Autonomous Local Manoeuvre and Scenario Orchestration Based on Automated Action Planning in Driving Simulation

Zhitao Xiong[1], Anthony G. Cohn [2], Oliver Carsten [3], Hamish Jamson [3]
(1) Institute for Transport Studies and School of Computing, 38 University Road, University of Leeds, Leeds, UK, LS2 9JT, E-mail: tszx@leeds.ac.uk
(2) School of Computing, University of Leeds, Leeds, UK, LS2 9JT, E-mail: a.g.cohn@leeds.ac.uk
(3) Institute for Transport Studies, 38 University Road, University of Leeds, Leeds, UK, LS2 9JT, O.M.J.Carsten@its.leeds.ac.uk, A.H.Jamson@its.leeds.ac.uk

Abstract *– Techniques for orchestrating scenarios with autonomous vehicles in driving simulation are not abundant and three problems are still open: Actor Management, Actor Preparation and Scenario Orchestration. Moreover, "failures" still happen in scenarios. In this paper, a decision-making algorithm based on automated action planning called NAUSEA(autoNomous locAl manoeUvre and Scenario orchEstration based on automated action plAnning) is proposed for the cognitive layer of a driver model called SAIL (Scenario-Aware drIver modeL). Autonomous drivers equipped with SAIL/NAUSEA can make decisions according to their memories of scenario instructions and personal features (e.g., personalities), so as to make their controlling vehicles not only follow the scenario requirements and perform pre-defined actions, but also tolerate interferences and endow the scenario with rich behaviours as permitted. An experiment was used to evaluate NAUSEA and its implementation. It shows that NAUSEA is working properly but the implementation needs improvement.*

Key words: *Scenario Orchestration, Driving Simulation, Automated Planning, Temporal Reasoning*

1. Introduction

In driving simulation, a scenario[1] is a pre-defined environment that experimenters need a participant or simulator driver to experience; it includes the physical scenes, pre-defined traffic flow, interactions with other vehicles and measures that need to be collected. There are two basic requirements regarding scenarios: 1) the simulated vehicles in driving simulation should behave in a natural and realistic manner and 2) the scenario in the driving simulation should be reproducible. Researchers always put a focus on the second requirement and materials about how to satisfy the two requirements in the meantime are still not rich, that is, the following three questions are still open: 1) Which vehicles should interact with the participant? (Actor management problem); 2) How should those vehicles go to their proposed position? (Actor preparation problem) and 3) How should those vehicles be instructed what to do when interacting with participants? (Scenario orchestration problem).

In [Kea1], an actor manager was included in their HTPS-based architecture but no detail of actor management was mentioned. In [Ols1], Johan Olstam discussed the actor preparation and actor management problems with a focus on producing reproducible start conditions of plays in his algorithm. In [Pap1], autonomous vehicles can be directed to perform specific actions by receiving orders from an external director object, in which case humans handle scenario orchestration and scenarios may fail, e.g., the vehicle fails to follow the scenario instructions of overtaking because of some trigger failure and the whole scenario may be destroyed.

In order to find a solution for the three problems above and deal with failures, a "The Matrix" metaphor has been taken to design a driver model for driving simulation (see [The1] for a description of the film trilogy "The Matrix"). In short, this research separates the (virtual) driver from the vehicles. Drivers are treated as "Agent Smith" [Age1], and vehicles in the simulation are treated as "simulated humans" in "The Matrix" (at least one of them is driven by a participant). More details about this metaphor can be found in [Xio1]. A driver model called SAIL (Scenario-Aware drIver modeL) is therefore proposed to create a (virtual) driver that can:

1) Take control of one vehicle or a flock of vehicles dynamically by using Role Matching (solution to problem 1 and in section 2.2);

[1] In driving simulation literature, terms may have different meanings, e.g., "scenario", so in this paper/research terms are defined in order to be consistent and unified, not to be accurate. The Ontology for Scenario Orchestration mentioned later is trying to standardize the definitions and scenarios among different groups and different platforms.

2) Understand the scenario instructions and navigate any vehicles to the proposed Formation[2] Position needed for the scenario (solution to problem 2 and in section 2.3);

3) Follow the scenario instructions intelligently with a general plan and the ability of replanning in order to deal with failures (solution to problem 3 and in section 2.1) and

4) Cooperate with other (virtual) drivers so several drivers can run in parallel and coordinate.

SAIL uses OSO (Ontology for Scenario Orchestration) [Xio1] as its data source, which is stored in SDF (Scenario Definition File). OSO can also standardize scenario procedures, descriptions and concepts. SAIL uses an algorithm called NAUSEA (autoNomous locAl manoeUvre and Scenario orchEstration based on automated action plAnning) to make decisions. SAIL and NAUSEA have been implemented in a program called Smith. The vehicle or flock Smith is controlling are termed as "ego-vehicle" or "ego-flock" respectively.

This paper will introduce NAUSEA in SAIL including an experiment and results. It is organised as follows: Section 2 will first have a description of NAUSEA followed by Section 3 that is dedicated to show the experiment and results. Section 4 will include a conclusion and a discussion about future research and enhancement.

2. Algorithm Description - NAUSEA

SAIL is derived from ECOM architecture [Eng1] and illustrated in Fig. 1. The Perception layer is used to sense the outside environment and make necessary interpretation. The Cognition layer is used to maintain Memory and make decisions. Memory includes the Individual Features (e.g., personalities) and the World Model (e.g., the logical road network, previous Memory of World Model etc.). More details about SAIL can be found in [Xio1]. NAUSEA is based on temporal reasoning proposed in [Had1] and works in the Decision-Making layer in the Cognition (see Fig. 1). In order to adopt not only the temporal constraints[3] but also triggers including monitors (pre-conditions), success conditions (post-condition) and failure conditions(post-condition), extra procedures have been added. NAUSEA is described in Fig. 2 and procedures of Plan Evaluation, Role Matching and Regulating are elaborated as follows:

2.1 Plan Evaluation Procedure

NAUSEA maintains a General Plan ***Gr***$_\alpha$ to guide Smith through a scenario. Before introducing ***Gr***$_\alpha$ and its evaluation procedure, three concepts are given first: ***Action, Assignment*** and ***Recipe***.

An ***Action*** can be an Assignment-action or a pre-defined action described in the next paragraph: α, β_0, β_1, β_2 and β_3. It can be a High-Level Action or a Low-Level Action[4] (which are complex action and basic action in [Had1] respectively; the names have been changed in order to reflect the hierarchical architecture of SAIL). Each action is associated with the following parameters: ***ID, d, D, r, s, f***. ***ID*** represents the name/id of the ego-vehicle/flock. ***d*** represents the deadline of the action. ***D*** represents the duration of the action. ***r*** represents the release time of the action. ***s*** and ***f*** represent the start time and finish time of the action. Moreover, each action has a type and an action profile, e.g., "change desired speed to 10 mph" is an Assignment-action whose type is "Low-Level" and "Adapt-Speed" and whose action profile is "Desired Speed" with a value of "10.0" (mph).

Assignments are stored as Motivations in the Memory layer and specify what Smith needs to do in a scenario. An ***Assignment*** has four main components: monitors, success conditions, failure conditions and Assignment-actions. Monitors are used to trigger the corresponding Assignment-actions and success/failure conditions are used to check if the Assignment-actions have succeeded or failed respectively. Assignment-actions are what Smith should do in scenarios and can be driving behaviours or non-driving behaviours such as "request new vehicle", etc.

In every scenario, each Smith needs to finish a top High-Level Action α that can be either ***Perform-scenario*** or ***Free***. ***Free*** makes Smith ignore any Assignment and autonomously navigate the world, in which case route or a destination will be randomly chosen. ***Perform-scenario*** has only one ***Recipe***[5] that contains four sub-actions, namely, β_0, β_1, β_2 and β_3 (Fig. 4). β_0 (Get-to-the-initial-state) adopts initial state (e.g., initial speed, initial target speed etc.). β_1 (Generate-formation) means that Smith should "drive" the ego-vehicle(s) to the proposed Formation Position in an unsuspicious manner [Ols1] in order to perform the corresponding Assignment-actions, which is the process called Autonomous Local Manoeuvre in NAUSEA. Because the recipe of β_1 will change according to the dynamic environment, this action will not be further divided into sub-actions, but it will be monitored throughout the scenario in order to make sure that the ego-vehicle(s) can get to the position in time or on time. β_2 means Perform-assignment Action, and can be further divided into several Assignment-actions, which are represented as γ_0

[2] A Formation is a set of pre-defined local positions around the simulator driver/participant. Vehicles in driving simulation always interact with the participants at these Formation Positions. See figure 3 for further information.

[3] They include metric and precedence constraints, e.g., "action α ***Before*** action β" is a precedence constraint and "***the start time of*** β − **the finish time** of $\alpha \leqslant 100$ (seconds)" is a metric constraint. Details can be found in [Had1].

[4] A Low-level action is defined as an action that can be only performed in one way, one sequence and by one vehicle/flock.

[5] A Recipe is a set of actions that specifies how to perform a complex/High-Level Action. Recipe is a term borrowed from [Had1].

through γ_n (n is the number of Assignment-actions a Smith needs to perform). Each Assignment-action (γ_0 through γ_n) is contained in the Assignments stored in Memory. β_3 (Clean-up) should be specified by experimenters as an Assignment-action in most circumstances; however, it can be an autonomous action by changing it to the top-action of Free. At present, Smith always ignores Clean-up/β_3 and Get-to-the-initial-state/β_0 when constructing or evaluating Gr_α.

By using the start time ***s*** and finish time ***f*** of every action in the recipe of ***Perform-scenario*****,** Smith, or specifically, NAUSEA can generate a General Plan in Memory - a temporal constraint graph Gr_α [Had1], so ***s*** and ***f*** are represented as nodes in the graph Gr_α. The plan evaluation procedure uses the Floyd-Warshall algorithm to check the consistency of Gr_α, which is to check if there is any conflict regarding temporal constraints. When Smith is making decision and trying to finish the Assignment-actions specified in $\beta_{2,}$ it may change: 1) performer(s) of the action (***ID*** of the ego-vehicle/flock); 2) Assignment-actions and 3) the action ***Recipe*** of $\beta_{1.}$ Change of performer(s) may lead to the change of temporal constraints if permitted. Change of Assignment-actions will lead to the change of the temporal constraints and nodes in Gr_α. Change of the action ***Recipe*** of β_1 is caused by any failure in β_2 and Smith may need to navigate the ego-vehicle(s) to the proposed Formation Position first. It will lead to the change of nodes in $Gr_{\alpha,}$ as it will add or delete actions. In such a case the temporal constraints will not be changed but Smith needs to predict how long it will take for the ego-vehicle(s) to get to the Formation Position (duration and proposed finish time of β_1) and check if the duration and proposed finish time of β_1 are consistent with the temporal constraints in $Gr_{\alpha.}$.

2.2 Role Matching Procedure

Role Matching has three steps: Matching of Formation Position (Fig. 3), Matching of vehicle model and Matching of Gr_α. When Smith needs to find a vehicle/flock to perform some Assignments, he will first find a vehicle that is near the proposed Formation Position (flocks are usually used for ambient traffic flow). There are now two versions of Formation Position available for Smith. Fig. 3a is the original version for all circumstances while Fig. 3b is the version for rural road. For instance, if an Assignment needs a leader, the “Formation Position” of that Assignment will be specified as “L” in SDF (according to Fig 3b). Smith will try to find a vehicle that is near position “L”, which means that he will first try to find a vehicle in position “L”; if no vehicle is found, then try “LL”, “NSL” etc. Smith will then match the vehicle model with specified parameters regarding its model, its manufacturer, its max speed etc. If he can find a right vehicle in “L”, it will proceed to Matching of Gr_α, if not, he will try other near positions such as “NSL” or “LL” until he finds one. If Smith fails in finding a vehicle/flock, it will broadcast “Failure” or request vehicles from SMM (Scenario Management Module), which will be discussed in section 4. The last step is the Matching of Gr_α, which is to use the values of vehicle parameters to re-evaluate the time Smith needs to navigate the ego-vehicle(s) to the proposed position and see if the time is consistent with Gr_α.

2.3 Regulating Procedure

At present, only three behaviours are adopted: lane changing, overtaking and speed adaption. Lane changing is used to get to the lane that the Formation Position requires while overtaking is used to overtake slower vehicles and get to the Formation Position in time. Speed adaptation is mainly used to make Smith obey the speed limit and maintain a realistic speed trajectory when performing a turning movement. The Regulating procedure is only active in the phase of β_1, which is to generate formation for Assignment-actions. In the phase of β_2, which is to perform Assignment-actions, lane changing and overtaking are both forbidden while speed-adaptation is allowed if there is no speed requirement in Assignment-actions.

3. Experiment and Results

An experiment was designed to see if NAUSEA can provide proposed output and if the implementation – Smith - can work properly and stably. The desktop version of UoLDS (University of Leeds Driving Simulator) [Jam1], which is called Babysim has been used to conduct the experiment. A laptop is used to run Smith. It is equipped with an Intel® T2130 CPU and 2GB of memory and runs Ubuntu Linux 11.10 32bit. Communication between Babysim and Smith is based on a wired 10Mb hub. A video camera is also used to record the animation of the screen so that the driving activities can be recorded, which are needed for future examination or papers/presentations. The experiment contains one scenario and two phases, which are described in the following sections.

3.1 Experiment Description

3.2.1 Scenario Description

The scenario contains a 13.7 miles long (22 km) rural road with some curved road segments. There are three villages and five junctions along the road. The speed limit on the open road is 60 mph but in villages the speed limit is 30 mph. Assignments of “**Coherence**”, “**Layby**” and “**Gap Acceptance**” are supposed to provide measurements regarding driver’s behaviours but for the purpose of this experiment, these measures have been ignored. Because participants may sabotage Assignments and Smith can generate extra actions to compensate the Assignments, participants may experience less than or more than three Assignments, which depends on whether or not the failed Assignment can cause the whole scenario to fail and whether it can be reattempted. Moreover, the temporal constraints of the scenario are generated by manual estimation at present. Assignments that a participant could experience are listed below and illustrated in Fig. 5.

- Assignment of **Acc-BL**: **Acc-BL** is short for "Accelerate and Be Participant's Leader". It is the first Assignment that participants experience. This Assignment needs a Formation Position of "L", so vehicle with the id "1" is chosen in the beginning as the ego-vehicle. Smith will accelerate vehicle 1 and maintain speed of 30 mph;
- Assignment of **CL-BL**: **CL-BL** is short for "Change Lane and Be Participant's Leader". It is actually an action generated by the Regulating Layer in order to navigate vehicle 2 to the position of "L" so that the participant can have a leader after he/she has failed **Acc-BL**. Vehicle 2 is the ego-driver in this Assignment. The failure of **CL-BL** will lead to the failure of the whole scenario;
- Assignment of "**Coherence**": After the adoption of a leader by performing **Acc-BL** or **CL-BL**, the **Coherence** Assignment will start and last for 50 seconds[6]. During this period, the ego-vehicle (vehicle 1 or vehicle 2) will be the leader and varies its speed according to a sinusoid. The participant will be told to match the leader's speed and maintain his/her favoured distance to the leader. Two sub-Assignments can be generated: **Coherence1** and **Coherence2**, which are short for "**Coherence** performed by vehicle 1" and "**Coherence** performed by vehicle 2" respectively. The failure of **Coherence** will lead to the failure of the whole scenario;
- Assignment of "**Free Traffic Flow**": This Assignment is used to generate a traffic flow in order to prevent the participant from overtaking but because the traffic flow is of low density, the participant can still have chance to overtake. This Assignment starts with "**Coherence**" and stops in "**Layby**" that is elaborated next;
- Assignment of "**Layby**": In this Assignment, Smith needs to find a vehicle and then pull it out suddenly without indication, in which case the participant may accidently overtake the vehicle. Vehicle 3 is chosen as the ego-vehicle first and if it fails, vehicle 4 will be chosen, so two sub-Assignments can be generated as well: **Layby-V3** and **Layby-V4**, which are short for "**Layby** performed by vehicle 3" and "**Layby** performed by vehicle 4" respectively. The failure of **Layby-V4** will lead to the failure of the whole scenario;
- Assignment of "**Gap Acceptance**": After the **Layby** Assignment, the participant will arrive at a junction with oncoming vehicles whose gap is increasing; he/she is instructed to turn right if the gap is considered safe by him/her. This Assignment is not supposed to fail so the participant will be instructed to turn right.

If we call a Test Case a set of Assignments that a participant will experience or sabotage and the corresponding information Smith receives, there are in total 9 Test Cases in this experiment. This number is generated by considering all the combinations of the seven Assignments and the fact that 1) failures of **Coherence1/Coherence2/Layby-V4/CL-BL** will lead to the failure of the whole scenario and 2) some Test Cases are actually the same, for instance, in order to perform **CL-BL**, **Acc-BL** should be failed first. Hence, the Test Case of "fail **Acc-BL** and then fail **CL-BL**" is actually the Test Case of "fail **CL-BL**". See Table 3 for all the 9 Test Cases and corresponding desired output. From Table 3, we can see that in the normal Test Case, **Acc-BL** should be triggered (M: √) and succeed (S: √). Failure condition of **Acc-BL** should not be triggered (F: ×). **Coherence1** should be triggered and succeed by considering its duration (S: D). Failure condition of **Coherence1** should not be triggered. **Layby-V3** should be triggered and succeed by considering its duration. Failure condition of **Layby-V3** should not be triggered. **Gap Acceptance** and **Free Traffic Flow** should both be triggered and since there are no success or failure condition in this Assignment (S: N; F: N), Smith will not check the status of the Assignment after they are triggered. Moreover, in the Normal Test Case, **CL-BL**, **Coherence2** and **Layby-V4** are not triggered.

3.2.2 Phase One

In Phase one, five participants were recruited within the Institute for Transport Studies (ITS) to act as software testers. All of them are male and four of them have taken part in a driving simulation experiment before. Every participant will drive in the scenario twice or three times in two rounds. In round one, the participant will try Test Case 1, which means the participant should drive normally and experience **Acc-BL**, **Coherence1**, **Layby-V3** and **Gap Acceptance** in sequence. In round two, the participant will try two other Test Cases randomly in two sections and overtaking the leader may be allowed according to the requirement of each Test Case. The participant may try Test Case 3, 4, 7 in section one and 2,5,6,8,9 in section two, so two participants will not try any Test Case in section one. The rule is that the Test Case that has been tried by the last participant will not be considered for the next participant. The Test Cases each participant has tried can be found in Table 1. The input to Smith, which is the information of every vehicle in the Babysim was recorded in a Data-Log during each participant's drive.

3.2.3 Phase Two

[6]It has been set to 70 seconds in temporal constraints in order to make sure that Smith has enough time to monitor the Assignment and will not trigger failure accidentally. 50 seconds is used in the Assignment definition for success condition.

In Phase two, an automatic software test is performed by using the Data-Log recorded in Phase one. In this phase, every record of Test Case is played by a Log player in order to re-construct every participant's drive. Every log has been played 10 times, so there are in total 90 tests (9 × 10). Test of the Normal Test Case used the Data-Log of participant one in round one. The time that Smith makes the decision has been recorded in each test and is termed "Order Release time".

3.2 Results

Increase in the number of Assignments and the number of action ***Recipe*** for a high-level action will certainly make Smith slower and may cause failure due to the complexity of the Plan Evaluation Procedure. However, Assignments can be distributed to several Smiths and the number of action ***Recipe*** can be restricted, so we can safely ignore the size effect. Although the time spent on decision-making is platform-dependent, it can still reflect whether Smith is working stably as in a specific platform with the same scenario, Smith should spend almost the same time to trigger the same Assignment so that every participant can experience the same context when the same Assignment is triggered.

3.3.1 Can Smith Generate the Desired Output? (Is the Algorithm Working?)

In Phase one, Smith generated the desired output (13 Participant-based tests) and in phase two, Smith generated the desired output with a success rate of 100% in all the 90 tests. Moreover, the plan evaluation procedure is also working properly and its output can be found in Fig. 6, Fig. 6a shows the output in the Normal Test Case and Fig. 6b shows the output when Acc-BL fails (Test Case 2,3,7,8,9). Numbers in the graph are temporal constraints, e.g., 255 means that "Coherence1"/ "Coherence2" should start before 255 seconds from the start of the simulation.

3.3.2 How well did Smith Generate the Output? (Is the Implementation Good?)

Order lag is measured in Phase one by using the time that Babysim receives an order from Smith to minus the time when Smith makes the decision. The latter is the time stamp of the package that makes the monitor become true and Action Execution procedure is evoked. The result is shown in Table 2 and shows that Smith needs 2 ± 1 frame(s) to make decisions, which is supposed to be a reference as the time is a platform-dependent value.

3.3.3 Is Smith Stable?

The statistics description of the results is shown in Table 4 and the time value is all Babysim-based, e.g., a time value of 300 means 300 seconds after the start of the Babysim simulation. It shows that Smith is not stable enough, as the time point when he makes decision in the same test varies a lot from less than 1 frame to more than 10 frames compared to the mean order release time. Since there is no difference regarding algorithm or data structure when performing the same Assignment on the same machine, the cause of the variance is the communication mechanism used between the Perception and Cognition layers.

3.3.4 Comments from Participants

Participants were encouraged to give some general comments after their drive. Some of them have noticed that 1) in Coherence1 and Cohehence2, the acceleration rate of the leader is relatively high; 2) vehicle 3 or vehicle 4 in Layby-V3 or Layby V4 pulls out without advance indications (although they are designed to be a surprise Assignment) and 3) Lane changing trajectory of the vehicle is not smooth enough.

Hence, SAIL/NAUSEA is working properly as desired but the implementation of Smith needs enhancement regarding the communication mechanism between the Perception and Cognition layers. Moreover, the behaviours need enhancement as well although Smith is not responsible for the low-level trajectory generation, e.g., the trajectory of lane changing.

4. Conclusions

In this paper, a decision-making algorithm called NAUSEA has been designed for SAIL to solve the three problems and deal with "failures". SAIL/NAUSEA is working properly with a 100% success rate but the implementation needs enhancement. This driver model is a component of a framework called SOAV (Scenario Orchestration with Autonomous simulated Vehicles), which is designed to be an architecture that can orchestrate scenarios with autonomous vehicles, so that participants can experience rich, appropriate and reproducible scenarios. For scenario description, OSO and SDF can standardize scenarios and make them shareable among different simulators; for scenario interpretation, the driver model can naturally combine autonomous actions and scenario actions; for scenario execution, a Scenario Management Module (SMM) is proposed to cooperate with the driver and meet any macroscopic requirement, including traffic flow generation and vehicle creation/destruction.

A further experiment will be carried out in order to test SOAV. The enhancement for the next experiment will be: 1) adoption of a new communication mechanism between Perception and Cognition; 2) adoption of multi-triggers so that Smith can monitor more than one state variable in the simulation; 3) adoption of a Scenario

Management Module (SMM) so that vehicles can be added/destroyed dynamically according to traffic flow or Assignment requirements, in which case actor preparation can be tested dynamically and 4) adoption of dynamic temporal constraints based on the work from [Ols1].

SAIL adopted a goal-oriented hierarchical architecture, which can be used to assist the design of in-vehicle devices and adopt findings in related areas, such as driver model, driving behaviour, psychology etc. NAUSEA can also be used when self-management is needed for goal-oriented decision-making. All data in this experiment including OSO, SDF, videos and original data logs are available upon request.

5. Acknowledgement

Thanks to China Scholarship Council and University of Leeds for funding this research and to people in ITS (alphabetically): Dan Gillett, Daz Hibberd, Mojtaba Moharrer for their kind help before the experiments. Special thanks to Tony Horrobin (ITS) who gave me many suggestions ranging from scenario preparation to software test.

6. References

[Age1] Agent Smith(2011). Agent Smith. Retrieved October 16, 2011. http://en.wikipedia.org/wiki/ AgentSmith

[Eng1] Engström, J. and Hollnagel, E. (2007). A General Conceptual Framework for Modelling Behavioural Effects of Driver Support Functions. In Cacciabue, P. C (Editor), Modelling Driver Behaviour in Automotive Environment (pp. 61 - 84). Springer.

[Had1] Hadad, M., Kraus, S., Gal, Y. and Lin, R.(2003).Temporal Reasoning For A Collaborative Planning Agent In A Dynamic Environment. Annals of Mathematics and Artificial Intelligence, vol37, pp. 331-379.

[Jam1] Jamson, A. H., Horrobin, A. J., and Auckland, R. A.(2007). Whatever Happened to the LADS? Design and Development of the New University of Leeds Driving Simulator. In Proceedings of Driving Simulation Conference America (DSCNA), DSC'2007.

[Kea1] Kearney, J., Willemsen, P., Donikian, S., Devillers, F., de Beaulieu, C., and Rennes, F. (1999). Scenario languages for driving simulation. In Proceedings of Driving Simulation Conference,DSC'99, pages 377–393.

[Ols1] Olstam, J., Espi, S., Mardh, S., Jansson, J. and Lundgren, J. (2011) An algorithm for Combining Autonomous Vehicles and Controlled Events in Driving Simulator Experiments. Transportation Research Part C: Emerging Technologies, Elsevier.

[Pap1] Papelis, Y., Ahmad, O., and Schikore, M. (2001). Scenario definition and control for the national advanced driving simulator. In International Conference on the Enhanced Safety of Vehicles (ESV). SAE International.

[The1] The Matrix (franchise) (2011). The Matrix (franchise).Retrieved on February 2, 2012. http://en.wikipedia.org/wiki/The_Matrix_(franchise)

[Xio1] Xiong, Z., Carsten, O., Cohn, A.G. and Jamson, H. (2012). Driving with Smith: A Scenario-Aware Driver Model for Driving Simulation. To appear in the Proceedings of BRIMS 2012, March 2012.

1. Initialization Procedure: Read SDF and store relevant information into Memory; build a General Plan Gr_α according to the Recipe in Figure 4 and Scenario Assignments from Memory along with metric and precedence constraints;
2. Manoeuvre Loop: Run the following procedure until the end of Gr_α:
   (a) Plan Evaluation Procedure: If a new plan is found, check it's consistency. If it's consistent, continue, if no, go to 2b;
   (b) Role Matching Procedure: if a vehicle/flock is needed, do role matching and if a vehicle/flock can be found, update Gr_α and go to 2a; if no, go to 4;
   (c) Targeting Procedure: find the shortest route in length between current position and destination, if a route is found, continue; if failed, go to 2b;
   (d) Regulating Procedure: Autonomously navigate the local area, and go to the formation position according to the temporal constraints in Gr_α; if failed, go to 2b;
   (e) Situation Assessment Procedure:
      i. Assignment Checker Procedure: Check if some Assignment has been triggered sequentially or according to its monitor, set the online release time r_β^{online} of its corresponding Assignment-action β and add the Assignment to Action Execution Queue;
      ii. Action Execution Procedure: Check if there is any Assignment in Action Execution Queue, if true, check if r_β^{online} of its Assignment-action is consistent with Gr_α, if yes, execute the Assignment-action whose r_β^{online} is earlier than or equal to present time by evoking 3, delete the Assignment from the Action Execution Queue and add it to Action Monitor Queue; if no, go to 2b;
      iii. Action Checker Procedure:
         - check postconditions of any Assignment in Action Monitor Queue, if true, set the adjusted deadline of its Assignment-action β: d_β^{adj}; if d_β^{adj} is consistent with Gr_α, delete the Assignment from the Action Monitor Queue and related nodes from Gr_α, set corresponding Assignment Status if necessary; if it's not consistent, go to 2b;
         - if the Assignment-action did not finish after its duration, set the corresponding Assignment to "Failure" and go to 2b;
         - Check if there is some "Failure" conditions regarding the Assignment, if yes and it becomes true, set the Assignment to "Failure" and go to 2b;
3. Action Procedure: Send out Orders, if success, go to 2;
4. Failure Broadcast procedure: Broadcast "Failure".

Fig.2 Algorithm Description of NAUSEA

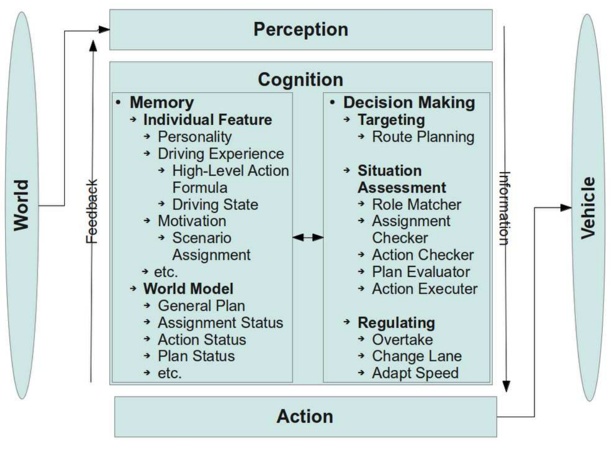

Fig. 1. Scenario-Aware Driver Model

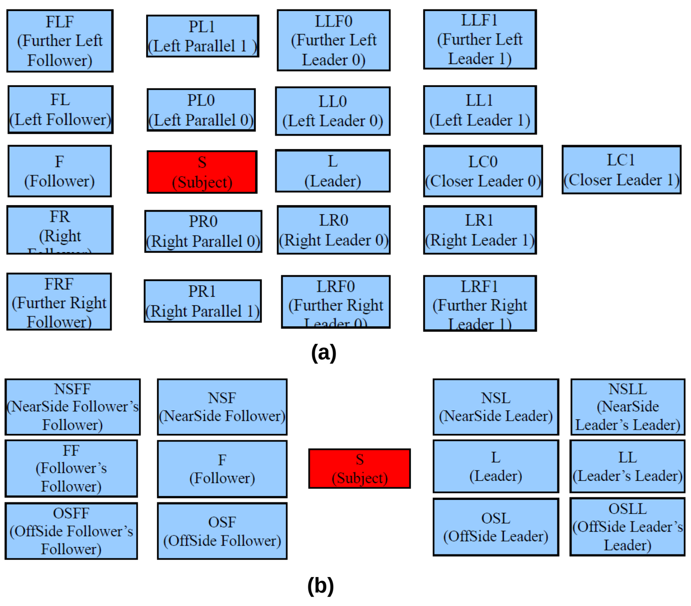

Fig. 3. Formation Position in SAIL/NAUSEA

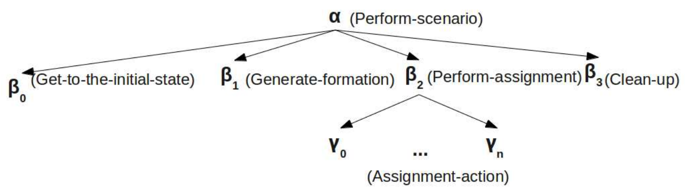

Fig. 4. Action Recipe for Smith (Perform-sceanrio)

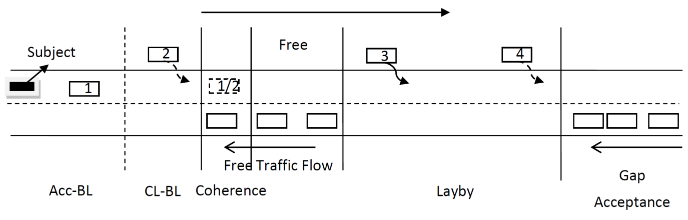

Fig. 5. Illustration of Scenario

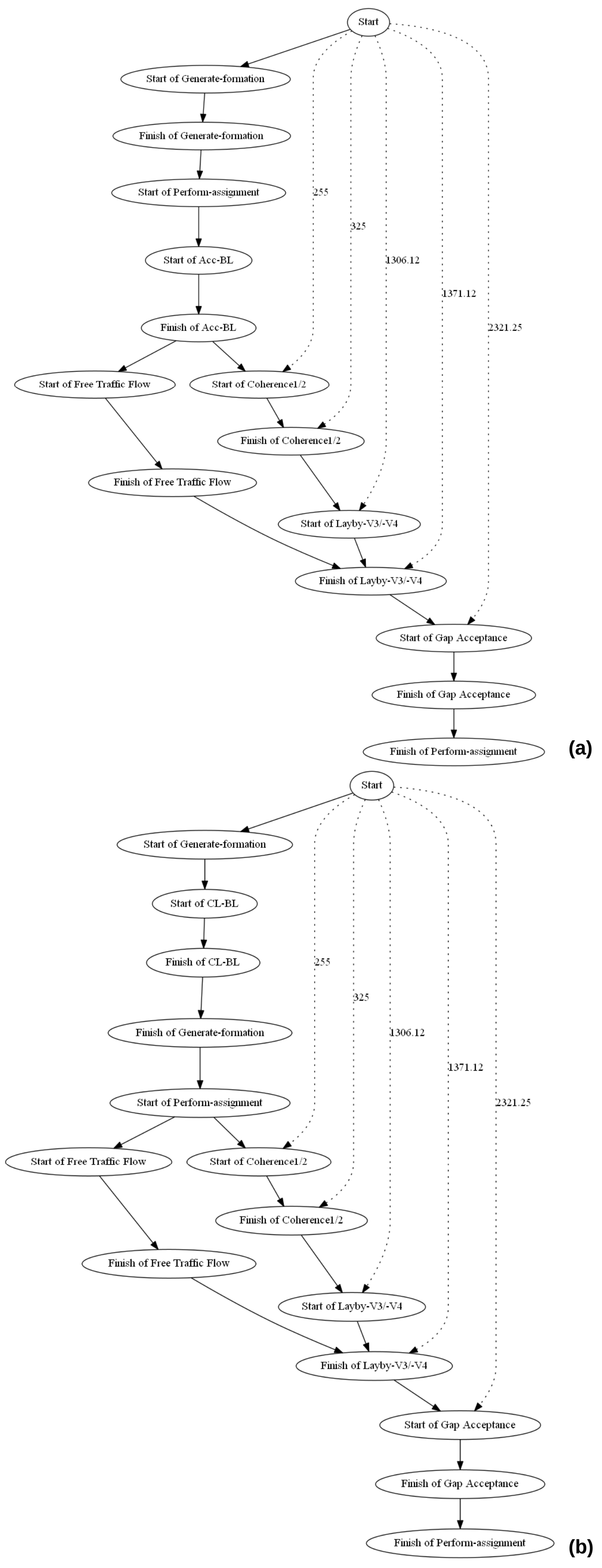

Fig. 6. Output of Plan Evaluation Procedure in Normal (a) and Failure (b) Test Case

Participant No.	Test Case in Round One	Test Case in Round Two	
		Section One	Section Two
1	1	7	9
2	1		8
3	1	4	2
4	1		6
5	1	3	5

Table 1. Test Case Tried by Every Participant

Table 2. Statistics of Order Lag in Phase One

Minimum	Maximum	Mean	Std. Deviation
.0000163	.1000000	.031511634	.0148830995

Table 3. Desired Output of Each Test Case

Table 4. Statistics of Order Release Time in Phase Two

Test Case		Total Trigger Number/ Order Number	Acc-BL			CL-BL			Coherence1			Coherence2			Layby-V3			Layby-V4			Gap Acceptance			Free Traffic Flow		
			M	S	F	M	S	F	M	S	F	M	S	F	M	S	F	M	S	F	M	S	F	M	S	F
1	Normal	6/5	√	√	×				√	D	×				√	D	×				√	N	N	√	N	N
2	Fail **Acc-BL**	8/6	√	×	√	√	√	×				√	D	×	√	D	×				√	N	N	√	N	N
3	Fail **CL-BL**	4/2	√	×	√	√	×	√																		
4	Fail **Coherence(1)**	5/3	√	√	×				√	×	√													√	N	N
5	Fail **Layby-V3**	8/6	√	√	×				√	D	×				√	×	√	√	D	×	√	N	N	√	N	N
6	Fail **Layby-V4**	8/5	√	√	×				√	D	×				√	×	√	√	×	√				√	N	N
7	Fail **Acc-BL** and **Coherence2**	7/4	√	×	√	√	√	×				√	×	√										√	N	N
8	Fail **Acc-BL** and **Layby-V4**	10/6	√	×	√	√	√	×				√	D	×	√	×	√	√	×	√				√	N	N
9	Fail **Acc-BL** and **Layby-V3**	10/7	√	×	√	√	√	×				√	D	×	√	×	√	√	D	×	√	N	N	√	N	N

M: Monitor

S: Success Condition

F: Failure Condition

√ **: Triggered**

× **: Not Triggered**

N **: Not Available**

D **: Duration-Based Success Condition**

Test Case	Coherence1				Coherence2				Layby-V3				Layby-V4				Gap Acceptance			
	Min	Max	Mean	Std. Deviation	Min	Max	Mean	Std. Deviation	Min	Max	Mean	Std. Deviation	Min	Max	Mean	Std. Deviation	Min	Max	Mean	Std. Deviation
1	247.067	247.183	247.108	0.037					710.400	710.433	710.414	0.012					869.350	869.367	869.361	0.009
2					200.350	200.917	200.541	0.177	661.467	661.567	661.500	0.031					822.350	822.383	822.365	0.012
3																				
4	244.000	244.333	244.074	0.103																
5	238.317	238.533	238.386	0.070					699.617	699.650	699.627	0.011	808.500	808.533	808.511	0.011	853.533	853.583	853.564	0.015
6	238.933	239.033	238.971	0.038					693.333	693.367	693.342	0.012	796.250	796.267	796.258	0.009				
7					181.317	181.983	181.574	0.194												
8					215.167	215.800	215.352	0.186	674.783	674.833	674.794	0.015	795.033	795.067	795.042	0.012				
9					195.817	196.583	196.147	0.269	662.850	662.917	662.882	0.020	784.533	784.633	784.562	0.030	829.533	829.683	829.618	0.040