToDeptLane

Combination of root node states				**Leaf node states**	
SOT-OVIS	SOT-OTRLVO	SOT-MinSpaceOT		$P_{COT-1} = True$	$P_{COT-1} = False$
False	False	False		0	1
Other combinations				1	0
PAV-OVIS	PAV-OTRLVO	PAV-MinSpaceOT	PAV	$P_{GOTODEPTLANE} = True$	$P_{GOTODEPTLANE} = False$
False	False	False	True	1	0
Other combinations				0	1

When all risks are safe, but either *SOT* or *MOT* is still actively used to indicate *HV*'s position, the action projection for this situation is denoted by node *KP* which means that *HV* will keep processing the current overtaking manoeuvre. However, when one of the measured risks is high while *HV*'s position is indicated by *MOT* or *PAV*, according to the overtaking process, it will trigger the overtaking cancelation but *HV* will stay in the overtaking lane as its new ego lane. This kind of action projection is represented by node *COT*-2. The detailed states of contributory nodes affecting nodes *KP* and *COT*-2 are presented in Table 6-16.

Table 6-16. CPT of node KP and COT-2

Combination of root node states								**Intermediate nodes states**	
SOT-OVIS	MOT-OVIS	SOT-OTRLVO	MOV-OTRLVO	SOT-MinSpaceOT	MOT-MinSpaceOT	PAV	SP	$P_{KP} = True$	$P_{KP} = False$
False	False	False	False	False	False	False	True	1	0
Other combinations								0	1
MOT-OVIS	PAV-OVIS	MOV-OTRLVO	PAV-OTRLVO	MOT-MinSpaceOT	PAV-MinSpaceOT			$P_{COT-2} = True$	$P_{COT-2} = False$
False	False	False	False	False	False			0	1
Other combinations								1	0

Furthermore, the *ASAS* model shown in Figure 6-8 also comes with two matrices for the fuzzy relation operation. As most nodes in the model have two contrary states, such as true/false, safe/unsafe, the two matrices also come with 1/0 values where 1 represents a true and safe state. As a consequence, in the fuzzy relation operation, the state which is

assigned the 0 value from the matrices will be eliminated. When all states in the node are critical, such as *HVPos* and Lane, all matrices corresponding to these nodes are all-ones matrices. In a special case like the *SP* node, which has *true*/*false*/NA states, all matrices will have elements [1, 0, 1] where the zero represents *false* state.

C) Description of the autonomy agent's situation understanding

After all nodes in the *ASAS* model are set and connected, the next step is to develop the explanation provider which defines the rationale of the autonomy agent's actions based on their contributory situations. The snippet autonomy agent's situation understanding in explanation provider can be seen in Table 6-17.

Table 6-17. Snippet information in Explanation Provider for overtaking situation

Detected agent's action projection	The representation of possible contributory situations	The meaning of situation representation
COT-1 = True	- SOT-OVIS=true; SOT-OTRLVO=false; SOT-MinSpaceOT=false	Just entering overtaking lane, the predicted overtaking speed exceeded the road speed limit as overtaken vehicle increased its speed
	- SOT-OTRLVO=true; SOT-OVIS=false; SOT-MinSpaceOT=false	Just entering overtaking lane, lead vehicle in overtaking lane slowed down
	- SOT-MinSpaceOT=true; SOT-OVIS=false; SOT-OTRLVO=false	Just entering overtaking lane, minimum space to go back to departure lane after passing is not met
COT-2 = True	- MOT-OVIS=true; MOT-OTRLVO=false; MOT-MinSpaceOT=false; PAV-OVIS=false; PAV-OTRLVO=false PAV-MinSpaceOT=false	In the middle of passing overtaken vehicle, the predicted overtaking speed exceeded the road speed limit as overtaken vehicle increased its speed
	- MOT-OTRLVO=true; MOT-OVIS=false; MOT-MinSpaceOT=false; PAV-OVIS=false; PAV-OTRLVO=false; PAV-MinSpaceOT=false	In the middle of passing overtaken vehicle, lead vehicle in overtaking lane slowed down
	- MOT-MinSpaceOT=true; MOT-OVIS=false; MOT-OTRLVO=false; PAV-OVIS=false; PAV-OTRLVO=false; PAV-MinSpaceOT=false	In the middle of passing overtaken vehicle, minimum space to go back to departure lane after passing is not met
	- PAV-OVIS=true; PAV-OTRLVO=false; PAV-MinSpaceOT=false; MOT-OVIS=false; MOT-	Just passed the overtaken vehicle, the predicted overtaking speed exceeded the road speed limit as overtaken vehicle increased its speed

Detected agent's action projection	The representation of possible contributory situations	The meaning of situation representation
	OTRLVO=false; MOT-MinSpaceOT=false	
	- PAV-OTRLVO=true; PAV-OVIS=false; PAV-MinSpaceOT=false; MOT-OVIS=false; MOT-OTRLVO=false; MOT-MinSpaceOT=false	Just passed the overtaken vehicle, lead vehicle in overtaking lane slowed down
	- PAV-MinSpaceOT=true; PAV-OVIS=false; PAV-OTRLVO=false; MOT-OVIS=false; MOT-OTRLVO=false; MOT-MinSpaceOT=false	Just passed the overtaken vehicle, minimum space to go back to departure lane after passing is not met
GoToDeptLane = True	- PAV-OVIS=false; PAV-OTRLVO=false; PAV-MinSpaceOT=false; MOT-OVIS=false; MOT-OTRLVO=false; MOT-MinSpaceOT=false	Just passed the overtaken vehicle, all risks are considerably low to go to departure lane

6.3.2.2. Self-explanation Generation for the Goal *Overtake Vehicles*

In this section, the proposed *ASAS*-based method to generate the behaviour explanation is presented using the three pre-determined scenarios. In this regard, a simulated real-driving dataset is used to feed the Bayesian network. In this method, after the observer captures the *ASAS* model states in the episodical memory, it will obtain the closest situation by referring to the explanation provider. Then, the description of the related situation will be generated.

A. Generating Explanation: Scenario 1

In this scenario, once the state $P_{COT-2} = True$ is detected, the behaviour explanation generator will examine the causal factors of this state by examining its contributory nodes in the episodical memory. As illustrated in Figure 6-9, after retrieving the related episode of the *ASAS* model states captured by the observer, the fuzzy relation operation is performed. The result of this operation, then, will be sent to the explanation provider to obtain the closest situation and its meaning. For this scenario, the closest causal situation for this episode is represented by six nodes, *MOT-OVIS*, *MOT-OTRLVO*, *MOT-*

MinSpaceOT, *PAV-OVIS*, *PAV-OTRLVO*, and *PAV-MinSpaceOT*. All of these nodes hold the *false* state, except *MOT-OVIS* which holds the *true* state. Thus, the corresponding description situation understanding is 'In the middle of passing overtaken vehicle, but overtaking is cancelled and stay in overtaking lane because the predicted overtaking speed exceeds the road speed limit as the overtaken vehicle increased its speed'. To generate this explanation, this method only needs to revisit six nodes of the 39 existing nodes in the *ASAS* model.

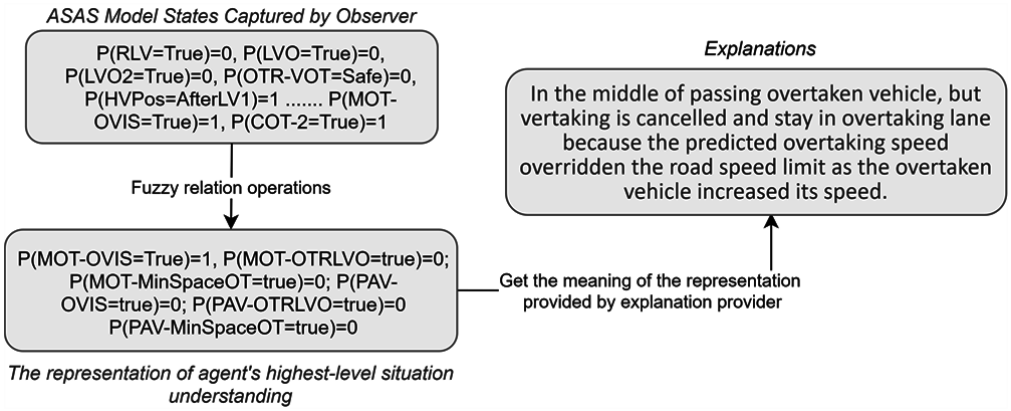


Figure 6-9. Illustration of generating agent's behaviour explanations for scenario 1 of the overtaking situation

B. Generating Explanation: Scenario 2

Similar to scenario 1, this scenario makes the state of *COT*-2 *true*. This means that the overtaking manoeuvre is cancelled and the vehicle stays in the overtaking lane. As it is similar to scenario 1, the closest situation for the cause of *COT*-2 is also represented by six nodes, *MOT-OVIS*, *MOT-OTRLVO*, *MOT-MinSpaceOT*, *PAV-OVIS*, *PAV-OTRLVO*, and *PAV-MinSpaceOT*. However, in this scenario, two parent nodes of *COT*-2, *PAV-OTRLVO* and *PAV-MinSpaceOT,* hold the *true* state ($P_{PAV-OTRLVO} = True$ and $P_{PAV-MinSpaceOT} = True$) while the other parent nodes hold *false* states. As a consequence, the corresponding description of situation understanding is 'Just passed the overtaken vehicle, but overtaking manoeuvre is cancelled and stay in the overtaking lane because the lead vehicle in the overtaking lane slowed down and the minimum space to go back to departure lane after passing is not met' (see Figure 6-10). To generate this explanation, this method only needs to revisit six nodes of the 39 existing nodes in the *ASAS* model.

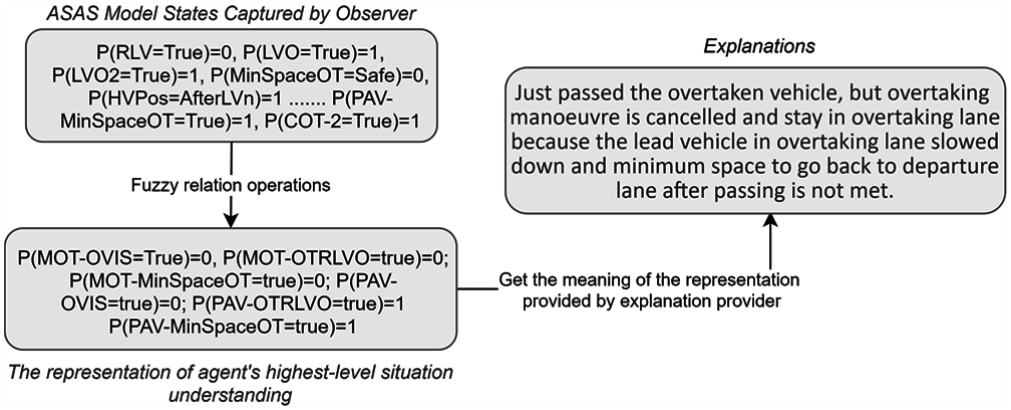


Figure 6-10. Illustration of generating agent's behaviour explanations for scenario 2 of overtaking situation

C. Generating Explanation: Scenario 3

In this scenario, we have $P_{COT-1} = True$. This means that the vehicle will go back to the departure lane. Based on the *ASAS* model, this action projection has three contributory nodes, *SOT-OVIS*, *SOT-OTRLVO*, and *SOT-MinSpaceOT*. For this scenario, these contributory nodes hold the *false* state except *SOT-OVIS*. Thus, referring to the explanation provider, the corresponding description of situation understanding is 'Just entered the overtaking lane, but the overtaking manoeuvre is cancelled and go back to the departure lane behind the overtaken vehicle because the predicted overtaking speed exceeds the road speed limit as the overtaken vehicle increased its speed' (see Figure 6-11). To generate this explanation, this method only needs to revisit three nodes of the 39 existing nodes in the *ASAS* model.

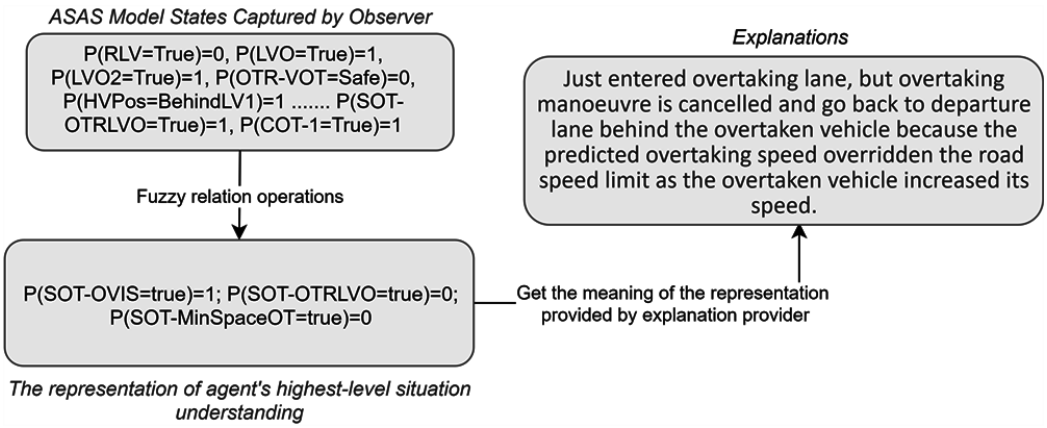


Figure 6-11. Illustration of generating agent's behaviour explanations for scenario 2 of overtaking situation

6.3.2.3. Sensitivity Analysis

To examine the degree of dependencies between the root nodes and leaf nodes in the *ASAS* model, sensitivity analysis is performed on the model and its results are presented in Figure 6-12. In this analysis, this study uses *COT*-1 and *COT*-2 as the target nodes with both nodes holding the *true* state ($P_{COT-1} = true$ and $P_{COT-2} = true$). The top figure shows that there are 6 states of contributory nodes which have a strong effect on triggering $P_{COT-1} = true$. These states are $P(Lane = OTLane) = 1$, $P(TLIM = True) = 1$, $P(HVPos = BehindLV) = 1$, $P(DSS = Yes) = 1$, $P(LV = True) = 1$, and $P(POTR - VOT = Safe) = 1$. When *HVPos* holds 'AfterLV1' and 'AfterLn' states, they have a negative contribution on achieving $P_{COT-1} = true$. Similarly, a negative contribution is also delivered by $P(Lane = EgoLane) = 1$ and $P(Lane = DepartureLane) = 1$.

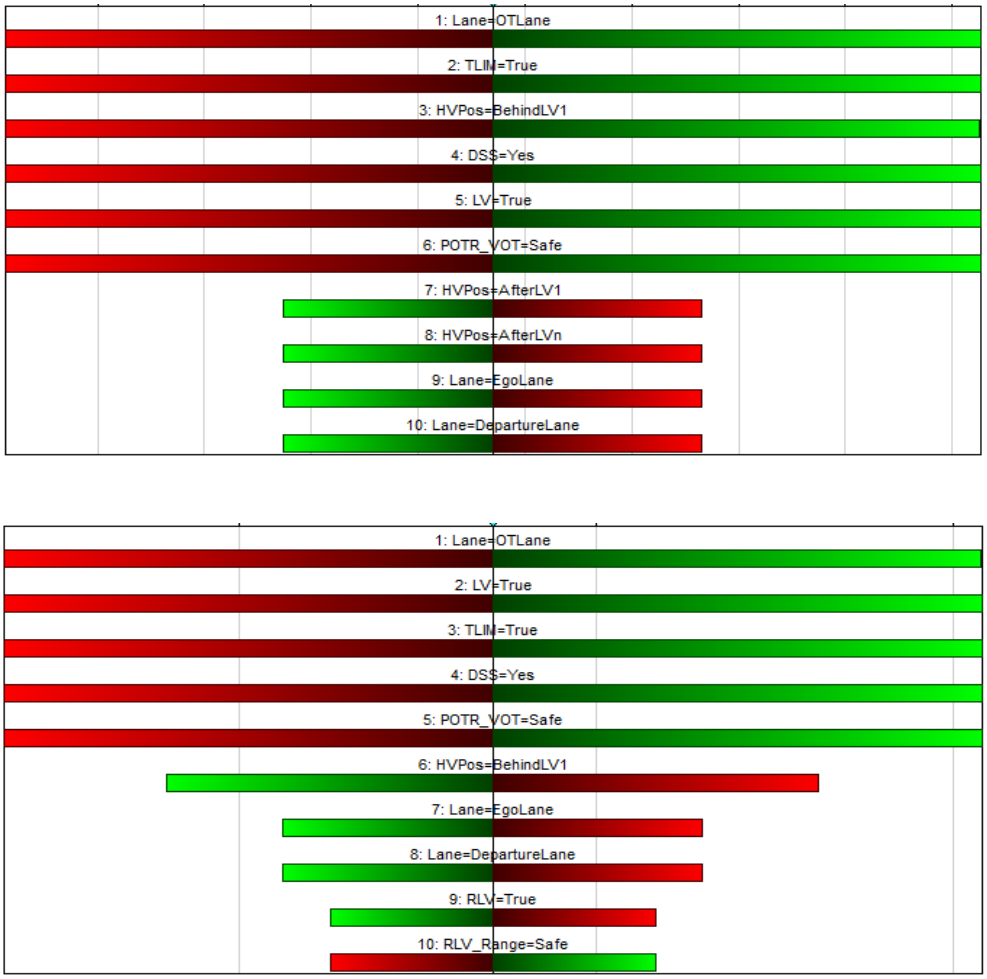


Figure 6-12. Sensitivity Analysis: Target state P(COT-1) = true (top) and Target state P(COT-2)=true (bottom)

Furthermore, the sensitivity analysis also shows that to achieve the target state of $P_{COT-2} = true$ (see the bottom figure), the positive contributions are given by six states, $P(Lane = OTLane) = 1$, $P(LV = True) = 1$, $P(TLIM = True) = 1$, $P(DSS = Yes) = 1$, $P(POTR - VOT = Safe) = 1$, and $P(RLV - Range = True) = 1$. Four states are indicated to be the opposed states to achieve $P_{COT-2} = true$. These four states are $P(HVPos = BehindLV) = 1$, $P(Lane = EgoLane) = 1$, $P(Lane = DepartureLane) = 1$, and $P(RLV = True) = 1$.

6.4. Evaluation of the Proposed ASAS-based Self Explanation Method

In previous sections, the proposed *ASAS*-based method to generate self-explanation is presented. The *ASAS*-based method is proposed with the aim of dealing with the complexities of the autonomy agent's logics. Even though a sensitivity analysis has been performed to validate the sensitiveness of the behaviour network in the *ASAS* model given the observable variables, an evaluation that shows how this proposed method can reduce the complexities in generating behaviour explanations has not been undertaken.

However, providing an evaluation to compare methods which practically generate the same information to support human situation awareness development is not an easy task, particularly if the compared method does not have attributes that can be directly measured such as accuracy. The situation awareness development itself can be measured with a combination of existing situation awareness measurement techniques such as performance measurement and observer ratings where both human and non-human agent situation awareness can be compared to know whether they have same situation understanding given the status of the surrounding salient objects, such as collision risk with an oncoming vehicle in front. Another example of a situation awareness measurement technique is process indices which can be used to measure an individual's focus while they are performing tasks to examine their level of alertness.

Even though the *ASAS*-based method is proposed to support human situation awareness development, its evaluation is less about the performance of generated information to enhance human situation awareness, and it is more about the effectiveness of generating information as it is intended to address the complexity issues during explanation

generation. Therefore, in this chapter, an evaluation framework is developed to provide appropriate comparable measurement variables from behaviour explanation generation methods, process-based methods and the proposed *ASAS*-based methods. After this, the advantages and disadvantages of the *ASAS*-based method are presented.

6.4.1. Comparison of Complexity of the ASAS-based and Process-based Methods

In this section, the *ASAS*-based and process-based methods are compared. This study does not consider a comparison with a component-based method because this method is focused on physical component problems and has a lack of support on goal-based behavioural problems. This evaluation is intended to show how the proposed method (*ASAS*-based) can reduce the complexities of process-based methods in generating behaviour explanations. The implementation of this process-based method can be seen in the Appendix 3.

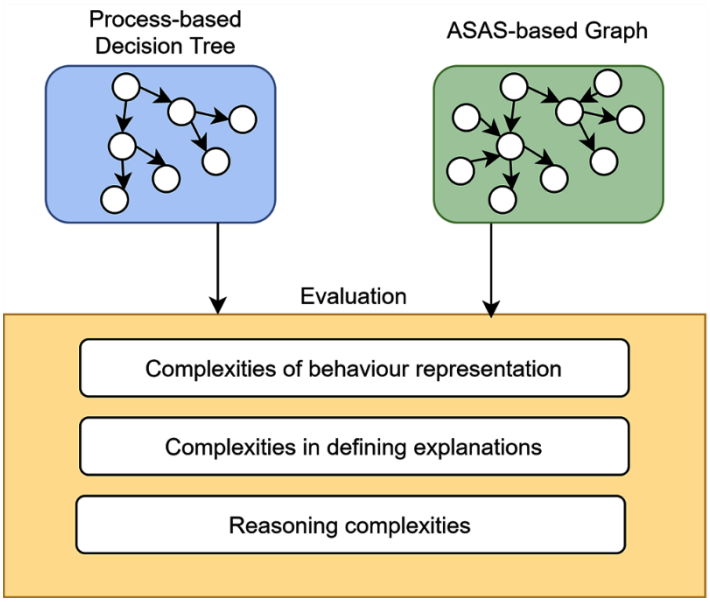


Figure 6-13. Evaluating complexities in generating behaviour explanation

There are three main evaluation contexts as illustrated in Figure 6-13. In the first evaluation part, the complexity of each model to represent non-human agent behaviour is reviewed. As the models in the two compared methods can be considered as a graph, the number of nodes becomes the main variable to indicate the complexity of behaviour representation. Secondly, the complexities in defining explanations are compared based on the variables called depth level and possible paths. Finally, the last evaluation reviews

the complexities in generating behaviour explanations. In this part, in addition to the depth level and possible paths, the number of revisiting nodes in the trace back process is considered as one indicator. At the end of this section, the pros and cons of the *ASAS*-based and process-based methods are presented.

6.4.1.1. Complexities of Behaviour Representation

As the actual behaviour of a non-human agent is driven by a process which consists of algorithms, logic, rules, and knowledge, the process model can be considered as the detailed behaviour representation of this agent. The backbone of this process model is its logic structure (algorithms). Algorithms process input in a certain way by applying determined rules and embedded knowledge to obtain an intended end state of an agent's actuator that regulates the agent's behaviour. For example, two case studies presented in previous chapters show that in the traffic light and overtaking situations, the end state of the process leading the agent behaviours is intended to determine the acceleration/deceleration value even though in the middle of the process, it also sets the steering angle value to perform lane changing in an overtaking situation. As the process model relies on its logics, the number of logic sentences within a process can be used to infer the complexity of this process. Thus, it can reflect the complexity of behaviour representation. Logics in this process model are also referred to as decision points (DP).

Referring to the process model developed for the above two study cases (see Appendix 3), the total number of nodes for the first study case (traffic light situation) is about 99 nodes which are divided into 38 DP nodes and 61 sub-process nodes. The second case study involves 105 nodes which consist of 49 DP nodes and 56 sub-process nodes. However, it should be noted that DP can also exist in sub-process nodes, and it is also possible that a sub-process node contains more than one decision node. By assuming that there are two DPs in each sub-process node, the number of DPs in the first study case will be 160 in total. Moreover, the total of the involved nodes in the process model presented in Table 6-18 only covers the automatic control routine process. This means that other processes within the whole system are not covered.

Table 6-18. Total number of nodes involved in two case studies based on automatic control routine only

Case Study	Process-based		ASAS-based
	DP Node	non-DP Node	Node
Traffic light situation	38	61	22
Overtaking situation	49	56	39

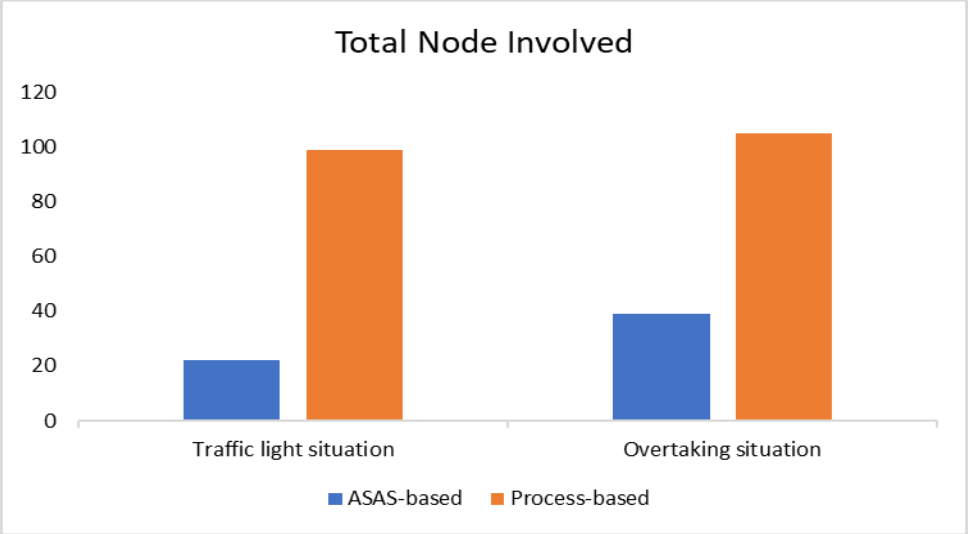


Figure 6-14. Total nodes involved to represent non-human agent's behaviours in each case study

Having less nodes than the process-based model, the *ASAS* model only needs 22 and 39 nodes to reflect an agent's behaviour in the traffic light and overtaking situations, respectively. Unlike the process model, there is no hidden node in the *ASAS* model in both case studies. The *ASAS* model is efficient in representing an agent's behaviour because it focuses on the abstraction level. For example, in the overtaking situation, the abstraction level states that when the space to go back to the departure lane is not safe, the *stay in overtaking lane* manoeuvre will be selected. The process level may translate such abstraction with so many rules to identify a safe space and to operate the vehicle to stay in the overtaking lane. These rules significantly increase the number of DP nodes in the process-based model. The *ASAS* model only needs one node to reflect the action taken which is to stay in the overtaking lane. The comparison of the total nodes involved in the two case studies can be seen in Figure 6-14.

6.4.1.2. Complexities in Defining Explanations

Defining explanations with a lot of nodes involved is not a simple task. As defining an explanation in a process-based method is about describing the meaning of every visited decision path, other variables can be used to measure the complexity in defining explanations in addition to the number of DP nodes. These variables include the depth level of a decision tree and the number of possible paths. The depth level refers to the number of nodes that should undergo a process from receiving input until the last state of the process. To achieve its last state, a process can go through the DP nodes in many different paths. This because a DP node has two states, true and false, and, therefore, after visiting a node, this process can go in two possible directions.

Table 6-19. Depth level and paths in process-based and ASAS-based methods

Case Study	Process-based		ASAS-based	
	Depth level	Possible Paths	Depth Level	Arcs
Traffic light	15-37	1440	3-4	32
Overtaking	11-60	993	3-7	64

Table 6-19 shows that in the first study case, the depth level of the process-based model is within the range of 15-37 while the depth level range in the second study case is 11-60. Within this depth level range, the first value indicates the shortest path of a process since it receives input until the final state of the process. The second value represents the longest path to the end of the process. Furthermore, this table also shows that for the case study on the traffic light situations, the automatic control routine process has at least 1440 possible paths. Regarding the case study on the overtaking situation, there are only 993 possible paths. This number seems to be lower than the first case study, but it could increase when all the DP nodes are presented during the generation of the overtaking plans.

The possible path variable is a critical factor contributing to the complexities of defining explanations. When a process has i.e., 1440 possible paths, in the worst case, the same number of reasons must be provided to explain every possible path. However, some paths might not be explained in detail, and it will depend on the system designer's considerations. Or, a system designer can also choose to generalize some paths by giving

the same explanations when these paths have the same meaning. However, still, when more decision point nodes are involved, defining explanations will become more difficult.

Different from a process-based model, the complexity of defining explanations in the *ASAS* model is reflected in the number of nodes in the model. This is because the process of defining explanations in this model starts from defining the knowledge or situation held by each node in the model. Higher-level nodes, such as intermediate nodes, can hold knowledge or situation summaries from their contributory nodes. Such summaries are called a higher-level of situation understanding. With only 22 and 39 involved nodes in the traffic light and overtaking case studies, respectively, defining explanations become less complex than in a process-based model.

6.4.1.3. Reasoning Complexities

In addition to becoming critical factors in defining explanations, the depth level and possible paths are also essential for reasoning complexity measurement. As both the process-based and *ASAS-based* models are graph-based models, reasoning complexity can be reflected in how many nodes should be revisited or evaluated in order to generate explanations. If the depth level value of a process model is within the range of 11 and 60, this means that the minimum number of revisited nodes to generate an explanation is 11 while the maximum is 60. Table 6-20 presents the number of revisited nodes during explanation generation in both case studies. Using a process-based method, the total number of revisited nodes for the traffic light case are 17, 33, and 18 for scenario 1, scenario 2, and scenario 3 respectively. For the overtaking case, the total number of revisited nodes are 45, 47, and 59 for scenario 1, scenario 2, and scenario 3, respectively. The total number of revisited nodes does not include the hidden DP nodes in the sub-process nodes.

Table 6-20. Total number of revisited nodes to generate behaviour explanations

Method	Traffic light			Overtaking		
	Scenario 1	Scenario 2	Scenario 3	Scenario 1	Scenario 2	Scenario 3
Process-based	17	33	18	45	47	59
ASAS-based	4	3	3	6	6	3

The *ASAS*-based method has a different way of generating reasons to the process-based method even though both are graph-based models. From Table 6-19, it can be seen that the nodes involved in the two case studies are linked with arcs, 32 and 64 arcs for the traffic light and overtaking vehicle study cases, respectively. Moreover, the depth level for the first case study is only 3-4 while the latter case is 3-7 (see Table 6-19). Even though these numbers are significantly lower than the process-based model, once the *ASAS* model receives its input, all arcs will be visited during the process. This is different from the process-based model which only goes to paths driven by logics. As a consequence, when all intermediate and root nodes become the contributory nodes of a leaf node, the worst case for generating explanations for the state of this leaf node in the *ASAS* model will be the necessity to revisit all arcs to examine the states of all contributory nodes.

However, since the *ASAS* model can be considered as a causal network where the higher-level nodes hold knowledge or a situation summary of the lower-level nodes, it will be necessary to revisit the contributory nodes if the state of one node is caused by two or more situations at the higher level. For example, in traffic light case scenario 1, the state of the *KG* node is affected by two situations, yellow-green light (represented by *S1PGU* node) and free ride in segment 2 (represented by *S2P* node). Therefore, the contributory nodes one level down from *KG*, which are *S2P* and *S1PGU*, should be evaluated first. In the case where the cause of the *true* state in node *KG* is coming from *S1PGU*, it should be clarified with its one level down nodes (*S1G* and *S1U*) to examine which light (yellow or green) becomes the contributory factor of the true state in node *S1PGU*. Therefore, the total number of nodes to be evaluated in scenario 1 of the traffic light case study is 4 nodes (*S2P*, *S1PGU*, *S1G*, and *S1U*). However, in the case where it is found that *S2P* is the contributory node of the true state in node KG, tracing back to nodes *S1G* and *S1U* is not necessary.

After finding *S2P* or both *S1G* and *S1U* as the causal factors, the tracing back to other contributory nodes is discontinued because these nodes already represent higher-level situation understanding from their parent nodes without ambiguity. Based on the two case studies, it is recommended that the episodical memory should focus on capturing the states' contributory nodes up to the level where there is no vagueness in relation to the

knowledge or situation they represent. This is significant for behaviour explanation generation using the *ASAS* model.

The revisiting mechanism in the *ASAS* model is similar to the breadth-first search approach (Al-Ajlan 2015). Table 6-20 shows that the total number of revisited nodes for scenario 1 in the overtaking case is six while for scenarios 2 and 3, the numbers are 6 and 3, respectively. Compared to the depth-level for the same case, which is 3-7, the number of revisited nodes is not associated with the depth level as in the process-based method. In the process-based model, when the total revisited nodes is 6, it can be inferred that the revisiting process goes to the same depth level number. However, in scenario 1 of the overtaking case, with the total six revisited nodes, it only goes one level down.

To summarize the differences between the *ASAS*-based method and process-based method in generating behaviour explanations, a process-based method traces back the executed logics while the *ASAS*-based method traces back the nodes in the Bayesian network as the representation of *ASAS*. Using the process-based method, all executed logics should be revisited, and each revisited node should be translated into a meaningful sentence to generate explanations. By assuming that each logic is a node, numerous numbers of nodes will be evaluated in the process-based method to generate behaviour explanations. On the other hand, the *ASAS*-based method revisits fewer nodes as the *ASAS* model is a causal network. Thus, the search space is reduced significantly.

With all these characteristics, we believe that behaviour explanation generation in the *ASAS*-based method is simpler that the process-based method.

6.4.1.4. Pros and Cons of ASAS-based Method in Generating Behaviour Explanation

To sum up the characteristics of the *ASAS*-based method, Table 6-21 presents its advantages and disadvantage derived from our case studies. In relation to the advantages, this method is more persistent than the process-based method. Algorithms determining the autonomy agent's behaviours might be updated regularly to obtain better responses in given situations, therefore, the process model needs to be updated. The *ASAS*-based method relies on situation abstraction to determine a non-human agent's behaviours, therefore, a change to the algorithms does not have a significant effect. In other words,

when the algorithms are updated, the *ASAS* model can still be used to explain a non-human agent's behaviours. To validate this advantage, we made some changes to the logics related to the stopping behaviour, and the results shows that the *ASAS* model is still reliable to represent given situations.

Table 6-21. Advantages and disadvantages of ASAS-based method

Advantages	Disadvantages
- More persistent, less dependent on the process	- Poor in providing detailed states
- Easy to manage	- Lack of capability to explain nested loop
- Speciality in generating behaviour explanations	- In some cases, all nodes must be observed
- More flexibility to express the agent's action projection	- Still relies on process variables to feed the model
- One intermediate node can represent more than one knowledge/situation	- Potential problems exist in synchronizing the activities in process level

Moreover, the *ASAS* model is easy to manage and design. Also, it has more flexibility in expressing the agent's action projection. Recalling the traffic light case study, the actual actions generated at the process level are either decelerating or accelerating. However, in the *ASAS* model, the system designer can deliberately determine more options to represent the action projection such as keep going, adjust speed based on the *LV*. In this way, a human teammate can enhance their comprehension of the machine teammate. Furthermore, the *ASAS*-based method specialises in generating behaviour explanations, particularly over a complex system. This capability is supported by the causal network where each node in this network can be used to represent more than one knowledge/situation.

The disadvantages of our proposed method are obtained from our experience during the method implementation. In this regard, the *ASAS*-based method is unable to present the detailed state of a non-human agent and explain the nested loop of the algorithms. However, the most significant disadvantage of the *ASAS*-based method is that the agent observer might need to observe all the nodes in the *ASAS* model to support explanation generation. In the process-based method, the observer only focuses on the visited nodes. Even though the total number of nodes in the *ASAS* model may still be lower than the total visited nodes in the process-based method, visiting all nodes will incur a space cost in the episodical memory. Furthermore, it is claimed that the *ASAS*-based method is less

dependent on algorithms. The only relation between the algorithms and the *ASAS* model is that the process variables generated by the algorithms supply the values of the root nodes in the *ASAS* model. With this relation, a potential problem may exist, particularly to synchronize the state in the *ASAS* model with the state in the process level.

Furthermore, there are no significant issues found during the implementation of our proposed self-explanation method as the *ASAS* model is fed by variables from the agent's perception states and flags generated by functions and logics. However, from our study cases, we found that extra effort might be needed to generate required flags; for example flags to indicate current vehicle position (i.e., overtaking lane, departure lane, after overtaken vehicle, or before overtaken vehicle).

6.5. Summary

This chapter presents the implementation of the *ASAS*-based methods to generate an autonomy agent's behaviour explanation which is required for coordination in a teaming with supervision in two case studies: achieving the goal *Pass traffic light* and *Overtake vehicles*. This chapter evaluates the performance of the process-based method and the proposed ASAS-based method in a collaborative driving context with these two case studies. The evaluation of the *ASAS*-based method focuses on how this method can be more efficient in generating explanations given the complexity of the process driving the autonomy agent's behaviours. Three complexity measurements are provided which cover the capability to deal with the complexities of behaviour representation, defining explanations, and reasoning complexities. As a result, the *ASAS*-based method can be considered to be more efficient in generating behaviour explanations than the process-based method because the proposed *ASAS*-based method can significantly reduce the search space.

Moreover, this chapter also presents the advantages and disadvantages of the *ASAS*-based method. Being more persistent, easy to manage, and less dependent on the algorithms are its advantages. Moreover, a system designer has more flexibility in defining the agent's action projection. However, the ASAS-based method is unable to provide the detailed states and lacks the capability to explain the nested loop within the algorithm.

Chapter 7 : Driver Distraction Recognition

7.1. Introduction

In Chapter 5, we highlighted that this study selects driver distraction recognition as one feature of an autonomy agent to support the human supervisor in the teaming without supervision in a collaborative driving context. In this teaming, the human driver is in charge. The ability to recognize that a human driver is distracted while driving is necessary to achieve all collaborative driving goals, such as the goals of *Pass traffic lights* and *Overtake vehicles*. This study identifies 10 classifications of driving behaviours: 1) safe driving, 2) texting using right hand, 3) talking on the phone using right hand, 4) texting using left hand, 5) talking on the phone using left hand, 6) operating radio, 7) drinking, 8) reaching behind, 9) doing makeup, 10) talking to passengers.

Previous studies have demonstrated the capability of the image learning method to achieve good accuracy in recognising these ten classifications using sample images from left-steering vehicles. However, as mentioned in Chapter 5, because our sample images were collected from a right-steering vehicle (referred to as the target vehicle), the direct adoption of an existing classifier trained using sample images from left-steering vehicles for driver distraction recognition in the target vehicle results in poor classification accuracy. In this chapter, we propose a new method to resolve this issue by using the transfer learning approach, which means that we take advantage of an existing classifier trained using images from left-steering vehicles (referred to as the source domain) to develop a new classifier for driver distraction recognition in the target vehicle (referred to as the target domain). As discussed in the literature review in chapter 2, transfer learning contains a special transfer learning problem called geometrical relation constraints where some images from the target domain fall into a different class in the source domain when they are augmented. For example, the sample images from the source domain which are classified as ‘talking on the phone using the right hand’ in a left-steering vehicle can have a geometrical relation to the sample images from the target domain which are labeled as ‘talking on the phone using the left hand’ in a right-steering

vehicle. Therefore, in this study, we solve a transfer learning problem with geometrical relation constraints to increase classification accuracy for driver distraction recognition.

In the rest of this chapter, Section 7.2 presents the transfer learning problem statement and Section 7.3 provides the proposed transfer learning approach with directive offline augmentation. Section 7.4 presents the way to implement and evaluate the proposed approach including the baseline methods, dataset, and deep neural network architecture, and hyperparameter settings. The result is presented in Section 7.5 followed by the chapter summary in Section 7.6.

7.2. Transfer Learning Problem Statement

Some definitions related to the transfer learning problem in this study are as follows.

Definition 1. (Source Domain). A source domain is denoted by *SD* consisting of n labeled data samples, $SD = \{(x_1^{SD}, y_1^{SD}), (x_2^{SD}, y_2^{SD}), \ldots, (x_n^{SD}, y_n^{SD})\}$ where n is the number of samples, and x and y represent the data and its label, respectively.

Definition 2. (Target Domain). A target domain is denoted by *TD* consisting of l labeled data samples and m unlabeled data samples, $TD = \{(x_1^{TD}, y_1^{TD}), (x_2^{TD}, y_2^{TD}), \ldots, (x_l^{TD}, y_l^{TD}) \cup (x_1^{TD}, ?), (x_2^{TD}, ?), \ldots, (x_m^{TD}, ?)\}$ where x and y represent the data and its label, respectively, and the question mark represents the unknown label. The size of the samples in TD is much smaller than the size of the samples in SD, so that $l + m \ll n$.

Definition 3. (Classification Tasks). The classification task (*T*) is a task to classify the domain's sample and it can be denoted by $T = \{f(\cdot), R\}$ where $f(\cdot)$ is a predictive function to generate response *R*.

Definition 4. (A Model). A model (*M*) is a classifier trained using the domain's samples to perform the classification task in this domain. The *SD's* and *TD*'s classifiers are denoted by M_{SD} and M_{TD}, respectively.

Definition 5. (Transfer Learning). Given a classification task T_{TD} based on *TD*, the learning process for T_{TD} can get help from *SD* for the classification task T_{SD}. Transfer learning aims to improve the performance of $f_{TD}(\cdot)$ as a predictive function in *TD* by

discovering and transferring the weight or latent knowledge from *SD* and T_{SD} where *SD* $\neq$ *TD* and/or $T_{SD} \neq T_{TD}$ but they are relevant to each other. A transfer learning task can be defined by a tuple (*SD,* T_{SD}*, TD,* T_{TD}*,* $f_{TD}(\cdot)$).

Definition 6. (Deep Transfer Learning). Given a transfer learning task defined by (*SD,* T_{SD}*, TD,* T_{TD}*,* $f_{TD}(\cdot)$), if $f_{TD}(\cdot)$ is a non-linear function reflecting a deep neural network, this task is a deep transfer learning task (Tan et al. 2018).

The transfer learning challenge in this study is how to maintain the accuracy level of an image classification model from SD when it is applied to new images in TD. Based on the idea of semi-supervised transfer learning (SSTL), SD has a rich set of labeled training images, and TD has fewer images than SD. Furthermore, only some of the sample images in TD are labeled and the remaining images are unlabeled. This chapter presents a solution to this challenge in a special situation in which the classification task is to classify the drivers' distraction images into a given number of classes and the drivers' images in the two domains have geometrical relation constraints. The objective is to develop a classifier model with good accuracy for the target domain.

The problem is described as follows:

Given a source domain, $SD = \{(x_1^{SD}, y_1^{SD}), (x_2^{SD}, y_2^{SD}), \ldots., (x_n^{SD}, y_n^{SD})\}$ and a target domain TD, $TD = TD_1 \cup TD_2 = \{(x_1^{TD}, y_1^{TD}), (x_2^{TD}, y_2^{TD}), \ldots., (x_l^{TD}, y_l^{TD}) \cup (x_1^{TD}, ?), (x_2^{TD}, ?), \ldots., (x_m^{TD}, ?)\}$, where x and y are the input data and its label, respectively, TD_1 and TD_2 are subsets of the target domain whose labels are known and unknown, respectively. Then, n, l and m are the number of samples in SD, TD_1, and TD_2 respectively. A classification model M_{SD} is trained to perform T_{SD} using SD. Then the problem is a classification problem that determines the weights of M_{TD} such that M_{TD} can perform T_{TD} using a pre-trained model M_{SD} from an SSTL approach where in SD and TD, geometrical relation constraints exist.

7.3. New Transfer Learning Method with Directive Offline Augmentation

This section presents the new transfer learning method from the SSTL approach based on the pseudo-label method using directive offline augmentation (referred to as Pseudo-DOA). The idea of this method is to build an image classifier in TD using the weights of M_{SD} first and then tune the weights using a new training dataset which is a mixture of a subset of the training dataset from SD (SD_{subset}) and TD. TD needs to be transformed by a new directive offline augmentation (DOA) technique, which handles the geometrical relation constraints between the two domains. A block diagram of this method is depicted in Figure 7-1, which consists of eight modules: source domain (SD), target domain (TD) which includes a small number of labeled data TD_1 and a large number of unlabeled data TD_2, a pretrained model M_{SD} which is obtained with supervised learning using SD, DOA which transforms samples TD, training dataset for M_{TD}, a training process which includes the deep learning network architecture, unlabeled samples handler, trained model M_{TD}, and labelling adjustment which restores the affected labels for TD_2.

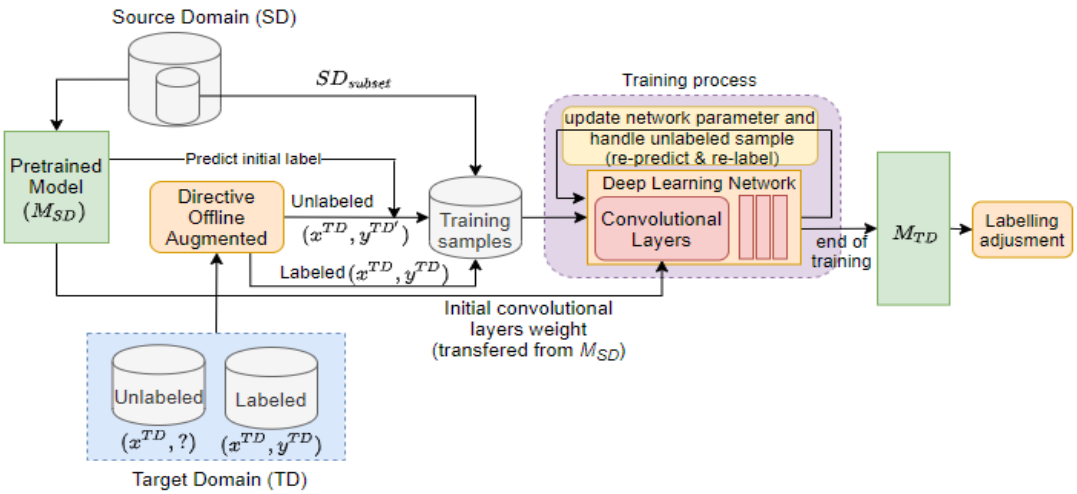


Figure 7-1. The framework of TL with Directive Offline Augmentation

7.3.1. Source Domain

The source domain (SD) contains a large number of sample images and these samples are labeled. SD has two main functions. The first function is to generate the classifier called the pretrained model (M_{SD}) through a supervised learning process. The second function is to provide a few samples (referred to as source domain subset SD_{subset}) in the training

samples for the transfer learning process. This subset plays a significant role because it provides a set of treated image samples from the other domain so that the learning weight can better match the knowledge in M_{SD}.

7.3.2. Pre-trained Model (M_{SD})

As the pre-trained model, M_{SD} is expected to have good performance to classify the images in SD. Without the good performance of M_{SD}, the transfer learning process may not produce a good result for the TD.

7.3.3. Target Domain

The target domain (TD) contains sample images that have geometrical relation constraints to SD, and the size of the samples in TD is significantly fewer than SD. Through the transfer learning process, the classifier for TD is constructed.

7.3.4. Directive Offline Augmentation

The Directive Offline Augmentation (DOA) module is designed to transform image samples from TD to be geometrically equivalent to the corresponding image samples in SD. For example, as shown in Figure 7-2, let's assume that I is an image sample from SD in the form of a set of arrays, $I = \{X, Y, Z\}$. X, Y, and Z represent the value of red, green, and blue (RGB) channels of the image respectively and have $a \times b$ dimensions representing the pixel size of the image. Similarly, now suppose $G = \{X, Y, Z\}$ is an image sample from TD which has a geometrical relationship with I. Consequently, the geometrical position of each element in array I will be related to one in G. Let's consider that the position of $\{X_{(0,0)}, Y_{(0,0)}, Z_{(0,0)}\}$ in I becomes the determinant attribute of a class in SD labeled with *top left*. The position of $\{X_{(0,0)}, Y_{(0,0)}, Z_{(0,0)}\}$ in G is the determinant attribute of a class in TD labeled *bottom right*. When G is transformed in a certain way so that the position of $\{X_{(0,0)}, Y_{(0,0)}, Z_{(0,0)}\}$ is on the top left, the class for the image in G will be equivalent with the one in I.

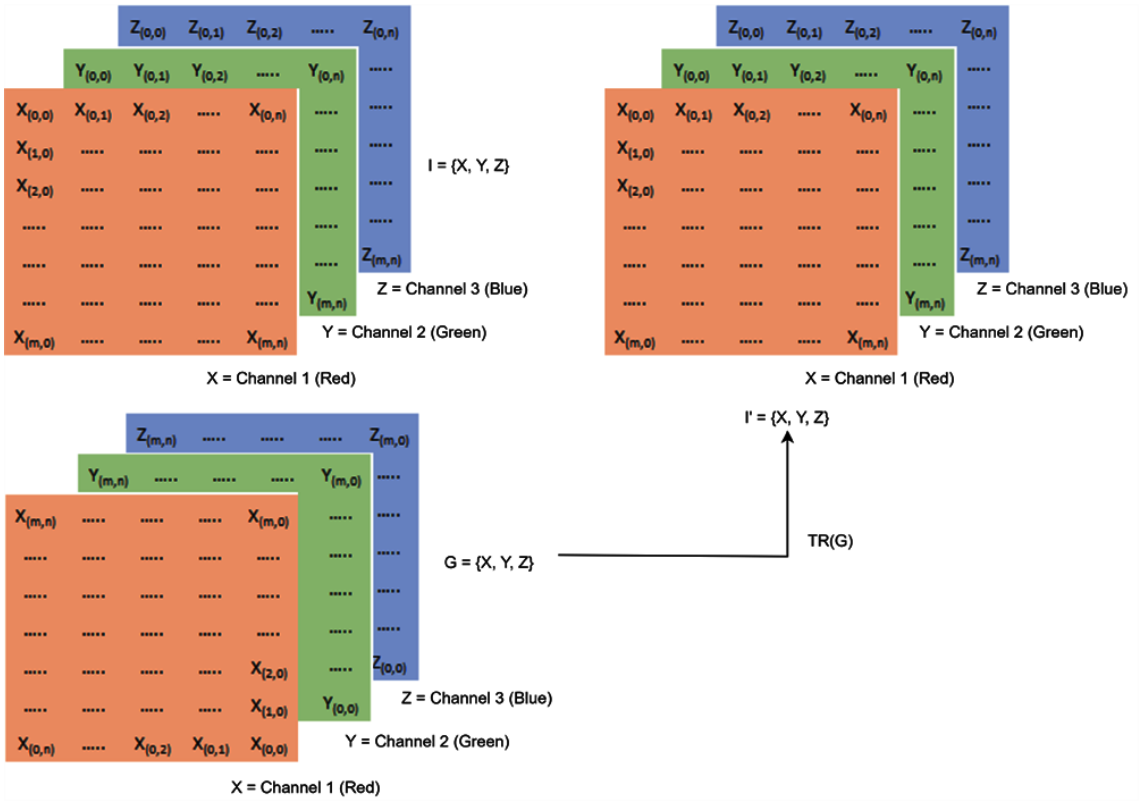


Figure 7-2. Transforming G to have geometrical equivalence with I

As the opposing geometrical relation in I and G is known, it can be represented in a directive transformation function $TR(\cdot)$ to reorder the elements of G to have an identical corresponding geometrical position with I. This can be denoted by $TR(G) \rightarrow I'$ where I' is the new set of arrays for G following the geometrical position of I. In this regard, $TR(\cdot)$ can also be an image augmentation function. With the changes of the element position in G, some labels of the image samples in TD are affected. As the training process for the samples in TD follows the labelling rules in SD, the label *bottom right* in G will be changed to the *top left* after the transformation function is applied. It is also possible that some labels in G will not be affected by the transformation function, depending on whether the geometrical position of the elements in the array are considered as the class attribute or not.

7.3.5. Training Samples

Transfer learning training samples contain three segments. The first segment is the labeled samples from SD_{subset}. The second segment is the labeled samples from TD. The third segment is the unlabeled samples from TD. It should be noted that all samples in TD have undergone the process in DOA. To handle the unlabeled sample, our approach is to

implement the Pseudo Label method from Lee (2013) to prepare the training samples. A pseudo label can be described as a class assignment for unlabeled data as if they were true labels during the training process. These labels are obtained by passing the unlabeled samples to classifier M_{SD} and selecting the label with the maximum predicted probability for each unlabeled sample. This function, then, is denoted as follows:

$$y_i = \begin{cases} 1, & if\ i\ =\ argmax\ (x_i) \\ 0, & otherwise \end{cases} \tag{7-1}$$

As a result, now the unlabeled samples in TD will have fake labels predicted from M_{SD}.

7.3.6. Training Process

To train classifier M_{TD}, handling unlabeled samples is critical so that they can be useful in the training process. Because the total number of labeled and unlabeled samples is quite different, their balance is critical to maintain network performance. Therefore, a coefficient ($\alpha(t)$) is set in order to balance them. Moreover, determining a proper $\alpha(t)$ schedule is also critical as too high a value can lead to disturbance for labeled samples during the training process. In contrast, when the $\alpha(t)$ is too small, it does not get any benefit from the unlabeled samples. In this chapter, $\alpha(t)$ is adopted from the Pseudo Label method and depends on the iteration number of the current epoch and is characterized as follows:

$$\alpha(t) = \begin{cases} 0, & t < 0 \\ \frac{t}{70} \times 3, & t \geq 70 \\ 3, & otherwise \end{cases} \tag{7-2}$$

The pre-determined constant values in equation (7-2) helps the $\alpha(t)$ to maintain a small value. The $\alpha(t)$, then, is used to calculate the overall loss ($l^{totalce}$) by applying it to the loss function for labeled samples l^{ce}. The overall loss is denoted in equation (7-3) where n' and y' represent the number of mini-batches in the unlabeled sample and their predicted labels respectively.

$$l^{totalce} = l^{ce} + \alpha(t) \times \frac{1}{n'} \sum_{i=1}^{n'} log\ p_{model}\ [y'_i \in C_{yi}] \tag{7-3}$$

The labels of the unlabeled samples, then, will be updated during the training process using the in-training-process classifier M_{TD} considering the overall loss. Training in pseudo labels can be considered as training under entropy regularization. This regularization is a way to obtain a benefit from unlabeled data with maximum posteriori estimation by minimizing the conditional entropy of the class probability for unlabeled data. Entropy regularization (H) used in the Pseudo Label method can be denoted as follows:

$$H(y|x) = -\frac{1}{n} \sum_{m=1}^{n'} \sum_{i=1}^{C} P(y_i^m = 1 \mid x'^m)\ log\ P(y_i^m = 1 \mid x'^m) \tag{7-4}$$

where n' is the number of unlabeled data, C is the number of classes, y_i^m is the unknown label of $m\text{-}th$ labeled sample, x'^m is the input vector of $m\text{-}th$ unlabeled sample. This entropy can also be viewed as a class overlap measurement. When the class overlap decreases, lower data point density will be obtained at the decision boundary. To maximize the posterior distribution, a parameter called maximizer ($C(\theta, \lambda)$) can be denoted as follows:

$$C(\theta, \lambda) = \sum_{m=1}^{n} log\ P(y^m \mid x^m;\ \theta) - \lambda H(y \mid x'; \theta) \tag{7-5}$$

where n is the number of labeled data, x^m is the $m\text{-}th$ labeled of sample, λ is a coefficient balancing two terms where the first term is used to maximize the conditional log-likelihood of labeled data, and the second term is used to minimize the entropy of the unlabeled data. In this way, it is expected that a better generalization performance will be obtained using unlabeled data.

7.3.7. Target Domain Model (M_{TD})

It should be noted that the changes to the labels are implemented during the training process to obtain M_{TD}. After the classifier for TD is acquired, it needs to be adjusted to switch the labels affected by the transformation function back into their original labels. Let's assume that TD has q classes with label $lbl = \{0, 1, \dots, q-1\}$ and then, the softmax layer of M_{TD} generates output $\emptyset_{softmax} = \{o_1, o_2, \dots, o_q\}$ where o_i is the weight for every class. In this output, its element indexes follow the order of the element indexes of lbl. Now, to obtain the label of the input image, an argmax function is applied on $\emptyset_{softmax}$ that can be denoted as follows:

$$\underset{\emptyset_{softmax} \in TD}{argmax}\ f(\emptyset_{softmax}) = \{\emptyset_{softmax} | f(\emptyset_{softmax}) \geq f(lbl), \forall lbl \in TD\} \tag{7-6}$$

7.3.8. Labelling Adjustment

Once the label is obtained from the argmax function, a mapping function examines this label to determine whether it is affected by $TR(\cdot)$ or not. If so, the label will be replaced with the appropriate one.

7.4. Implementation and Evaluation of Pseudo-DOA Method

7.4.1. Baseline Methods

For evaluation purposes, this study compares the classification accuracy of the proposed transfer learning method (Pseudo-DOA) with three baseline methods, namely the original Pseudo Label method, the deep adaptation network (DAN), and the join adaptation network (JAN). DAN is a transfer learning method which can be used in either unsupervised or semi-supervised settings, introduced by Long et al. (2015). Similar to DAN, the JAN method can be implemented in semi-supervised or unsupervised transfer learning settings. This method was introduced by Long et al. (2017) and is based on the Joint Maximum Mean Discrepancy (JMMD) technique. These baseline methods are selected for their highly accurate performance in the classification of images generated in the SSTL environment. DAN and JAN also use the online augmentation technique in their learning processes. In the case of image transfer learning without geometrical relation constraints, the online augmentation technique is helpful in improving the classification performance.

7.4.2. Transfer Learning Dataset

This study uses two datasets as SD and TD for transfer learning purposes; each dataset contains a total of 10 classes of driver behaviours (see Figure 7-3). The samples in SD are from the State Farm Distracted Driver Detection (SFDDD) dataset obtained from Kaggle (2016). This dataset is taken from left-steering vehicles with 22424 images for all classes. The samples in TD are from a dataset collected by the researcher using a camera attached within a right-steering vehicle with the total number of images being 7093 (see Table 7-1). Each domain has a normal distribution throughout its classes.

Figure 7-3. Ten classes of driver behaviours (left side: source domain, right side: target domain)

Table 7-1. The number of target domain datasets and SD classifier performance

Class labels	Source Domain	Target Domain
C0	2489	711
C1	2267	719
C2	2317	715
C3	2346	699
C4	2326	703
C5	2312	706
C6	2325	713
C7	2002	704
C8	1911	707
C9	2129	716
∑ samples	22424	7093

To construct a new classifier for TD, some labeled data from SD (eg.,100 samples from each category were taken as the training dataset and 30 samples from each category were taken as the test dataset) are mixed with the TD dataset. Some of the TD datasets are labeled (TD_1), and the rest remains unlabeled (TD_2). The training iteration is set to three times and each iteration increases the number of labeled samples from each category in TD (n) starting from 100, 300, and the last iteration uses 400 labeled images from each category (see Table 7-2). This number of labeled samples, then, becomes the number of sample images in the training dataset. For the test dataset, it takes 0.33% of the labeled samples. The implementation of the baseline methods (Pseudo, DAN, and JAN) and Pseudo-DOA in the transfer learning setting will use the dataset division as presented in this table. This division is intended to highlight the effect of geometrical relation constraints.

Table 7-2. Number of labeled images in TD for every n samples in the training process

Class	Number of Samples	Number of labeled images in TD (n)					
		n=100		n=300		n=400	
		Train	Test	Train	Test	Train	Test
C0	711	100	33	300	99	400	132
C1	719	100	33	300	99	400	132
C2	715	100	33	300	99	400	132
C3	699	100	33	300	99	400	132
C4	703	100	33	300	99	400	132
C5	706	100	33	300	99	400	132
C6	713	100	33	300	99	400	132
C7	704	100	33	300	99	400	132
C8	707	100	33	300	99	400	132
C9	716	100	33	300	99	400	132
Sub-Total		1000	330	3000	990	4000	1320
Total	7093	1330		3990		5320	
Percentage	100%	19%		56%		75%	

7.4.3. Deep Neural Network Architectures

To implement the selected methods, two deep neural network architectures, VGG-16 and AlexNet, are used. The VGG-16 architecture was used in the implementation of the Pseudo Label and the Pseudo-DOA methods to train both the SD and TD classifiers, while AlexNet was used for the DAN and JAN methods. VGG-16 (see Figure 7-4) has the input

of the convolutional layer which is a tensor of dimension 128 × 128 × 3 representing the RGB image of size 128 × 128. For every convolutional layer, the stride is set to 1 pixel and multiple kernel-sized filters of size 3 × 3 are used, which is the minimal size to detect up/down or left/right. Furthermore, the spatial resolution is 1-pixel fixed for 3 × 3 convolutional layers. Five max-pooling layers are used to perform spatial pooling over a 2 × 2 pixel window with the stride set to 2.

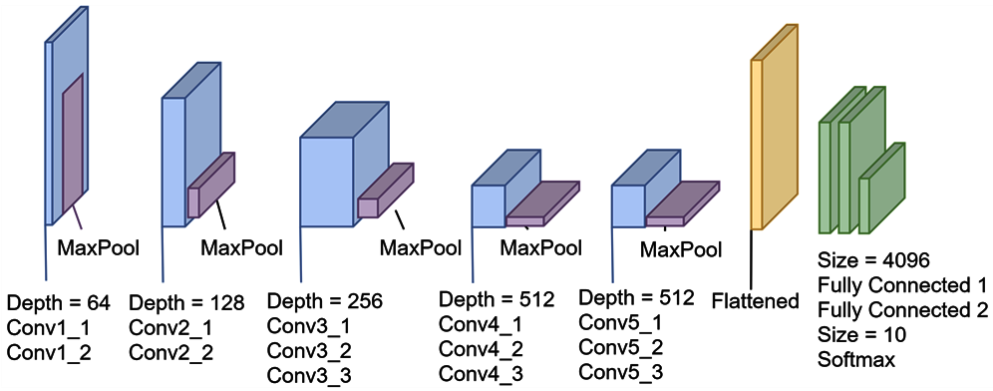


Figure 7-4. The macro architecture of the VGG-16 Model

After the output of the convolutional network layers are flattened, three fully connected (FC) layers are added in which the first two layers have 4096 channels and the last one (also called the softmax layer) has 10 channels. All the hidden layers apply an activation function called the Rectified Linear Unit (ReLU). Mathematically, ReLU (R) for input x is defined as:

$$R(x) = max(0, x) \tag{7-7}$$

The softmax function is equipped in the last layer which turns the logits scores into probabilities that sum to one. The output values of this function are in the range [0,1] where the total sums to 1. Intuitively, for each value (x_i) in an input vector, the softmax value equals the exponent of x_i divided by a sum of the exponents of all inputs. This is denoted as follows:

$$Softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{7-8}$$

A summary of the VGG-16 network is given in Table 7-3.

Table 7-3. VGG-16 network summary

Layer	Output Shape	Parameter numbers
conv2d_1	(None, 128, 128, 64)	1.792
conv2d_2	(None, 128, 128, 64)	36.928
max_pooling2d_1	(None, 64, 64, 64)	0
conv2d_3	(None, 64, 64, 128)	73.856
conv2d_4	(None, 64, 64, 128)	147.584
max_pooling2d_2	(None, 32, 32, 128)	0
conv2d_5	(None, 32, 32, 256)	295.168
conv2d_6	(None, 32, 32, 256)	590.080
conv2d_7	(None, 32, 32, 256)	590.080
max_pooling2d_3	(None, 16, 16, 256)	0
conv2d_8	(None, 16, 16, 512)	1.180.160
conv2d_9	(None, 16, 16, 512)	2.359.808
conv2d_10	(None, 16, 16, 512)	2.359.808
max_pooling2d_4	(None, 8, 8, 512)	0
conv2d_11	(None, 8, 8, 512)	2.359.808
conv2d_12	(None, 8, 8, 512)	2.359.808
conv2d_13	(None, 8, 8, 512)	2.359.808
max_pooling2d_5	(None, 4, 4, 512)	0
flatten_1	(None, 8192)	0
dense_1 (Fully connected)	(None, 4096)	33.558.528
dense_2 (Fully connected)	(None, 4096)	16.781.312
dense_3 (Softmax)	(None, 10)	40.970
Total params:	65.095.498	
Trainable params:	65.095.498	
Non-trainable params:	0	

AlexNet is presented in Figure 7-5 and contains eight layers with the weights, 5 layers being convolutional layers and the other three layers being fully connected (FC) layers. The output of the last FC layer (the softmax layer) in AlexNet can cover up to 1000 class labels. The kernels of the second, fourth, and fifth convolutional layers are only connected to those kernel maps in the previous layer, while the third layer's kernel is connected to all kernel maps in the second layer. The neurons in the FC layers are connected to all the previous layers' neurons. The first and the second convolutional layers are followed by the response-normalization layers. The max-pooling layers follow the response-normalization layers and the fifth convolutional layer. ReLU non-linearity is applied to every convolutional and FC layer's output.

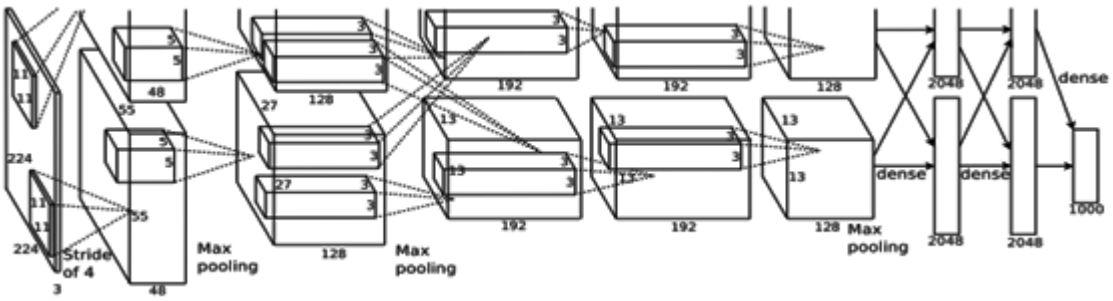


Figure 7-5. AlexNet

The input layer of AlexNet filters 224 × 224 × 3 input images with 96 kernels of size 11 × 11 × 3 with a stride size of 4 pixels. The second convolutional layer receives the output of the input layer and filters it using 256 kernels of size 5 × 5 × 48. The third to fifth convolutional layers connect each other without any normalization layers or pooling. The third convolutional layer applies 384 kernels of size 3 × 3 × 256, and they connect to the pooling and normalization layer which exist in the second convolutional layer. Furthermore, the fourth and the fifth convolutional layers have 384 and 256 kernels respectively, and the size of both kernels is 3 × 3 × 192. Lastly, the FC layers have 4096 neurons each and in total, the AlexNet architecture has 60 million parameters.

7.4.4. Setting Hyperparameters

Hyperparameters are the variables to govern the overall training process and they are divided into two groups, the hyperparameters to train the SD model and TD model. In Table 7-4, the hyperparameters for the proposed method and baseline methods are presented, and each hyperparameter value is obtained by referring to the setting used in the original study of each method or the default value of the implementation backend. To train the *TD* classifier, the Pseudo Label and Pseudo-DOA methods use the same *SD* classifier which is trained under the same hyperparameter preferences. Hyperparameters to train the *TD* classifier in these two methods are also the same even though they are different from the ones used to train the *SD*. For DAN and JAN which use the AlexNet architecture, all hyperparameters to train *SD* and *TD* classifiers have the same preference. However, the *SD* classifiers for DAN and JAN are trained separately.

Table 7-4. Hyperparameters for each implemented method

Hyperparameters	VGG-16 Architecture			Alexnet Architecture			
	SD	TD	TD	SD	TD	SD	TD
	Pseudo label and Pseudo-DOA	Pseudo label	Pseudo-DOA	DAN	DAN	JAN	JAN
Learning rate	1.00E-03	0.001	0.001	1	1	1	1
Initial learning rate	-	-	-	0.0003	0.0003	0.0003	0.0003
Momentum	0.9	-	-	0.9	0.9	0.9	0.9
Decay	1.00E-06	-	-	0.0005	0.0005	0.0005	0.0005
Optimizer	SGD	Adam	Adam	SGD	SGD	SGD	SGD
Early Stopping	Accuracy-based; patience=5	-	-				
Gamma	-	-	-	0.0003	0.0003	0.0003	0.0003
Power	-	-	-	0.75	0.75	0.75	0.75
Beta_1	-	0.9	0.9	-	-	-	-
Beta_2	-	0.999	0.999	-	-	-	-
Epsilon	-	1.00E-07	1.00E-07	-	-	-	-
Batch size	32	400	400	-	-	-	-
Epochs	50	50	50	50	50	50	50
Validation step	200	16	16	-	-	-	-
Step per epoch	800	4	4	500	500	500	500
Online image augmentation	shear range = 0.2; zoom range = 0.2; horizontal flip = true	No	No	crop; random horizontal flip	crop; random horizontal flip	crop; random horizontal flip	crop; random horizontal flip

The VGG-16 deep neural network architecture in this study is implemented using Keras API with tensorflow as the backend. In contrast, the AlexNet for DAN and JAN uses PyTorch as the backend. From the hyperparameters, the learning rate is set to a reasonable value to avoid an excessive training time per epoch. Momentum and decay are the common parameters for stochastic gradient descent (SGD). While momentum accumulates the gradient of the past steps, decay is an additional term for a network weight update rule that causes the weights to exponentially decay to zero. Decay can also be considered as a regularization technique to avoid overfitting. In addition to using a decay parameter, the early stopping and image augmentation parts also contribute to preventing overfitting. As the Adam optimizer is used, other hyperparameters such as

beta_1, beta_2, and epsilon must be configured as well. Beta_1 and beta_2 values can be described as a constant float tensor that contributes to estimate the exponential decay rate for the first and second moments, respectively. Epsilon can be described as a small constant to maintain numerical stability. In this training process, the values of beta_1, beta_2, and epsilon are set by referring to the Keras API default value.

7.4.5. Source Domain Models

We used three SD classifiers. One classifier (M_{SD1}) is used by the pseudo label and the Pseudo-DOA method and the other two classifiers are for DAN (M_{SD2}) and JAN (M_{SD3}). These three SD classifiers were trained using the SFDDD dataset which is divided into 75% for the training dataset and 25% for the test dataset. As the results, M_{SD1} achieves 98 % accuracy on the SFDDD dataset while M_{SD2} and M_{SD3} achieve 81% and 80.9% accuracy respectively.

Table 7-5. Confusion matrix of the MSD1 classifier on the SFDDD test dataset

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9	accuracy
c0	616	0	0	2	2	2	0	0	0	0	0.9903
c1	0	566	0	0	0	0	0	0	0	0	1
c2	0	0	574	0	0	0	1	1	3	0	0.913
c3	0	0	1	583	1	0	0	0	1	0	0.9948
c4	3	0	1	2	572	0	1	0	1	1	0.9845
c5	3	0	0	0	1	574	0	0	0	0	0.993
c6	0	0	0	0	0	0	580	0	1	0	0.998
c7	0	0	0	0	0	0	0	498	2	0	0.996
c8	4	0	2	1	1	0	2	2	461	4	0.9664
c9	4	1	1	0	0	1	0	1	4	520	0.9774
										Average	0.98134

The confusion matrix of M_{SD1} is presented in Table 7-5, where the safe driving class (C0) obtains 99.03% accuracy. From 622 sample images, 6 images fall into different categories, C3, C4, and C5 (two images for each category). The highest accuracy (100%) is obtained from C1 where all 566 images are accurately classified. The other categories that achieved more that 99% accuracy are C3, C5, C6, and C7. For C4, C8, and C9, the classifier performance is only around 96-98%. Of all the categories above, the poorest performance is achieved by C2 with only 91.3% accuracy (see Table 7-5).

When the M_{SD1} was used directly to classify images in the target domain dataset, the classification accuracy is only 8.1%, which is very poor. Based on the confusion matrix in Table 7-6, most images from the TD dataset fall into the C5 class. This table also shows that the SD classifier failed to assign correct labels to the images in C0 and C9.The best accuracy was achieved by C5 which has 62% accuracy.

Table 7-6. Confusion matrix of MSD1 classifier on TD dataset

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9	accuracy
c0	0	42	10	3	32	539	57	26	2	0	0
c1	0	40	16	4	125	409	25	99	1	0	0.05
c2	0	35	27	55	93	402	13	89	1	0	0.0377
c3	0	27	9	3	53	544	21	42	0	0	0.004
c4	0	20	5	9	21	628	6	14	0	0	0.0298
c5	0	10	18	0	44	440	34	159	1	0	0.623
c6	0	18	3	13	207	401	13	56	2	0	0.018
c7	0	122	22	26	114	355	32	32	1	0	0.045
c8	0	28	12	6	24	589	14	33	1	0	0.0014
c9	0	29	3	7	28	448	66	85	50	0	0
										Average	0.08089

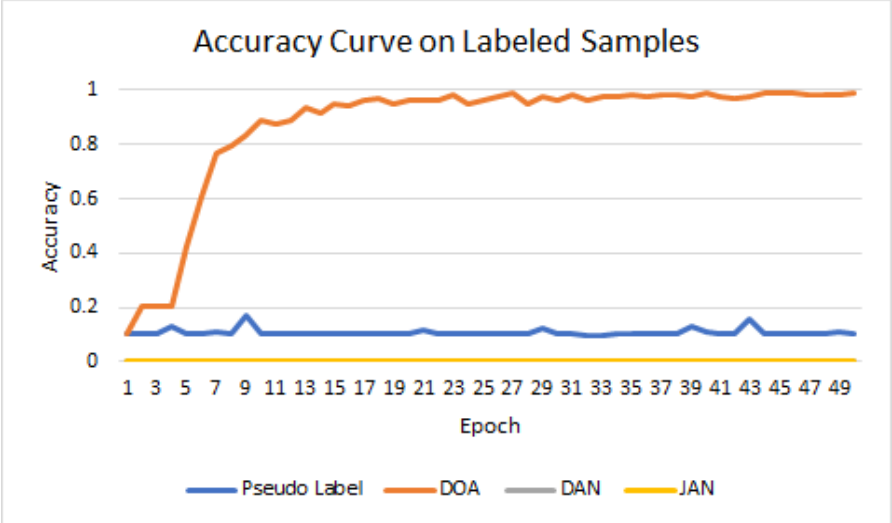
7.5. Experiment Comparisons

After the new transfer learning method was implemented along with the three baseline methods, we run a set of experiments on transfer learning with various amounts of labeled data in *TD*. From these experiments, it is noticed that when *n* (the number of labeled samples in *TD*) is set to 100 and 300, the models M_{TD} generated by all the implemented methods have a very poor classification performance on both labeled and unlabeled data (see Table 7-7). Pseudo-DOA and JAN provide zero correct answers when predicting all the labeled and unlabeled data in *TD*. The Pseudo Label and Pseudo-DOA method achieve around 10% accuracy for the given labeled and unlabeled data in the target domain. The lower accuracy achieved by DAN and JAN might be caused by the implementation of online image augmentation during training. Fake labels generated for unlabeled data are the primary factor in the poor classification performance of Pseudo-DOA and Pseudo Label.

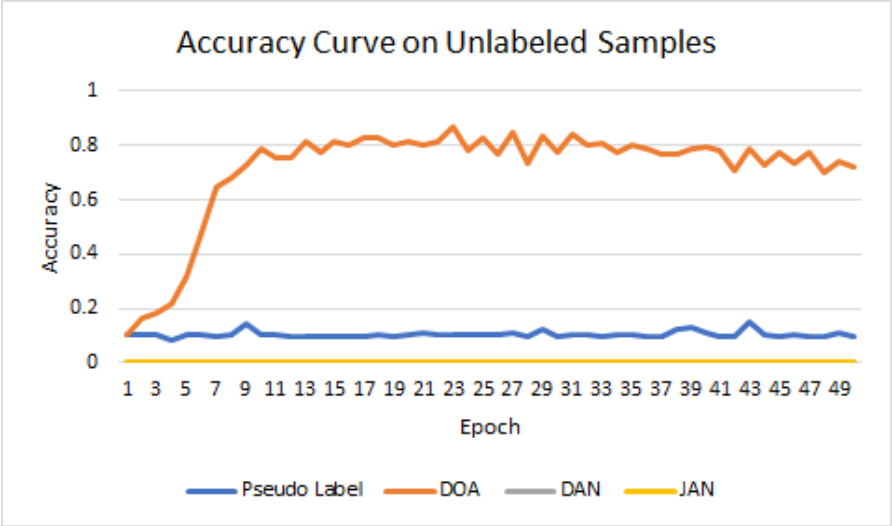
Table 7-7. Classification performance of MTD generated by the implemented methods for each n labeled samples

Method Classifier	Accuracy on TD (n=100)		Accuracy on TD (n=300)		Accuracy on TD (n=400)	
	Labeled	Unlabeled	Labeled	Unlabeled	Labeled	Unlabeled
Pseudo label	10%	9.7%	10.3%	10.2%	10%	9.7%
Pseudo-DOA	10%	10.1%	10%	9.8%	99%	71.8%
DAN	0	0	0	0	0	0
JAN	0	0	0	0	0	0

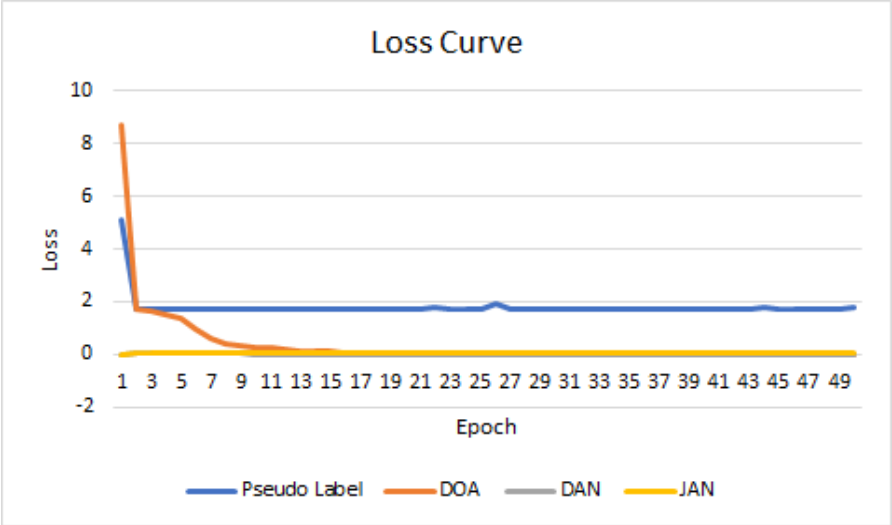
The effect of implementing offline augmentation in the Pseudo-DOA method can be seen in n = 400. While the other methods continue to demonstrate poor classification performance, the accuracy of Pseudo-DOA significantly increases, achieving 99% and 71.8% accuracy for the labeled and unlabeled samples, respectively. This highlights that geometrical relation issues should be carefully managed during the learning between two datasets. The graphical comparison of classification performance in n = 400 can also be seen in Figure 7-6a (accuracy for labeled images) and Figure 7-6b (accuracy for unlabeled images).



(a)



(b)



(c)

Figure 7-6. Accuracy and loss curve comparison given n=400 and 50 epochs

Another indicator used to evaluate the training process is the decrement of loss values during training. As presented in Figure 7-6c, even though the proposed approach shows the highest loss value at first, it significantly drops after a few epochs and in the end, its value is 0.0052. The loss values of the Pseudo Label method tend to be stable at a value of around 1.7 from the second epoch which indicates that it faces difficulties in setting classification boundaries on the given datasets. Even though it may have lower values, the loss variable of DAN and JAN which ended with 0.0024 and 0.0328, respectively, always stay around the zero (0). For DAN, its lost is around the range of -0.0126 to 0.0511, and JAN it is around 0 to 0.0533. Therefore, they tend to exhibit stable behaviour.

Furthermore, to visualize the classification boundaries, Figure 7-7 shows the distribution of the learned features using t-SNE. In this visualization, only the Pseudo Label method and the proposed approach are compared. Figure 7-7a and Figure 7-7b indicates the distribution of the Pseudo Label on labeled and unlabeled samples, respectively. Similarly, for the proposed approach, the distribution on labeled and unlabeled samples is presented in Figure 7-7c and Figure 7-7d, respectively. This suggests that the directive offline augmentation has a significant impact on the production of discriminative features when samples in the two domains have opposing geometrical relations.

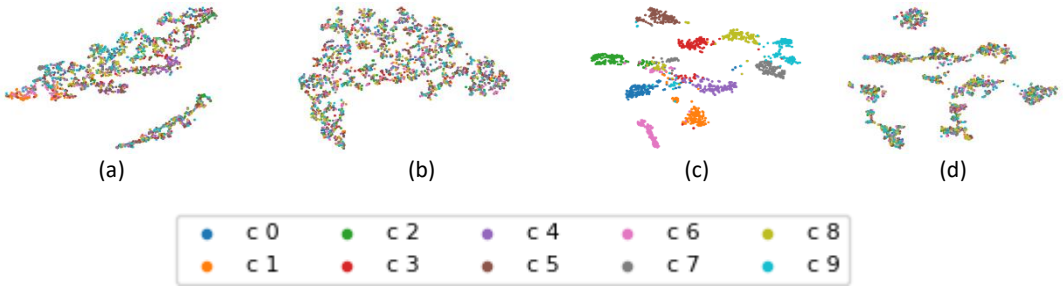
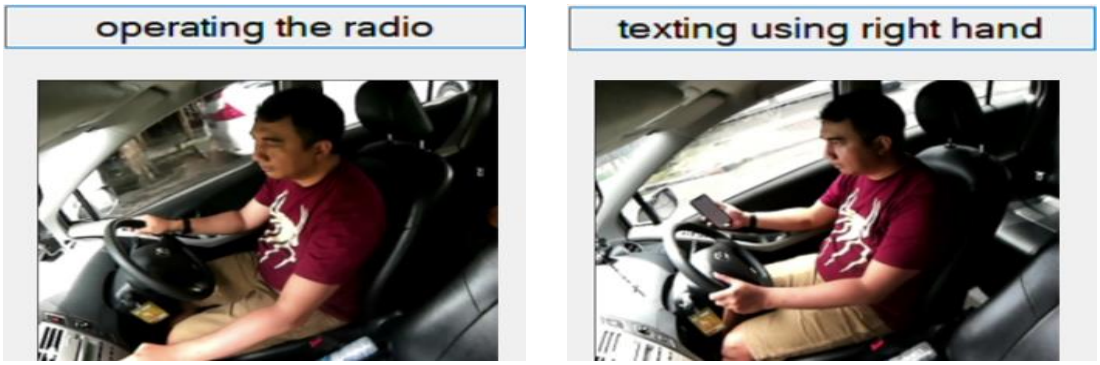


Figure 7-7. Visualization with t-SNE for Pseudo Label for labeled data (a) and unlabeled data (b); and the proposed method for labeled data (c) and unlabeled data (d)

7.6. Showcase of the agent's ability for Driver Distraction Recognition

This section demonstrates the usage of the driver distraction classifier as a feature of the agent as part of its coordination ability in teaming without supervision in the collaborative driving context. There are ten labels of the distracted driver: 1) safe driving, 2) texting using right hand, 3) calling using right hand, 4) texting using left hand, 5) calling using left hand, 6) operating radio, 7) drinking, 8) reaching behind, 9) doing makeup, 10) talking to passenger.



(a) (b)

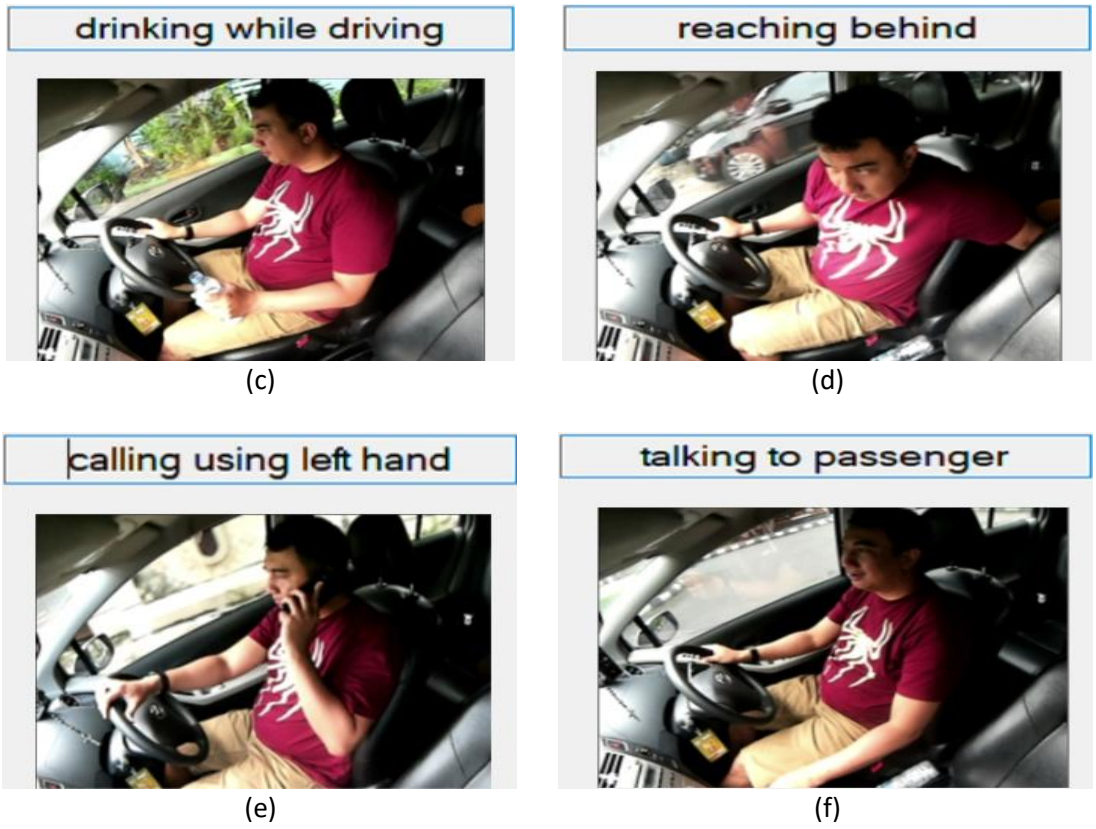


Figure 7-8. Demonstration of driver distraction classifier

As presented in Figure 7-8, when the video frame shows an image of the human driver operating the radio, the agent detects that the driver is distracted and responds with an associated label "turning on the radio" as a message to the driver (see Figure 7-8a). When the video frame shows another image in Figure 7-8b, the agent detects that the driver is distracted and responds with an associated label 'texting using right hand' as a message to the driver. In Figure 7-8c, the video frame presents that the human driver is holding a bottle of water, and the agent detects that the driver is distracted and responds with an associated label "Drinking while driving" as a message to the driver. The human driver in Figure 7-8d is trying to look for something behind them, and the classifier recognizes this activity and presents the label 'reaching behind' as a message to the driver. In Figure 7-8e, the driver is using a mobile phone and the classifier detects this action and responds to the driver with a message "calling using left hand". In Figure 7-8f, the frame shows the driver is talking to a passenger and the agent detects this action and responds to the driver with a message "talking to passenger". From this showcase, it can be seen that the agent can recognise the driver's distraction types correctly so this can be an effective assistant to the human driver.

7.7. Summary

In this chapter, we present a new transfer learning method with directive offline augmentation called Pseudo-DOA to realize an example of the autonomy agent's ability to assist for a teaming without coordination in SSA-based HAT. In this example, the sample images in both SD and TD have geometrical relation constraints, which results in the poor performance of the common transfer learning approach. The new method addresses this geometrical relation constraints problem effectively with directive offline augmentation (DOA).

DOA is different from online augmentation even though it has the same augmentation basis. In nature, online augmentation produces additional batches during a training process containing augmented images. DOA replaces all images in the dataset with new ones which have been transformed based on the parameters provided in the directive information. Therefore, the number of samples will be increased in online augmentation but not in DOA. A comparison with other methods indicates that transfer learning with DOA can produce a more accurate classifier when geometrical relation constraints exist.

This new transfer learning method enables the agent to detect when a driver is distracted when driving, caused by secondary tasks such as using a mobile phone, putting on make up, or operating the in-vehicle radio. This case study also shows that in SSA-based HAT, an autonomy agent can support and maintain the situation awareness of its human teammates when they perform their tasks. Without such a support, the role of an autonomy agent in a team can be considered as a passive team member which cannot provide any feedback to its human teammate. From the team theory perspective, such a passive team member can lead to poor communication and coordination within the team. As in our case study shows that the autonomy agent can be an active team member, it can be inferred that overall human autonomy teaming performance can be enhanced.

Chapter 8 : Time-Constraint-Driven Transparency Model

8.1. Introduction

Once we proposed the methods to enable an autonomy agent to have a self-explanation ability as detailed in Chapter 6, we consider the question of how to make the agent's behaviours transparent to the human driver in collaborative driving. Referring to the literature review in Section 2.5.1 of Chapter 2, there are various recommendations regarding the kind of information that should be presented to a human teammate to ensure the autonomy agent's transparency. However, as the duration of a situation may vary, not all information about the agent's behaviours can be well-absorbed by the human teammate to understand the autonomy agent's behaviours during the live period of a situation. Hence, in this chapter, we propose a new transparency model for the autonomy agent's behaviours in collaborative driving by introducing a time-constraint to reveal what the agent is doing and why. This method is referred to as time-constraint-driven transparency (TCDT) model. Based on this TCDT model, we present two simulation cases to demonstrate the self-explanation ability of the agent in collaborative driving using a CARLA simulator. The first case presents the autonomy agent's coordination ability in teaming with supervision using an example which is to achieve its goal of *Pass traffic lights*. The other case is to achieve its goal of *Overtake vehicles*.

This chapter is structured as follows. Section 8.2 presents the TCDT model to guide the transparency presentation of the agent's behaviours to a human teammate. Section 8.3 presents the simulation cases in achieving the goals of *Pass traffic lights* and *Overtake vehicles*. Lastly, Section 8.4 summarises the chapter.

8.2. Time-Constraint-Driven Transparency Model

This section presents the time-constraint-driven transparency (TCDT) model. The formal description of the TCDT model is presented first and then followed by the detailed explanation of each TCDT group in the collaborative driving context.

8.2.1. Transparency Themes

In order to make an agent's behaviours transparent to its human teammate in a given situation, we first define 11 transparency themes of information that need to be presented to a human teammate to deliver the autonomy agent's transparency as follows: DP (Decision projection), LoF (Likelihood of task failures), FOS (Focal object status), TI (Task intention), EP (Non-focal object status), UP (Updated plan), CF (Confirmation), RC (Recommendation), PS (Process state), RS (Reasoning state) and MI (Miscellaneous information). The details of these transparency themes are as follows.

A) Theme *DP* (decision projection)

DP is about explaining what the agent will do in the immediate future. It is not necessarily in a detailed form that consists of all actions that will be taken, i.e., it can only represent the essence of these actions. For example, when an agent is performing driving tasks in the autopilot mode in a tailing situation (following a leading vehicle in the same lane), it needs to make various manoeuvres including slowing the vehicle, stopping the vehicle, switching into a lower gear, and so on to keep a safe margin from the leading vehicle, but the DP can only be about keeping the safe margin from the leading vehicle by skipping these manoeuvres.

B) Theme *LoF* (likelihood of task failures)

LoF refers to the probability of failures in performing real-world tasks assigned to an agent. In an autonomy agent system, the agent's tasks are often implemented by a set of instructions written in a software program controlling how the agent behaves and therefore, the successful execution of these instructions becomes the measure of task performance. There are two types of instructions: the first type is instructions for the agent to perform real-world tasks and the second one is instructions for a software program to

implement these real-world tasks. Even though the instructions written in the agent program are intended to ensure the agent accomplishes the designated real-world task, several lessons can be learned from the trade-off case in the traffic light situation. In this case, to achieve the goal of *Pass traffic lights*, by design, an agent is instructed in the program to keep going when it fails to recognize the traffic light state even though the actual state is red. In this regard, there is a trade-off between safety and other road users' convenience. From a program instruction perspective, performing the keep going manoeuvres in an unknown traffic light state can be considered as successfully completing the instructed action. However, from a real-world task perspective, it can be considered as a failure to follow the traffic light state. Hence, *LoF* does not focus on successfully executing program instructions, but in accomplishing real-world tasks.

C) Theme *FOS* (focal object status)

FOS can be described as objects that have a significant relation to the agent's tasks. Traffic light state and margin from the leading vehicle are examples of the focal objects or situations. *FOS* represents the states of focal objects or situations including uncertain states. Conveying uncertain states is critical to provide a human teammate with insight that there is a problem with the agent's recognition process. The successful accomplishment of a task usually starts from well-recognized states of focal objects.

D) Theme *TI* (tasks intention)

TI refers to any cues provided to monitor the tasks given to the agent. A task can consist of many activities. This study identifies two types of tasks: the first type is a task that is performed by the agent all the time and the second type is a task triggered by a particular event which can be a human instruction or a circumstance from the environment. In an autopilot driving context, keeping the vehicle in the lane is an example of the first type of task and adjusting manoeuvres based on traffic light states is an example of the second type of task because it is triggered by the existence of traffic lights.

E) Theme *EP* (non-focal object status)

Other objects' states that do not have a significant relation to tasks are denoted by *EP* which can be useful in helping understand an agent's perception of the overall situation, mainly when the agent is surrounded by moving objects within an uncontrolled environment. It should be noted that even though an agent has a camera to capture objects in the surrounding environment and visualizes these objects on a presentation screen, it doesn't mean that all the captured objects affect the agent's behaviour. Some of them are captured by the camera by chance so they are not considered as the agent's perception. In this regard, the vehicle behind can be an example of *EP*.

F) Theme *UP* (updated plan)

Depending on the cases, plans can either be at a strategical, a tactical, or an operational level. In a driving context, a route to the intended destination is an example of a strategic plan. Determining the overtaking route is one example of tactical plans. Furthermore, adjusting gears, increasing speed, or pressing the brake pedal are types of operational plans. The plan in this transparency theme group is considered as a strategic plan. Then, a route change after the vehicle's navigation system detects traffic jams ahead is an example of a strategic plan update.

G) Theme *RC* (recommendations)

RC is a form of coordination in HAT in which the agent may need human consent to perform an action. For example, an autonomy agent in autopilot mode might want to make an overtaking manoeuvre, but in order to do this, it needs the human teammate to send a signal representing their consent. Yet, *RC* does not change the main plan such as the route to destination.

H) Theme *CF* (confirmation)

CF is slightly different from *RC* in which an agent may need confirmation from a human teammate as to whether its prediction-based action is correct. When there is no response from the human teammate, the agent will assume that nothing is wrong with its action. An example of *CF* is as follows: when there are no road lines detected, an autopilot agent

will predict a correct path and ask for the driver's confirmation. As a response, the driver can correct the agent's actions if necessary.

I) Theme *PS* (process state/activities)

PS refers to the agent's activities as written in the program instructions to accomplish tasks such as calculating a safe margin to the vehicle ahead, activating the camera, obtaining image samples to identify the traffic light state and location, and scanning threats from surrounding objects crossing the vehicle's path. However, it should be noted that in some cases like autopilot driving, information about these activities might be difficult to follow because they occur very quickly.

J) Theme *RS* (reasoning state)

RS is about generating an explanation on an agent's behaviour. An example of an explanation can be "the probability of successfully following the traffic light state is still high even though the traffic light state is not recognized because the manoeuvre will be based on the leading vehicle". This means that when the leading vehicle stops, the agent will also stop.

K) Theme *MI* (miscellaneous information)

MI is used to convey miscellaneous information such as hardware failures or other abnormal behaviours. Also, it can be any information such as a task performance report or the weather.

In general, each transparency theme has its own purposes to make the agent's states and behaviour transparent so that the human teammate will be able to comprehend it. However, it can be seen from the above transparency theme examples that not all themes are suitable for a particular context because many factors such as infobesity, limited decision time, or situations with a very tight duration. For example, in an autopilot driving context, it would be very difficult for a driver to make use of *PS* and *RS* because the driver doesn't have enough time to read this information, particularly in situations that last for a short period of time.

Similarly, conveying *RS*, which can be considered as the core of transparency to comprehend an agent, can be a cognitive burden for human teammates, particularly when it is a very complex explanation. Most complex explanations are presented in text form. Using a combination of cues can provide a simple explanation. For example, a timer icon is used to indicate the remaining time for the agent to arrive at the destination which implicitly conveys the possibility of arriving on time.

8.2.2. Transparency Theme Groups

Based on the three-level SA model and teaming context, we group the 11 themes into four groups. The first transparency theme group is 'intention' (*INT*). *INT* focuses on the information related to the status of the situation awareness elements. The transparency themes included in the INT group are *FOS*, *TI*, and *EP*. This group is associated with the autonomy agent's SA-Level 1. The second transparency group is 'explanation and activities' (*EXP*). *EXP* contains information to explain the autonomy agent's higher-level understanding and activities. Transparency themes *RS*, *PS*, and *MI* are considered as the member of group *EXP*. This group is associated with the autonomy agent's SA-Level 2. The third transparency group is 'outcome prediction' (*OP*). *OP* contains cues that explain what the consequences of the current action/decision will be in the near future based on the current states. The *OP* group contains two transparency themes which are *DP* and *LoF*. This group is associated with the autonomy agent's SA-Level 3. Lastly, the fourth transparency group is 'plans, decision, and coordination' (*PDC*). The *PDC* group contains transparency themes *RC*, *CF*, and *UP*. This group is specifically formed to accommodate the teaming context.

8.2.3. Key Ideas of TCDT Model

There are three key ideas behind the TCDT model. One is it views that a situation is only alive for a certain time-length. Consequently, we introduce a time constraint as an attribute to a situation as follows: this attribute can have one of the following three linguistic values: strong, medium, or low. If a situation has a strong time constraint, it will only be alive for a short period or have a short validity period (e.g., under 15 seconds). If a situation has a low time constraint, it will be alive for a longer period or have a long validity period (e.g., more than 5 minutes). If a situation has a medium time constraint, it

will be alive for a period between a short and a long period or have a medium period (e.g., a minute or two).

The other idea is that a transparency theme has a visibility index, which indicates when this theme should be presented to the human teammate. This index has three discrete values, which are 0, 1 and 2. If a theme has a visibility index of 0, this theme will be only displayed when the situation has a long validity period. If a theme has a visibility index of 2, this theme will be displayed in all situations. If a theme has a visibility index of 1, this theme will be presented for situations that have a medium validity period or more (a long validity period).

Furthermore, the TCDT model defines four transparency themes as mandatory to be presented namely *DP*, *FoS*, *TI*, and *RS*. The reason to set the four themes as mandatory is because this study views that to comprehend the agent, a human teammate should know "what the agent's decisions/actions are" (represented by DP), "what its intention is" (represented by *FoS* and *TI*), and "what the reason behind its decision is" (represented by *RS*). Other themes are situational as not all situations contain information for a particular theme. For example, in traffic light situations, *RC* may not be available. The third idea is that the transparency presentation is situation-specific. This means that to make the agent's behaviours transparent to a human teammate, the transparency presentation to this human teammate will depend on the TC of a given situation. For example, if TC = "strong", the presentation includes only essential information. Similarly if TC="low", the presentation can include more information.

8.2.4. Formal Description of TCDT Model

A) Time constraint TC

The TCDT model defines a situation using its TC which represents how long this situation is alive. TC can be one of three possible values: strong, medium, or low, i.e., $TC = [strong, medium, low]$.

B) Transparency themes and theme's properties

The TCDT model defines a list of 11 transparency themes, denoted as $TT = \{DP,\ LoF,\ FOS,\ TI,\ EP,\ UP,\ CF,\ RC,\ PS,\ RS,\ MI\}$, and each theme (an element in the list) has two properties, the visibility index (V) and the nature of the transparency theme (M). We can use a pair (V, M) to denote these two properties for a theme. V has three possible values: $V = [0, 1, 2]$ where index 0 indicates that this theme will be visible in a situation with a low time constraint, index 1 indicates that this theme will be visible in a situation with a medium and a low time constraint, and index 2 indicates that this theme will be visible in any situation no matter what its time constraint is. M has two possible values, one is mandatory and the other is situational, i.e., $M = [mandatory, situational]$. A theme with a '*mandatory*' value means this theme must be presented to the human teammate subject to the TC of the given situation. A theme with a '*situational*' value means this theme is not applicable to all situations and only occurs in relevant situations. For example, we use notation $DP(2, mandatory)$ to represent that *DP* has a visibility index 2 (suitable for all time constraint modes) and *DP* is a mandatory theme which must be visible to the human teammate.

C) TCDT model description

The TCDT model is defined as a pair $(TC,\ TT_{subset})$ where TC is the time-constraint attribute of a given situation, and TT_{subset} is a customised theme list based on the TC value. The theme list will be generated to include mandatory themes with a visibility index that matches with this TC and any situational themes for this situation. The matched visibility index and TC are tabulated in Table 8-1.

Table 8-1. The relation between TC and visibility in the TCDT model

TC	Visibility
Low	0, 1, and 2
Medium	1 and 2
Strong	0

8.2.5. Visual Presentation of TCDT Model

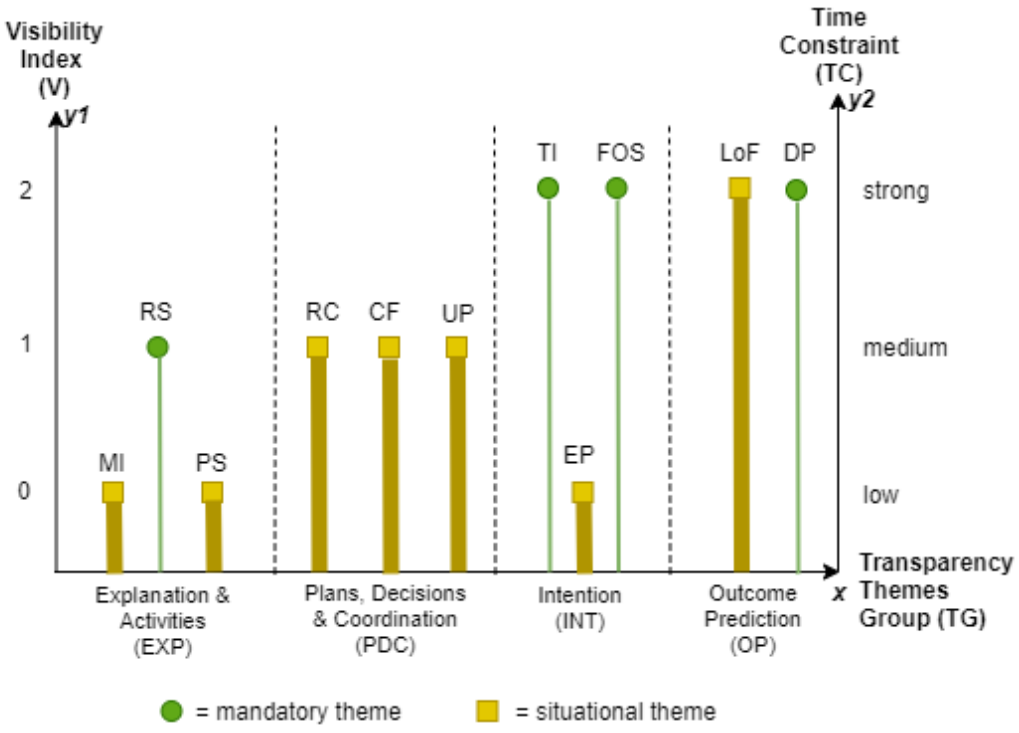


Figure 8-1. Time-constraint-driven transparency model

We visually present this TCDT model in Figure 8-1, where there are three axes: the horizontal axis *x* representing the transparency theme groups, the primary vertical axis y_1 representing the visibility index, and the secondary vertical axis y_2 representing the time constraint. The transparency themes are clustered based on their group and located along the *x*-axis. The location of each transparency group does not matter. The scale of the visibility index shown on the y_1-axis is from 0 to 2. The scale of the time constraint as shown on the y_2-axis ranges from low to high. Also, each transparency theme is displayed using a thin bar with a marker at the top to indicate it as a *mandatory* (circle) or *situational* theme (square). Based on the mapping relations between TC and the visibility index, the y_1-axis and y_2-axis should be scaled in such a way that visibility index 2 corresponds to TC="strong", visibility index 1 corresponds to TC ="medium" and visibility index 0 corresponds to TC = "low".

When a situation is given, based on its TC, we can determine its theme list TT_{subset} by drawing a horizontal line in the $x - y_1 - y_2$ plan that passes the known TC, and the themes that intersect with this line are the candidates in TT_{subset}. Of these candidates, the

mandatory ones and situational ones that are relevant to the given situation will be chosen to be in the theme list (TT_{subset}) to be presented to the human teammate.

8.3. Implementation of the TCDT Model

In this section, we present the TCDT model for the goals *Pass traffic lights* and *Overtake vehicles* using the CARLA simulator.

8.3.1. Transparency for the Goal *Passing Traffic Lights*

In this study, to specify the time constraint, we use two situations to achieve the goal *Pass Traffic Lights.* In Situation 1, the target vehicle is in segment 2, and in Situation 2, the target vehicle is in segment 1 approaching the TL (see Figure 4-6). We set the time constraint of Situation 1 as TC = medium and TC of Situation 2 as TC = strong.

Referring to Table 8-1, the TL situation 1 with TC = medium category is linked to the transparency themes with a visibility index 1 and 2. We can use the visual illustration of the TCDT model in Figure 8-1 to determine the list of transparency themes for this situation. *First*, we draw a horizontal line (l_1) that passes TC = medium, as shown in Figure 8-2. Then we find the candidate themes. It can be seen that this line intersects with the following themes: *DP*, *FOS*, *TI*, *RS*, *RC*, *CF*, *UP*, and *LoF*. *Lastly,* we select the themes. Of these theme candidates, the first four themes are *mandatory* and the last four are *situational*. Of these four situational themes, only *LoF* is relevant to this situation for the following reason, the theme *LoF* is used to reflect a critical issue due to the trade-off between safety and other road users' convenience in the autopilot feature design. This trade-off refers to the behaviour of the autonomy agent (as the one which is responsible for the autopilot feature) in response to a situation where it fails to recognize the traffic light state. By design, this agent will utilize the keep going manoeuvre in the event of such failures. Such a design increases the potential for a road incident, i.e., violating a red light and colliding with other vehicles. Therefore, it is very important to include *LoF* to explicitly capture such a failure.

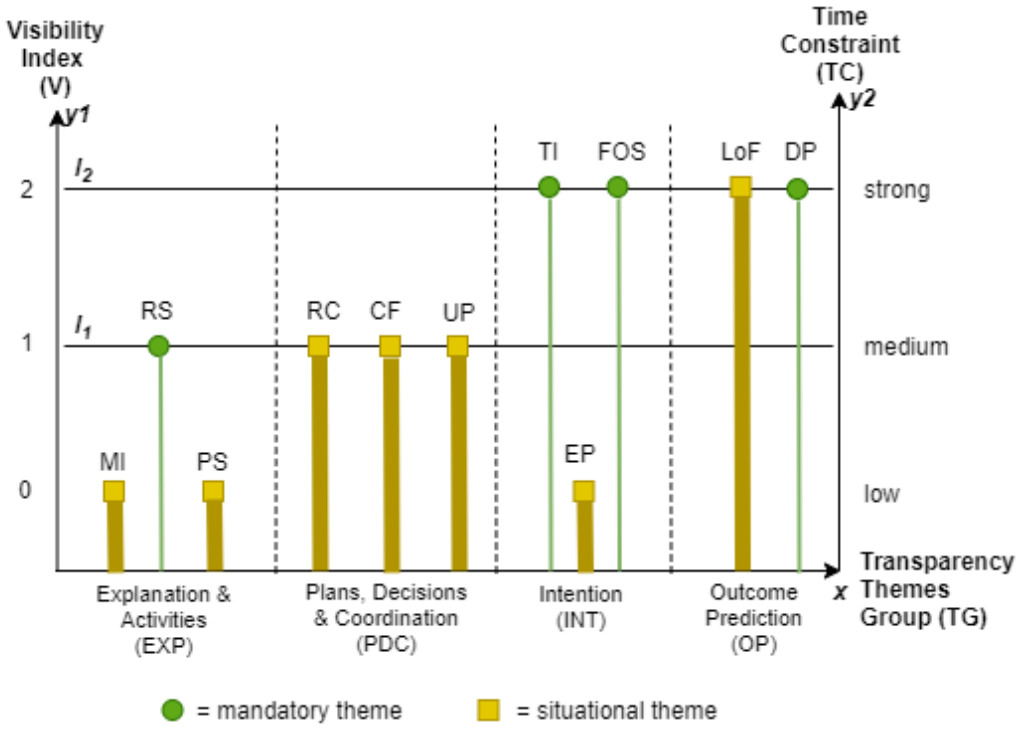


Figure 8-2. Getting transparency themes by relating the visibility index and time constraint

Therefore, the four mandatory themes and this *LoF* situational theme are chosen to form the theme list for this situation. These themes and their designed transparency information are presented in Table 8-2.

Table 8-2. Transparency themes and information for the traffic light scenario

Themes	Designed transparency information
DP	Keep going, TL-state-based manoeuvre, LV-based manoeuvre
LoF	Possibility of failures
FOS	The HV and LV status (speed, existence and distance with LV), TL states
TI	Traffic light situation
RS	The reason for DP

The theme *DP* is obtained from Level 3 awareness which is described in Table 4-12 in Chapter 4. *FOS* is selected from the situation awareness elements. *TI* is triggered when traffic lights are recognized. *RS* is obtained from the self-explanation ability developed in Chapter 6.

However, when the target vehicle enters Segment 1, Situation 2 becomes alive. The candidate themes are determined based on the horizontal line l_2 in Figure 8-2. It can be seen that the candidate themes for this situation are *DP*, *FOS*, *TI*, and *LoF* with the first

being the mandatory themes and the last being a situational theme. For the same reason as in Situation 1, *LoF* is relevant. Therefore, four themes are chosen to be in the theme list for presentation.

8.3.2. Showcase of Transparency for the Goal *Passing Traffic Lights*

Figure 8-3. Unrecognized TL state with no LV in Segment 2

Now, let's consider Figure 8-3 which illustrates TL Situation 1 with the target vehicle still in segment 2 and the autonomy agent failing to recognize the TL state. Therefore, this situation is a TL situation with medium TC. Transparency information for this situation is "The vehicle keeps going and is about to fail to follow TL as it enters TL situation under free ride situation but TL state unrecognized". In this information, the statement "The vehicle keeps going" is the representation of transparency theme *DP* while "is about to fail to follow TL" is the representation of transparency theme *LoF*. Furthermore, the TL icon in the left panel will pop out when the vehicle is entering the TL situation. This icon helps the human driver to know that currently, the autonomy agent is aware of the TL situations. Moreover, this TL icon becomes the representation of transparency theme *TI*. Furthermore, the icon state will depend on the TL state which is read by the inference engine. This state represents information for transparency theme *FOS*. With the scenario illustrated in Figure 8-3, we can see that the use of four default themes (*DP*, *LoF*, *TI*, and *FOS*) helps the human driver to calibrate their trust in the autonomy agent's behaviour. In addition to the four default themes, another transparency

theme, *RS*, is added to provide the reasoning of the DP. This is represented by the statement "as it enters TL situation under free ride situation but TL state unrecognized". In another type of TL situation (TC = strong), the statement representing RS is not provided.

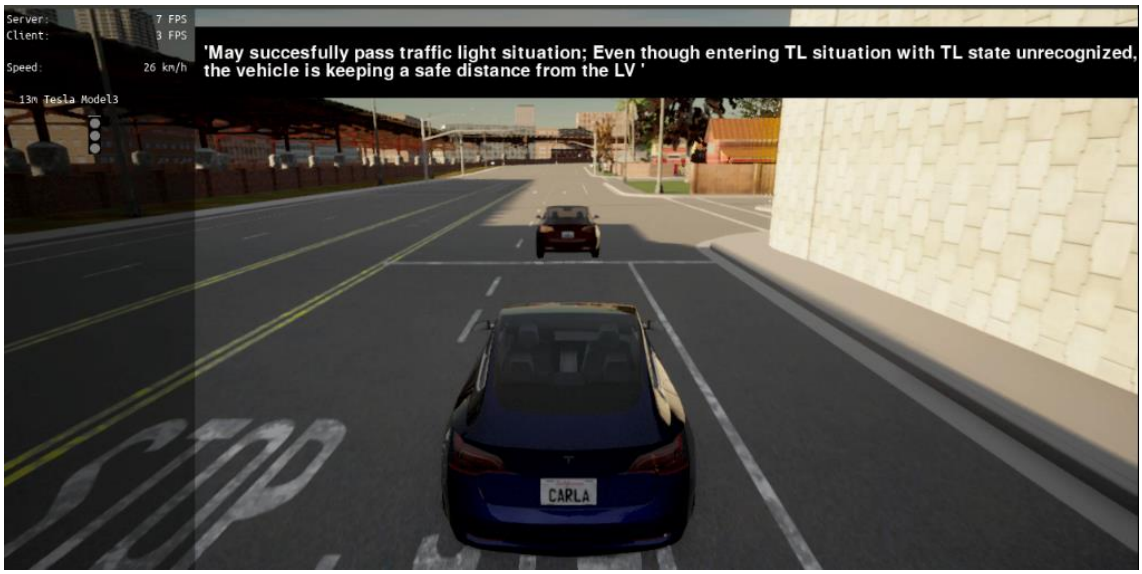


Figure 8-4. Unrecognized TL state with LV in segment 2

Now let's consider that LV exists when the TL state is unrecognized as illustrated in Figure 8-4. In this situation, the designated transparency information will be "May successfully pass traffic light situation; even though entering TL situation with TL state unrecognized, the vehicle is keeping a safe distance from LV". The statements "May successfully pass traffic light situation" and "the vehicle is keeping a safe distance from the LV" are representations of the transparency theme *LoF* and *DP*, respectively whereas the rest of the statement represents transparency theme *RS*. For this kind of situation, when the actual traffic light is red, the LV will stop. Hence, even though the autonomy agent cannot recognize the TL state, the target vehicle can still stop if LV stop.

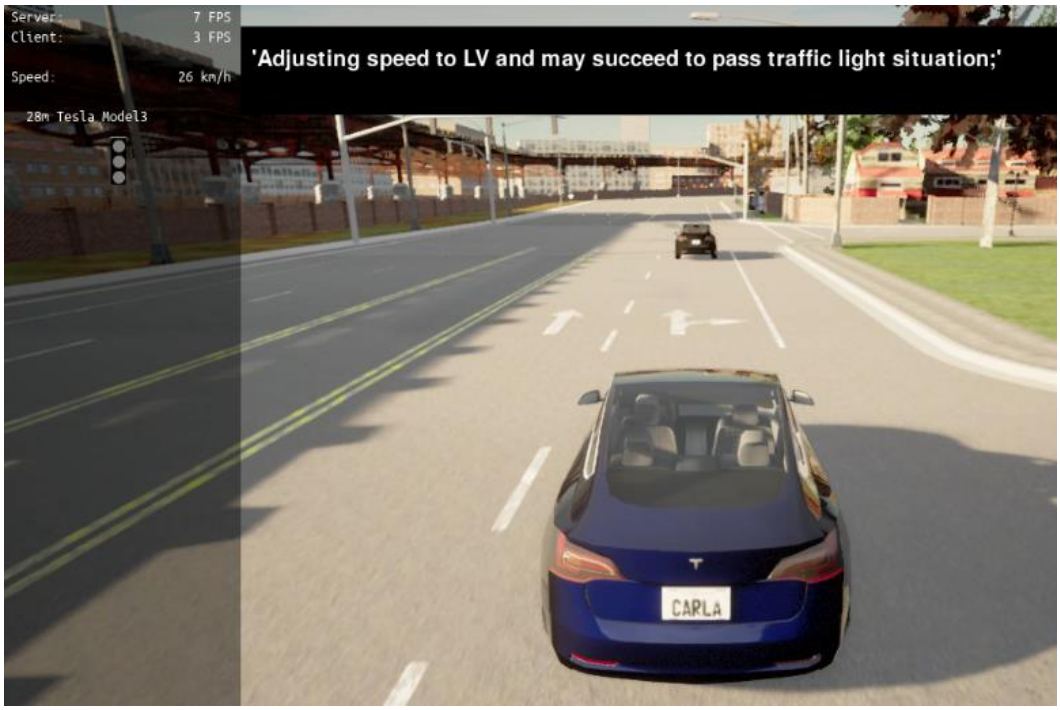


Figure 8-5. Unrecognized TL state with LV in segment 1

Figure 8-5 illustrates TL Situation 2 (target vehicle in segment 1) which has TC = 'strong'. We can see in Figure 8-2 that the valid transparency themes for strong TC are *DP, TI, LoF,* and *FOS*. Unlike TL situation 1, there is no *RS* for TL situation 2. Therefore, referring to Figure 8-2, the statement 'Adjusting speed to LV' represents *DP* and *FOS*, the TL icon represents *TI*, and the statement 'may succeed in passing traffic light situation' represents *LoF*.

8.3.3. Transparency for the Goal *Overtake Vehicles*

We define a situation to achieve the goal *overtaking vehicles* as a situation with a medium time constraint (TC = medium). Therefore, with a similar mechanism to that explained in Section 8.3.1 to determine the candidates of the transparency themes based on the TC category, we get the candidates *DP*, *FOS*, *TI*, *RS*, *RC*, *CF*, *UP*, and *LoF*. The first four are *mandatory* transparency themes and the rest are situational themes. Since achieving the goal *Overtake vehicles* requires coordination activities such as overtaking suggestion and getting overtaking approval, two *situational* transparency themes *CF* and *RF* become relevant for this situation. The chosen themes and their designated transparency information for the goal of *Overtake vehicles* are presented in Table 8-3.

Table 8-3. Transparency information for overtaking scenario

Themes	Designed transparency information
DP	Overtaking suggestion, perform overtaking, overtaking cancelation
FOS	The HV and LV status (speed, existence and distance with LV)
TI	Overtaking status/situation
CF	Confirmation
RC	Overtaking suggestion
RS	The reason for DP

8.3.4. Showcase of Transparency for the Goal *Overtake Vehicles*

Figure 8-6. Safe to overtake situation

Figure 8-6 shows the initial stage of the overtaking scenario. In the left column, the information from the FOS theme which is about the HV's speed and distance to LV is presented. Furthermore, the information "Overtaking suggested as all parameters indicate safe to overtake" aggregates information from several themes. As the information contains an overtaking suggestion, this implies *DP*, *TI*, and *RC*. In this regard, the autonomy agent has the new task intention (*TI*) which is to overtake, and therefore, its first action is to acknowledge the human driver by sending a suggestion (*DP* and *RC*).

Furthermore, the statement "all parameters indicate safe to overtake" represents the theme *RS* which is the cause of *DP*.

Once the human driver agrees to overtake, the next stage of the overtaking scenario is performed as illustrated in Figure 8-7. Information from the *FOS* theme, which is HV's speed and distance to LV will always be presented. The information in association with this stage is "Proceed with lane changing manoeuvre as overtaking task confirmed by driver". In this information, the statement 'Proceed with lane changing manoeuvre' represents the theme *DP* and also indicates the task intention, (*TI* theme), of the autonomy agent. Furthermore, the statement "overtaking task confirmed by driver" is the reasoning state (*RS* theme) of why such a manoeuvre is performed and indicates the example of the theme *CF* (i.e., driver confirmation).

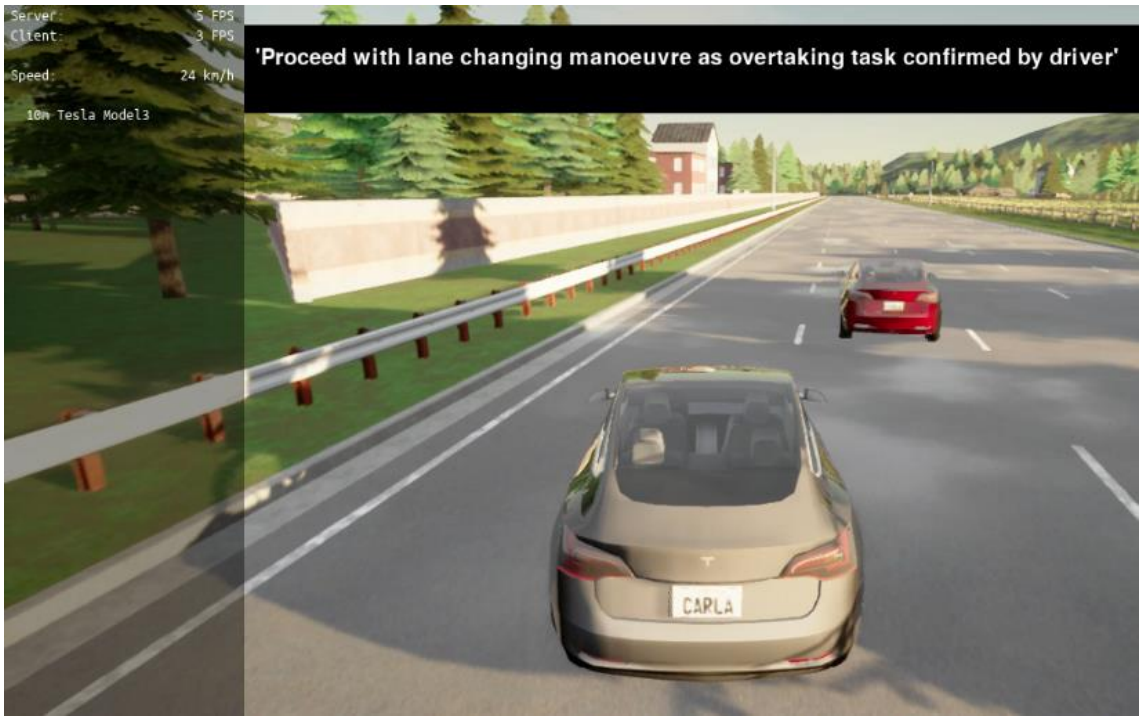


Figure 8-7. Transparency information for overtaking initialization

Figure 8-8. Transparency information for overtaking cancelation

Now, let's assume that the overtaken vehicle increases its speed, and therefore the overtaking task is cancelled by the autonomy agent (see Figure 8-8). The transparency information for such a situation is "Overtaking cancelled and stay in overtaking lane as overtaken vehicle increased its speed so the predicted overtaking speed is overridden by road speed limit". The statement "Overtaking cancelled and stay in overtaking lane" is the representation of themes *DP* and *TI*. The overtaking cancelation can also be considered as a change of plan which represents the theme *UP*. Furthermore, the rest statement in the transparency information indicates the reasoning state (*RS* theme).

8.4. Summary

In this chapter, we propose a new TCDT model to guide the determination of themes to delivery transparency in relation to the autonomy agent's behaviour based on the situation time constraint. Increasing the transparency of the autonomy agent is a part of the solution to make it easier for the human superior to supervise an autonomy agent. To showcase the TCDT model, we developed a simulation to make the autonomy agent's three-level

SA transparent to the human driver using two cases: achieving the goal *Pass traffic lights* and *Overtake vehicles*. This simulation is developed using the CARLA simulator. The simulation shows that the new TCDT model can help choose adequate information to ensure the agent's behaviour is transparent to the human driver.

Chapter 9 : Conclusions and Future Work

9.1. Conclusions

In human autonomy teaming (HAT) the autonomy agent acts as a teammate, therefore establishing coordination between the agent and its human teammates becomes a critical factor to ensure good teaming performance.

The proposed autonomy agent's coordination abilities in this study consist of two main modules. The first module focuses on helping the human teammate in supervising this agent by providing transparency through self-explanation ability. The other module focuses on the agent's ability to participate in maintaining human teammate's SA. Even though this study uses collaborative driving context for the showcase, the proposed coordination ability is also applicable for other HAT cases other than collaborative driving, such as the collaborative work between human and autonomous robot squad as illustrated by Selkowitz, Lakhmani & Chen (2017). However, it should be noted that the proposed coordination ability in this study is specifically developed for SSA-based HAT where the autonomy agent is the sub-ordinate of human team member. Also, the self-explanation ability part is only designed for a logic-based autonomy agent, and therefore, it may need further adjustments to implement it in a machine-learning-based agent.

This study has fulfilled its aim which is to develop the coordination ability of an autonomy agent in a HAT context in which the agent is a subordinate of a human superior by achieving five objectives. Each objective is concluded in the following sub-sections along with their limitations.

9.1.1. Develop a Framework to Synthesize the Characteristics of SA Models in Different HAT Contexts.

In fulfilling this objective, we examined team formation including the team goals and team members, and then determined individual goals for team members through goal

distribution using a hierarchical structure. Based on the goal provision hierarchy, we defined two coordination types in a team, which are vertical and horizontal coordination. According to the team coordination types, we specified four teaming situation awareness models to guide the development of the situation awareness model for each team member in a teaming. Lastly, we synthesized these efforts to propose a situation awareness modelling framework for teaming situations.

This framework presents a uniform way of modeling teaming situation awareness through a series of four blocks, namely the team specification block, goal provision block, teaming formation block, and teaming situation awareness specification block. Based on this framework, the SA of a given teaming can be represented by one of the following four models: supportive model (also called supportive situation awareness (SSA) model), team model, shared model, and mutual model. This framework is significant in developing an autonomy agent's capabilities based on its situation awareness requirements in the teaming context.

The limitation of this framework is due to there are many ways to develop the goal hierarchy for a SA model, and they can be very subjective. For this regard, the developed SA modelling framework can be optimally used when the goal hierarchy is developed under task-based perspective.

9.1.2. Develop a SA Development Process to Implement SA models in a Teaming with Vertical Coordination

In fulfilling this objective, we first selected collaborative driving as a representative HAT example, then we chose two teaming instances to develop the autonomy agent's SA. Based on the framework of situation awareness modelling for teaming situations in Chapter 3, we found that these two teaming instances fall into the category of vertical coordination as the human driver and the ADAS agent have a superior-subordinate relationship, in other words, the agent's individual goals are a subset of the human driver's goals, or the agent has no goals other than the ones shared with the human superior. Therefore, we used the supportive SA model to develop the agent's SA. We first proposed a situation awareness development process (SADP) framework, which comprises three phases, namely the perception phase, comprehension phase, and

projection phase. Through these three phases, the SA elements and SA requirements for the agent in these two teaming instances were determined. Based on these SA requirements, the agent's SA in these two teaming instances were simulated in a CARLA simulator. The achievement of this objective laid the foundation for the other objectives as SA is a critical part in fulfilling the autonomy agent's individual goal and supporting its coordination ability in a collaborative driving context. A simulation is developed using the CARLA simulator. The simulation results show that the autonomy agent can react based on its SA to achieve the goals *Pass traffic light* and *Overtake vehicles*. As a limitation, this framework is only suitable for a logic-based autonomy agent.

9.1.3. Develop a New Method to Improve an Autonomy Agent's Self-Explanation Abilities to Make Its Behaviours Understandable

To accomplish this objective, we first chose a mode in which the autonomy agent is in charge of the vehicle (i.e., the autopilot mode is on) but is supervised by the human driver. This is teaming with supervision. We then proposed an *ASAS*-based method with which the autonomy agent's behaviour is modeled based on its artificial situation awareness states (*ASAS*) and embedded this method as the *ASAS* module into the SADP framework. We further implemented the ASAS model using a Bayesian network (BN) to represent the autonomy agent's behaviours. It is this BN-based *ASAS* model that enables this new method to explain the agent's behaviours differently from the other methods. It has the following three highlights: 1) an episodical memory to capture one episode of a situation in time *t*; 2) fuzzy relation operations to present the *ASAS* model in a compact form to simplify the process to ascertain the reason for a particular behaviour, 3) an explanation provider and the Manhattan similarity function for explanation generation. In the existing process-based method, behaviour explanation is generated by tracing the executed paths of autonomy agent's logics. Therefore, this *ASAS*-based method can overcome the complexity issues in the process-based methods.

There are several advantages of using the *ASAS* model to represent the autonomy agent's behaviour. First, the *ASAS* model is more persistent than the process model because it is less vulnerable to changes in the algorithms. Second, this model is easier to design and can deal with system complexities. Third, it gives more flexibility to a system designer to express the action projection of the autonomy agent. The experiments using two case

studies in a collaborative driving context show that the proposed *ASAS*-based method can reduce the complexities in generating behaviour explanations significantly by reducing the search space in generating explanations compared to a process-based method.

The achievement of this objective enables an autonomy agent to self-explain its behaviours and addresses a critical issue in HAT to help the human teammate comprehend their artificial teammate.

To use the ASAS-based method more effectively, we also provide eight design principles for developing an *ASAS* model. These principles are particularly useful in configuring the node states, the conditional probabilities of the linked node in the BN and the BN model, which links the autonomy agent's perception, comprehension, and action projection. Using CARLA simulator, the simulation results show that the autonomy agent can generate a self-explanation on its behaviour based on artificial situation awareness states.

We identified two limitations for our proposed method. First, the ASAS-based method proposed in this study is only used to generate the rationale of the autonomy agent's actions in the BCE (backward chaining explanations) context. Secondly, the proposed method has been evaluated using only two typical teaming cases in collaborative driving based on simulations using the CARLA simulator. In this regard, we only used less variables for simulated environment. A real-world implementation can be more challenging as the real-world environment has more complex variables to be considered.

9.1.4. Develop an Approach for the Autonomy Agent to Observe Its Human Teammate's Behaviours in a Collaborative Driving Situation

To achieve this objective, we first chose a situation in which the human driver is in charge of the vehicle and the autonomy agent plays the role of an assistant to support the human driver's SA development or maintenance. Based on the fact that we collected a small number of sample images of driver distraction using the vehicle's on-board camera, which is far less than the images needed for training a good classifier, we then proposed a new transfer learning approach to generate a model to classify the driver's distraction types by taking advantage of the existing large number of sample images of distracted drivers in the source domain. This new approach addresses the transfer learning problem with

geometrical relation constraints which can significantly degrade the classification performance of the model generated from the transfer learning process, even though online augmentation is used to multiply the sample of images in various positions, if they are not handled properly. This new method handles these constraints using directive offline augmentation. The results of an experiment in which the target domain is the target vehicle which is a right-steering car and the source domain is a set of left-steering cars show that this new method achieves better classification performance than the baseline methods. The proposed method is significant for the promotion of safe driving by preventing the human driver from being distracted. However, it is noticed that the distracted driver simulation was undertaken using the on-board camera-based driver distraction recognition model which requires a computation engine with high specifications. This requirement becomes the limitation of the developed model.

9.1.5. Develop a Time-Constraint-Based Transparency Model to Control the Visibility of an Autonomy Agent's Behaviour for a Given Situation

To fulfill this fifth objective, we developed the following four novel proposals: 1) To identify 11 transparency themes from the agent's SA and the representation of its coordination ability, namely DP (Decision projection), LoF (Likelihood of task failures), FOS (Focal object status), TI (Task intention), EP (Non-focal object status), UP (Updated plan), CF (Confirmation), RC (Recommendation), PS (Process state), RS (Reasoning state) and MI (Miscellaneous information), and then divided them into four groups. The first group called 'Intention' (*INT*) is associated with the transparency of the autonomy agent's SA level 1. The second group called 'explanation and activities' (EXP) is associated with the transparency of the autonomy agent's SA level 2. The third group called 'outcome prediction' (*OP*) is related to the transparency of the autonomy agent's SA level 3. The last group called 'plans, decisions, and coordination' (*PDC*) accommodates the teaming context. 2) To define two attributes for each transparency theme, which are the visibility index (0,1 or 2) and the nature of the theme (mandatory or situational). 3) To define an attribute for a situation, which is time constraint (TC) and has the three categorical values of "low", "medium" and "strong". 4) To define the mapping rules between the visibility index of a theme and the TC of a situation. Based on

these proposals, we developed a transparency model for agent behaviours and a visual representation of this model.

This model provides guidance on the transparency information to be presented to a human about the autonomy agent's behaviour, driven by the time constraints of situations. The transparency information suggested by this model is to make the three levels of autonomy situation awareness transparent. The showcases in the traffic light and overtaking situations indicate that this model is useful to gauge the necessary information for the human teammates in a given situation. The proposed model is significant to ensure the autonomy agent's behaviours are transparent to help the human driver understand the agent's behaviours so that they can calibrate their trust in the agent.

One critical factor in the proposed TCDT model is classifying situations into low, medium, or strong time constraint. The limitation in our study is that we only focused on demonstrating transparency related to a certain goal. Meanwhile, there are possibilities in HAT context that more than one critical goal is simultaneously performed by an autonomy agent. Hence, situation classification needs to be designed thoroughly.

9.2. Future Work

The future directions of this research can be summarized from the following perspective:

1) Implement the proposed framework and methods in real-world applications with other teaming cases.

2) Develop a lighter classifier for a lower computation engine such as a mobile phone so that a manual vehicle without on-board camera can also take advantage of driver distraction recognition to promote safe driving.

3) Develop a method to generate data-driven explanations which is also called forward chaining explanations (FCE) and further develop a new method which is capable of combining goal-driven with data-driven explanations in the HAT context as sometimes it is necessary to deliver explanations using both BCE and FCE.

4) Develop a method to address transparency-false alarm problems. Ensuring transparency on one hand can be used to calibrate human trust on their artificial teammate, but on the other hand, the human may view transparency information as false alarms. For example, the human and the autonomy agent may have a different situation understanding as the autonomy agent may have a weakness in its situation recognition, especially when the environment is not specifically designed to support the task of this agent. For example, on a curved road, an autonomy agent may falsely recognize a vehicle in front in the adjacent lane as a lead vehicle within its path. When an unsafe margin with such an incorrectly recognized lead vehicle is detected, an alarm is delivered to the human driver. In this regard, the human driver may consider this alarm as a false alarm rather than an effort to present transparency on the autonomy agent's situation awareness. Thus, further research is desirable to address these problems.

5) In SSA-based human autonomy teaming, the autonomy agent might work in a remote location. Moreover, the team can have many autonomy agents as team members under human supervision. In this scenario, team members can have relations as described in either a team, mutual, shared or supportive situation awareness model. Therefore, developing coordination abilities for teaming with multi autonomy agents as team members can be a future challenge.

References

Abdul, A., Vermeulen, J., Wang, D., Lim, B.Y. & Kankanhalli, M. 2018, 'Trends and trajectories for explainable, accountable and intelligible systems: An HCI research agenda', *Conference on Human Factors in Computing Systems - Proceedings*.

Adams, M.J., Tenney, Y.J. & Pew, R.W. 1995, 'Situation awareness and the cognitive management of complex systems', *Human factors*, vol. 37, no. 1, pp. 85–104.

Aggarwal, C.C. 2018, *Neural Networks and Deep Learning A Textbook*, 1st ed. 20., Springer International Publishing, Cham.

Al-Ajlan, A. 2015, 'The comparison between forward and backward chaining', *International Journal of Machine Learning and Computing*, vol. 5, no. 2, p. 106.

Allen, J. & Ferguson, G. 2002, 'Human-machine collaborative planning', *International NASA Workshop on Planning*.

Anjomshoae, S., Najjar, A., Calvaresi, D. & Främling, K. 2019, 'Explainable agents and robots: Results from a systematic literature review', *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1078–88.

Arnold, A., Nallapati, R. & Cohen, W.W. 2007, 'A comparative study of methods for transductive transfer learning', *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, pp. 77–82.

Asfoor, H., Srinivasan, R., Vasudevan, G., Verbiest, N., Cornells, C., Tolentino, M., Teredesai, A. & De Cock, M. 2015, 'Computing fuzzy rough approximations in large scale information systems', *Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014*.

Bainbridge, L. 1983, 'Ironies of automation', *Analysis, design and evaluation of man--machine systems*, Elsevier, pp. 129–35.

Barber, K.S. & Martin, C.E. 1999, 'Agent autonomy: Specification, measurement, and dynamic adjustment', *Proceedings of the autonomy control software workshop at autonomous agents*, vol. 1999, pp. 8–15.

Beavers, G. & Hexmoor, H. 2003, 'Types and limits of agent autonomy', *International Workshop on Computational Autonomy*, pp. 95–102.

Bedny, G. & Meister, D. 1999, 'Theory of activity and situation awareness', *International Journal of cognitive ergonomics*, vol. 3, no. 1, pp. 63–72.

Bencomo, N., Welsh, K., Sawyer, P. & Whittle, J. 2012, 'Self-explanation in adaptive systems', *Proceedings - 2012 IEEE 17th International Conference on Engineering of Complex Computer Systems, ICECCS 2012*.

Berthold, M. 1999, 'Fuzzy logic', *Intelligent data analysis*, Springer, pp. 269–98.

Bhaskara, A., Duong, L., Brooks, J., Li, R., McInerney, R., Skinner, M., Pongracic, H. & Loft, S. 2021, 'Effect of automation transparency in the management of multiple unmanned vehicles', *Applied Ergonomics*, vol. 90, p. 103243.

Bhaskara, A., Skinner, M. & Loft, S. 2020, 'Agent Transparency: A Review of Current Theory and Evidence', *IEEE Transactions on Human-Machine Systems*, Institute of Electrical and Electronics Engineers Inc., pp. 215–24.

Biran, O. & Cotton, C. 2017, 'Explanation and justification in machine learning: A survey', *IJCAI-17 workshop on explainable AI (XAI)*, vol. 8, p. 1.

Brandt, S.L., Lachter, J., Russell, R. & Shively, R.J. 2017, 'A human-autonomy teaming approach for a flight-following task', *International Conference on Applied Human Factors and Ergonomics*, pp. 12–22.

Castelfranchi, C. & Falcone, R. 2003a, 'Founding autonomy: the dialectics between (social) environment and agent's architecture and powers', *International Workshop on Computational Autonomy*, pp. 40–54.

Castelfranchi, C. & Falcone, R. 2003b, 'From automaticity to autonomy: the frontier of artificial agents', *Agent Autonomy*, Springer, pp. 103–36.

Caterini, A.L. 2018, *Deep Neural Networks in a Mathematical Framework*, D.E. Chang

(ed.),SpringerBriefs in Computer Science, 1st ed. 20., Springer International Publishing, Cham.

Chanda, J., Kanjilal, A., Sengupta, S. & Bhattacharya, S. 2009, 'Traceability of requirements and consistency verification of UML use case, activity and Class diagram: A Formal approach', *2009 Proceeding of International Conference on Methods and Models in Computer Science (ICM2CS)*, pp. 1–4.

Chapelle, O., Weston, J. & Schölkopf, B. 2003, 'Cluster kernels for semi-supervised learning', *Advances in neural information processing systems*, pp. 601–8.

Chen, F.-C. 1990, 'Back-propagation neural networks for nonlinear self-tuning adaptive control', *IEEE control systems Magazine*, vol. 10, no. 3, pp. 44–8.

Chen, J.Y., Procci, K., Boyce, M., Wright, J., Garcia, A. & Barnes, M. 2014, *Situation awareness-based agent transparency*, Army Research Lab Aberdeen Proving Ground MD Human Research And Engineering Directorate.

Chen, J.Y.C., Lakhmani, S.G., Stowers, K., Selkowitz, A.R., Wright, J.L. & Barnes, M. 2018, 'Situation awareness-based agent transparency and human-autonomy teaming effectiveness', *Theoretical Issues in Ergonomics Science*, vol. 19, no. 3, pp. 259–82.

Christoffersen, K. & Woods, D.D. 2002, '1. How to make automated systems team players', *Advances in Human Performance and Cognitive Engineering Research*.

Cox, M.T. 2013, 'Goal-driven autonomy and question-based problem recognition', *Second Annual Conference on Advances in Cognitive Systems 2013, Poster Collection*, pp. 29–45.

Dai, W., Chen, Y., Xue, G.-R., Yang, Q. & Yu, Y. 2009, 'Translated learning: Transfer learning across different feature spaces', *Advances in neural information processing systems*, pp. 353–60.

Deering, R.K. 1992, *Automatic vehicle deceleration*, Google Patents.

Demir, M., Likens, A.D., Cooke, N.J., Amazeen, P.G. & McNeese, N.J. 2019, 'Team coordination and effectiveness in human-autonomy teaming', *IEEE Transactions on Human-Machine Systems*.

Demir, M., McNeese, N.J. & Cooke, N.J. 2017, 'Team situation awareness within the context of human-autonomy teaming', *Cognitive Systems Research*, vol. 46, pp. 3–12.

Deshmukh, A.A. & Laftchiev, E. 2018, 'Semi-supervised transfer learning using marginal predictors', *2018 IEEE Data Science Workshop (DSW)*, pp. 160–4.

Diakopoulos, N. 2016, 'Accountability in algorithmic decision making', *Communications of the ACM*, vol. 59, no. 2, pp. 56--62.

Elizalde, F., Sucar, L.E., Luque, M., Díez, F.J. & Reyes, A. 2008, 'Policy explanation in factored Markov decision processes', *Proceedings of the 4th European Workshop on Probabilistic Graphical Models, PGM 2008*, pp. 97–104.

Endsley, M.R. 1995, 'Toward a theory of situation awareness in dynamic systems', *Human Factors*, vol. 37, no. 1, pp. 32–64.

Endsley, M.R. 1999, 'Level of automation effects on performance, situation awareness and workload in a dynamic control task', *Ergonomics*, vol. 42, no. 3, pp. 462–92.

Endsley, M.R. 2001, 'Designing for situation awareness in complex systems', *Proceedings of the Second International Workshop on symbiosis of humans, artifacts and environment*.

Endsley, M.R. 2017, 'From here to autonomy: Lessons learned from human–automation research', *Human Factors*, vol. 59, no. 1, pp. 5–27.

Endsley, M.R., Bolstad, C.A., Jones, D.G. & Riley, J.M. 2003, 'Situation awareness oriented design: From user's cognitive requirements to creating effective supporting technologies', *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 47, no. 3, pp. 268–72.

Endsley, M.R. & Jones, D.G. 2011a, 'Automation and Situation Awareness', *Designing for Situation Awareness*, CRC Press, pp. 169–92.

Endsley, M.R. & Jones, D.G. 2011b, 'Creating Situation Awareness-Oriented Designs', *Designing for Situation Awareness*, CRC Press, pp. 61–2.

Endsley, M.R. & Robertson, M.M. 2000, 'Situation awareness in aircraft maintenance

teams', *International Journal of Industrial Ergonomics*, vol. 26, no. 2, pp. 301–25.

Fähndrich, J., Ahrndt, S. & Albayrak, S. 2013, 'Towards self-explaining agents', *Trends in Practical Applications of Agents and Multiagent Systems*, Springer, pp. 147–54.

Fastenmeier, W. & Gstalter, H. 2007, 'Driving task analysis as a tool in traffic safety research and practice', *Safety Science*, vol. 45, no. 9, pp. 952–79.

Feng, Y.-H., Teng, T.-H. & Tan, A.-H. 2009, 'Modelling situation awareness for Context-aware Decision Support', *Expert Systems with Applications*, vol. 36, no. 1, pp. 455–63.

Fukuchi, Y., Osawa, M., Yamakawa, H. & Imai, M. 2017, 'Autonomous self-explanation of behavior for interactive reinforcement learning agents', *HAI 2017 - Proceedings of the 5th International Conference on Human Agent Interaction*.

Galushkin, A.I. 2007, *Neural network theory*, Springer Berlin Heidelberg.

Ghaffari, A., Khodayari, A.R., Alimardani, F. & Sadati, H. 2012, 'Modeling and Intelligent Control System Design for Overtaking Maneuver in Autonomous Vehicles', *Iranian Journal of Mechanical Engineering Transactions of the ISME*, vol. 13, no. 1, pp. 68–87.

Göritzlehner, R., Borst, C., Ellerbroek, J., Westin, C., van Paassen, M.M. & Mulder, M. 2014, 'Effects of transparency on the acceptance of automated resolution advisories', *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2965–70.

Gorman, J.C., Cooke, N.J. & Winner, J.L. 2006, 'Measuring team situation awareness in decentralized command and control environments', *Ergonomics*, vol. 49, no. 12–13, pp. 1312–25.

Hansen, L.K. & Salamon, P. 1990, 'Neural network ensembles', *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001.

Harbers, M., Van Den Bosch, K. & Meyer, J.J. 2010, 'Design and evaluation of explainable BDI agents', *Proceedings - 2010 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2010*.

Harbers, M., Bradshaw, J.M., Johnson, M., Feltovich, P., van den Bosch, K. & Meyer, J.-J. 2011, 'Explanation and coordination in human-agent teams: a study in the BW4T testbed', *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 03*, IEEE Computer Society, pp. 17–20.

Hecht-Nielsen, R. 1992, 'Theory of the backpropagation neural network', *Neural networks for perception*, Elsevier, pp. 65–93.

Hexmoor, H. 2001, 'Stages of autonomy determination', *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 31, no. 4, pp. 509–17.

Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R.R. 2012, 'Improving neural networks by preventing co-adaptation of feature detectors', *arXiv preprint arXiv:1207.0580*.

Hoffman, R.R., Johnson, M., Bradshaw, J.M. & Underbrink, A. 2013, 'Trust in Automation', *IEEE Intelligent Systems*, vol. 28, no. 1, pp. 84–8.

Hunt, J.E., Lee, M.H. & Price, C.J. 1993, 'Applications of qualitative model-based reasoning', *Control Engineering Practice*, vol. 1, no. 2, pp. 253–66.

Inagaki, T. 2003, 'Automation and the cost of authority', *International Journal of Industrial Ergonomics*, vol. 31, no. 3, pp. 169–74.

Jacobson, I. 1993, *Object-oriented software engineering: a use case driven approach*, Pearson Education India.

Janssen, C.P., Donker, S.F., Brumby, D.P. & Kun, A.L. 2019, 'History and future of human-automation interaction', *International Journal of Human Computer Studies*, vol. 131, pp. 99--107.

Johannsdottir, K.R. & Herdman, C.M. 2010, 'The role of working memory in supporting drivers' situation awareness for surrounding traffic', *Human Factors*, vol. 52, no. 6, pp. 663–73.

Kaggle 2016, *State Farm Distracted Driver Detection*, viewed 3 April 2020, <https://www.kaggle.com/c/state-farm-distracted-driver-detection/data>.

Kang, G., Jiang, L., Yang, Y. & Hauptmann, A.G. 2019, 'Contrastive Adaptation Network for Unsupervised Domain Adaptation', *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4893–902.

Kaptein, F., Broekens, J., Hindriks, K. & Neerincx, M. 2017, 'Personalised self-explanation by robots: The role of goals versus beliefs in robot-action explanation for children and adults', *RO-MAN 2017 - 26th IEEE International Symposium on Robot and Human Interactive Communication*.

Kaya, M. & Bilge, H. \cSakir 2019, 'Deep metric learning: a survey', *Symmetry*, vol. 11, no. 9, p. 1066.

Klien, G., Woods, D.D., Bradshaw, J.M., Hoffman, R.R. & Feltovich, P.J. 2004, 'Ten challenges for making automation a" team player" in joint human-agent activity', *IEEE Intelligent Systems*, vol. 19, no. 6, pp. 91–5.

Körber, M., Baseler, E. & Bengler, K. 2018, 'Introduction matters: Manipulating trust in automation and reliance in automated driving', *Applied Ergonomics*, vol. 66, pp. 18–31.

van Koten, C. & Gray, A.R. 2006, 'An application of Bayesian network for predicting object-oriented software maintainability', *Information and Software Technology*, vol. 48, no. 1, pp. 59–67.

Kridalukmana, R., Lu, H.Y. & Naderpour, M. 2020, 'A supportive situation awareness model for human-autonomy teaming in collaborative driving', *Theoretical Issues in Ergonomics Science*, vol. 0, no. 0, pp. 1–26.

Kumar, A., Saha, A. & Daume, H. 2010, 'Co-regularization based semi-supervised domain adaptation', *Advances in neural information processing systems*, pp. 478–86.

Langley, P., Meadows, B., Sridharan, M. & Choi, D. 2017, 'Explainable agency for intelligent autonomous systems', *Twenty-Ninth IAAI Conference*.

Le, C.D. 2002, *System transparency through self-explanation*, AAAI Technical Report FS-02-03. The Association for The Advancement Of Artificial Intelligence (AAAI).

Lee, D.-H. 2013, 'Pseudo-label: The simple and efficient semi-supervised learning

method for deep neural networks', *Workshop on challenges in representation learning, ICML*, vol. 3, p. 2.

Liu, W., Kim, S.-W., Pendleton, S. & Ang, M.H. 2015, 'Situation-aware decision making for autonomous driving on urban road using online POMDP', *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1126–33.

Long, M., Cao, Y., Wang, J. & Jordan, M.I. 2015, 'Learning transferable features with deep adaptation networks', *arXiv preprint arXiv:1502.02791*.

Long, M., Zhu, H., Wang, J. & Jordan, M.I. 2017, 'Deep transfer learning with joint adaptation networks', *34th International Conference on Machine Learning, ICML 2017*.

Lu, J., Behbood, V., Hao, P., Zuo, H., Xue, S. & Zhang, G. 2015, 'Transfer learning using computational intelligence: A survey', *Knowledge-Based Systems*.

Lundberg, J. 2015, 'Situation awareness systems, states and processes: a holistic framework', *Theoretical Issues in Ergonomics Science*.

Madhavan, P. & Wiegmann, D.A. 2005, 'Cognitive anchoring on self-generated decisions reduces operator reliance on automated diagnostic aids', *Human Factors*, vol. 47, no. 2, pp. 332–41.

Mamdani, E.H. 1976, 'Application of fuzzy logic to approximate reasoning using linguistic synthesis', *Proceedings of the sixth international symposium on Multiple-valued logic*, IEEE Computer Society Press, Logan, Utah, USA, pp. 196–202.

Markowski, A.S., Mannan, M.S., Kotynia, A. & Pawlak, H. 2011, 'Application of fuzzy logic to explosion risk assessment', *Journal of Loss Prevention in the Process Industries*, vol. 24, no. 6, pp. 780–90.

Matasci, G., Volpi, M., Kanevski, M., Bruzzone, L. & Tuia, D. 2015, 'Semisupervised transfer component analysis for domain adaptation in remote sensing image classification', *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 7, pp. 3550–64.

McAree, O. & Chen, W.-H. 2013, 'Artificial situation awareness for increased autonomy of unmanned aerial systems in the terminal area', *Journal of Intelligent & Robotic*

Systems, vol. 70, no. 1–4, pp. 545–55.

McNeese, N.J., Demir, M., Cooke, N.J. & Myers, C. 2018, 'Teaming with a synthetic teammate: insights into human-autonomy teaming', *Human Factors*, vol. 60, no. 2, pp. 262–73.

Mizumoto, M. & Tanaka, K. 1981, 'Fuzzy sets and their operations', *Information and Control*, vol. 48, no. 1, pp. 30–48.

Mostafa, S.A., Darman, R., Khaleefah, S.H., Mustapha, A., Abdullah, N. & Hafit, H. 2018, 'A general framework for formulating adjustable autonomy of multi-agent systems by fuzzy logic', *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, pp. 23–33.

Mu, G., Xinyu, Z., Deyi, L., Tianlei, Z. & Lifeng, A. 2015, 'Traffic light detection and recognition for autonomous vehicles', *The Journal of China Universities of Posts and Telecommunications*, vol. 22, no. 1, pp. 50–6.

Mudalige, U.P. & Losh, M. 2015, *Efficient data flow algorithms for autonomous lane changing, passing and overtaking behaviors*, Google Patents.

Naderpour, M., Lu, J. & Zhang, G. 2013, 'A fuzzy dynamic bayesian network-based situation assessment approach', *2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, pp. 1–8.

Neuhaus, R., Laschke, M., Theofanou-Fülbier, D., Hassenzahl, M. & Sadeghian, S. 2019, 'Exploring the impact of transparency on the interaction with an in-car digital AI assistant', *Proceedings of the 11th International Conference on Automotive User Interfaces and Interactive Vehicular Applications: Adjunct Proceedings*, pp. 450–5.

Niesser, U. 1976, *Cognition and reality: Principles and implications of cognitive psychology*, Freeman, San Francisco.

Onisko, A., Druzdzel, M.J. & Wasyluk, H. 2001, 'Learning Bayesian network parameters from small data sets: application of Noisy-OR gates', *International Journal of Approximate Reasoning*, vol. 27, no. 2, pp. 165–82.

Ososky, S., Sanders, T., Jentsch, F., Hancock, P. & Chen, J.Y.C. 2014, 'Determinants of system transparency and its influence on trust in and reliance on unmanned robotic

systems', *Unmanned Systems Technology XVI*, vol. 9084, International Society for Optics and Photonics, p. 90840E.

Pan, S.J. & Yang, Q. 2009, 'A survey on transfer learning', *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–59.

Panwai, S. & Dia, H. 2007, 'Neural agent car-following models', *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 60–70.

Parasuraman, R. 2000, 'Designing automation for human use: empirical studies and quantitative models', *Ergonomics*, vol. 43, no. 7, pp. 931–51.

Parasuraman, R. & Riley, V. 1997, 'Humans and automation: Use, misuse, disuse, abuse', *Human Factors*, vol. 39, no. 2, pp. 230–53.

Pearl, J. 1985, 'Bayesian networks: A model of self-activated memory for evidential reasoning', *Proceedings of the 7th Conference of the Cognitive Science Society, 1985*, pp. 329–34.

Peffers, K., Tuunanen, T., Rothenberger, M.A. & Chatterjee, S. 2007, 'A Design Science Research Methodology for Information Systems Research', *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77.

Pereira, L.A.M. & da Silva Torres, R. 2018, 'Semi-supervised transfer subspace for domain adaptation', *Pattern Recognition*, vol. 75, pp. 235–49.

Perez, L. & Wang, J. 2017, 'The effectiveness of data augmentation in image classification using deep learning', *arXiv preprint arXiv:1712.04621*.

Puri, M.L. & Ralescu, D.A. 1986, 'Fuzzy random variables', *Journal of Mathematical Analysis and Applications*, vol. 114, no. 2, pp. 409–22.

Radecký, M., Gajdoš, P. & Fasuga, R. 2010, 'Agent Behavior Diagram for Intelligent Agents', *International Conference on Networked Digital Technologies*, Springer, pp. 333–41.

Reynolds, O. 2019, 'Towards Model-Driven Self-Explanation for Autonomous Decision-Making Systems', *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pp. 624–8.

Salas, E., Prince, C., Baker, D.P. & Shrestha, L. 1995, 'Situation awareness in team performance: Implications for measurement and training', *Human Factors*, vol. 37, no. 1, pp. 123–36.

Salas, E., Sims, D.E. & Burke, C.S. 2005, 'Is there a "big five" in teamwork?', *Small group research*, vol. 36, no. 5, pp. 555–99.

Saner, L.D., Bolstad, C.A., Gonzalez, C. & Cuevas, H.M. 2009, 'Measuring and predicting shared situation awareness in teams', *Journal of Cognitive Engineering and Decision Making*, vol. 3, no. 3, pp. 280–308.

Sanodiya, R.K. & Mathew, J. 2019, 'A framework for semi-supervised metric transfer learning on manifolds', *Knowledge-Based Systems*, vol. 176, pp. 1–14.

Sanodiya, R.K., Mathew, J., Saha, S. & Thalakottur, M.D. 2019, 'A new transfer learning algorithm in semi-supervised setting', *IEEE Access*, vol. 7, pp. 42956–67.

Sarter, N.B. & Woods, D.D. 2000, 'Team play with a powerful and independent agent: A full-mission simulation study', *Human Factors*.

Schmidt, K.W. & Boutalis, Y.S. 2012, 'Fuzzy discrete event systems for multiobjective control: Framework and application to mobile robot navigation', *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 5, pp. 910–22.

Selkowitz, A.R., Lakhmani, S.G. & Chen, J.Y.C. 2017, 'Using agent transparency to support situation awareness of the Autonomous Squad Member', *Cognitive Systems Research*, vol. 46, pp. 13–25.

Seppänen, H., Mäkelä, J., Luokkala, P. & Virrantaus, K. 2013, 'Developing shared situational awareness for emergency management', *Safety Science*, vol. 55, pp. 1–9.

Shapiro, A.F. 2009, 'Fuzzy random variables', *Insurance: Mathematics and Economics*, vol. 44, no. 2, pp. 307–14.

Shen, W. & Liu, S. 2003, 'Formalization, testing and execution of a use case diagram', *International Conference on Formal Engineering Methods*, Springer, pp. 68–85.

Shen, Z., Gay, R.K.L. & Miao, C. 2002, 'Goal-Based Modeling for Intelligent Business Forecasting Agents.', *The 1st International Conference on Fuzzy Systems and*

Knowledge Discovery: Computational Intelligence for the E-Age, pp. 534–8.

Shi, Y., Lan, Z., Liu, W. & Bi, W. 2009, 'Extending semi-supervised learning methods for inductive transfer learning', *2009 Ninth IEEE international conference on data mining*, pp. 483–92.

Shively, R.J., Lachter, J., Brandt, S.L., Matessa, M., Battiste, V. & Johnson, W.W. 2017, 'Why human-autonomy teaming?', *International conference on applied human factors and ergonomics*, pp. 3–11.

Shu, Y. & Furuta, K. 2005, 'An inference method of team situation awareness based on mutual awareness', *Cognition, Technology & Work*, vol. 7, no. 4, pp. 272–87.

Sklar, A.E. & Sarter, N.B. 1999, 'Good vibrations: Tactile feedback in support of attention allocation and human-automation coordination in event-driven domains', *Human factors*, vol. 41, no. 4, pp. 543–52.

Smith, K. & Hancock, P.A. 1995, 'Situation awareness is adaptive, externally directed consciousness', *Human factors*, vol. 37, no. 1, pp. 137–48.

Smithers, T. 1997, 'Autonomy in robots and other agents', *Brain and cognition*, vol. 34, no. 1, pp. 88–106.

Song, S., Yu, H., Miao, Z., Zhang, Q., Lin, Y. & Wang, S. 2019, 'Domain adaptation for convolutional neural networks-based remote sensing scene classification', *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 8, pp. 1324–8.

Stanton, N.A. 2006, 'Hierarchical task analysis: Developments, applications, and extensions', *Applied Ergonomics*, vol. 37, no. 1, pp. 55–79.

Stanton, N.A., Salmon, P.M., Walker, G.H., Salas, E. & Hancock, P.A. 2017, 'State-of-science: Situation awareness in individuals, teams and systems', *Ergonomics*, vol. 60, no. 4, pp. 449–66.

Stanton, N.A., Stewart, R., Harris, D., Houghton, R.J., Baber, C., McMaster, R., Salmon, P., Hoyle, G., Walker, G., Young, M.S., Linsell, M., Dymott, R. & Green, D. 2006, 'Distributed situation awareness in dynamic systems: Theoretical development and application of an ergonomics methodology', *Ergonomics*, vol. 49, no. 12–13, pp. 1288–311.

Stephenson, T.A. 2000, *An introduction to Bayesian network theory and usage*, IDIAP.

Sting, F.J. & Loch, C.H. 2016, 'Implementing operations strategy: How vertical and horizontal coordination interact', *Production and Operations Management*, vol. 25, no. 7, pp. 1177–93.

Sugeno, M. 1985, *Industrial applications of fuzzy control*, Elsevier Science Inc.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C. & Liu, C. 2018, 'A survey on deep transfer learning', *International conference on artificial neural networks*, pp. 270–9.

Taylor, R.M. 1988, 'Trust and awareness in human-electronic crew teamwork', *T. Emmerson, et al*.

Theodorou, A., Wortham, R.H. & Bryson, J.J. 2016, 'Why is my robot behaving like that? Designing transparency for real time inspection of autonomous robots', *AISB Workshop on Principles of Robotics*, University of Bath.

Vattam, S., Klenk, M., Molineaux, M. & Aha, D.W. 2013, 'Breadth of Approaches to Goal Reasoning : A Research Survey', *Goal Reasoning: Papers from the ACS Workshop*.

de Vries, P., Midden, C. & Bouwhuis, D. 2003, 'The effects of errors on system trust, self-confidence, and the allocation of control in route planning', *International Journal of Human Computer Studies*, vol. 58, no. 6, pp. 719--735.

Wang, N., Pynadath, D. V & Hill, S.G. 2016, 'The impact of pomdp-generated explanations on trust and performance in human-robot teams', *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*, pp. 997–1005.

Wang, X., Li, L., Ye, W., Long, M. & Wang, J. 2019, 'Transferable Attention for Domain Adaptation', *Proceedings of the AAAI Conference on Artificial Intelligence*.

Wijngaards, N., Kempen, M., Smit, A. & Nieuwenhuis, K. 2006, 'Towards sustained team effectiveness', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

Wortham, R.H. & Theodorou, A. 2017, 'Robot transparency, trust and utility', *Connection Science*, vol. 29, no. 3, pp. 242–8.

Wright, J.L., Chen, J.Y.C. & Lakhmani, S.G. 2019, 'Agent Transparency and Reliability in Human–Robot Interaction: The Influence on User Confidence and Perceived Reliability', *IEEE Transactions on Human-Machine Systems*.

Zhang, Y. & Yeung, D.-Y. 2012, 'Transfer metric learning with semi-supervised extension', *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, pp. 1–28.

Zhong, N., Liu, J., Bradshaw, J.M., Beautement, P., Breedy, M.R., Bunch, L., Drakunov, S. V., Feltovich, P.J., Hoffman, R.R., Jeffers, R., Johnson, M., Kulkarni, S., Lott, J., Raj, A.K., Suri, N. & Uszok, A. 2004, 'Making Agents Acceptable to People', *Intelligent Technologies for Information Analysis*.

Zhu, J., Liapis, A., Risi, S., Bidarra, R. & Youngblood, G.M. 2018, 'Explainable AI for Designers: A Human-Centered Perspective on Mixed-Initiative Co-Creation', *IEEE Conference on Computatonal Intelligence and Games, CIG*.

Zimmermann, H.-J. 2012, *Fuzzy sets, decision making, and expert systems*, vol. 10, Springer Science & Business Media.

Zuo, H., Lu, J., Zhang, G. & Liu, F. 2019, 'Fuzzy Transfer Learning Using an Infinite Gaussian Mixture Model and Active Learning', *IEEE Transactions on Fuzzy Systems*.

Appendix 1: Abbreviations

SA	=	Situation awareness
ASAS	=	Artificial situation awareness state
BCE	=	Backward chaining explanations
CPT	=	Conditional probability table
e-GDTA	=	Extended goal directed task analysis
FCE	=	Forward chaining explanations
GPS	=	Global positioning system
HAT	=	Human-autonomy teaming
NHTSA	=	National Highway Traffic Safety Administration
RPM	=	Revolution per minutes
SAF	=	Situation awareness failures
TL	=	Traffic light
TTC	=	Time to collision
TCDT	=	Time-constraint-driven transparency

Appendix 2: Fundamental Techniques

1. Introduction

Appendix 2 presents fundamental techniques used in this study including Bayesian Network, Fuzzy Theory, Manhattan Distance, and Neural Networks. The first three techniques are applied in the proposed self-explanation method. The fourth technique is used to develop an image classifier. The detail of these techniques is presented in the following sections.

2. Bayesian Network

Bayesian networks, firstly introduced by Pearl (1985), is a mathematical technique using graphical representation to connect nodes (variables) that can be categorized into three groups: root nodes, intermediate nodes, and leaf nodes. The input for Bayesian networks, *root nodes,* also called predictive variables, can be marginal, unconditional, or prior probabilities. *Intermediate nodes* are latent variables with conditional probabilities to represent the variables' states while *leaf nodes* represent the response variables, where their states are calculated as posterior probabilities. In Bayesian networks, all variables should be discretized where continuous values are transformed into a value range or discrete states. In nature, all these nodes can have discrete and mutually exclusive states.

Bayesian networks also consist of a set of directed edges among nodes which have three characteristics: no undirected edges, no edge with both directions among any nodes, and no edge that forms a cycle by going back to the original node (Stephenson 2000). The relationship links parent nodes and child nodes to develop causal connections where each parent node has an effect on its child node. Therefore, for any Bayesian network with given nodes $X = X_1, X_2, \ldots, X_n$, the function of join probability distribution $P(X)$ is represented as follows:

$$P(X) = \prod_{i=1}^{n} P\left(X_i \middle| parents(X_i)\right) \qquad (1)$$

When the parent node $parents(X_i)$ is an empty set, then $P(X_i|parents(X_i)) = P(X_i)$ and X_i is a root node which indicates its previous probability (Naderpour, Lu & Zhang 2013). The probability of an event caused by another occurrence that has already occurred is defined as conditional probability (van Koten & Gray 2006). Therefore, according to Bayes' theorem, the conditional probability of, for example, node B given the outcome of occurrence in node A, $P(B|A)$, can be denoted as follows:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \quad (2)$$

where $P(A|B)$ is the conditional probability of event A given B, $P(A)$ and $P(B)$ is the probability of A and B, respectively. To maintain this dependency, Bayesian network provides a Conditional Probability Table (CPT) to give a complete probabilistic interaction specification between child nodes and their parent nodes (Onisko, Druzdzel & Wasyluk 2001). After CPT is set for all possible options of related events, Bayesian networks can calculate the joint probability distributions $P(B, A)$ as follows:

$$P(B, A) = P(B|A)P(A) \quad (3)$$

$P(A)$ and $P(B|A)$ are the prior probability distribution and likelihood probability distribution respectively. These two probability distributions are what we have in order to get a posterior probability distribution that can be denoted as $P(B = b, A = a)$. Furthermore, the marginal probability in node B ($P(B = b)$) can be written as follows:

$$P(B = b) = \sum_a P(B = b, A = a) = \sum_a P(B = b \mid A = a)P(A = a) \quad (4)$$

To validate the Bayesian network model, sensitivity analysis can be performed by calculating entropy reduction with categorical variables or ordinal scale. Entropy reduction, I, becomes the expected reduction within the mutual information of Q which is obtained from a finding for variable F and is denoted as follows:

$$I = H(Q) - H(Q|F) = \sum_q \sum_f \frac{P(q, f)log_2[P(q, f)]}{P(q)P(f)} \quad (5)$$

where $H(Q)$ and $H(Q|F)$ represent the entropy of Q before and after the new findings from variable *F,* respectively. Sensitivity analysis produces a rank-order of evidence nodes allowing a quantitative comparison so that entropy (uncertainty) or variance can be reduced in a target outcome variable.

3. Fuzzy Theory

Sensing observable variables of objects from the environment often contains fuzziness which indicates there are no certain boundaries to describe the object's states given a group of elements x as the result of sensor reading (Berthold 1999). Based on fuzzy theory, a fuzzy set can be used to represent such fuzziness by using membership functions. For example, let Z be a fuzzy set so that one can define the membership of $\forall x \in Z$ using the characteristic function $\mu_Z(x) \in [0,1]$, $0 \leq \mu_Z(x) \leq 1$ (Zimmermann 2012). While $\mu_Z(x) > 0$ indicates the membership of x to Z, $\mu_Z(x) = 0$ shows non-membership. For instance, let x represent a temperature which will be mapped to the fuzzy set Z representing a hot temperature. When $x = 40$, it might be considered as still warm, but it is almost hot. Therefore, this temperature may have the degree of membership $\mu_Z(x) = 0.7$ in Z. In real applications, the most commonly used membership function shapes are triangular, trapezoidal, singleton, and Gaussian. Figure 1 represents an example of triangular shapes which is denoted by the triple (a_1, a_2, a_3).

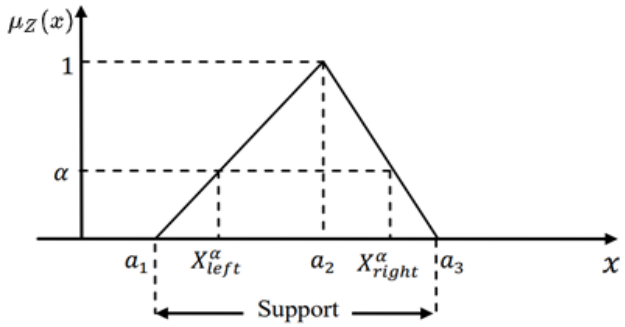


Figure 1. An example of fuzzy α-cut and fuzzy set support

In the universe of discourse X, the support of a fuzzy set Z is a crisp set that comprises all elements that have nonzero membership values in Z, so that $supp(Z) = \{x \in X | \mu_Z(x) > 0\}$ where $supp(Z)$ denotes the support of fuzzy set Z. Then, the α-cut or α-level set of

the fuzzy set Z is the crisp set Z_α defined by $Z_\alpha = \{x \in X | \mu_Z(x) \geq \alpha\}$ (Shapiro 2009). A fuzzy set Z in $\mathbb{R}$ satisfies the following conditions:

- Z is normal
- Z_α is a closed interval for every $\alpha \in (0,1]$.
- The support of Z is bounded

Figure 1 shows an example of the support of fuzzy set Z which is all x such that $\mu_Z(x) > 0$, and its α-cut is X^α_{left} from to X^α_{right}. Values outside this range are considered as a cutout values which might be excluded from consideration., a set of elements within the interval of α-cut is considered as the domain under consideration. Several α-cuts can be applied, and they produce the crisp state set $\mathcal{P} = (p_1, p_2, \dots, p_n)$. Each fuzzy state can be written as a vector $q = [q_1, q_2, \dots, q_n]$, where $q_i \in [0,1]$. Thus, each fuzzy state can be considered as a fuzzy set $\mathrm{q} \in \mathcal{F}(\mathcal{P})$ where $\mathcal{F}(\mathcal{P})$ is the set of all fuzzy subsets defined for $\mathcal{P}$ (Schmidt & Boutalis 2012). Alternatively, a fuzzy state can also be viewed as a possibility distribution.

An example of the use of a fuzzy set can be found in a fuzzy logic system environment (see Figure 2). In this system, the fuzzification process forms all the input variables as fuzzy sets. Then, an inference engine processes all the input variables by applying fuzzy logic operations representing the logic relations among variables to generate the output fuzzy set. Through the defuzzification process, this output is converted into a crisp value (Markowski et al. 2011).

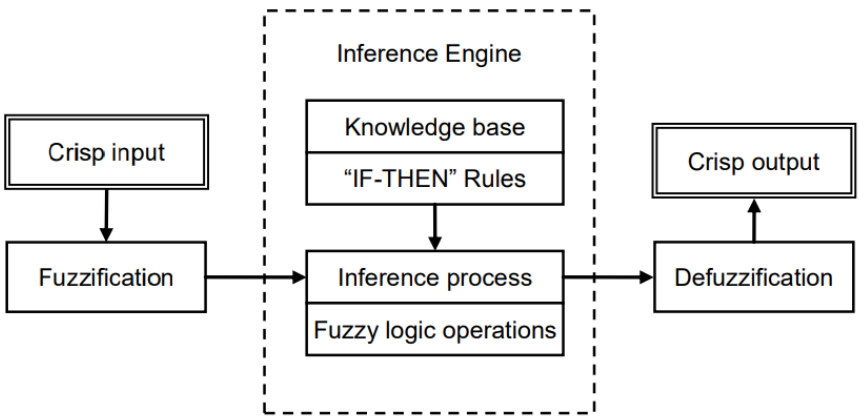


Figure 2. Fuzzy logic system

The most commonly used inference methods for a fuzzy logic system are Mamdani (1976) and Sugeno (1985). Mamdani's method is also known as the Max-Min fuzzy-rule-based inference, and Table 1 shows its characteristics.

Table 1. Characteristics of Mamdani's inference method

Operation	Operator	Formula
Union (OR)	MAX	$\mu_C(x) = \max(\mu_A(x), \mu_B(x)) = \mu_A(x) \vee \mu_B(x)$
Intersection (AND)	MIN	$\mu_C(x) = \min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x)$
Implication	MIN	$\min(\mu_A(x), \mu_B(x))$
Aggregation	MAX	$\max(\min(\mu_A(x), \mu_B(x)))$
Defuzzification	CENTROID	$COA = Z^* = \frac{\int z\, \mu_C(z)\, \mathrm{dz}}{\int \mu_C(z)\, \mathrm{dz}}$

$\mu_C(x)$ = value of the resultant membership function.
$\mu_A(x)$ = value of the membership function where the input belongs to the fuzzy set A.
z = abscissa value, ($\mu_C(z)$ is the ordinate)

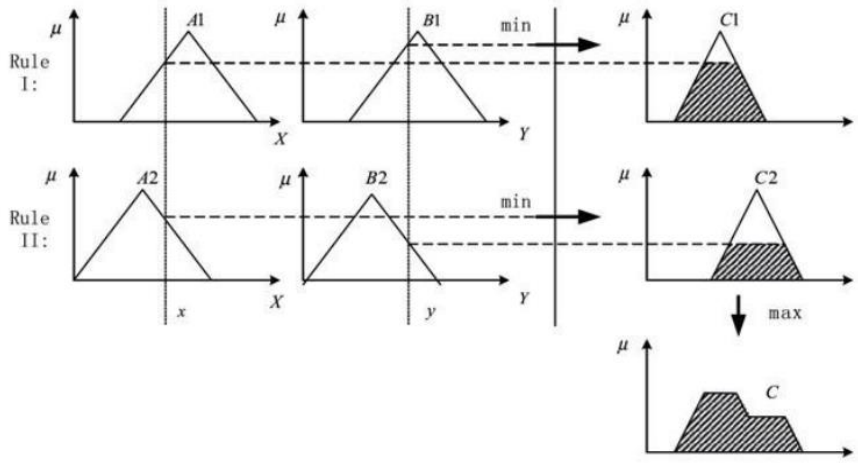


Figure 3. Mamdani's inference method with two inputs and one output

In summary, Figure 3 illustrates an example case of Mamdani's fuzzy inference method. This case has two rules, two inputs, and a single output. The fuzzification process in this method transforms the two inputs to fuzzy sets by finding the intersection between the crisp input value and its membership function. The inference engine uses the MIN operator to compute the fuzzy, AND operator to combine the two fuzzified inputs in order to obtain a rule strength. Then, this engine clips the output membership function at the rule strength. Finally, it uses the MAX operator to compute the fuzzy and OR operator to combine the outputs of the two rules. Sugeno's method is similar to Mamdani's. However, the main difference is that the output consequence is not computed by clipping the output membership functions at the rule strength. In fact, Sugeno's method has no

output membership function at all. The output in Sugeno's method is a crisp number obtained by multiplying each input by a constant and adding up the results.

A fuzzy relation can be described as the strength of the relation mapping between two fuzzy sets. This relation is expressed by the degree of the membership function of the relation (μ_R). Let M in $X \times Y$ be a fuzzy set representing an object which is related to another object N in $Y \times Z$. The relation composition between M and N can be described either using max-min composition or max-product composition. In max-min composition, the relation $M \circ N$ will be in $X \times Z$ such that:

$$M \circ N \leftrightarrow \mu_{M \circ N}(X,Z) = max\{min\{\mu_M(X,Y), \mu_N(Y,Z)\}\}/\{X,Z\} \quad (6)$$

The max-product composition of $M \circ N$ can be denoted by:

$$M \circ N \leftrightarrow \mu_{M \circ N}(X,Z) = max\{\mu_M(X,Y) \cdot (Y,Z)\}/\{X,Z\} \quad (7)$$

The fuzzy set operations $min(a,b)$ and $max\ (a,b)$ are widely known as logical sum and logical product, respectively (Mizumoto & Tanaka 1981).

4. Manhattan Distance

Manhattan distance is a similarity function f which compares two sets of attributes i.e. $q_{i,y}$ and $q_{j,y}$ so that $f(q_{i,y}, q_{j,y})$ can be denoted by Equation (8) (Asfoor et al. 2015). In this regard, i and j indicate the index of the attributes to be compared, while y represents the size of the matrices of each attribute set. The $range\ (a_y)\ is$ defined in Equation (9).

$$f(q_{i,y}, q_{j,y}) = 1 - \frac{|q_{i,y}, q_{j,y}|}{range\ (a_y)} \quad (8)$$

$$range\ (a_y) = \max\{q_{i,y} |\ i\ \in \{1,..,n\}\} - \min\{q_{i,y} |\ i\ \in \{1,..,n\}\} \quad (9)$$

$$f(q_{i,y}, q_{j,y}) = \begin{cases} 1, & if\ q_{i,y} = q_{j,y} \\ 0, & otherwise \end{cases} \quad (10)$$

When the result of similarity function $f(q_{i,y}, q_{j,y})$ is characterized by Equation (10), if the output of function equals 1, this means that the two sets are equal.

5. Neural Network

The simplest neural network consists of a single input layer and an output node, and this network is called the perceptron (Aggarwal 2018), as illustrated in Figure 4. In this figure, the input layer contains *d* features $\bar{X} = [x_1, x_2, \ldots, x_d]$ which are connected to an output node by the edges of weight $\bar{W} = [w_1, w_2, \ldots, w_d]$. The computation in the perceptron is a linear function that computes $\bar{W} \cdot \bar{X} = \sum_{i=1}^{d} w_i x_i$ at the output node. Now, assume that a sign function is used to map a real value to either +1 or –1 in a binary classification, the prediction $\hat{y}$ as a dependant variable of $\bar{X}$ will be computed as $\hat{y} = sign\{\bar{W} \cdot \bar{X}\}$.

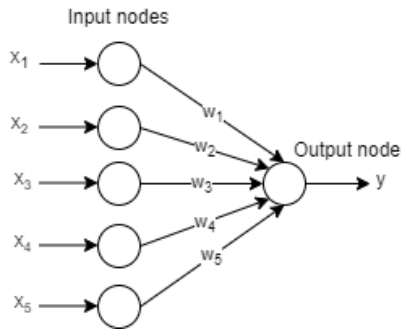


Figure 4. The basic architecture of perceptron

A neural network imitates the function of the human brain where neurons are connected to each other and are composed of a weighted sum of an activation function (Caterini 2018). Selecting an activation function is an essential part of a neural network design, particularly when the network becomes more complex with multilayered architectures. The sign function is one example of a non-linear activation function. Now, let use the notation Φ to denote the activation function, so that $\hat{y} = \Phi\{\bar{W} \cdot \bar{X}\}$. There are at least 5 popular variants of activation functions namely the linear function, sigmoid function, tanh function, rectified linear unit (ReLU), and softmax function (Galushkin 2007).

A linear function is the most basic activation function and it is also called identity activation which provides no nonlinearity, so that $\Phi(v) = v$. This function has range of –infinity to +infinity and only the output node uses this function. Furthermore, the sigmoid function can be denoted by $\Phi(v) = \frac{1}{1+e^{v}}$ and is a non-linear activation function. This function has value range from 0 to 1. Therefore, this function is popular for binary

classification and is computed in the output layer. When the result of this function is greater than 0.5, $\hat{y}$ will be predicted as *true* (1) and 0 otherwise (Hansen & Salamon 1990).

Furthermore, the tanh function is a non-linear activation function, and is formulized by $\Phi(v) = tanh(v)$. While linear and sigmoid functions are computed at the output node, the tanh function is used in the hidden layers. The value range of the tanh function is from –1 to 1. This function helps in maintaining the mean of data close to 0, and as a consequence, learning for the next layer would be much easier. In addition to the tanh function, ReLU is also implemented in the hidden layers and it is denoted by $\Phi(v) = max(0, v)$. Therefore, the range value of this function is from 0 to infinity. ReLu is the most widely used activation function as it has a simpler mathematical operation and low computational cost than tanh and sigmoid. Like sigmoid but more useful in handling the classification problem, the softmax function is widely known for multiple class prediction. The softmax function is ideally computed in the output layer and its range value is from 0 to 1. An example of a neural network with multilayers including input layers, hidden layers, and softmax layers can be seen in Figure 5.

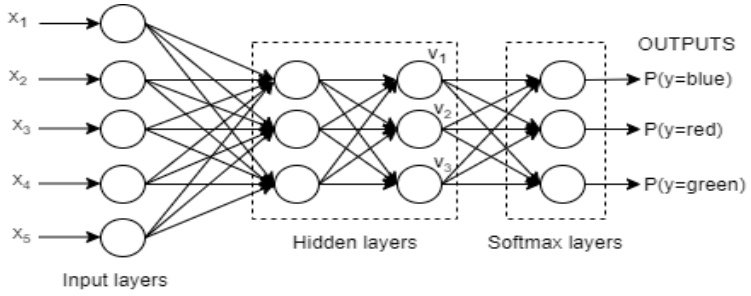


Figure 5. Neural network with multilayers and multi-outputs for categorical classification with the softmax function

Multilayered neural networks have additional layers between the input layer and output layer, and these intermediate layers are referred to as hidden layers (Hinton et al. 2012). Different from input and output layers, the computations in the hidden layers are not visible to the user. One type of multilayered neural network architecture is the feed-forward network. This architecture assumes that the data flows in the forward direction from input to output. However, the problem with multilayered neural networks is that the error can be calculated as a direct function allowing an easy gradient in its computation

(Aggarwal 2018). As the error gradient can be considered as a summation of local-gradient products which are computed over the various path from earlier nodes to the output node, this gradient contains an exponential number of paths. In this regard, an efficient computation can be undertaken by implementing dynamic programming which is widely known as the backpropagation algorithm.

There are two phases in the backpropagation algorithm, namely forward and backward phases. In the forward phase, all the inputs are sent to the neural networks, and the computation across layers in the networks produces a set of weights (Aggarwal 2018; Hecht-Nielsen 1992). The predicted output from this phase will be compared to the training instance and generates the loss value. This loss value is computed using a loss function. The second phase is responsible for learning the gradient of the loss function, and the backpropagation algorithm uses this gradient to update the network weights (Chen 1990). As the gradient learning process starts from the output node and goes in a backward direction, this process is called the backward phase. In most deep learning platforms such as Keras, the training process uses a backpropagation neural network.

Appendix 3: Process-based Method

1. Introduction

The process-based method used to evaluate the newly proposed method is adopted from Al-Ajlan (2015). In this method, the behaviour explanations are generated by observing the algorithms and logics of an autonomy agent. Let assume that a system (*S*) of an autonomy agent (see Figure 1) has three major components, namely user interface, inference engine, and knowledge base. The input of an autonomy agent is the state of objects from the surrounding environment that related to its task. The actual output is the action of the autonomy agent and its rationale produced by the self-explanation feature. With this system, only two parts – knowledge base and inference engine – are reviewed.

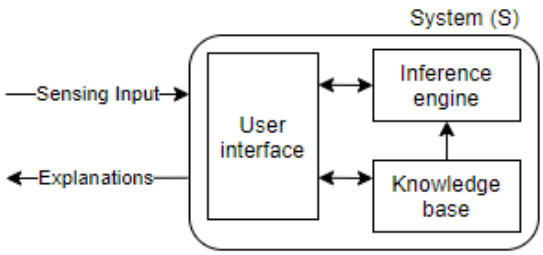


Figure 1. A system of an autonomy agent

a. Knowledge Base

In this process-based method, Knowledge base (*KB*) holds the necessary facts and knowledge for a specific assignment to autonomy agent, and a set of rules for applying these facts and knowledge. A rule itself can be described as a conditional (if – then – else) structure that provides logical expression where the *antecedent* is given after *if*, the *consequent* is delivered after *then*, and the *alternative* is specified after *else*. When the conditions are met, the consequent will be executed. Otherwise, the alternative will be delivered. Mathematically, let x denote the notation for *condition*, y and z represent *consequent* and *alternative,* respectively. The function of conditional structure can then be denoted as follow:

$$f(x,y,z) = \begin{cases} y, & x = true \\ z, & x = false \end{cases} \quad (1)$$

A production rule for the knowledge base, can also be defined as a statement which has the following forms:

<production rule> ::= **if** <antecedent> **then** <consequent> **else** <alternative>
<antecedent> ::= <disjunction> {**and** <disjunction>}*
<disjunction> ::= <condition> {**or**<condition>}*
<consequent> ::= <conclusion> {**also**<conclusion>}*
<alternative> ::= <conclusion> {**also**<conclusion>}*
<condition> ::= <predicate>(<variable>,<constant>)
<conclusion> ::= <action>(<variable>,<constant>)
<predicate> ::= same | not same | greater than | …
<action> ::= action1 | action2 | …

b. Inference Engine

The inference engine allows the system to use the knowledge in *KB* to infer the conclusion by relating the input to rules stored in *KB*. In this regard, conclusion can be an action of an artificial agent. While the system processes the input to produce a conclusion through a set of decision points, it can be seen as a forward chaining process. Consequently, the role of inference engine in this context is to deduce input using the encoded rules. Let assume that $P \rightarrow R$ be a rule so that:

$$(P;\ P \rightarrow R)\ \Rightarrow\ R$$

which means that if P is *true*, and the rule $P \rightarrow R$ is *true* as well, that it can be derived that R is also *true*. This method of reasoning is also called *modus ponens*. Now, let assume that INP is a set of sensing inputs, $INP = \{inp_1, inp_2, \dots, inp_k\}$ where k is the total number of input. Then, let RL be a set of rules $RL = \{rl_1, rl_2, \dots, rl_j\}$ where j is total the number of rules. Generating a conclusion can be done by the following steps:

1) Examine the conditions of the rules which match the inference engine input INP. In this regard, the set of applicable rules is referred to as a conflict set.

2) Perform incremental forward chaining until no more rules in RL that can be applied for INP

3) Stop incremental forward chaining when the condition holds, otherwise go back to step 1

When the autonomy agent receives input from its sensory tools, its actions will be determined by the forward chaining process. Based on the conclusion generated in this process, an explanation about why the autonomy agent executed this action will be produced. The behaviour explanation generator can be considered as a part of inference engine which provides explanations in the backward chaining mode. For every specific assignment, the rule-based knowledge can be modeled in a decision tree as a graphical representation of rules to make decisions. Therefore, performing backward chaining explanation can be done by tracing back the decision path within the decision tree which has been used to obtain the conclusion/action of the autonomy agent. When tracing back, the abduction technique is used to check the relation among decisions. Let assume $(R;\ P \rightarrow R)\ \Rightarrow\ P$ where R is the conclusion and P is its antecedent. By knowing that the state of R is true and it is known from the decision tree that there is a rule $P \rightarrow R$ which is also true, then it can be derived by abduction that P is also true.

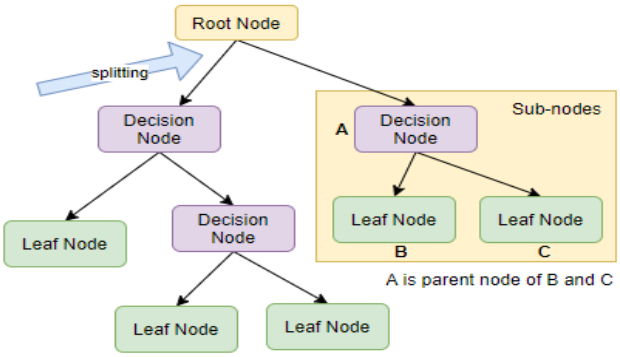


Figure 2. Decision Tree

Furthermore, to crawl every decision node in the decision three, the depth-first search (DFS) technique is used. Let $G = (V, E)$ be a graph of the decision three with n vertices where V and E represent vertex set and edge set, respectively. In this regard, vertices are used to represent root node, decision nodes, and leaf nodes. Root node is a term to represent first decision point which receives the input from agent's sensory tools. Nodes in the decision tree are connected by the edge which can be assumed as a splitting process of a decision node. Nodes which do not split are called leaf nodes; these nodes represent

the conclusion/action of the autonomy agent. Nodes which are divided into sub-nodes are called parent nodes and the sub-nodes are called child nodes (see Figure 2).

Crawling the decision tree is started from a leaf node. This leaf node is an end state of a forward-chaining process which can be considered as an action of an autonomy agent. It should be noted that in the forward-chaining process, the visited decision nodes within the tree are also stored in a working memory. Now, let assume that in G, a set of visited elements of V in forward-chaining process, $ð = (v_1, v_2, \dots v_n)$, exists for $v \in V$. The search process in DFS is intended to examine the connections of each v either to the selected conclusion vertex or to other visited elements as written in the adjacency list (ADJ) linking an edge to every vertex in G. Finding this connection is critical as it creates causal links following the conclusion states. After visiting the conclusion vertex, the search will check other vertices in ð as listed in the adjacency list. The search, then, will check the edge that connects current vertex with another vertex in ð. If it exists, it will proceed to the next linked vertex. When all connections among nodes in ð have been established, the search will be terminated. This process can be represented by the algorithm as follows:

Algorithm 1: Examine the visited nodes to create causal links

Input: $ð = (v_1, v_2, \dots v_n)$ and $root$
Output: $causal\_links[]$, i.e. $[conclusion \leftarrow v_1 \leftarrow v_2 \dots \leftarrow v_n]$
Procedure:

```
1   parent = root
2   causal_link.add(parent)
3   foreach v ∈ ð
4     If check_adj((parent, v)) = true
5       causal_links.add(v)
6       parent = v
```

Once the causal links are created, explanations will be generated by giving the meaning on the causal connections. This step is important as ð only has information about the conditions in a decision node. Based on the production rule, a condition consists of *predicate*, *variable*, and *constant* where the *predicate* represents basic operations such as equal to, greater than, and less than. Consequently, a reference which describes what is the purpose of every variable in *predicate* should be defined. This reference should also describe the meaning of combination of variables as listed in causal links. In other words, the description of each decision path must be well-defined in this reference. The

description of all possible decision paths, then, becomes the behaviour explanations of the autonomy agent.

2. Implementation in Traffic Light Scenario

a. Modelling the Autonomy Agent's Logics at Process Level

In this section, the processes determining the autonomy agent's behaviours in TL situations are modeled by referring to the United States patent proposed by Deering (1992). This patent is intended to determine the deceleration/acceleration value based on a free ride or tailing situation. The process flow from this patent is adjusted for TL situation.

Based on the type of traffic light situation in Chapter 4, the autonomy agent's behaviours in TL situations can be divided into three basis manoeuvres, namely *LV*-based manoeuvre, TL-based manoeuvre, and free ride manoeuvre. The *LV*-based manoeuvre (also referred to as tailing manoeuvre) is illustrated in Figure 3 which also presents the main variables used to determine the autonomy agent's behaviours. Initially, the process in *LV*-based manoeuvres reads the speed (V_0) and acceleration (A_0) of the vehicle controlled by the autonomy agent, which is the host vehicle (*HV*). From *LV*, the most critical variable is its travelling speed (V_T). Based on the relative speed of both *LV* and *HV*, a value for a variable called *LV* range rate (*LVRR*) can be generated including the actual distance between *HV* and *LV* which is denoted by $Range_A$. Based on these variables, the main target of this *LV*-based manoeuvre process is to determine the acceleration/deceleration of *HV* so that *HV* can be within desired range ($Range_D$) to *LV* with minimum distance of D_{min}.

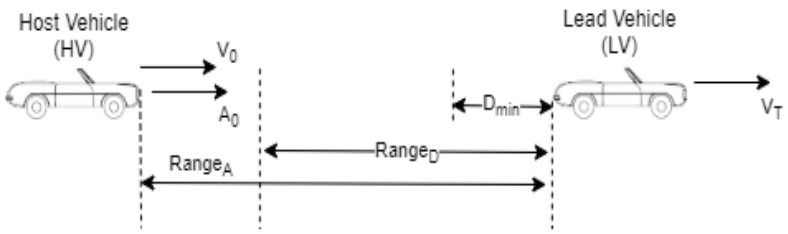


Figure 3. Behaviour's main variables for LV-based manoeuvres

Furthermore, the TL-state-based manoeuvre (see Figure 4) is intended to perform an appropriate behaviour according to TL colour recognized by the autonomy agent. However, in this regard, this study refers to TL-state-based manoeuvre as stopping manoeuvre because it is only applied when the combination of both red light and free ride situation is detected. Other than such combination, the behaviours will follow the free-ride manoeuvres or tailing manoeuvres depending on whether there is a LV or not. Like LV-based-manoeuvre, the process in stopping manoeuvre is started by reading the V_0 and A_0 of HV, and also the TL location. However, while in LV-based manoeuvre LV has variable V_T, it is assumed that this variable equals to zero (0) in TL-based manoeuvre as it is a static object. This is a critical assumption to measure the range rate between HV and TL, which referred to as TL range rate ($TLRR$). the actual distance between HV and TL is represented by $dHVTL$. HV can be in segment 2 or segment 1. When $dHVTL$ is greater than the $S1 - Range$, which is a variable to denote the segment 1 range, it means that HV is in segment 2. The target of TL-state-based manoeuvre process is to provide the deceleration value of HV due to the red light so that it can stop at an estimated point before TL denoted by DTL_{min}.

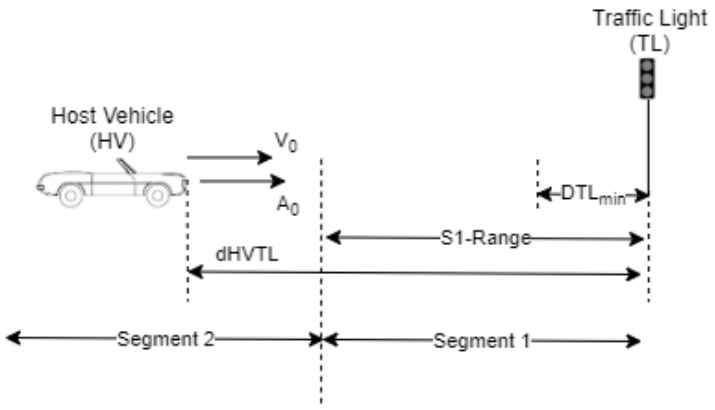


Figure 4. Behaviour's main variables for TL-state-based manoeuvres

When no LV is detected and the TL state is not red, the free ride manoeuvre is executed. Unlike the other two manoeuvres, stopping and tailing manoeuvres, the process of free ride manoeuvre is simply intended to accelerate or decelerate vehicle so that the target speed can be achieved (see Figure 5). In this regard, the target speed can be either the road speed or the desired speed set by the driver. The road speed is also depending on whether the vehicle is currently on, for example, urban area or highway. However, in this study, the process to acquire the road speed will not be detailed.

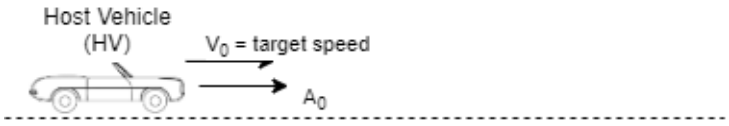


Figure 5. Behaviour's main variables in free ride situations

This study determines three main processes of the autonomy agent as presented in Figure 6. First, the initialization process is executed which is intended to establish the communication link to sensory tools, to allocate the memory for the running process, and to synchronize data acquisition from sensors. The second process is to make the overall process aware of the acknowledgement from the engine interrupt. This interrupt is useful to propagate the control switching information, i.e., from the manual control to autopilot control. the background loop process is intended to perform all the manoeuvre processes including stopping manoeuvre, tailing manoeuvre, and free ride manoeuvre. This process runs repeatedly following the sensor reading synchronization frequency.

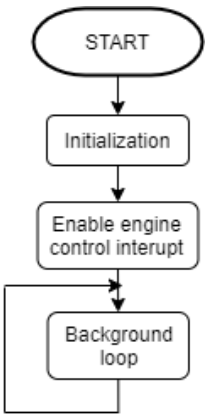


Figure 6. Main processes

Due to the complexities, only selected processes that will be modeled. For the main processes, this study only highlights the detail of background loop process. the detail of the other two main processes is not covered in the behaviours model. Similarly, even though the background loop process has numerous sub-processes which consist of routines, not all of them will be accommodated in the behaviour model. This study only focuses on a routine called automatic control routine.

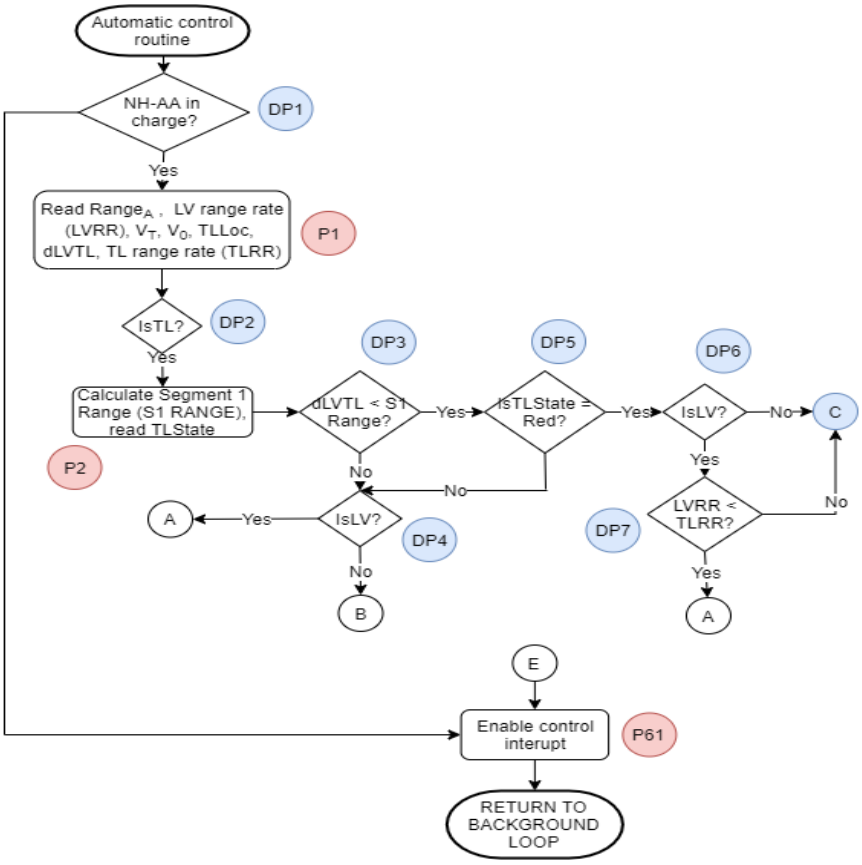


Figure 7. A Process within background loop: Automatic control routine

Automatic control routine is intended to select a manoeuvre that will be taken given input from sensor reading (see Figure 7). This routine starts from continuously checking on whether the autonomy agent is activated to control the vehicle. If the autonomy agent is in control, it will be supplied with information associated with variables $Range_A$, $LVRR$, V_t, V_0, $TLLoc$, $dlLVTL$, and $TLRR$. After obtaining this information, the routine undergoes five decision points before it determines the appropriate manoeuvre. The description of each decision point state can be seen in Table 1.

Table 1. Decision points in automatic control routine

Decision point symbol	Decision point	States and description
ACR-D1	The autonomy agent in charge?	*Yes*: the vehicle is controlled by autonomy agent *No*: the vehicle is in manual mode
ACR-D2	Is TL?	*Yes*: TL is detected for oncoming traffic *No*: Vehicle is not under TL situation
ACR-D3	Is dLVTL < S1 Range	*Yes:* the vehicle is in segment 1 *No:* the vehicle is in segment 2
ACR-D4	Is TL State = red?	*Yes*: red light

Decision point symbol	Decision point	States and description
		No: green light, or yellow light, or unknown
ACR-D5	Is LV?	*Yes*: LV ahead is detected No: No LV ahead
ACR-D6	Is LVRR < TLRR?	*Yes*: LV before TL location *No*: LV after TL location

It can be seen in Figure 7 that either in segment 1 or segment 2, when no LV situation is combined with all TL states except red light, this combination will cause the routine to execute free ride manoeuvre (proceeds to B). no matter in segment 1 or segment 2, as long as LV exists, tailing manoeuvre is selected (proceeds to A). When no LV before TL and TL state is red, stopping manoeuvre is taken (proceeds to C). With all these decision points, all types of TL situations identified in this study have been accommodated. After selected manoeuvres have been executed, the process proceeds to enable control interrupt. At this point, the automatic control routine will be restarted as the background loop process is going to the next cycle. In automatic control routine, every decision point is denoted by the symbol DP along with its number. the process is denoted by the P and the process number. When the process symbol and number is not given to a certain sub-process of the automatic control routine, such as calculating A_D, it means that this process will be detailed in another figure. In total, this routine consists of 37 DP and 61 P. It should be noted, that within every P, DP may exist, but it is not presented and discussed.

Within the automatic control routine, this study details the manoeuvre execution processes (tailing, stopping, and free ride) as the core of autonomy agent's behaviours. Under the tailing situations, the behaviour of autonomy agent is driven by the process illustrated in Figure 8. After tailing manoeuvre is selected, the process reads T_{REACT} from the system setting. T_{REACT} is a variable which stores a predetermined driver reaction time. This variable is a prerequisite in calculating the desired space between LV and HV represented by $RANGE_D$. $RANGE_D$ itself is generated by a function which combines the relative speed and range of HV and LV, including the range rate and T_{REACT}. When the value of T_{REACT} is large, it results in a larger following distance. However, when it is too small, it will be overridden by a variable called D_{MIN}.

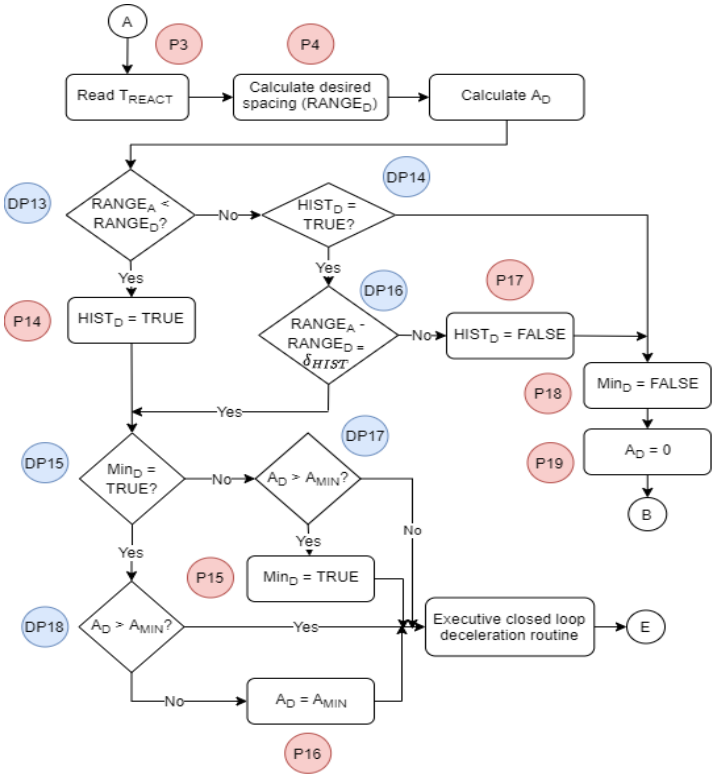


Figure 8. LV-based (tailing) behaviours

After $RANGE_D$ is calculated, the routine process proceeds to calculate a necessary deceleration value (A_D). After the routine obtains A_D value, the routine once again, compares the $RANGE_D$ with the actual range $RANGE_A$. If the actual range is less than the desired range, then a flag called $HIST_D$ will be set to *true* to give a mark for the next cycle of background loop process. In the next cycle, when both flags $HIST_D$ are *true* and the actual range equals to or is greater than the desired range, the actual range value will be subtracted by the desired range value. In this regard, the tailing manoeuvre only exists when the $RANGE_A$ exceeds the sum of the hysteresis factor (δ_{HIST}) and $RANGE_D$. Otherwise, the flag will be set to *false*, the value of A_D will be cleared, and free ride manoeuvres are executed. While the flag $HIST_D$ is active and the hysteresis factor exists, the process goes to examining whether the minimum deceleration active flag (MIN_D) is checked. If checked, A_D will be compared to A_{MIN} which is the minimum predetermined level of deceleration. If it is less than A_{MIN}, A_{MIN} will be set as A_D. But, if the flag MIN_D is not checked and A_D is greater than A_{MIN}, this flag will be activated.

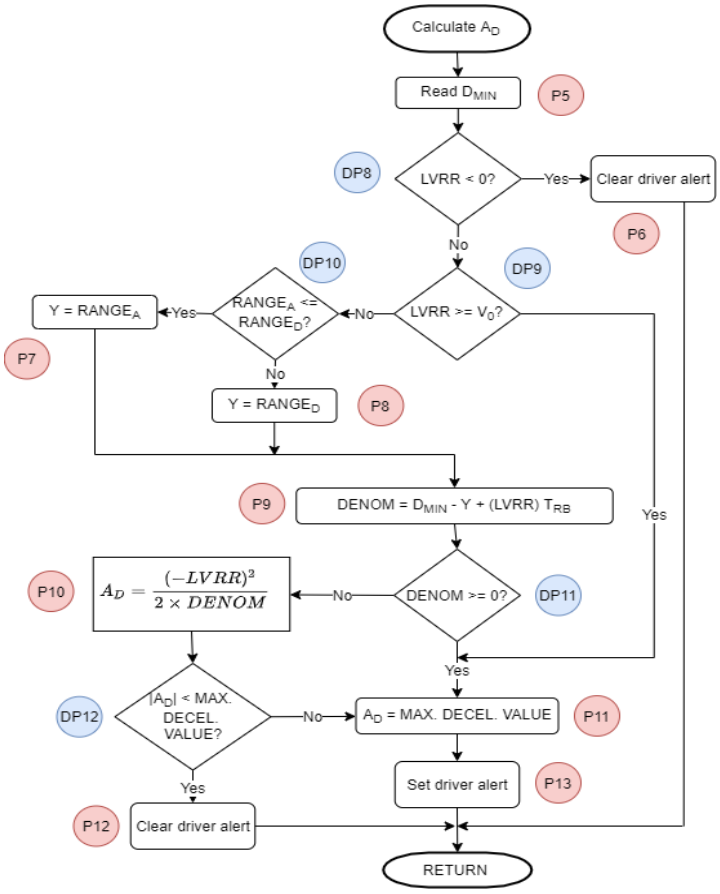


Figure 9. Deceleration process in LV-based manoeuvres

The detail process in calculating A_D is presented in Figure 9. First, this process reads D_{MIN} to infer whether $LVRR$ is less than zero (0) which means that HV is moving away from LV so that it is not necessary to calculate A_D. Calculating A_D is only needed when HV is approaching LV. When HV is not closing on LV, the routine can clear the driver alert. If $LVRR$ equals to or is greater than zero, this range rate will be compared to the current vehicle speed (V_0).

When $LVRR$ is less than V_0, $RANGE_A$ and $RANGE_D$ values are contrasted to select which range between these two that will be used to calculate A_D. The selected range will be stored in variable Y. Based on the value of variable D_{MIN}, Y, $LVRR$, and T_{RB}, the denominator ($DENOM$) is calculated. $DENOM$ is an important expression to obtain the value of A_D that considers the driver's brake reaction time factor (T_{RB}). T_{RB} is a setting regulating the responsiveness of the vehicle braking system which is proportionally associated with $RANGE_D$. The higher value of $RANGE_D$ will produce the larger braking reaction time for a situation where the braking is needed.

The $DENOM$ equals to or greater than zero means that the maximum deceleration value will be used as A_D. In this regard, driver alert will be sent. Otherwise, A_D is produced from the division of $-LVRR^{\,2}$ by $2 \times DENOM$. Taking the maximum deceleration value as A_D is also applied when $LVRR$ equals to or greater than V_0. After the value of A_D is obtained from the process in Figure 9, the routine will return to the process in Figure 8 which performs necessary checking for A_D value that has been generated.

Furthermore, stopping manoeuvres are driven by the process illustrated in Figure 10. In general, the process is similar to LV-based manoeuvres with any necessary adjustment regarding variables used in the process. For example, in LV-based manoeuvre process, V_T is used to represent the speed of LV. In this process, variable V_T is replaced by V_{TL} and this variable has a static value which is zero as TL is a static object. Moreover, while TL-based manoeuvre uses $RANGE_A$ and $RANGE_D$, in this process they are replaced by $dLVTL$ and $S1\ RANGE$ respectively. Therefore, the calculation of A_{DTL} (see Figure 11) will be conceptually the same as the calculation of A_D.

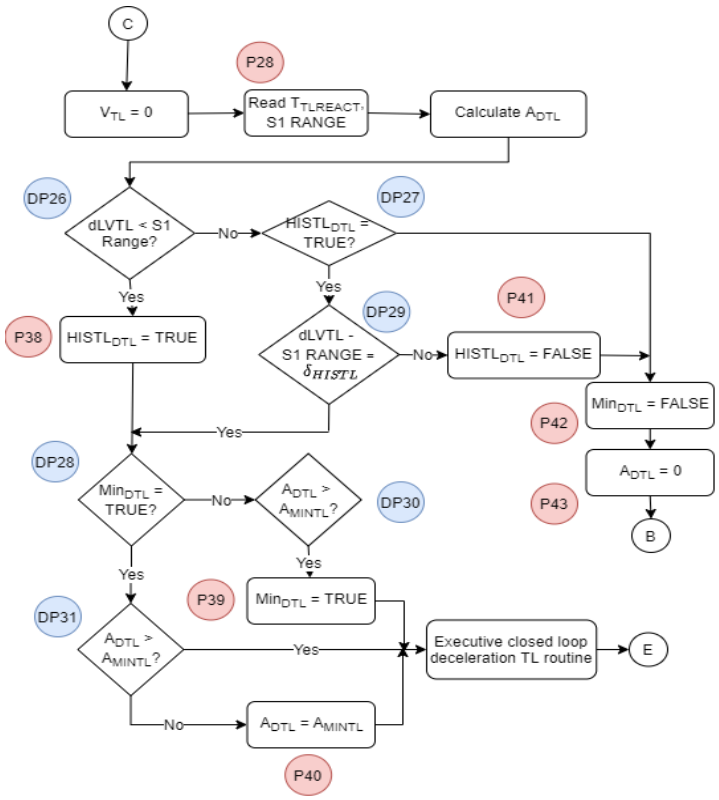


Figure 10. TL-state-based behaviours (stopping manoeuvres)

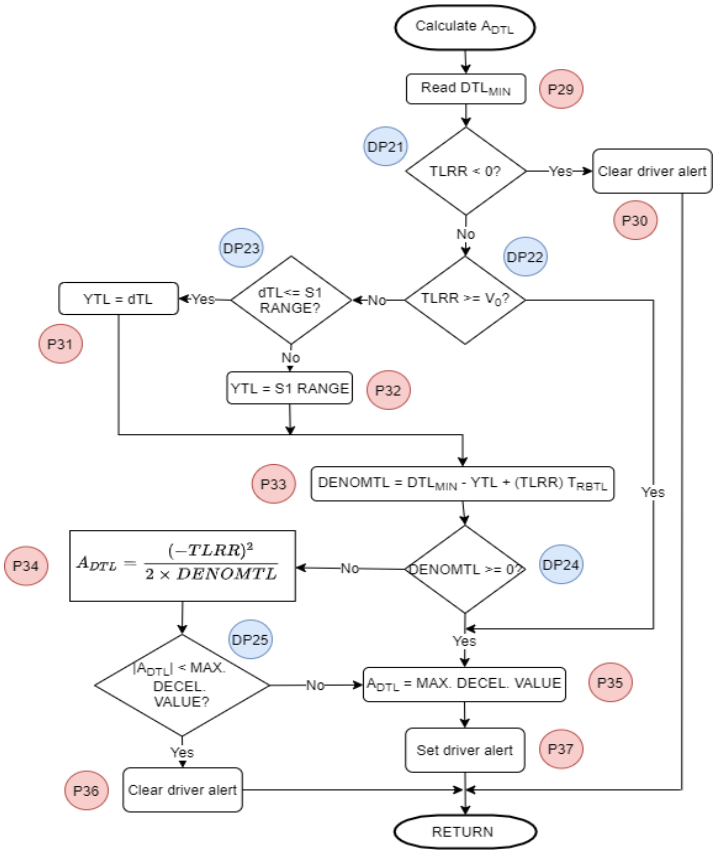


Figure 11. Deceleration process in stopping manoeuvres

The process of free ride manoeuvre is simpler than the stopping and tailing manoeuvres (see Figure 12). It starts by calculating the intended speed V_{0D} which requires the driver's desire speed (V_{SET}) and the road speed (V_{RS}). In this process, there are only two maximal values for V_{0D}, either it will be set to V_{SET} or V_{RS}. Once HV is running with the speed of V_{0D}, the goal of this process is achieved, and the brake active flag is deactivated.

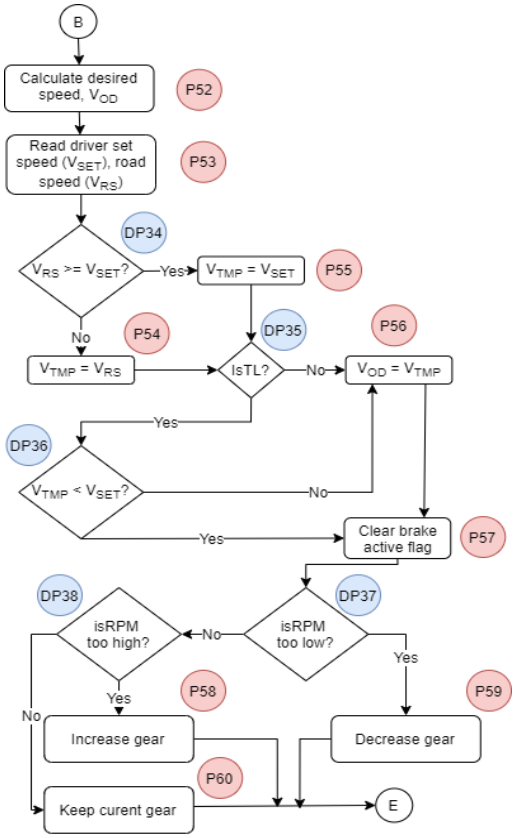
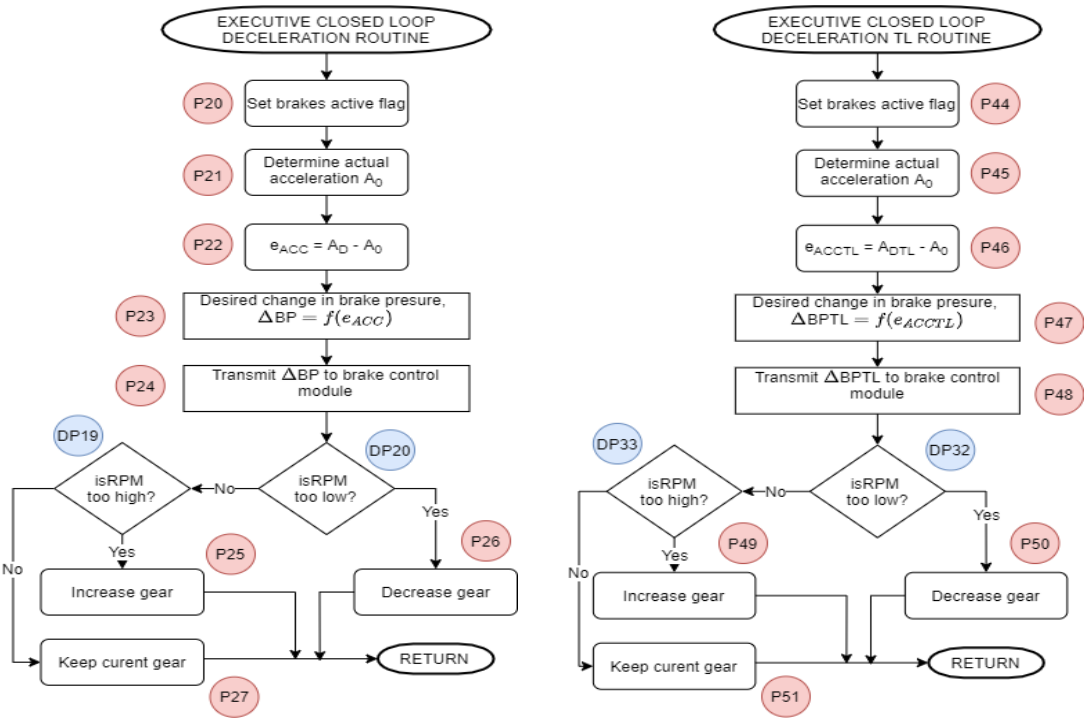


Figure 12. Behaviours in free ride situations

The last sub-process in background loop that will be presented in detail is the executive closed loop deceleration routine (see Figure 13). This routine is intended to obtain the desired change of value brake pressure (ΔBP), which is necessary to apply the generated deceleration value, and send the brake pressure value to brake control module. During this process, a simple deceleration error, e_{ACC}, is produced from the difference between A_D and A_0. Thus, the ΔBP is calculated as a function proportional to e_{ACC} to reduce the difference between A_D and A_0 to zero. When decelerating, the RPM might be too high or too low and, therefore, the control module will adjust the vehicle's gear into higher or lower positions. Even though the process in this figure illustrates the executive closed loop deceleration routine for tailing manoeuvres (Figure 13a), it will also be applied for the stopping manoeuvres by adjusting variable used in the process, such as replacing A_D with A_{DTL} (see Figure 13b).



(a) (b)

Figure 13. The process of executive closed loop deceleration routine for:(a) tailing manoeuvre; (b) stopping manoeuvre

After communicating brake pressure value change to the brake control module, the process returns to the automatic control routine process denoted by *E* symbol and enables the control interrupt. This routine, then, performs its next cycle of the loop. For the purpose of behaviour explanation generation using process-based method, the observation will be focused on the background loop process, particularly on automatic control routine as the autonomy agent's behaviour model.

b. Extracting Behaviour Explanation from the Process Model

In order to generate BCE, it is necessary to identify the final state of the process whose rationale will be investigated and every possible path of the process driving the autonomy agent's behaviours. The automatic control routine has two final states. The first state is to set the level of brake pressure based on the deceleration value obtained from executive closed loop deceleration routine. Then, the second state is to set the vehicle speed either to road speed or driver's target speed as a result of the free ride manoeuvre process.

Therefore, there are two main *why*-questions based on the final state of automatic control routine:

1) Why does the vehicle decelerate?

2) Why does the vehicle achieve the selected target speed?

Combined with the previously determined three implementation scenarios, this section will generate behaviour explanations when the agent fails to recognize TL state and no *LV* is detected (scenario 1), the agent fails to recognize TL state, but *LV* is detected (scenario 2), and the agent recognized TL state (green) and no *LV* is detected (scenario 3). For the first and the third scenario, the relevant *why*-question is the question number 2. the relevant *why*-question for the second scenario is the question number 1. After the final states of the process are identified, the next step is to link every possible path to the final state given the logics of the process. These possible paths are presented in Figure 14. In this figure, all the paths end at P61 and, therefore, all connected links to P61 are considered as the process carrying the final state of the automatic control routine.

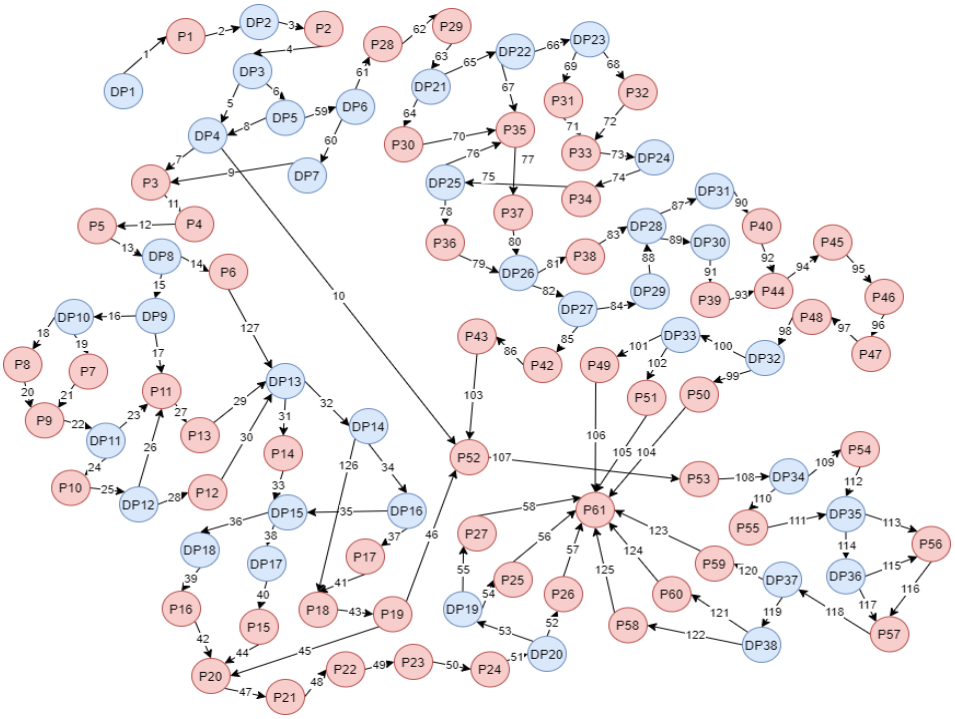


Figure 14. The possible paths of background loop

As the DP(s) inside the P(s) is not considered, there are 1440 different paths from $DP1$ to land at $P61$. Based on these paths, an adjacency list is created. Each list element contains information about two connected nodes and its path number. Also, an annotation is assigned for every path number. When a tracing-back procedure is executed to connect the path of current final state of the process, the annotation from connected path will be concatenated. The detail of adjacency list and its path number can be seen in Table 2 which will be used to generate behaviour explanations on three provided scenarios.

Table 2. Adjacency list and path number based on the process of TL situations

Connected Nodes	Path number	Symbol	Path number assigned annotation
$(DP1, P1)$	1	$((DP1, P1) \longrightarrow 1)$	
$(P1, DP2)$	2	$((P1, DP2) \longrightarrow 2)$	
…….	……	……..	
$(DP3, DP4)$	5	$((DP3, DP4) \longrightarrow 5)$	In segment 2
$(DP3, DP5)$	6	$((DP3, DP5) \longrightarrow 6)$	In segment 1
$(DP4, P3)$	7	$((DP4, P3) \longrightarrow 7)$	detected LV
$(DP5, DP4)$	8	$((DP5, DP4) \longrightarrow 8)$	but TL green, yellow, or state unknown
…….	……	……..	
$(DP4, P52)$	10	$((DP4, P52) \longrightarrow 10)$	while no LV detected
…….	……	……..	
$(P14, DP15)$	33	$((P14, DP15) \longrightarrow 33)$	as starting to approach TL with
$(DP14, DP15)$	34	$((DP14, DP15) \longrightarrow 34)$	
$(DP16, DP15)$	35	$((DP16, DP15) \longrightarrow 35)$	as still approaching
…….	……	……..	
$(P18, P19)$	43	$((P18, P19) \longrightarrow 4\ 3)$	as condition change from approaching to away from
…….	……	……..	
$(P19, P52)$	46	$((P19, P52) \longrightarrow 46)$	switch from tailing situation
$(P20, P21)$	47	$((P20, P21) \longrightarrow 47)$	Gradually
…….	……	……..	
$(P23, P24)$	50	$((P23, P24) \longrightarrow 50)$	and braking
…….	……	……..	
$(P25, P61)$	56	$((P25, P61) \longrightarrow 56)$	Decelerating
$(P26, P61)$	57	$((P26, P61) \longrightarrow 57)$	Decelerating
$(P27, P61)$	58	$((P27, P61) \longrightarrow 58)$	Decelerating
$(DP5, DP6)$	59	$((DP5, DP6) \longrightarrow 59)$	
$(DP6, DP7)$	60	$((DP6, DP7) \longrightarrow 60)$	red light with LV detected
$(DP6, P28)$	61	$((DP6, P28) \longrightarrow 61)$	red light no LV detected
…….	……	……..	
$(P38, DP28)$	83	$((P38, DP28) \longrightarrow 83)$	as stopping manoeuvre started because
…….	……	……..	
$(P42, P43)$	86	$((P42, P43) \longrightarrow 86)$	as TL situations just passed while previously
$(DP28, DP31)$	87	$((DP28, DP31) \longrightarrow 87)$	
$(DP29, DP28)$	88	$((DP29, DP28) \longrightarrow 88)$	until stopping point
…….	……	……..	
$(P44, P45)$	94	$((P44, P45) \longrightarrow 94)$	Gradually
…….	……	……..	
$(P46, P47)$	96	$((P46, P47) \longrightarrow 96)$	and braking

Connected Nodes	Path number	Symbol	Path number assigned annotation
…….	……	……..	
$(P43, P52)$	103	$((P43, P52) \longrightarrow 103)$	switch from red light situation
$(P50, P61)$	104	$((P50, P61) \longrightarrow 104)$	Decelerating
$(P51, P61)$	105	$((P51, P61) \longrightarrow 105)$	Decelerating
$(P49, P61)$	106	$((P49, P61) \longrightarrow 106)$	Decelerating
…….	……	……..	
$(DP34, P54)$	109	$((DP34, P54) \longrightarrow 109)$	as road speed less than driver set speed,
$(DP34, P55)$	110	$((DP34, P55) \longrightarrow 110)$	as target speed less than road speed,
$(P55, DP35)$	111	$((P55, DP35) \longrightarrow 111)$	driver set speed
$(P54, DP35)$	112	$((P54, DP35) \longrightarrow 112)$	road speed
$(DP35, P56)$	113	$((DP35, P56) \longrightarrow 113)$	
$(DP35, DP36)$	114	$((DP35, DP36) \longrightarrow 114)$	current speed, which is the
…….	……	……..	
$(P59, P61)$	123	$((P59, P61) \longrightarrow 123)$	Achieving/keeping
$(P60, P61)$	124	$((P60, P61) \longrightarrow 124)$	Achieving/keeping
$(P58, P61)$	125	$((P58, P61) \longrightarrow 125)$	Achieving/keeping
$(DP14, P18)$	126	$((DP14, P18) \longrightarrow 126)$	
$(P6, DP13)$	127	$((P6, DP13) \longrightarrow 127)$	

It should be noted that the assumption used in the three scenarios is that the vehicle currently is entering segment 1. Explanation generation using process-based method starts from examining the states of working memory which memorizes the visited nodes during the process execution. After that, this method inspects the connections among nodes from the working memory using the adjacency list and queries the path number. Finally, this method generates explanation based on the path number's annotation.

A. Generating Explanation: Scenario 1

In this scenario, the vehicle is entering segment 1, but the autonomy agent fails to recognize TL state and no LV is detected. The process in generating explanation for this scenario is illustrated in Figure 15. In working memory, there are 17 nodes captured during one cycle of background loop process executing automatic control routine. Using the **Algorithm 1**, these nodes are examined by referring to the adjacency list in Table 2. This examination takes node P61 as the root node.

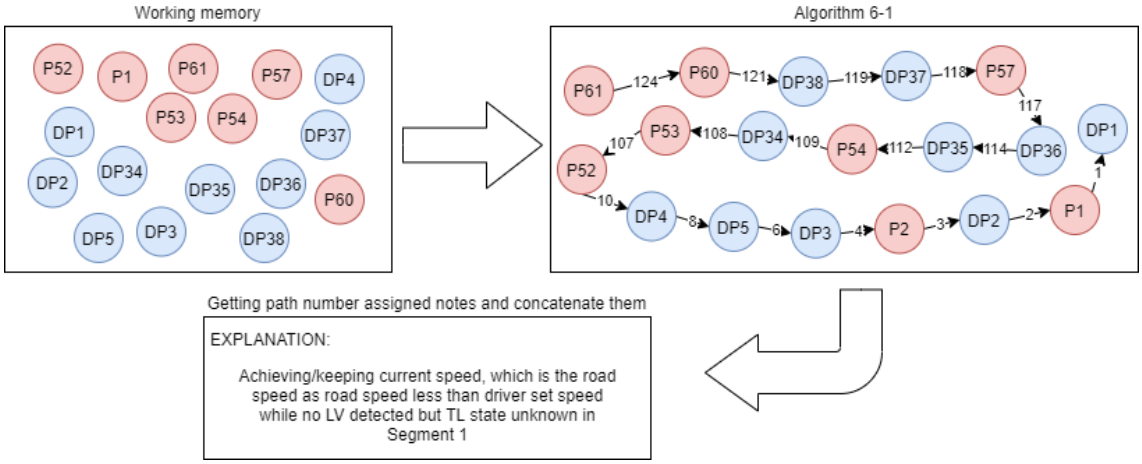


Figure 15. Generating explanation for scenario 1 using process-based method

Initially, the algorithm seeks which node stored in working memory that is connected to P61 by sending a query to the adjacency list. This query returns P60 as the result and makes this node as the root node replacing P61. Besides getting the adjacent node, this query also gets the path number between the root node and its adjacent node. For example, the path number of nodes P61 to P60 is 124 and then, the path number 121 connects the node P60 to DP38.

Based on the 2, the assigned note for path number 124 is 'achieving/keeping'. the path number 121, 119, 118, and 117 have no annotations. The explanation text, then, continues with the annotation from path number 114 and 112, 'current speed, which is the road speed'. Furthermore, the annotation 'as road speed less than driver set speed' is obtained from the path number 109. Finally, 'while no *LV* detected but TL state unknown is detected in Segment 1' is the concatenation of annotation from path number 10, 8, and 6. Thus, the full explanation text for this scenario is 'achieving/keeping current speed, which is the road speed as road speed less than driver set speed while no *LV* detected but TL state unknown in Segment 1'. To generate this explanation, it needs to revisit 17 nodes without considering the existence of DP nodes within P nodes.

B. Generating Explanation: Scenario 2

There are 33 revisited nodes while generating explanation for scenario 2 as shown in Figure 16. In scenario 2, TL state unknown with detected *LV* in Segment 1 presents. This scenario produces the vehicle deceleration value to adjust the safe margin to *LV*. Furthermore, the tracing back of process paths is started from node P61 as the root node,

which is then connected to node 27. This connection has path number 58. After that, P27 becomes the root node linked to node DP19 through path number 58. Node DP19 itself has a preceding node called DP20. At the end of the link is path number 1 connecting node P1 with the last node, DP1.

The explanation text ‘Decelerating and braking gradually’ is acquired from the path number 58, 50, and 47. Then, it is followed by ‘as starting to approach TL with detected LV’ which comes from path number 33 and 7. Lastly ‘but TL state unknown in segment 1’ from path number 8 and 6. After fully connected, the explanation is ‘Decelerating and braking gradually as starting to approach TL with detected *LV* but TL state unknown in segment 1’.

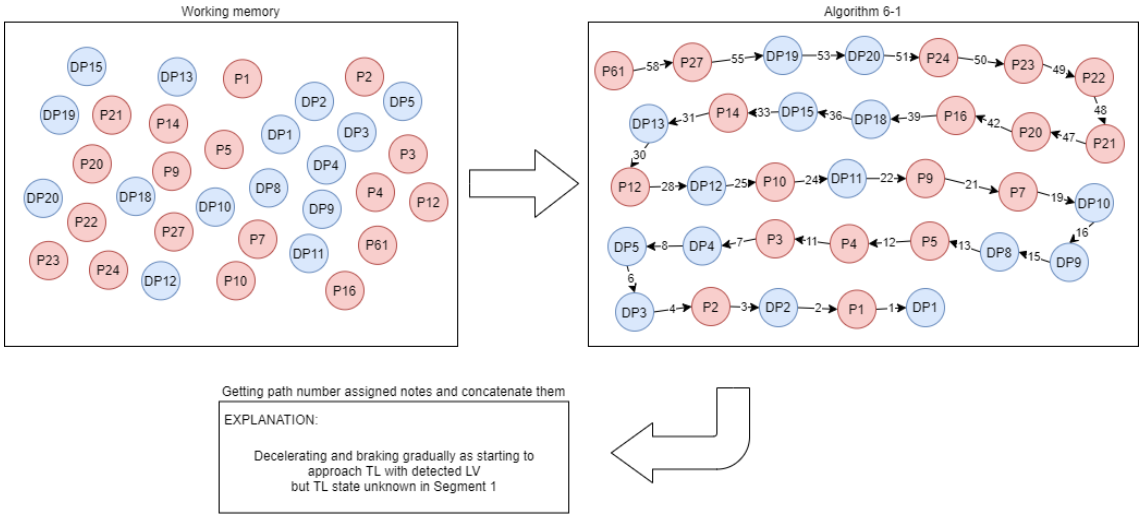


Figure 16. Generating explanation for scenario 2 using process-based method

C. Generating Explanation: Scenario 3

With the same way of generating explanation for scenario 1 and scenario 2, this scenario connects 18 nodes after tracing back the process path (see Figure 17). The tracing back process starts from node P61 which is connected to node P60 with path number 124. The explanation text ‘Achieving/keeping current speed, which is the road speed’ is generated from path number 124 combined with path number 114 and 112 which connect DP36 to DP35 and DP35 to P54, respectively. the path number 109 connecting P54 to DP34 produces explanation text ‘as road speed less than driver set speed’. Finally, ‘while no LV detected but TL green in segment 1’ is obtained from path number 10, 8, and 6. Hence, the full text explanation for this scenario is ‘Achieving/keeping current speed, which is

the road speed, as road speed less than driver set speed while no LV detected but TL green in segment 1'.

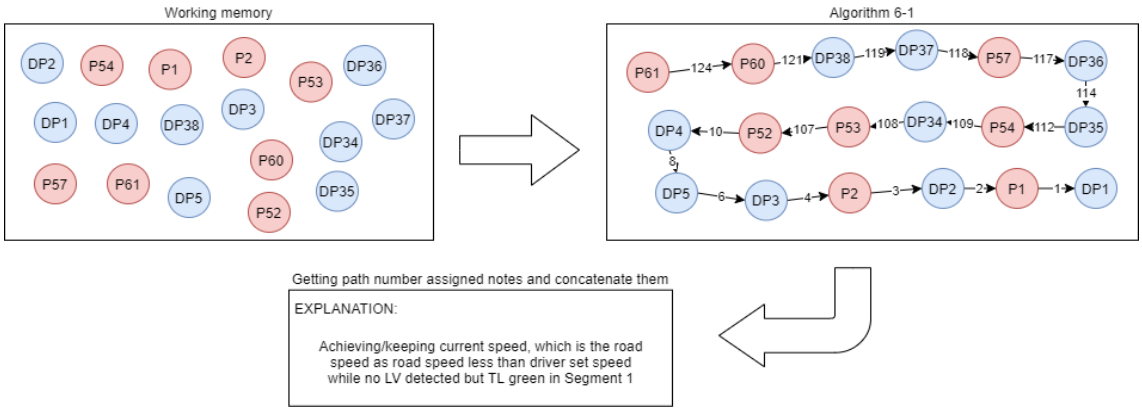


Figure 17. Generating explanation for scenario 3 using process-based method

3. Implementation in Overtaking Scenario

In this study case, the explanation generation in overtaking situations is to clarify the causal of overtaking cancelation and *HV*'s positioning after cancelation. For this purpose, this study provides three scenarios in association with the circumstances causing the cancelation. In all scenarios, it is assumed that overtaking suggestions have been received and approved by the driver and the first lane changing which takes *HV* to overtaking lane is already executed. These three scenarios can be described as follows:

1) **Scenario 1:** The first scenario is a multi-vehicle overtaking manoeuvre. There are two vehicles to be overtaken, but *LVO* does not present. However, after overtaking the first vehicle the second vehicle adds its speed which makes the overtaking speed will override the maximum road speed limit. Therefore, overtaking is cancelled but *HV* will stay at the overtaking lane.

2) **Scenario 2:** In the second scenario, *HV* just passed the overtaken vehicle (one vehicle only) but before going back to its departure lane, *LVO* reduces its speed. Consequently, the minimum space to perform lane change to *HV*'s departure lane is not met. Therefore, overtaking is cancelled and *HV* will stay at the overtaking lane.

3) **Scenario 3:** In the third scenario, *HV* just started the overtaking manoeuvre and finished the first lane change to overtake *LV*. However, before passing *LV*, *LVO* slows

down so that it becomes unable to continue overtaking under such circumstances. Therefore, *HV* will go back to its departure lane.

The implementation of explanation generation methods is intended to produce clarification as described in the determined scenarios.

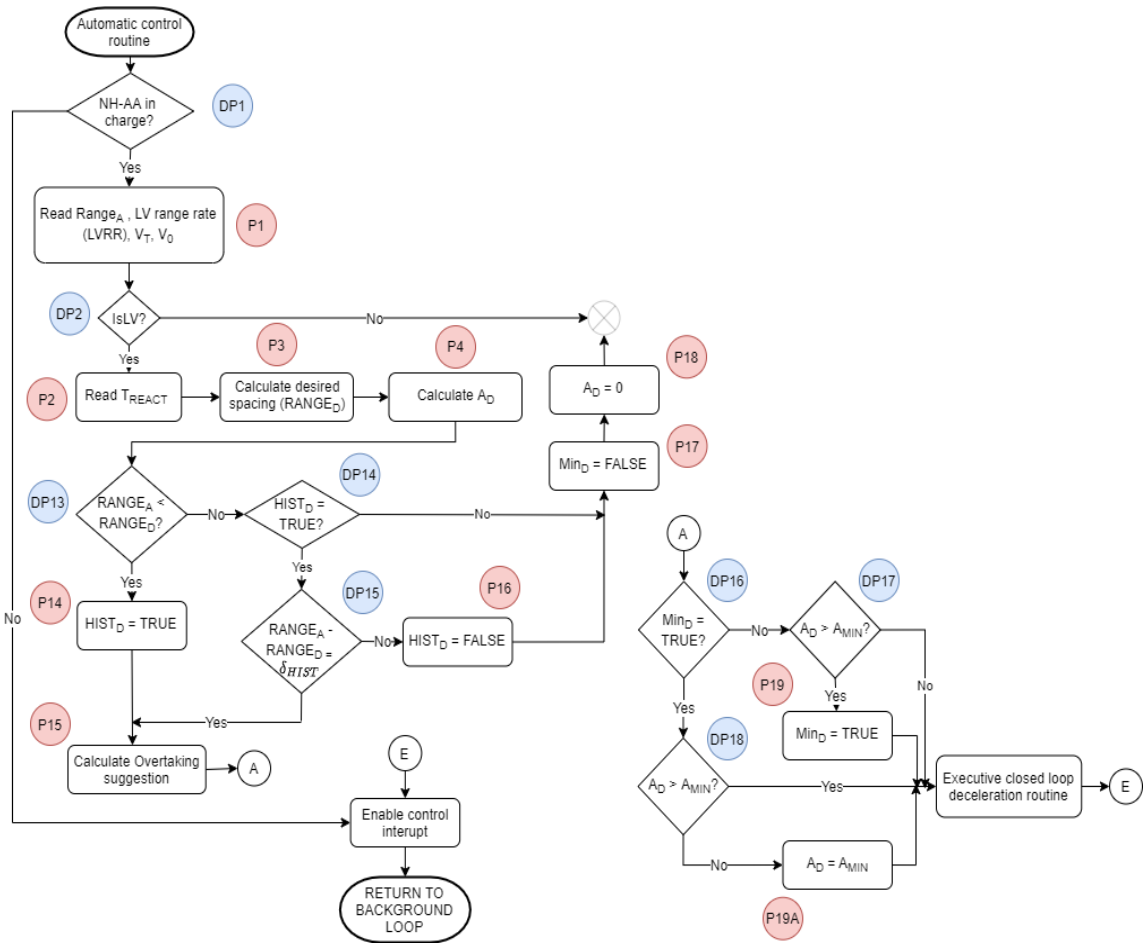


Figure 18. Automatic control routine with overtaking suggestion

a. Modelling Agent's Decision Logic at Process Level

To model the autonomy agent's behaviours, this study uses two main processes which cover pre-overtaking and overtaking performance. In these two processes, the automatic control routine previously used in TL situation is also applied because it is associated with determining deceleration when vehicles with lower speed in front exist. In addition, we also combine the automatic control routine with the US google patent proposed by

Mudalige & Losh (2015) to handle overtaking task. In pre-overtaking situations, this routine is expanded by comparing the *HV*'s speed after deceleration with road speed and driver's target speed to produce overtaking suggestion. in overtaking situations, this routine contributes to determine *HV*'s deceleration when *LVO* and *LVH* present.

Figure 18 shows the automatic control routine process that includes the overtaking suggestion. It is started by detecting *LV*, when its existence is valid, a process called calculate overtaking suggestion will examine surrounding situation so that notifications for human agent can be generated. The detail of this process can be seen in Figure 19. First, the calculation process reads the range between *HV* and *LVO* and the range between *HV* and *RLV*, which is referred to as $Range_{LVO}$ and $Range_{RLV}$, respectively. As a result, range rate of *LVO* (*LVORR*) and *RLV* (*RLVRR*) can be estimated including the speed of *LVO* (V_{LVO}) and *RLV* (V_{RLV}). Besides *LVO* and *RLV*'s speed, this process also needs to read the road speed (V_{road}) supplied by a navigation system, V_T, and V_{SET}. V_T and V_{SET} represent HV's speed and driver's desired speed (also called target speed), respectively.

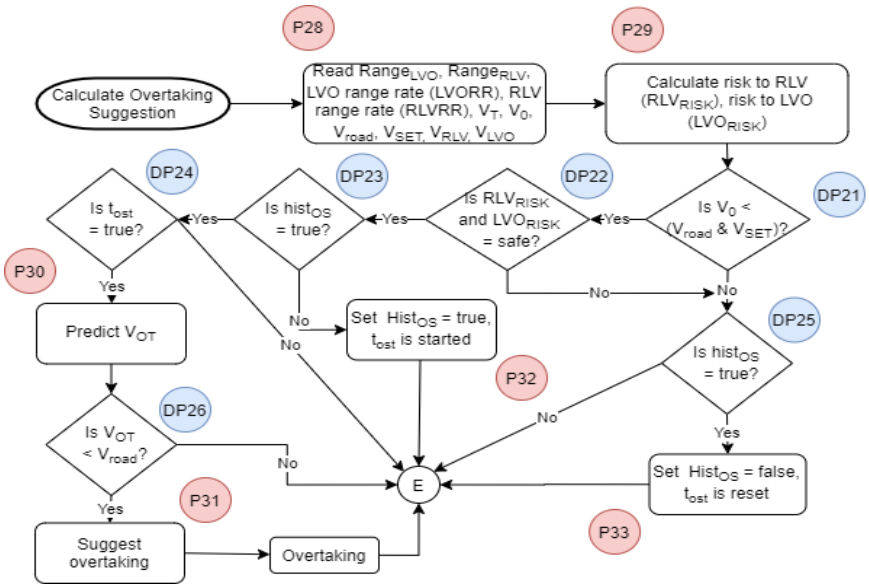


Figure 19. Calculating overtaking suggestion process

Once the range with all associated vehicles and their speed variables are obtained, the collision risk to *RLV* (RLV_{RISK}) and *LVO* (LVO_{RISK}) are calculated. All these variables, then, will be examined in several decision logic points to produce overtaking suggestion. The first decision logic point is to compare *HV*'s current speed (V_0) with V_{road} and V_{SET}. When the requirement of V_0 does not equal and is greater than the compared speed, the

second decision logic point will check whether RLV_{RISK} and LVO_{RISK} can be inferred as safe. After that, the third decision logic inspects the history flag which initially is not activated ($hist_{OS} = false$). Once the three requirements (V_0 below V_{road} and V_{SET}, RLV_{RISK} and LVO_{RISK} are safe) of overtaking suggestion are met, it will change the history flag to *true* state ($hist_{OS} = true$). Besides changing the history flag, the fulfillment of the three requirements also makes the time limiter of tailing situations (t_{OST}) is started. At this point, the overtaking suggestion is not generated yet.

When the overtaking suggestion process undergoes the third decision logic, but the time limiter value is still, i.e., 1 second, it needs to wait for a certain amount of time before generating overtaking suggestion. This amount of time is pre-determined, such as 10 seconds or more. When the state of three requirements can endure until time limiter achieves its pre-determined value, overtaking suggestion is generated. However, before this suggestion is passed to the driver, the process will check whether the overtaking speed prediction will excess the V_{road} or not. If not exceeding V_{road}, a suggestion will be delivered to human agent, but will not be delivered when otherwise. Even though this notification is received by human agent, the overtaking manoeuvre is not started yet.

After the suggested overtaking obtains human's consent, the overtaking process (see Figure 20) is started by performing a lane change manoeuvre called *LC1* which takes *HV* from ego lane to the overtaking lane. For this regard, a *LC*1 plan is generated including the route steps, which consist of route nodes, and node-to-node steering executions. After that, this process examines whether *HV* is in *LC*1 zone or not. If *HV* is not in *LC*1 zone, whether *HV* already passed from *LC*1 zone will be checked while continuing the *LC*1 route plan. If not passed yet, the process will go back to *LC*1 zone examination. However, if *HV* is still in *LC*1 zone, *LC*1 flag will be checked whether this flag is safe to be cleared now. This flag is intended to avoid re-entering *LC*1 process while the autonomy agent is still executing it. When this flag is cleared but overall route is not completed yet, this flag will be reactivated, and the process goes back to *LC*1 zone examination. Normally, the flag will not be cleared if *HV* is still in *LC*1 zone. When *LC*1 flag is still active, the process will continue the route plan and go to the decision point which inspects whether *HV* has finished the *LC*1 route by passing LC1 zone. If passed, the process will proceed to manoeuvre in overtaking lane.

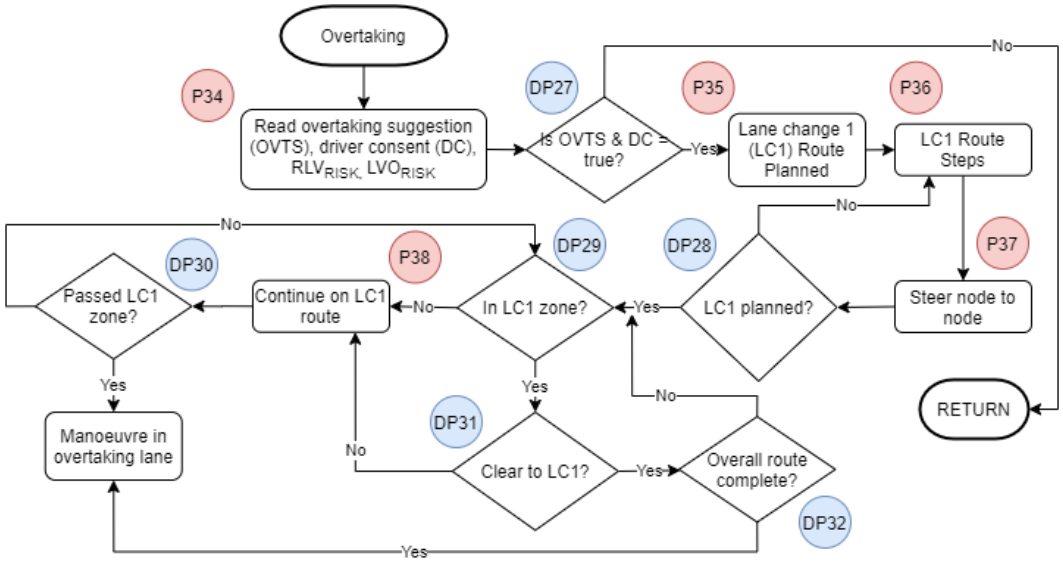


Figure 20. Overtaking Process: Lane change from departure lane to overtaking lane

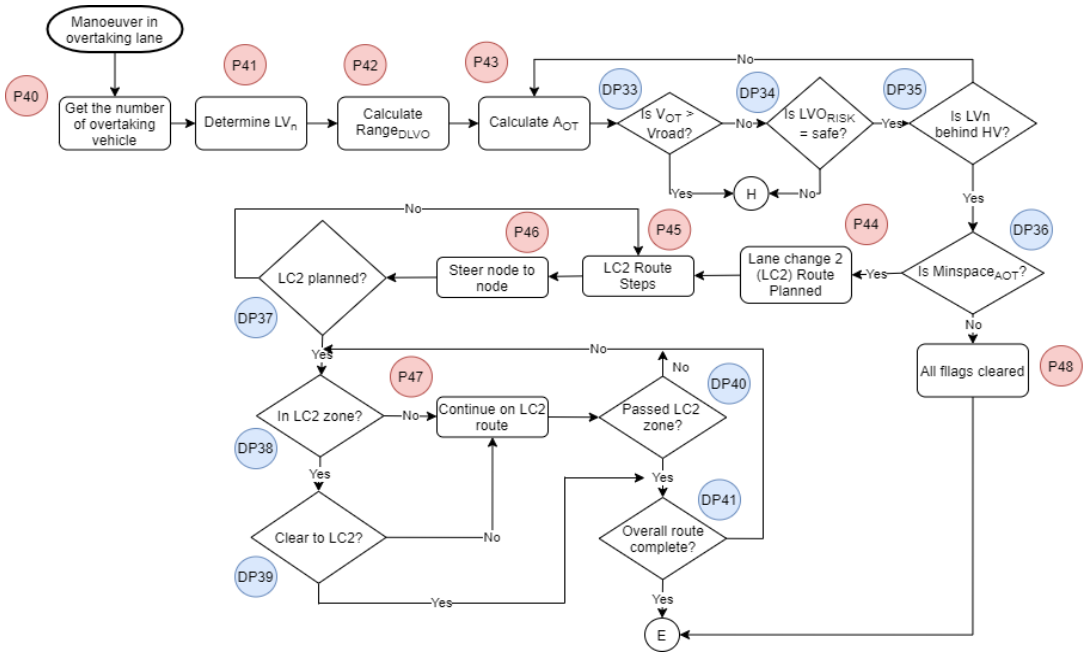


Figure 21. Overtaking process: Manoeuvre in overtaking lane

The algorithms of manoeuvre in overtaking lane are presented in Figure 21. First, the autonomy agent will evaluate the number of vehicles and determine the last vehicle (LV_n) that will be overtaken. Once these variables are acquired, the distance between HV and LVO ($Range_{DLVO}$) and the appropriate overtaking acceleration (A_{OT}) are calculated. If the produced overtaking speed (V_{OT}) does not exceed the road speed limit and the collision

risk with LVO (LVO_{RISK}) is low, this process will continuously check whether LV_n is already behind HV. Once HV precedes LV_n, the minimum space ($Minspace_{AOT}$) to return to departure lane is examined. When there is an enough space, the second lane change manoeuvre ($LC2$) will be proceeded by generating $LC2$ route plan. The process of $LC2$ is similar to $LC1$ which requires the examination of whether HV in $LC2$ zone, $LC2$ flag can be cleared, HV already passed $LC2$ zone, and overall route is completed. When the planned route is finished, then the process returns to background loop process.

When the requirement of $Minspace_{AOT}$ is not met, all flags associated with overtaking manoeuvre are cleared. This means that the autonomy agent will not perform $LC2$ due to all given risks. In other words, overtaking is cancelled. In this kind of cancelation, HV will stay in overtaking lane, and this lane will be referred to as the new ego lane by the autonomy agent. As consequences, all the references will change including ones related to LVO. In this regard, LVO will be referred to as the new LV in HV's ego lane. After that, the process returns to automatic control routine.

During overtaking, the LVO_{RISK} and V_{OT} might change and call the cancelation process. For example, LVO suddenly decreases its speed or the positive acceleration of overtaken vehicle is detected. To overcome such situations, there are two possible options whether to go back to previous position at departure lane (behind LV) or stay at overtaking lane. Figure 22 shows the process which guide the HV to these two options. Regarding the first option, the process will perform a lane change manoeuvre called $LV3$. However, before $LC3$ is executed, it needs to check whether the space behind LV is safe enough to be placed and, therefore, this process will check the existence of vehicle before LV and its range ($Range_{COT}$). When it is safe, a deceleration value (A_{COT}) will be determined to perform $LC3$. Otherwise, all overtaking manoeuvre flag is cleared and go to automatic control routine.

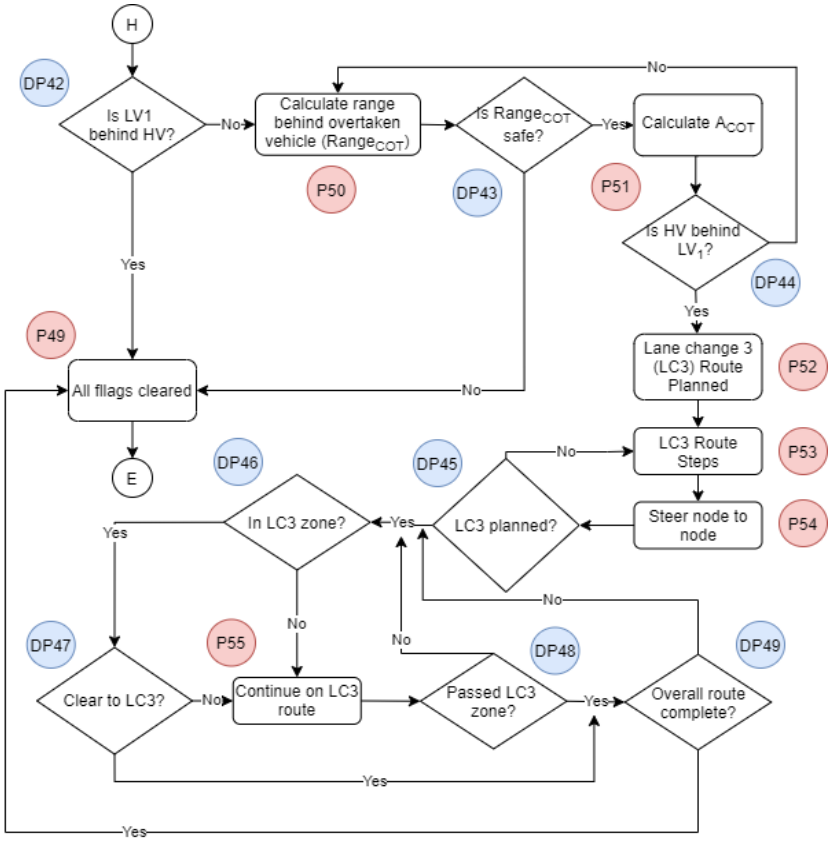


Figure 22. Overtaking process: Handling the changes of LVO_{risk} and predicted V_{OT} speed

Like the other lane change manoeuvre (*LC*1 and *LC*2), *LC*3 also follows generated lane change plans and routes. *LC*3 also undergoes several processes to examine current lane change zone, lane change flag, and its plan completion. However, the main difference between *LC*3 and the other two is that *LC*3 is performed under deceleration mode while the others are the contrary. After deceleration is started, the process will continuously check whether the current position of *HV* is already behind LV_1 and all risks are low to perform *LC*3. When all these requirements are fulfilled, *LC*3 is executed. After *LC*3 plans are accomplished, the process returns to automatic control routine (from node DP49 to P49 in Figure 22). In this returning process, the ego lane of *HV* is the departure lane before overtaking manoeuvre is executed. This is different from returning to automatic control routine from the first decision point DP42 (is LV_1 behind *HV*). In this case, the new ego lane of *HV* is the overtaking lane.

The first decision point in Figure 22 is crucial point in overtaking cancellation, particularly when there are multiple vehicles to be overtaken. Let assume that according to the overtaking plan, *HV* will pass three vehicles and currently its position is just after

LV_3. When the collision risk with *LVO* becomes high, this process will not consider going behind LV_3 within the departure lane even though there is enough space between LV_2 and LV_3. This process will refer to the position of LV_1 in which in this situation, *HV* is far behind LV_1 and, therefore, to stay on overtaking lane is selected. Now, assuming that *HV* will overtake three vehicles, and its position is already in overtaking lane but still somewhere behind LV_1. At this state, this process will lead the *HV* to go back behind LV_1 in the departure lane. Furthermore, the number of vehicles to be passed might not be unlimited. This number will depend on the ability of vehicle's sensory tools to recognize LV_n as it has maximum distance limitation. However, giving the ability to deal with numerous overtaken vehicles is still possible but it requires a more sophisticated and complex process.

b. Extracting Behaviour Explanations from the Process Level

After the process leading the autonomy agent's behaviours in overtaking situations is modelled, the next step is to identify the final state of the process which can be seen as a taken action of this agent given the dynamic changes of circumstances within the environment. However, as this overtaking situation process is under automatic control routine, its final states will be similar to ones from the traffic light situation process which are decelerating or accelerating the vehicle. However, as the transparency for the overtaking situation in this study is focused on overtaking cancelation, the *why*-question for the explanation generation will only consider the first final state of the process, decelerating the vehicle.

In relation with the determined implementation scenario, the deceleration will occur when the vehicle is already in overtaking lane and the overtaken vehicle increases its speed so that $V_{OT} > V_{road}$, *HV* will decelerate and stay in overtaking lane (scenario 1). in scenario 2, the *HV* decelerates, and it is unable to perform *LC*2 as the minimum space is not met. In scenario 3, while *HV* has just entered overtaking lane and it is still behind *LV*, *LVO* slows down. This circumstance makes *HV* decelerate.

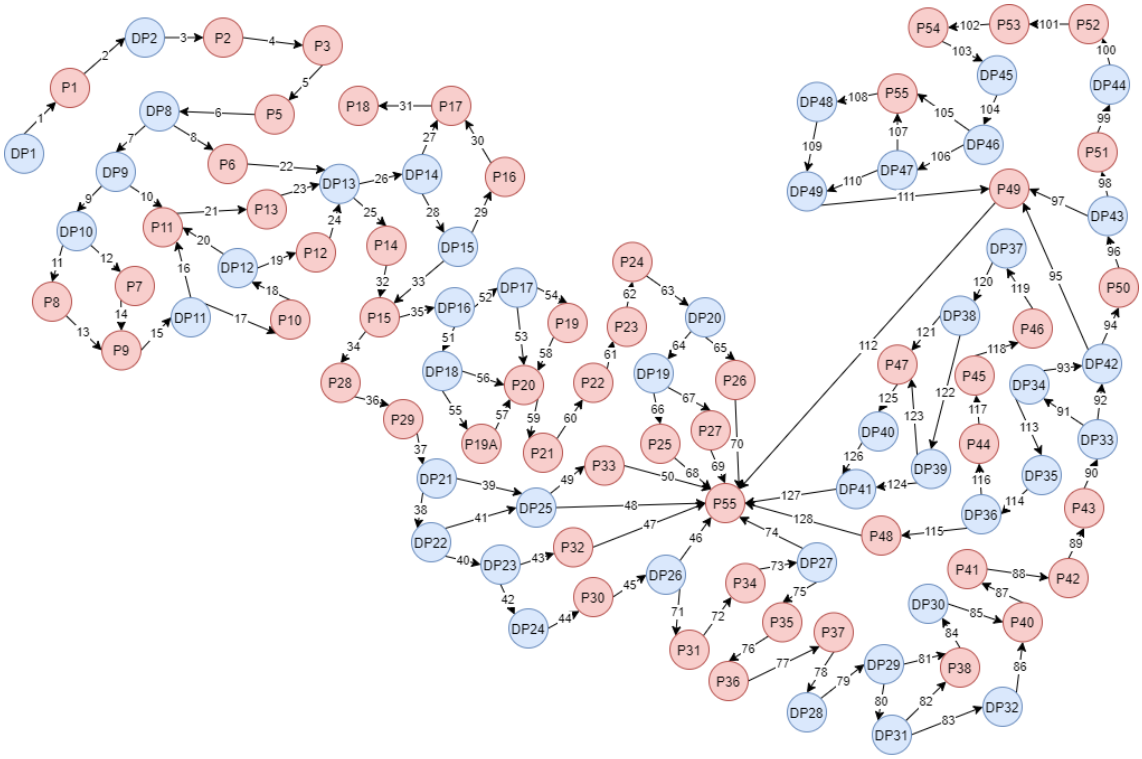


Figure 23. The possible paths of the behaviour's process in overtaking situations

In Figure 23, every possible path of the autonomy agent's behaviours is drawn, and there are 993 possible paths in total. In this figure, the DP number and P number are also referring to Figure 9 and Figure 13a which present the process of calculating A_D and executive closed loop deceleration routine respectively, because these two processes also become the part of overtaking manoeuvre process. Furthermore, it can be seen in Figure 23 that the whole process is ended at $P55$ which represent the end of background loop. In Table 3, the adjacency list and the assigned path number connecting nodes involved in the overtaking process are presented. Moreover, this table also presents the annotation for each path number.

Table 3. Adjacency list and path number based on the process of overtaking situations

Connected Nodes	Path number	Symbol	Path number assigned annotation
$(DP1, P1)$	1	$((DP1, P1) \longrightarrow 1)$	
$(P1, DP2)$	2	$((P1, DP2) \longrightarrow 2)$	
…….	……	……..	
$(DP15, P15)$	33	$((DP15, P15) \longrightarrow 33)$	during tailing situations
…….	……	……..	
$(DP22, DP23)$	40	$((DP22, DP23) \longrightarrow 40)$	while safe overtaking risks (collision and overtaking speed limitation) are indicated
$(DP22, DP25)$	41	$((DP22, DP25) \longrightarrow 41)$	

Connected Nodes	Path number	Symbol	Path number assigned annotation
$(DP23, DP24)$	42	$((DP23, DP24) \longrightarrow 42)$	which detects the vehicle speed below the target speed for a certain period of time
…….	……	……	
$(P31, P34)$	72	$((P31, P34) \longrightarrow 72)$	based on system suggestions
$(P34, DP27)$	73	$((P34, DP27) \longrightarrow 73)$	
$(DP27, P55)$	74	$((DP27, P55) \longrightarrow 74)$	
$(DP27, P35)$	75	$((DP27, P35) \longrightarrow 75)$	even though this overtaking was approved by driver
…….	……	……	
$(DP33, DP42)$	92	$((DP33, DP42) \longrightarrow 92))$	because the speed of overtaken vehicle is increased which makes overtaking speed exceeds road speed limit
$(DP34, DP42)$	93	$((DP34, DP42) \longrightarrow 93)$	because collision risk with vehicle in overtaking lane is changed to high
$(DP42, P50)$	94	$((DP42, P50) \longrightarrow 94)$	as overtaking was cancelled
$(DP42, P49)$	95	$((DP42, P49) \longrightarrow 95)$	and stay in overtaking lane as overtaking was cancelled
…….	……	……	
$(DP44, P52)$	100	$((DP44, P52) \longrightarrow 100)$	behind LV1
…….	……	……	
$(DP49, P49)$	111	$((DP49, P49) \longrightarrow 111)$	and just backed to departure lane
$(P49, P55)$	112	$((P49, P55) \longrightarrow 112)$	Decelerating
$(DP34, DP35)$	113	$((DP34, DP35) \longrightarrow 113)$	
$(DP35, DP36)$	114	$((DP35, DP36) \longrightarrow 114)$	even though we have passed the overtaken vehicle
$(DP36, P48)$	115	$((DP36, P48) \longrightarrow 115)$	and stay in overtaking lane as overtaking was cancelled because the minimum space to go back to departure lane is not met
…….	……	……	
$(P48, P55)$	128	$((P48, P55) \longrightarrow 128)$	Decelerating

Similar to the adjacency list for *TL* situations, some path numbers are not annotated. This means that it has been generalized in the next path. For example, the annotation for the process of assessing overtaking risk which involves nodes *P*28, *P*29, *DP*1, *DP*22 and *DP*23 has been summarized at the path connecting *DP*23 and *DP*24 using path number 40. Even though the annotation is generalized, it is still necessary to crawl all nodes to obtain all the notations explaining the autonomy agent's behaviours. Based on the developed adjacency list, the behaviour explanation generation for each overtaking situation implementation scenario is presented below.

A. Generating Explanation: Scenario 1

In this scenario, overtaking is cancelled, and the vehicle stays in overtaking lane. As illustrated in Figure 24, explanation generation for this scenario involves 45 nodes captured in working memory, and the relation among these nodes will be examined using

Algorithm 1 to find the rationale behind autonomy agent's actions. In this scenario, the paths connecting node *P*55 to node *P*40 reveal the autonomy agent's action and the causal behind it. In detail, the path number 112 which connects P55 to P49 represents the decelerating action. The first explanation of why such deceleration occurs can be found in path number 95 which connects *P*49 to *DP*42. The annotation for this path number is 'stay in overtaking lane as overtaking was cancelled'. The reason is being cancelled can be seen in the annotation of path number 92 linking *DP*33 to *DP*42. This annotation says, 'because the speed of overtaken vehicle is increased which makes overtaking speed exceeds road speed limit'.

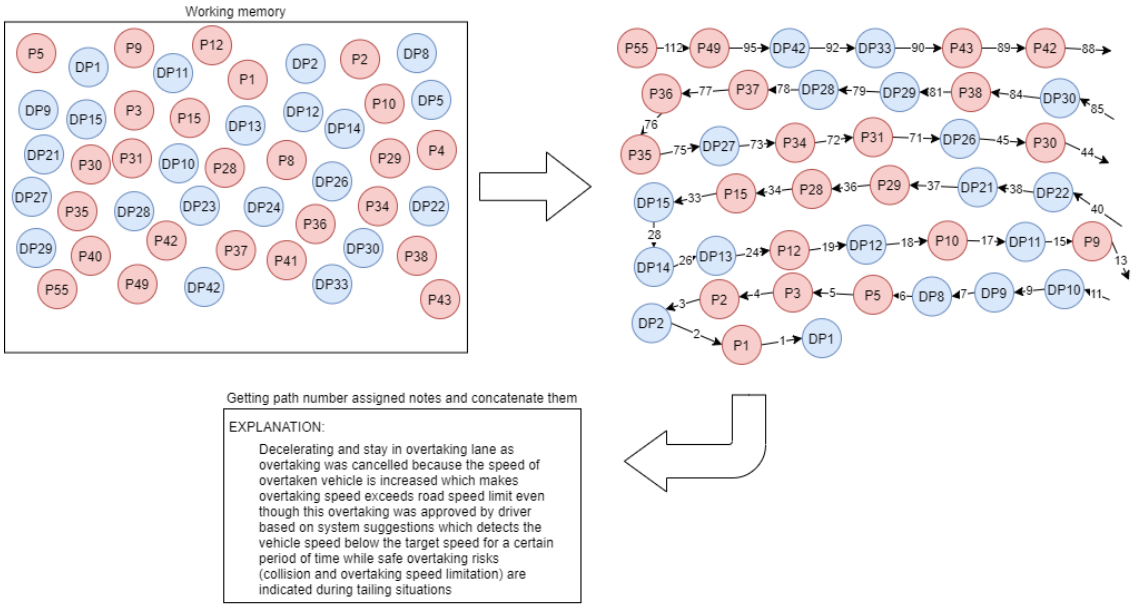


Figure 24. Explanation generation for scenario 1 of overtaking situations using process-based method

The further explanation, then, tells the process history of the driver approval (as pointed the path number 75) as suggested by system suggestions (path number 72) which previously detects the vehicle speed is lower than the target speed for a certain period (path number 42) while overtaking risk measurement that include collision risk and speed limit violation risk indicates a safe state (path number 40) when the vehicle is under tailing situations (path number 33). By connecting all the annotation, the complete explanation for scenario 1 generated by this process based method is 'Decelerating and stay in overtaking lane as overtaking was cancelled because the speed of overtaken vehicle is increased which makes overtaking speed exceeds road speed limit even though this

overtaking was approved by driver base on system suggestion which detects the vehicle speed below the target speed for a certain period of time while safe overtaking risks (collision and overtaking speed limitation) are indicated during tailing situations'.

B. Generating Explanation: Scenario 2

There are 47 nodes involved to generate explanations for scenario 2 as illustrated in Figure 25. While scenario 1 starts from connecting *P*55 to *P*49 (path number 112), the first tracing back path in this scenario is path number 128 which links *P*55 to *P*48. However, these two paths, 112 and 128, have the same meaning which represents the vehicle deceleration. The reason behind the deceleration in this scenario is indicated by path number 115 which relates node *P*48 to *DP*36. The annotation for this path number is 'stay in overtaking lane as overtaking was cancelled because the minimum space to go back to departure lane is not met'. After that, further explanation is obtained from the path number 114's annotation which tells that actually the vehicle has already passed the overtaken vehicle.

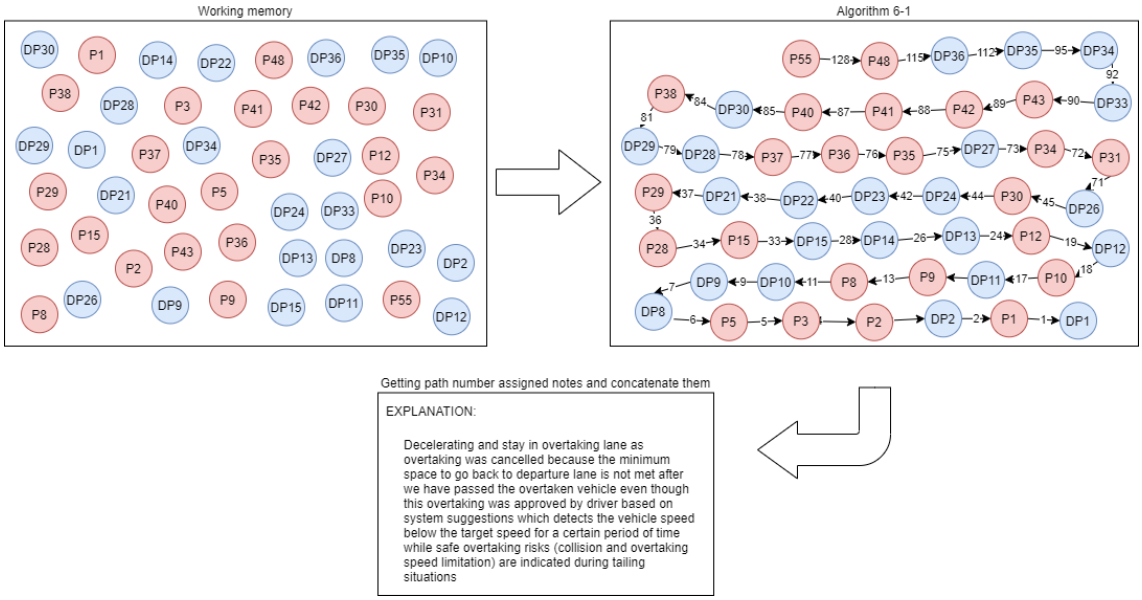


Figure 25. Explanation generation for scenario 2 of overtaking situations using process-based method

Similar to scenario 1, after node *P*40, the tracing back continues to reveal the process history. By combining to the process history with previously generated explanations, the complete explanation for scenario 2 is 'Decelerating and stay in overtaking lane as the minimum space to go back to departure lane is not met after we have passed the overtaken

vehicle even though this overtaking was approved by driver base on system suggestion which detects the vehicle speed below the target speed for a certain period of time while safe overtaking risks (collision and overtaking speed limitation) are indicated during tailing situations'.

C. Generating Explanation: Scenario 3

This scenario involves more nodes (59 nodes) than the other two scenarios (see Figure 26). The first tracing back path in this scenario is pointed to path number 112 which is the same as scenario 1. From this path, the tracing back continues to path number 111 which establishes the link from *P*49 to *DP*49 and provides additional information about the vehicle which just returns to the departure lane. In path number 100, the vehicle's position in departure lane is mentioned, which is behind LV_1. The causal of the behaviours described by annotations from path number 111 and 100 is provided by path number 94 which says that overtaking was cancelled.

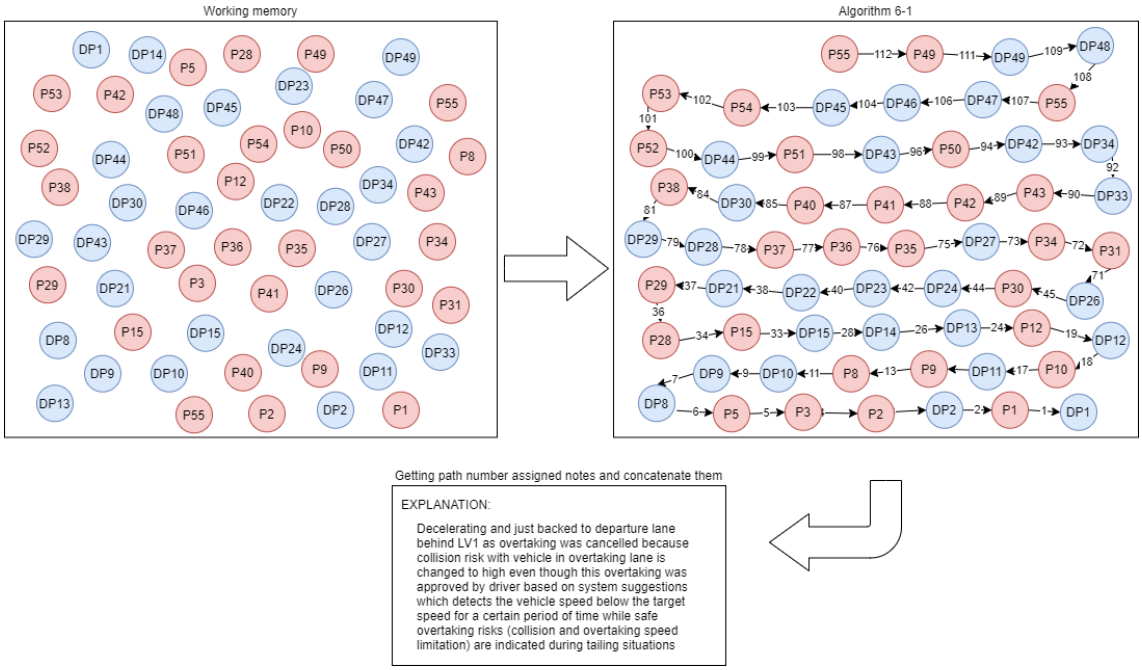


Figure 26. Explanation generation for scenario 3 of overtaking situations using process-based method

the reason of the cancelation can be seen by the annotation of path number 93 which links nodes *DP*42 and *DP*34. This annotation describes that the cancelation caused by the change of collision risk with vehicle in overtaking lane into high risk. Then, by combining

with the process history, the complete explanation for this scenario is ‘Decelerating and just returned to the departure lane behind LV_1 as overtaking was cancelled because collision risk with vehicle in overtaking lane is changed to high even though this overtaking was approved by the driver based on system suggestions which detects the vehicle speed below the target speed for a certain period of time while safe overtaking risks (collision and overtaking speed limitation) are indicated during tailing situations’.

To be more useful, the explanation for this scenario can be started from *DP*44. *DP*44 is the process which generates the route plan to perform *LC*3. In other words, the vehicle is in waiting state to go back to the departure lane while this plan is generated. During this waiting, it would be very helpful for the driver to understand what is happening by sending an explanation which says, ‘Waiting to go back to the departure lane behind LV_1 as overtaking was cancelled because collision risk with vehicle in overtaking lane is changed to high even though this overtaking was approved by the driver based on system suggestions which detects the vehicle speed below the target speed for a certain period of time while safe overtaking risks (collision and overtaking speed limitation) are indicated during tailing situations’. Once the process is completed at *P*55, the previous version of explanation can be delivered.

Appendix 4: Publications of This Research

The research outcomes have been published in peer-reviewed international journals and conferences. The details are as follows:

1) Rinta Kridalukmana, Hai Yan Lu & Mohsen Naderpour (2020), A supportive situation awareness model for human-autonomy teaming in collaborative driving, Theoretical Issues in Ergonomics Science, DOI: 10.1080/1463922X.2020.1729443 (Q2, SJR = 0.53, Cite Score = 2.7)

2) Kridalukmana, R., Lu, H.Y., Naderpour, M. An object-oriented Bayesian Network approach for unsafe driving manoeuvres prevention system, Proceedings of the 12th International Conference on Intelligent Systems and Knowledge Engineering, ISKE 2017 (ERA Tier B Conference)

3) Kridalukmana, R., Lu, H.Y., Naderpour, M. Component-Based Transparency to Comprehend Intelligent Agent Behaviour for Human-Autonomy Teaming, Proceedings of the 14th International Conference on Intelligent Systems and Knowledge Engineering, ISKE 2017 (ERA Tier B Conference)

In addition, two manuscripts have been submitted:

1) Kridalukmana, R., Lu, H.Y., Naderpour, M. 'Self-Explanation Abilities of an Intelligent Agent for Transparency in Human-Autonomy Teaming', IEEE Transactions on Human-Machine Systems (Q1, SJR = 0.82, Cite Score = 5.55) (Decision: Resubmit)

2) Kridalukmana, R., Lu, H.Y., Naderpour, M. 'Transfer Learning with Directive Offline Augmentation', Pattern Recognition Journal (Q1, SJR = 2.323, Cite Score = 13.1) (Under review)