# Assessing Drivers' Situation Awareness in Semi-Autonomous Vehicles

## ASP based Characterisations of Driving Dynamics for Modelling Scene Interpretation and Projection

Jakob Suchan Jan-Patrick Osterloh
German Aerospace Center (DLR)
Institute for Systems Engineering for Future Mobility, Oldenburg, Germany
jakob.suchan@dlr.de, jan-patrick.osterloh@dlr.de

Semi-autonomous driving, as it is already available today and will eventually become even more accessible, implies the need for driver and automation system to reliably work together in order to ensure safe driving. A particular challenge in this endeavour are situations in which the vehicle's automation is no longer able to drive and is thus requesting the human to take over. In these situations the driver has to quickly build awareness for the traffic situation to be able to take over control and safely drive the car. Within this context we present a software and hardware framework to asses how aware the driver is about the situation and to provide human-centred assistance to help in building situation awareness. The framework is developed as a modular system within the Robot Operating System (ROS) with modules for sensing the environment and the driver state, modelling the driver's situation awareness, and for guiding the driver's attention using specialized Human Machine Interfaces (HMIs).

A particular focus of this paper is on an Answer Set Programming (ASP) based approach for modelling and reasoning about the driver's interpretation and projection of the scene. This is based on scene data, as well as eye-tracking data reflecting the scene elements observed by the driver. We present the overall application and discuss the role of semantic reasoning and modelling cognitive functions based on logic programming in such applications. Furthermore we present the ASP approach for interpretation and projection of the driver's situation awareness and its integration within the overall system in the context of a real-world use-case in simulated as well as in real driving.

## 1 Introduction

With the rise of automated and semi-automated driving, a range of new challenges have moved into sight of developers. In particular semi-autonomous vehicles pose important questions when it comes to the interplay between driver and vehicle, since for these systems it is critical that the cooperation of the human and the automation system is seamless. To ensure this, the automation system needs to be designed with the needs of the human in mind and ideally have the functionality to asses the mental state of the driver and the driver's interaction with the system. Such functionality is especially important in situations when the driver and the automation have to function together, in order to safely operate the vehicle. A particular challenge in this context, and of importance with respect to this paper, is the question of how to enable the driver to quickly take over control from the automation system in situations, in which the automation cannot ensure safe driving, e.g., in unforeseen situations, or in situations outside of the Operational Design Domain (ODD) of the automation.

S. Costantini, E. Pontelli, A. Russo, F. Toni,
R. Calegari, A. D'Avila Garcez, C. Dodaro, F. Fabiano,
S. Gaggl, A. Mileo, (Eds.): ICLP 2023
EPTCS 385, 2023, pp. 300–313, doi:10.4204/EPTCS.385.29

The SituWare project presented in this application paper aims at developing a system to provide the driver with assistance in such situations, by modelling the driver's awareness of the situation and guide their attention to critical elements of the situation possibly missed by the driver. The focus of this paper is on highlighting the application of declarative logic programming methods as a means to model the driver's mental processing of the situation and demonstrate its integration within a large-scale modular assistive system. The work presented in this paper is driven by considerations in the fields of human-centred design, cognition, and formal semantic reasoning.

**Human-centred design for (semi-)autonomous vehicles.** With the above challenges in mind human factors play a central role in the development of novel mobility systems and autonomous vehicles [12] and development of novel systems is often accompanied and driven by empirical research investigating how humans interact with these systems, either for external communication with vulnerable road users, e.g. [18, 26], and for internal communication e.g. in handover scenarios [7, 8, 16] or for acceptance and user experience [20].

**Cognitive abilities for driving assistance systems.** Research in the area of cognitive systems and cognitive modelling is concerned with developing methods and tools that reflect cognitive abilities. Most directly connected to the topic of this paper and the development of SituSYS is the research on Situation Awareness and its modelling in computational systems [3]. In the context of driving assistance systems research on Situation Awareness has mostly been concerned with modelling e.g. by [11, 17, 19].

**Semantic reasoning about dynamic situations.** A key factor in developing such capabilities is the ability to abstractly represent the driving environment and the traffic dynamics within it, to interpret and reason about them on a semantic level. In this context logic programming and Answer Set Programming (ASP) [4, 5, 6] in particular has evolved as a powerful tool for semantic reasoning about dynamic scenes. For instance, ASP has been used in the area of stream reasoning [2] as a general tool for reasoning about large scale dynamic data, e.g, in the driving domain [13], for decision-making [9], or for recognition and reasoning about events [24]. Furthermore, [23] has developed a general method for visual abduction based on ASP, and applied it in the area of online semantic sense-making with commonsense knowledge in the context of safety-critical situations in driving [21, 22].

Within this paper we build on these works to develop an ASP-based approach for assessing the driver's situation awareness by characterising driving dynamics for modelling and reasoning about the driver's interpretation and projection of the scene dynamics.

## 2 SituWare: An Online System for Assessing Situational Awareness

The general focus of the SituWare project is f build and evaluate the technological basis for developing a usable software and hardware system (*SituSYS*), targeted at the reliable detection, interpretation, and consideration of the driver's situational awareness. In this section we discuss the relevance of situation awareness in (semi-)autonomous driving and provide an overview of the full system and its components.

### 2.1 Situation Awareness in (Semi-)Autonomous Driving

Today, in the automotive domain and within the framework of the SAE J3016 automation level system, it is generally assumed that the driver observes the environment and is thus immediately available as a fallback level (SAE level 2 - partial automation), can react to a request to intervene (SAE level 3 -

conditional automation) or can take over the vehicle control outside the operational area of the automation (SAE level 4 - high automation). Previous studies on handover scenarios, especially in the context of conditional automation, show that a considerable amount of time of at least 7 to 10 seconds must be reserved for the driver to safely take over the driving task. This time is needed by the driver(s) to gain an accurate situation awareness so that a safe handover is accomplished. Current approaches for handover scenarios from the system to the driver consider the current driver's state only peripherally and very roughly. However, the potential of such recognition and interpretation is very large. Adaptive interaction sequences can be used to optimally return the driver to the driving task. In addition, a more detailed picture of the driver's state also allows for the adaptation of the driving behavior of the automation system, so that larger time reserves can be kept available for a highly distracted driver, while time reserves can be reduced for an attentive driver without reducing safety. One of the most prominent models for Situation Awareness is the one developed by Mica Endsley [3]. The idea behind this model is that Situation Awareness is built in three Stages, namely the *Perception*, *Interpretation*, and *Projection* stage, also known as Level 1 to Level 3 Situation Awareness (L1 - L3):

L1. At Level 1, the data and elements of the environment that build the current situation are perceived. The main factor that influences Level 1 Situation Awareness is the focus of attention that is apparent in the situation, which in turn is guided by the goals and objectives of the user, as well as their expectations. Previous Experience and Training can influence the attention process and thus the ability of the user to perceive the environment correctly. Drivers need to perceive the cars around their own car, signs along the road, the road curvature and condition, and lots of other things.

L2. Level 2 Situation Awareness builds upon Level 1 through interpretation and comprehension of the perceived elements. At Level 2, a holistic picture of the environment including the significance of objects and events is formed. This interpretation is influenced by the goals of the user, previous experience as well as the workload and stress. In the automotive example, drivers need to understand that proximity of other cars might indicate a risk, that there are speed limits to keep, or the importance of the other signs on the road.

L3. At Level 3, the user builds a projection of the current situation into the future, based on the comprehension built at Level 2. Level 3 Situation Awareness is mainly influenced by the previous experience and training of the user, since this builds his knowledge on how the entities in the environment may act in the near future. For drivers, this could mean to predict the future speed and positions of other cars as well as possible actions the other drivers might take (like changing lanes, or breaking) as well as upcoming speed limits and the risk associated with the road conditions ahead.

## 2.2 A Modular System implemented in ROS

Objective of the SituWare project is to build *SituSYS*, a system that predicts the situation awareness of a driver, calculates possible deviations from an optimal situation awareness and finally uses specialized interaction techniques to improve the driver's situation awareness. Figure 1 shows a conceptual overview of the architecture of *SituSYS*: *SituSYS* consists of three parts, the sensor layer *SituSENSORS* which measures the driver and environment, the model layer *SituMODEL* which calculates the situation awareness, and *SituHMI* for the interaction. Starting at the sensor-layer *SituSENSORS*, multiple *Vehicle Sensors* are used to sense the environment and vehicle state, i.e. surrounding objects, signs, current vehicle speed and automation state. In addition to the vehicle sensors, an eye-tracker is used to detect the gaze of the driver. The calculated gaze vector is then used in the *Perception*-Model within *SituMODEL* to predict which elements in the environment and in the car has been looked at, i.e. other cars, street signs, or

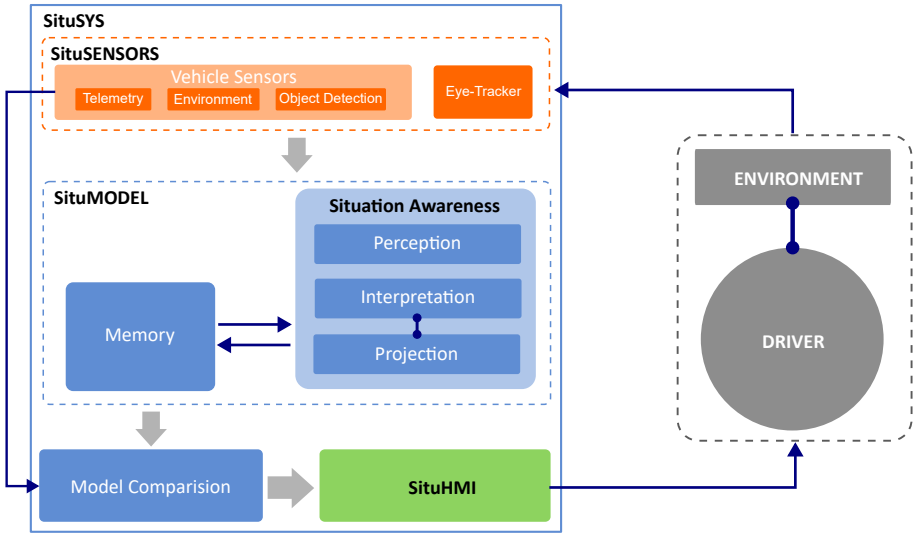


Figure 1: **SituSYS**: Conceptual Overview

any cockpit elements. Based on this information, the objects that have been perceived are then written into the *Memory*-Model. The *Memory*-Model implements retrieval and forgetting processes, and can be accessed by the other *SituMODEL* components. The *Interpretation*- and *Projection*-Models are for the calculation of the situation interpretation and the projection of the future state (A detailed description can be found in Section 3.2). The outputs of the *Perception*-, *Interpretation*-, and *Projection*-Models are then used by the *Model Comparison* to calculate the overall situation awareness (Level 1, Level 2 and Level 3 in Endsley's Model), by comparing the content in the memory, that has been produced by the components of the *SituMODEL*, with the data gathered by the vehicle sensors. Although the vehicle sensors have an error probability associated with them, we assume in this case that this is the *ground truth* for *SituSYS*. The *Model Comparison* then weights the differences between the memory and the ground truth and calculates a list of diverging elements sorted by priority. This list is then used by *SituHMI* to direct the focus of attention to the most important element. In SituWare we tested different interaction methods for that purpose [1].

To organize and coordinate the implementation of the different components of *SituSYS*, the *ROS* framework has been used. Each component of *SituSYS* has been implemented as separate node, allowing a modularization of the system. *ROS messages* describing the data exchanged between the components are used to provide a standardised interface between the different modules. The use of the ROS framework also facilitates the use of different sources for the input, i.e. we connected *SituSYS* to two different driving simulators as well as to a real car, by implementing a dedicated node that collects the data from the simulator or car and then sends them in the defined messages to *SituSYS*, and receives the messages from *SituHMI* to implement the selected interaction method in the vehicle/simulator.

## 2.3 Technical Setup and Data

The SituWare project integrates SituSYS within two different simulators and one real semi-autonomous vehicle and provides the hardware basis for sensing the ego vehicle and the environment together with mobile tracking of the driver's gaze. Figure 2 shows the simulated scenario within one of the two simulators used within the SituWare project.

a) Simulated scene of construction site after takeover b) Driving Simulator

Figure 2: **Simulation Environment:** The simulated scene and the simulator setup.

► Scene Data The vehicle sensors implemented in the specific platform are used to sense the state of the ego vehicle and to sense the environment including the vehicles within it. This information is published as *ROS messages* and used by the modules of *SituSYS*. The messages contain data on the ego vehicle, the automation system, and other vehicles in the scene as listed in table 1.

► Eye-Tracking Data The driver's perception is sensed using mobile eye-tracking that provides gaze coordinates and fixations within an ego view of the scene, as captured by the camera of the eye tracker. These coordinates are translated to the 3-dimensional space of the scene, which allows mapping of gaze data to scene elements. Within *SituSYS* the gaze data is provided as a 3D vector. The modular design of the system abstracts from the actual eye-tracking system, however, in the case study presented in this paper the PupilLabs mobile eye-tracker is used, which is equipped with a eye camera capturing the gaze with up to $120Hz$ and providing an accuracy of $0.60°$.

## 3 Assessing the Driver's Situation Awareness

Within *SituSYS* situation awareness is modelled by the perception module and the interpretation & projection module. The perception module implements 1st level Situation Awareness using scene data from the *SituSENSORS* together with eye-tracking data to calculate fixation probabilities for each scene object.

**Perception**

Level 1 Situation Awareness – *perception of the scene*[1] – is estimated based on the gaze data of the driver which is recorded by *SituSENSORS*. From this information a probability is computed for each scene object giving the likelihood that the driver has fixated the object together with a fixation time providing the duration of the fixation on the object. These measures are then published within the *ROS message* as an input for the interpretation & projection module, as well as the model comparison.

2nd and 3rd level Situation Awareness are implemented together in the interpretation & projection module. These consist of:

- Level 2 Situation Awareness – *interpretation of the scene* – models the awareness of the driver regarding scene elements in the current situation;

[1]Technical details on the used method for calculating the probabilities are out of the scope of this paper. For the examples of this paper we consider the perception module to be a black-box system providing the information if an object was perceived by the driver.

| Attribute | Type | Description |
|---|---|---|
| | Ego Vehicle | |
| ID | *string* | The *ID* of the ego vehicle. |
| type | *string* | The *type* of the ego vehicle. |
| position | *{float, float, float}* | The *position* of the ego vehicle as 3D vector. |
| orientation | *{float, float, float}* | The *orientation* of the ego vehicle as 3D vector. |
| velocities | *{float, float, float}* | The *velocity* of the ego vehicle as 3D vector. |
| indicator_left | *bool* | Truth value whether the *left indicator* is active. |
| indicator_right | *bool* | Truth value whether the *right indicator* is active. |
| acceleration | *float* | The *acceleration* of the ego vehicle. |
| current_speed_limit | *int* | The *current speed limit* holding for the ego vehicle. |
| current_lane | *float* | The *current lane* the ego vehicle is on. |
| | Ego Automation | |
| takeover_request | *bool* | Truth value whether the *takeover request* is active. |
| time_until_odd_boundary | *float* | *Time* until the driver has to take over. |
| criticality_level | *int* | *Criticality level* of the takeover request. |
| takeover_reason | *string* | *Reason* for the takeover request. |
| ego_automation_state | *bool* | Truth value whether the *automation* is active. |
| | Other Vehicles | |
| ID | *string* | The *ID* of the traffic vehicle. |
| type | *string* | The *type* of the traffic vehicle. |
| position | *{float, float, float}* | The *position* of the traffic vehicle as 3D vector. |
| orientation | *{float, float, float}* | The *orientation* of the traffic vehicle as 3D vector. |
| velocities | *{float, float, float}* | The *velocity* of the traffic vehicle as 3D vector. |
| acceleration | *{float, float, float}* | The *acceleration* of the traffic vehicle as 3D vector. |
| dimension | *{float, float, float}* | The *dimension* of the traffic vehicle as 3D vector. |
| lane | *int* | The *lane* the traffic vehicle is on. |
| fixation_probability | *float* | The *probability* that the driver fixated the vehicle. |
| fixation_time | *int* | The *duration* of the driver's fixation on the vehicle. |

Table 1: **Scene Data:** Relevant data-points from *SituSYS*.

- Level 3 Situation Awareness – *projection of scene dynamics* – models the expectations of the driver how the scene will evolve, i.e., the scene dynamics with respect to the task of the driver.

The implementation of these levels is based on semantic characterisations of the dynamics of the driving domain, which are declaratively defined within answer set programming (ASP) [4, 5, 6]. In particular, the interpretation & projection module of *SituSYS* consist of an online Python process maintaining a representation of the scene and uses an integrated ASP solver to generate interpretation and projection models based on the scene data from the *SituSENSORS* and the perception data from the perception module. In the following we provide the formal characterisation of the driving domain, and describe the interpretation and projection process in detail.

## 3.1 The Driving Domain

The domain is characterised by $\Sigma<\mathcal{O}, \mathcal{E}, \mathcal{R}, \mathcal{T}, \Phi, \Theta>$, which is used to formalise the driver's representation of the scene dynamics. In particular, the driver's belief state is represented on the one hand by the static and dynamic properties of the scene and the elements within it given by the domain objects, the basic entities representing these objects, and the spatial and temporal aspects of the scene $< \mathcal{O}, \mathcal{E}, \mathcal{R}, \mathcal{T} >$. On the other hand it is represented by the high-level event dynamics of perceived events and possible future events given by $< \Phi, \Theta >$.

| Events | Description |
|---|---|
| $change\_lane(Entity, Lane_1, Lane_2, Location)$ | An Entity changing the lane from $Lane_1$ to $Lane_2$ to a specific *Location*. |
| audio_signal_start | The audio signal indicating that the driver has to take over started. |
| audio_signal_end | The audio signal indicating that the driver has to take over ended. |
| takeover_manual | The driver takes over control of the vehicle. |
| takeover_automation | The automation takes over control of the vehicle. |

| Fluents | Description |
|---|---|
| curr_task(Task) | The current task, the driver has to perform. |
| automation | If the driving automation of the vehicle is on or not. |
| audio_signal | If the audio signal indicating that the driver has to take over control of the vehicle is on or not. |
| on_lane(Entity, Lane) | The lane a scene entity is driving on. |

Table 2: **Event Dynamics:** Exemplary events and fluents applicable in take-over situations of the use-case described in section 3.2.

▶ **Domain Objects ($\mathcal{O}$) and Spatial Entities ($\mathcal{E}$).** The scene consists of different scene elements, in particular we are considering the ego vehicle and other vehicles in the scene, as well as the lanes on the road and gaps between vehicles.

*Vehicles.* We distinguish between the ego vehicle and other vehicles in the scene, constituting the traffic. We use the following objects for representing vehicles:

- The *ego vehicle*: $\mathcal{O}_{ego} = ego$;
- other vehicles in the traffic: $\mathcal{O}_{trf} = \{trf_1, ..., trf_n\}$.

These elements are geometrically represented as spatial entities $\mathcal{E} = \{\varepsilon_1, ..., \varepsilon_n\}$ within the 3-dimensional scene space. Additionally they are assigned dynamic and static attributes as obtained from the egos driving system (for the attributes of the ego vehicle) and the ego vehicle sensors (for estimated attributes of other vehicles) as detailed in section 2.3.

*Lanes.* These are based on the OpenDRIVE standard [25] , in which lanes are numbered with positive and negative numbers and 0 represents the middle of the road. Lanes are adjacent when the lane *ids* are consecutive.

- Lanes: $\mathcal{O}_{lanes} = \{lane_1, ..., lane_n\} \cup \{lane_{-1}, ..., lane_{-n}\}$.

We define the adjacency of lanes $lane_i, lane_j \in \mathcal{O}_{lanes}$ on the road using the predicate $adjacent(lane_i, lane_j)$.

*Gaps.* Of particular interest for the task at hand are gaps between vehicles in the scene. Therefore we introduce objects representing these gaps.

- Gaps: $\mathcal{O}_{gaps} = \{gap_{trf_i,trf_j}, ..., gap_{trf_p,trf_q}\}$.

Gaps are declaratively defined based on the vehicles in the scene using the predicate $gap(trf_i, trf_j)$, where $trf_i, trf_j \in \mathcal{O}_{trf}$. Additionally we define the predicate $gap\_size(gap_{trf_i,trf_j}, size)$, where $gap_{trf_i,trf_j} \in \mathcal{O}_{gaps}$ and *size* is a number representing the distance between $trf_i$ and $trf_j$.

▶ **Spatial Configuration ($\mathcal{R}$).** The spatial arrangement of the entities in the scene is represented from an egocentric perspective in the context of the road, i.e., representing whether vehicles are ahead or behind the ego vehicle, on the same lane, or on the lane to the left or right of the ego vehicle, and how many other vehicles are between a vehicle and the ego vehicle. To this end we define the following relations in $\mathcal{R}$ holding between the ego vehicle and the other scene elements.

- *Relative longitudinal direction*: $R_{rel\_long} = \{ahead, behind, overlapping\}$, representing the direction of an object on the longitudinal axis based on the road layout.
- *Relative lane*: $R_{rel\_lane} = \{same, left, right\}$, representing the lane an object is on, relative to the lane the ego vehicle is on.
- *Relative ordering*: $R_{rel\_order} = n+1$, where $n$ is the number of other vehicles between the vehicle and the ego vehicle, representing the position of an object with respect to the ego vehicle and the other vehicles on the road.

▶ **Time ($\mathcal{T}$).** We represent time using time points $\mathcal{T} = \{t_1, ..., t_n\}$. These are used to denote that dynamic object properties and spatial relations between basic entities representing scene objects hold at a certain time, as well as to describe temporal aspects of event dynamics.

▶ **Driving Event Dynamics (<Φ, Θ>).** We use the event calculus notation to define the event dynamics in the driving domain. Towards this we define **fluents** $\Phi = \{\phi_1, ..., \phi_n\}$ and **events** $\Theta = \{\theta_1, ..., \theta_n\}$ to characterise dynamic properties of the scene objects and high-level events (e.g., Table 2). We use the axioms $occurs\_at(\theta, t)$ denoting that an event occurred at time $t$ and $holds\_at(\phi, v, t)$ denoting that $v$ holds for a fluent $\phi$ at time $t$.

## 3.2 Reasoning about Situation Awareness Level 2 & 3: Interpretation and Projection

The interpretation and projection module is based on declarative characterisations of driving dynamics (as defined in section 3.1 ) implemented in ASP, using Event Calculus [10] for reasoning about events in the scene. In particular, we are building on the Event Calculus (EC) as formalised in [14, 15]. The module is implemented as a hybrid system, in which an online Python process maintains a representation of the mental belief state of the driver and uses an integrated ASP solver for generating the interpretation and the projection model. For this, the module uses the scene data $\mathcal{S}$ containing the ego vehicle data and the environmental data including the gaze data of the driver, together with the characterisations of the driving domain in $\Sigma$.

**Application: The Case of Take-Over Situations in a Construction Environment**

As a use case we have applied *SituSYS* in the context of a case study conducted in the driving simulator. The case study implemented a situation in which a driver had to take over the control of a highly automated vehicle at a section of a highway, where the current lane ended because of a construction site and the vehicle had to change lanes. The driver got notified about the upcoming take-over and had a certain time window to get familiar with the situation and take over control to manually perform the lane change.

As a test case the overall system is applied in the context of the above case study, in which the driver has to take over control from the automation to safely drive the vehicle. Fig. 3 shows an exemplary situation in the context of the case study and depicts the interpretation and the projection step.

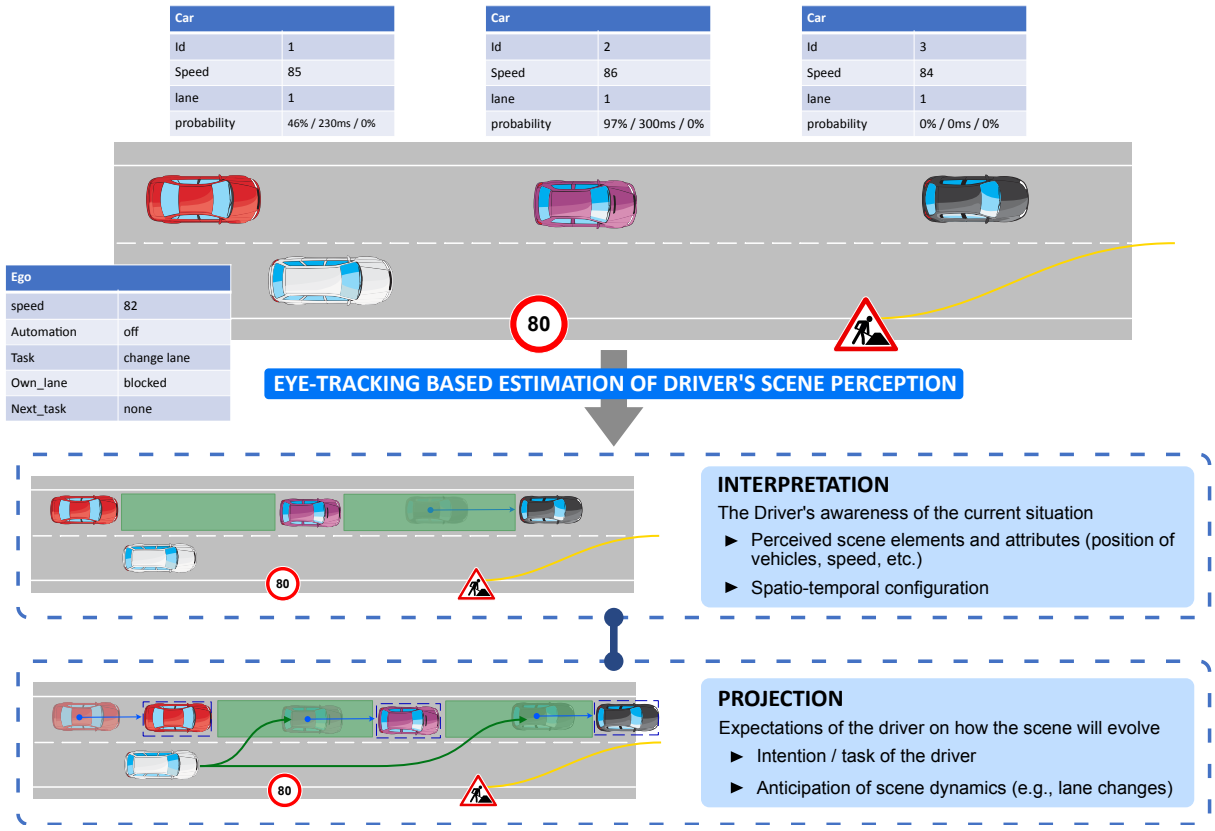


Figure 3: **Application Example:** Interpretation & Projection

**Computational Steps for Situation Awareness Level 2 & 3.** The overall process for interpretation and projection consists of the following steps (S1-S3, also refer to Alg. 1) performed at each time point:

S1. Update Scene Elements and Predict Current State. Update the driver's mental belief state ($\mathcal{MBS}$) with the scene elements for which the fixation probability is above the fixation threshold, and predict the current position of all scene elements that are part of the driver's mental belief state, using a Kalman Filter base motion model, assuming constant velocity.

S2. Generate the Problem Specification. Generate an ASP problem statement, containing the scene elements that are part of the driver's mental belief state, and the characterisations of the driving domain in $\Sigma$.

S3. Integrated ASP Solving. Generate the interpretation model ($\mathcal{IM}$) and the projection model ($\mathcal{PM}$) by solving the generated problem specification using the Clingo solver integrated within the Python process.

Within this process the interpretation and the projection are implemented as follows:

► Interpretation. The interpretation level of situation awareness is modelled as an extrapolated representation of the scene, consisting of the perceived scene elements, together with the arrangement of these elements and the events happening in the scene. When the probability that the driver has fixated a particular object exceeds a certain threshold we register it within the scene representation and use a Kalman Filter based motion model assuming constant velocity to maintain a representation of the object

**Algorithm 1:** Interpret_and_Project($\mathcal{S}$, $\Sigma$)

**Data:** Scene data ($\mathcal{S}$), and the characterisation of the driving domain $\Sigma$

**Result:** Interpretation Model ($\mathcal{IM}$), Projection Model ($\mathcal{PM}$)

$\mathcal{MBS} \leftarrow \varnothing$
**for** $t \in T$ **do**
  **for** $object \in \mathcal{MBS}$ **do**
    $position_{object} \leftarrow kalman\_predict(object)$
  **for** $object \in \mathcal{S} \wedge not\ object \in \mathcal{MBS}$ **do**
    **if** $fixation\_probability_{object} > fixation\_threshold$ **then**
      $\mathcal{MBS} \leftarrow \mathcal{MBS} \cup object$
  **for** $object \in \mathcal{S} \wedge object \in \mathcal{MBS}$ **do**
    **if** $fixation\_probability_{object} > fixation\_threshold$ **then**
      $position_{object} \leftarrow kalman\_update(object)$
  $< \mathcal{IM}, \mathcal{PM} > \leftarrow ASP\_solve(\mathcal{MBS}, \Sigma)$
**return** $< \mathcal{IM}, \mathcal{PM} >$

while the driver is not fixating on it. When the driver is fixating the object again the estimated position is updated with the sensed one. In this way we estimate the driver's mental belief state about the movement of scene elements.

This estimated mental representation of the scene is then used together with the characterisations of the driving domain in $\Sigma$ to generate the interpretation model ($\mathcal{IM}$). To this end we declaratively model scene artefacts, spatial configuration, and events.

**Scene Artefacts.** These are elements of the scene that are indirectly obtained from the sensed objects. For instance, gaps between vehicles the driver is aware of, are declaratively defined using the following rule, stating that there is a gap between two entities if they are on the same lane and there is no other entity between these two entities.

```
gap(entity(ID1), entity(ID2)) :-
    entity(ID1), entity(ID2), ID1 != ID2,
    same_lane(entity(ID1), entity(ID2)),
    in_front_of(entity(ID1), entity(ID2)),
    not between(_, entity(ID1), entity(ID2)).
```

**Spatial Configuration.** The relational spatial structure holding between the scene elements the driver is aware of, is represented using the spatial relations defined in $\mathcal{R}$.

**Events.** We use the event calculus to detect driving events based on the driver's mental belief state of the scene. Towards this we define fluents representing dynamic scene properties, e.g, the fluent $on\_lane(entity(ID), lane(Lane))$ denotes that an entity is on a particular lane, or the fluent $curr\_task(Task)$ denotes the current task of the driver.

```
fluent(on_lane(entity(ID), lane(Lane))) :- entity(lane(Lane)), entity(ID).
fluent(curr_task(Task)) :- task(Task).
```

Additionally, we define events changing these scene properties. For instance the following definition of the event *takeover_manual* states that the event occurs if the fluent *automation* is *true* and the sensed state of the automation is *false*. Further, it states that the event initiates the fluent $curr\_task(change_lane)$, and terminates the fluent *automation* and $curr\_task(build\_sit\_aware)$.

```
event(takeover_manual).
initiates(takeover_manual, curr_task(change_lane), T) :- timepoint(T).
terminates(takeover_manual, automation, T) :- timepoint(T).
terminates(takeover_manual, curr_task(build_sit_aware), T) :- timepoint(T).
```

```
occurs_at((takeover_manual, T) :- holds_at(automation, T), ego_automation_state(false), timepoint(T).
```

**Generating the Driver's Scene Interpretation.** Solving this Answer Set Program with the scene elements the driver is aware of results in a model of the scene based on the drivers subjective perception, which constitutes the interpretation model ($\mathcal{IM}$).

In particular this model includes the following elements:

*Driving events the driver is aware of.*

```
occurs_at(audio_signal_start,119208). ... occurs_at(takeover_manual,135258).
```

*Fluents representing the driver's belief state about the properties of the scene.*

```
holds_at(curr_task(lane_change),136275). ...
holds_at(on_lane(entity(trf(1)),lane(1)),136275). holds_at(on_lane(entity(trf(2)),lane(1)),136275). ...
```

*Gaps between the vehicles the driver is aware of.*

```
gap(entity(trf(1)),entity(trf(2))). gap(entity(trf(2)),entity(trf(3))). ...
```

*The qualitative spatial configuration of the vehicles the driver is aware of.*

```
rel_longitudal_direction(ahead, entity(3)). rel_lane(left, entity(3)). rel_order(0, entity(3)). ...
```

This generated interpretation model ($\mathcal{IM}$) serves as input to the model comparison module to compare the drivers interpretation of the scene to the scene sensed by the SituSENSORS. Additionally these serve as abstractions needed for the projection step.

▶ **Projection.** Within SituWare the projection step is used to explore possible future states by generating the set of events, which are possible in the current situation, and which are consistent with the task of the driver. For instance in our use-case the driver has to perform a lane change after taking over control from the automation system. To model this we define the lane change event within our domain characterisation and use the event calculus to generate possible lane change events the driver could perform for each time point. This set of possible events constitute the projection model ($\mathcal{PM}$) and can be used to identify scene elements the driver needs to be aware of in order to safely perform the given task.

**Event Anticipation.** To anticipate scene events we define the events relevant for the driving task within the event calculus. For instance, the lane change event is defined based on the changes the event causes to the fluents and the constraints defining in which situations the event is possible.

```
event(change_lane(entity(ego(ID)), lane(Lane1), lane(Lane2), Location)) :-
    entity(ego(ID)), entity(lane(Lane1)), entity(lane(Lane2)),
    gap(entity(ID1), entity(ID2)).
```

```
initiates(change_lane(entity(ego(ID)), lane(Lane1), lane(Lane2), Location),
        on_lane(entity(ego(ID)), lane(Lane2)), T) :-
                entity(ego(ID)), entity(lane(Lane1)), entity(lane(Lane2)), timepoint(T).
terminates(change_lane(entity(ego(ID)), lane(Lane1), lane(Lane2), Location),
        on_lane(entity(ego(ID)), lane(Lane2)), T) :-
                entity(ego(ID)), entity(lane(Lane1)), entity(lane(Lane2)), timepoint(T).
```

```
poss(change_lane(entity(ego(ID)), lane(Curr_Lane), lane(Lane), Location)) :-
    holds_at(on_lane(entity(ego(ID)), lane(Curr_Lane)), T), curr_time(T),
    adjacent(lane(Lane), lane(Curr_Lane)), free(Location),
    on_lane(gap(entity(ID1), entity(ID2)), lane(Lane)).
```

In particular, a lane change is possible, when the lanes are adjacent to each other and there is a free location on the target lane. This free location may be either a gap between two vehicles, a free space behind or in front of a vehicle, or a completely empty lane. For the projection of the scene we generate all events that initiate the goal of a particular task and that are possible in the current situation.

```
anticipate(Event) :-
    initiates(Event, Goal, T), holds_at(curr_task(Task), T), goal(curr_task(Task), Goal), curr_time(T).

:- anticipate(Event), not poss(Event).
```

*For the example scene this results in two possible lane changes.*

```
anticipate(change_lane(ego(1),lane(2),lane(1),gap(entity(trf(1)),entity(trf(2))))).
anticipate(change_lane(ego(1),lane(2),lane(1),gap(entity(trf(2)),entity(trf(3))))).
```

## 4 Results and Future Directions

We presented an online and real-time capable approach for modelling drivers' situation awareness in the context of takeover situations as they happen in semi-autonomous driving. The approach is based on declarative characterisations of driving dynamics and is implemented as a ROS module within Python and Answer Set Programming (ASP) as a part of *SituSYS*, a modular system for measuring the drivers' situation awareness and guiding their attention based on the results. We highlighted the application of the presented approach as part of the *SituSYS* framework, which has been integrated in simulated as well as real-world driving. And we presented a case study concerned with situation awareness in take-over situations.

▶ Application Results. Application of the interpretation and projection model in the context of the case study has shown that the model is capable of maintaining a representation of the scene and provide the necessary information for comparing the estimated scene representation of the driver with the sensed information, and to provide individual guidance using the specialized attention guidance devices implemented in the *SituHMI*. The module works in real-time (approx. 30 Hz) with up to 20 vehicles plus the ego vehicle in the scene. However, to fully assess the effectiveness of the system it is required to perform a long term empirical study with the complete integrated system and measure the effect of *SituSYS* on the driving performance in take-over situations. To conduct such a study is not in the scope of this paper and subject of ongoing and future research.

▶ Future Directions. The presented *SituSYS* framework constitutes an ideal basis for applying logic based reasoning within simulated and real-world driving tasks. In this context future developments are aiming for more elaborate approaches for projecting scene dynamics, including the generation of possible and expected trajectories. Aside from this, an ongoing effort is the long term empirical assessment of the performance of *SituSYS* with respect to the cognitive adequacy of the generated mental models within simulated as well as real-world driving.

## Acknowledgments

We acknowledge partial funding by the *Federal Ministry for Economic Affairs and Climate Action* (BMWK) as part of the *SituWare* project (reference no. 19A19011C and 19A19011F).

## References


[1] Mark Colley, Lukas Gruler, Marcel Woide & Enrico Rukzio (2021): *Investigating the Design of Information Presentation in Take-Over Requests in Automated Vehicles*. In: *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction*, MobileHCI '21, Association for Computing Machinery, New York, NY, USA, doi:10.1145/3447526.3472025.

[2] Thomas Eiter & Rafael Kiesel (2020): *Weighted LARS for Quantitative Stream Reasoning*. In *ECAI 2020 - 24th European Conference on Artificial Intelligence, Santiago de Compostela, Spain, Frontiers in Artificial Intelligence and Applications* 325, IOS Press, doi:10.3233/FAIA200160.

[3] Mica R. Endsley (1995): *Toward a Theory of Situation Awareness in Dynamic Systems*. *Human Factors* 37(1), pp. 32–64, doi:10.1518/001872095779049543.

[4] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub & Philipp Wanko (2016): *Theory Solving Made Easy with Clingo 5*. In Technical Communications of the 32nd International Conference on Logic Programming (ICLP 2016), *OpenAccess Series in Informatics (OASIcs)* 52, SchlossDagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 2:1–2:15, doi:10.4230/OASIcs.ICLP.2016.2.

[5] Martin Gebser, Roland Kaminski, Benjamin Kaufmann & Torsten Schaub (2012): *Answer Set Solving in Practice*. Morgan & Claypool Publishers, doi:10.1007/978-3-031-01561-8.

[6] Martin Gebser, Roland Kaminski, Benjamin Kaufmann & Torsten Schaub (2014): *Clingo = ASP + Control: Preliminary Report*. *CoRR* abs/1405.3694, doi:10.48550/arXiv.1405.3694.

[7] Remo MA van der Heiden, Shamsi T Iqbal & Christian P Janssen (2017): *Priming drivers before handover in semi-autonomous cars*. In: *Proceedings of the 2017 CHI conference on human factors in computing systems*, ACM, New York, NY, USA, pp. 392–404, doi:10.1145/3025453.3025507.

[8] Lei Hsiung, Yung-Ju Chang, Wei-Ko Li, Tsung-Yi Ho & Shan-Hung Wu (2022): *A Lab-Based Investigation of Reaction Time and Reading Performance Using Different In-Vehicle Reading Interfaces during Self-Driving*. In: *Proceedings of the 14th* , AutomotiveUI '22, ACM, New York, NY, USA, p. 96 – 107, doi:10.1145/3543174.3545254.

[9] Suraj Kothawade, Vinaya Khandelwal, Kinjal Basu, Huaduo Wang & Gopal Gupta (2021): *AUTO-DISCERN: Autonomous Driving Using Common Sense Reasoning*. In: *Proceedings of the International Conference on Logic Programming 2021 Workshops, Porto, Portugal (virtual)* , *CEUR Workshop Proceedings* 2970, CEUR-WS.org, doi:10.48550/arXiv.2110.13606.

[10] Robert Kowalski & Marek Sergot (1989): *A Logic-Based Calculus of Events*, pp. 23–51. Springer-Verlag, Berlin, Heidelberg, doi:10.1007/978-3-642-83397-7_2.

[11] Josef F. Krems & Martin R. K. Baumann (2009): *Driving and Situation Awareness: A Cognitive Model of Memory-Update Processes*. In Masaaki Kurosu, editor: *Human Centered Design*, Springer Berlin Heidelberg, pp. 986–994, doi:10.1007/978-3-642-02806-9_113.

[12] M. Kyriakidis, J. C. F. de Winter, N. Stanton, T. Bellet, B. van Arem, K. Brookhuis, M. H. Martens, K. Bengler, J. Andersson, N. Merat, N. Reed, M. Flament, M. Hagenzieker & R. Happee (2019): *A human factors perspective on automated driving*. *Theoretical Issues in Ergonomics Science* 20(3), pp. 223–249, doi:10.1080/1463922X.2017.1293187.

[13] Danh Le-Phuoc, Thomas Eiter & Anh Le-Tuan (2021): *A Scalable Reasoning and Learning Approach for Neural-Symbolic Stream Fusion*. *Proceedings of the AAAI Conference on Artificial Intelligence* 35(6), pp. 4996–5005, doi:10.1609/aaai.v35i6.16633.

[14] Jiefei Ma, Rob Miller, Leora Morgenstern & Theodore Patkos (2014): *An Epistemic Event Calculus for ASP-based Reasoning About Knowledge of the Past, Present and Future*. In: *LPAR: 19th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, EPiC Series in Computing* 26, EasyChair, pp. 75–87, doi:10.29007/zswj.

[15] Rob Miller, Leora Morgenstern & Theodore Patkos (2013): *Reasoning About Knowledge and Action in an Epistemic Event Calculus*. In: *COMMONSENSE 2013.*

[16] Phillip L. Morgan, Chris Alford, Craig Williams, Graham Parkhurst & Tony Pipe (2018): *Manual Takeover and Handover of a Simulated Fully Autonomous Vehicle Within Urban and Extra-Urban Settings*. In *Advances in Human Aspects of Transportation*, Springer International Publishing, Cham, pp. 760–771, doi:10.1007/978-3-319-60441-1_73.

[17] Jan-Patrick Osterloh, Jochem W. Rieger & Andreas Lüdtke (2017): *Modelling Workload of a Virtual Driver*. In: *Proceedings of the 15th International Conference on Cognitive Modeling.*

[18] Amir Rasouli & John K. Tsotsos (2020): *Autonomous Vehicles That Interact With Pedestrians: A Survey of Theory and Practice*. *IEEE Transactions on Intelligent Transportation Systems* 21(3), pp. 900–918, doi:10.1109/TITS.2019.2901817.

[19] Umair Rehman, Shi Cao & Carolyn MacGregor (2019): *Using an Integrated Cognitive Architecture to Model the Effect of Environmental Complexity on Drivers' Situation Awareness*. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 63(1), pp. 812–816, doi:10.1177/1071181319631313.

[20] Christina Rödel, Susanne Stadler, Alexander Meschtscherjakov & Manfred Tscheligi (2014): *Towards Autonomous Cars: The Effect of Autonomy Levels on Acceptance and User Experience*. In: *Proceedings of the 6th* , AutomotiveUI '14, ACM, New York, NY, USA, p. 1 – 8, doi:10.1145/2667317.2667330.

[21] Jakob Suchan, Mehul Bhatt & Srikrishna Varadarajan (2019): *Out of Sight But Not Out of Mind: An Answer Set Programming Based Online Abduction Framework for Visual Sensemaking in Autonomous Driving*. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China*, ijcai.org, pp. 1879–1885, doi:10.24963/ijcai.2019/260.

[22] Jakob Suchan, Mehul Bhatt & Srikrishna Varadarajan (2021): *Commonsense visual sensemaking for autonomous driving - On generalised neurosymbolic online abduction integrating vision and semantics*. *Artificial Intelligence* 299, p. 103522, doi:10.1016/j.artint.2021.103522.

[23] Jakob Suchan, Mehul Bhatt, Przemyslaw Andrzej Walega & Carl Schultz (2018): *Visual Explanation by High-Level Abduction: On Answer-Set Programming Driven Reasoning About Moving Objects*. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA*, AAAI Press, doi:10.1609/aaai.v32i1.11569.

[24] Efthimis Tsilionis, Nikolaos Koutroumanis, Panagiotis Nikitopoulos, Christos Doulkeridis & Alexander Artikis (2019): *Online Event Recognition from Moving Vehicles: Application Paper*. *Theory and Practice of Logic Programming* 19(5-6), doi:10.1017/S147106841900022X.

[25] ASAM e. V. (2021): *OpenDRIVE Format Specification*. Available at `https://www.asam.net/standards/detail/opendrive/`.

[26] Raphael Zimmermann & Reto Wettach (2017): *First step into visceral interaction with autonomous vehicles*. In: *Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pp. 58–64, doi:10.1145/3122986.3122988.