Using Traffic Sequence Charts for Knowledge Formalization and AI-Application

Philipp Borchers[1], Willem Hagemann[2], Dominik Grundt[1], Tino Werner[1], and Julian Müller[1]

[1] German Aerospace Center (DLR)
Institute of Systems Engineering for Future Mobility
Escherweg 2, 26129 Oldenburg, Germany
philipp.borchers@dlr.de
[2] Fraunhofer Research Institution for Energy Infrastructures and Geothermal Systems IEG
Gulbener Straße 23, 03046 Cottbus

Abstract. In recent years, the integration of knowledge into AI training processes has been shown as a promising approach to improve AI performance, training costs and resource efficiency. Here, the formalization of knowledge is a key challenge. In this article, we discuss the capabilities of the visual yet formal specification language called Traffic Sequence Charts (TSC) on formalizing multimodal knowledge, in particular procedural knowledge about traffic maneuvers. Finally, we present an approach using the formalized knowledge to train reinforcement learning (RL) agents, aiming to transform the descriptive knowledge on traffic maneuvers in TSCs into performative knowledge in AI traffic agents. To this end, we were able to train an agent to control a vehicle through a pass-by maneuver and apply it successfully to an untrained overtaking maneuver.

Keywords: Knowledge Formalization, Reinforcement Learning, Traffic Scenarios, Abstract Scenario Specification

This research was partly funded by the German Federal Ministry of Economic Affairs and Climate Action (BMWK) through the "KI Wissen" project under grant agreement No. 19A20020M.

1 Introduction

The training of classical AI approaches is usually based on a huge amount of data, which is inevitably associated with high effort and costs. In contrast, hu-

mans do not learn through an immense amount of data, but much more by generalizations of experiences. The knowledge from experiences, observations and additional theoretical considerations can be distributed to others, such that they are able to decide and act on problems adaptively without making the experiences individually by themselves. In the context of autonomous driving, stakeholders like lawyers, safety experts, psychologists and physicists offer additional relevant knowledge about the driving environment and the expected behaviour of traffic participants. For instance, the laws of physics tell us that it is not very likely, that a car will teleport to an adjacent lane. When it comes to AI training, such knowledge should be integrated into the training process. Then, the AI does not need to learn individually, making the training more efficient and less costly. Furthermore, it increases the safety in the complex environment in which an AI-based driving function needs to make decisions.

In order to make use of such knowledge, researchers develop methods and concepts to infuse knowledge into AI training processes [17, 26]. A fundamental challenge here is to formalize knowledge so that it can be integrated into training process. In many cases, knowledge such as that of traffic experts is only available in fuzzy natural language. Such knowledge must be machine-readable for integration. Here, formal languages may help [26]. However, in case of complex multi-stakeholder knowledge, as it is the case in the transportation domain, it is important that the formalized knowledge is consistent. Otherwise the integrated knowledge may lead to undesirable and, in worst case, safety-critical AI behaviour. Here, a specification in one single and graphical formal language is beneficial. First, an extensive crosscheck of knowledge takes place at an early stage if different knowledge is formalized in a single language. Second, a formal specification is accessible to a wider circle of stakeholders due to the graphical interface. To use the formalized knowledge for AI training, it must be possible to derive necessary AI inputs like training environment and observation space. Otherwise the learning agent could not sense the environment. Furthermore reward or loss functions must be derivable to guide and check knowledge compliant behaviour.

In this paper, we discuss the abstract scenario description language "Traffic Sequence Charts" (TSCs) [5] as a knowledge formalization tool. This language, originally developed for abstract scenario descriptions, allows a visual specification of knowledge from several stakeholders as spatio-temporal knowledge building block. We will introduce the TSC language in Section 2.

To address the formalization, we continue with a classification of traffic-related knowledge in Section 3. This includes knowledge about time and space, ontological knowledge such as the characterization of lanes and traffic participants, their physical dynamics and properties, and also procedural knowledge for traffic scenes like an overtaking maneuver (later called *script knowledge*). Then we discuss possible formalization approaches with the TSC language in Section 3. The collection of knowledge that is formalized with the TSC language is called a *Knowledge Building Block* (KBB). The term *Building Block* indicates that it is intended to be used for the training processes of different AI-based

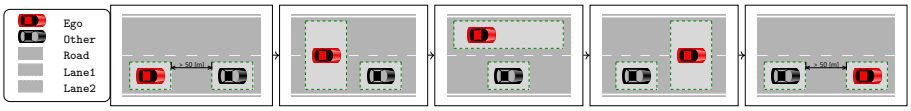


Fig. 1: TSC depicting an overtaking maneuver as a sequence $\boxed{A} \rightarrow \boxed{B} \rightarrow \boxed{C} \rightarrow \boxed{D} \rightarrow \boxed{E}$ of invariant nodes

driving functions. For instance, an AI-based controller to overtake can profit from a TSC knowledge building block with knowledge about involved traffic participants, their (physical) properties and the phases of an overtaking process.

All this knowledge in a KBB is declarative and not performative. For instance, the TSC in Figure 1 explains *what* an overtaking maneuver is, as it formalizes a phenomenon that we call an overtaking maneuver, but it does not explain *how* this maneuver is to be carried out by the actors. By analogy, a novice driver usually knows *what* an overtaking maneuver is long before he or she knows *how* to actually perform such a maneuver. To transform the knowledge *what* to do into a knowledge *how* to do, we will present the TSC2RL approach in Section 4. Here, we describe along the knowledge classes how an environment for reinforcement learning could be deduced in order to learn a reinforcement agent in controlling through an overtaking maneuver.

We will introduce the TSC language in Section 2, continue in Section 3 with elaborating on knowledge types and their formalization in TSCs, followed by our application within Reinforcement Learning in Section 4. The challenges and limitations of existing approaches will be discussed in corresponding subsections.

2 The Traffic Sequence Charts Language

Traffic Sequence Charts (TSCs) are a visual description language for abstract traffic scenarios with formal semantics. The basic building blocks of TSCs are *invariant nodes* (cf. Table 1). Each invariant node $\boxed{A}$ consists of a *spatial view* A, that visually depicts a traffic situation and a frame $\square$ stipulating that the spatial constraints of A must hold during a non-vanishing but typically short duration. A traffic sequence chart is a combination of invariant nodes by logical-temporal operators as found in Table 1. Hence, TSCs decouple spatial and temporal-logical aspects of traffic scenarios into spatial views and logical-temporal operations. Note that our usage of the adjective *spatial* refers to a general notion of space and is not restricted to geometric quantities such as relative positions or distances. It also might include other quantities that can be evaluated for a single point in time like the instantaneous velocity. The translation of spatial views and their detailed semantics are out of the scope of this article and can be found in [7, 8].

2.1 An Exemplary TSC

We consider an exemplary specification of an overtaking maneuver as shown in Figure 1. The principal reading direction is from left to right. On the left most

we find the *bulletin board* that introduces the involved entities of the TSCs, which are a road, two lanes, and two vehicles called *Ego* and *Other*. Right of the bulletin board we find a *sequence* of five *invariant nodes* which we name $\boxed{A}$, $\boxed{B}$, $\boxed{C}$, $\boxed{D}$, and $\boxed{E}$ according to the reading direction. Each invariant node depicts spatial constraints that have to hold for a nonempty interval of time, i.e., the invariant node $\boxed{A}$ consists of two parts, the *spatial view* A that depicts purely spatial constraints, and its surrounding time frame $\square$ representing a nonempty interval of time. Therefore, in the following description, we use verbs like 'stay' or 'remain' rather than 'is' to indicate that the depicted state is maintained for a period of time.

The invariant node $\boxed{A}$ depicts a traffic situation where *Ego* stays behind *Other*. Each vehicle remains somewhere in the right lane, depicted by the surrounding *somewhere-boxes* having dashed green border. The distance between both vehicles, or more precisely, the distance of the somewhere-boxes, remains at least 50 meters, depicted by the *distance arrow* and its annotation. Node $\boxed{B}$ depicts a traffic situation where *Ego* stays somewhere on the left or right lane, but still behind *Other*. In invariant node $\boxed{C}$, *Ego* remains on the left lane, somewhere behind, next, or in front of *Other*, which itself remains somewhere on the right lane. In node $\boxed{D}$, *Ego* stays now in front of *Other* and somewhere on the left or right lane. Finally, node $\boxed{E}$ depicts a traffic situation where *Ego* stays somewhere on the right lane in front of *Other* and the distance to *Other* remains at least 50 meters.

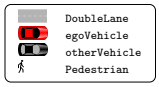
Overall, the TSC states a sequence of five temporally seamless situations which may be assumed to characterize almost any reasonable (and even some unreasonable) overtaking maneuvers of two vehicles. Hence, TSCs provide us an intuitive way to formalize abstractions of concrete traffic scenarios.

2.2 Satisfaction of Basic Charts and Premise-Consequence Charts

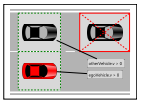
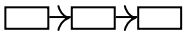
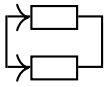
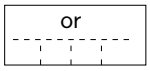
The set of all *basic charts* is defined as the least set that contains all invariant nodes and is closed under the operation *sequence*, *concurrency*, *choice*, and *negation*. The *satisfaction* of basic charts is defined for a time bounded trajectory of the state space and recursively defined over the chart operators: Let π be a time-bounded trajectory over the time interval $[b, e)$. The satisfaction of an invariant node $\boxed{A}$ is reduced to the satisfaction of the spatial view for all $t \in [b, e)$. The *true* invariant node $\boxed{\top}$ is somewhat special as it is satisfied by any trajectory. Concurrency of two charts $C \wedge D$ is reduced to the simultaneous satisfaction of both charts and the choice of two charts $C \vee D$ is satisfied if and only if at least one chart is satisfied. A negated chart $\neg C$ is satisfied, if and only if the chart C is not satisfied. Finally, a sequence $C \rightarrow D$ of two charts is satisfied if and only if there exists some $m \in (b, e)$ such that the first chart C is satisfied over the time-interval $[b, m)$ and the second chart D is satisfied over the time-interval $[m, e)$. [3] We call m a switching point of $C \rightarrow D$.

[3] The presented semantic account follows a previously unpublished simplification of [7, 8] proposed by Jan Becker.

Common Elements

Feature	*Description*
	**World Model:** An explicit formalization of a conceptualization of the relevant concepts as well as their relations, properties, and dynamics for a given domain.
	**Bulletin Board:** Introduces the involved symbols in the given TSC scenario. Each symbol is interpreted as an instance of a world model class with more attributes like position, velocity, size or heading.

Basic Charts

Feature	*Description*
	**Invariant node:** An invariant node defines spatial constraints that have to hold for the duration of the invariant node. Invariant nodes are the atoms of basic charts. In this article, we will also use the notation $\boxed{A}$, where A is a predicate that is defined by the spatial constraints.
	**True Chart:** Special spatial view constraint without any specification. It is true in any case and everything is possible.
	**Sequence:** Operation on basic charts that specifies that spatial views have to take place consecutively. In this article, sequences are noted as $C \rightarrow D$, where C and D are arbitrary basic charts.
	**Choice:** Operation on basic charts that specifies that at least one of its subcharts has to take place. In this article, choices are also denoted as $C \vee D$, where C and D are arbitrary basic charts.
	**Concurrency:** Operation on basic charts that specifies that both subcharts have to take place in parallel. In this article, concurrencies are also denoted as $C \wedge D$, where C and D are arbitrary basic charts.
	**Negation:** Operation to exclude spatial views or basic charts. In this article, negations are also denoted as $\neg C$, where C is an arbitrary basic chart.
	**Logical Operation:** Logical operations like "OR", "AND" and "NOT" to combine spatial view constraints.

Premise-Consequence Charts

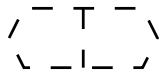
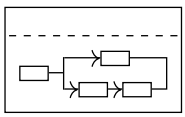
Feature	*Description*
	**Premise:** Split into two parts: The left part of the premise is called **history**, the right part is called **future**. The premise specifies the "if" condition for the **consequence** in the TSC: If the history has happened in the past, and the future will take place, then the consequence has to be true concurrently to the future.
	**Consequence:** Specifies what has to be happen, if the conditions in the premise are true. Typically, the consequence is a concatenation of invariant traffic situations that hold for a while.

Table 1: Summary of basic TSC features

Premise-consequence charts are built on top of basic charts. They are intended for the specification of requirements. A premise–consequence chart consists of three basic charts H, F and C, called *history*, *future*, and *consequence*. Its meaning is a follows: whenever the *premise* $H \rightarrow F$ holds, then also the consequence C has to hold simultaneously with F, i.e., a premise-consequence chart is satisfied by a time-bounded trajectory π over $[b, e)$ if and only if either $H \rightarrow F$ is not satisfied, or $H \rightarrow F$ is satisfied by π over $[b, e)$ and also C is satisfied by π over $[m, e)$ for all switching points m of $H \rightarrow F$.

The satisfaction of a TSC with respect to a given time-bounded trajectory π is the central concept on which TSC-monitors are based. A TSC-monitor is a mechanism that checks, either during runtime or after runtime, whether the recorded trajectory data satisfy certain TSCs. A research preview that describes the construction of runtime monitors from TSCs is presented in [15].

2.3 Satisfiability, Validity and World Models

Building on the notion of satisfaction, we may now introduce the dual notions of *satisfiability* and *validity* of a TSC. The idea is to call a TSC satisfiable exactly when there is a trajectory that satisfies the TSC. A TSC is called valid exactly when it is satisfied by any trajectory. Equivalently, but stressing the duality of satisfiability and validity, a TSC is valid if and only if its negation is unsatisfiable. Examples for valid basic charts are $\boxed{\top}$ or $\boxed{A} \vee \neg\boxed{A}$, as these charts are satisfied by any trajectory. However, often we want to restrict the class of admissible trajectories. For example, instead of referring to arbitrary trajectories, one might want to talk specifically about trajectories that can be generated by real existing vehicles. To do this, we enrich TSCs with *world models*.

A world model includes, but is not limited to, a specification of all admissible trajectories. There is no fixed formalism in which this specification must be made, it is only important that it uniquely defines the set of admissible trajectories. For example, the world model could be restricted to continuous trajectories, or further restricted to continuous trajectories that satisfy certain differential (in-)equations. In the introductory example, we implicitly made a continuity assumption on the trajectories. Nontrivial examples of world models can be found in [7, 8] where the admissible trajectories are generated by the parallel composition of hybrid automata, or in [5] where the set of admissible trajectories is restricted to Bézier curves. Note, that the satisfiability and validity of TSCs strongly depend on the given world model. In general, the more the set of admissible trajectories is restricted, the fewer TSCs are satisfiable and, dually, the more TSCs are valid. Another mean to restrict the set of admissible trajectories are premise-consequence charts, since they distinguish those trajectories from the world model that satisfy a desired (or required) behaviour. We now relate the conceptual pair of validity and satisfiability to premise-consequence charts in particular: Since the intended use of a premise-consequence chart C is to specify a requirement, C should be satisfiable over the world model, otherwise the specified requirement cannot be realized. However, C should not be

valid with respect to the world model, as in this case the requirement is trivially satisfied for any trajectory.

In addition, the world model associates entities that potentially populate the scenario with their individual trajectories. The world model specifies object types and their properties and attributes. Typically, the world model contains types for traffic participants, such as cars or pedestrians and environmental infrastructure such as roads, lanes, or even traffic lights. While the movement of a vehicle over time usually describes a continuous trajectory, this is not the case for a traffic light signal. Here, we would expect continuous phases of the same signal, interrupted by discrete state changes. Accordingly, the world model provides information on how the possible overall trajectories of the scenario are composed of individual, entity-specific trajectories. In case of the overtaking maneuver, the trajecories are the evolution of the continuous variables *position*, *velocity*, *acceleration* and *heading* of the two vehicles *Ego* and *Other*. Also the bulletin board refers to the world model, as it defines concrete instances of world model classes that actually act in the depicted scenario.

In Section 3, we discuss various knowledge types that are relevant for traffic situation and demonstrate, how the TSC language can be used for knowledge formalization. Additionally, the TSC elements from Table 1 will be described further on.

2.4 Related Work on Specification Languages for Transportation Domain

In recent years, several specification languages for the transportation domain have emerged. One reason is the progress that has been made in artificial intelligence and the associated possibilities for increasing automation of driving functions. These developments lead to the need of modeling the transportation domain as accurately and completely as possible in order to increase the safety and capabilities of automated driving functions. In particular, visual specification languages, which allow for an intuitive specification and linking of ontological aspects with physical and temporal properties, are of particular interest for the research field of knowledge specification in transportation. The multi-lane spatial logic (MLSL) and its extensions [28] deal with reasoning about authority of movement. Here, specifications for controllers of cooperative systems can be specified visually. Since the problem of reasoning about authority of movement mainly occurs at intersections, the focus is so far on urban road intersections, whereby extensions are already planned. With the M-SDL language of foretellix [14], abstract traffic scenarios can be specified, which can consequently be used to create a large variety of concrete simulation runs. Here, the focus is on designing test cases. The visual elements have no semantics, which complicates an intuitive and efficient specification of different transportation domain properties.

3 Formalizing Traffic Related Knowledge

In the following subsections, we list different types of declarative knowledge that we consider relevant in road traffic. These types emerged by an analysis of the artifacts described in the previous section, based on the formalization possibilities of the different types in the TSC language. We will consider the extent to which the TSC formalism (see Section 2) provides us with the means to capture the type of knowledge. In the literature, the categorization of knowledge does not take place at a single level of abstraction, but a suitable level is often developed, and consequently knowledge is clustered. In [26], the focus is on the integration of knowledge and the knowledge categories *world knowledge*, *expert knowledge* and *scientific knowledge* are introduced, which are suitable for the focus of knowledge representation and integration for AI driving functions. In this definition of knowledge categories, we find intersections of different modalities, which is a suitable level of abstraction for the focus of the work in [26]. Our approach for knowledge categorization is derived from by the Epistemic Deontic Axiological (EDA) model [12], where ontological aspects are also considered. The EDA model has already been applied to agent systems in medicine [29].

To identify more generally the relevant knowledge in the transportation domain, we will categorize knowledge along modalities. A modality is a linguistic device that relates the possibility of a statement to the actual reality or truth. Examples are modal verbs, such as "must", "can", "should", and "may", adverbs, such as "necessarily" and "possibly", or main verbs as "to know" or "to believe". However, there are various other linguistic devices to express modalities. In linguistics, often the classical distinction between *root* and *epistemic modalities* is made, see [19] and further references therein. Root modalities refer to the circumstances in the world, rather than what is perceived by the speaker or a community to which the speaker belongs. Hence, we use root modalities to characterize knowledge about ontologies, (physical) abilities, obligations, and situations, where any knowledge about the individual perception, goals or exchange of information via communication is characterized by epistemic modalities.

It should be noted that the list of different knowledge types is influenced by our research on TSCs and their expressiveness and does not claim to be exhaustive.

3.1 Fundamental Knowledge about Space and Time

Fundamental Knowledge about space and time includes any form of knowledge necessary for interpreting spatial views, invariant nodes and sequences thereof. This type of knowledge describes the meaning of relative positions, such as in front of or within an area. It contains fundamental knowledge about temporal relations like before, after and concurrently. In addition, it helps us to relate and compare quantities such as distances or speeds. In other words, this type of knowledge is *not the information that is depicted* in a TSC, but the *knowledge that is necessary to understand* a TSC. This knowledge is implicitly built into

the formalism of TSCs and is a prerequisite for the correct evaluation of spatial relationships. In Section 4, this knowledge is used to derive reward functions.

3.2 Ontological Knowledge

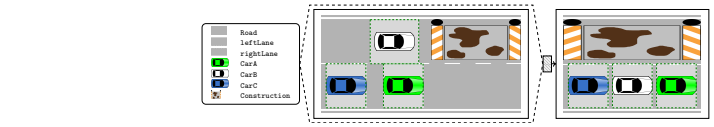
An ontology provides knowledge about entities that are potentially or actually involved in traffic-related situations. For example, in an overtaking maneuver such as the one presented in Section 2, we know that there are vehicles, lanes and roads, all of them are objects with a position as well as lateral and spatial extent. Furthermore, we also know that the road is a static object that cannot move, the vehicles instead are dynamic objects with a continuous position, velocity, acceleration and heading. Besides this attributive knowledge there is knowledge about the relation between the traffic entities. Lanes are placed beside other lanes on a road and vehicles drive on lanes and roads. Typically, ontological knowledge is expressed by the verb "to be" and describes the relation of entity classes, as in "A lane is part of a road."

Ontological knowledge about entity classes belongs to the world model of TSCs. Individual objects are named and instantiated in the bulletin board. We consider description logic to be a suitable formalization language for specifying ontological knowledge in TSCs. The distinction between entity classes and concrete objects is closely related to description logic [2] and its distinction between the TBox (terminological box), which contains sentences defining concepts and roles, and the ABox (assertional box), which contains sentences assigning individuals to these concepts and roles.

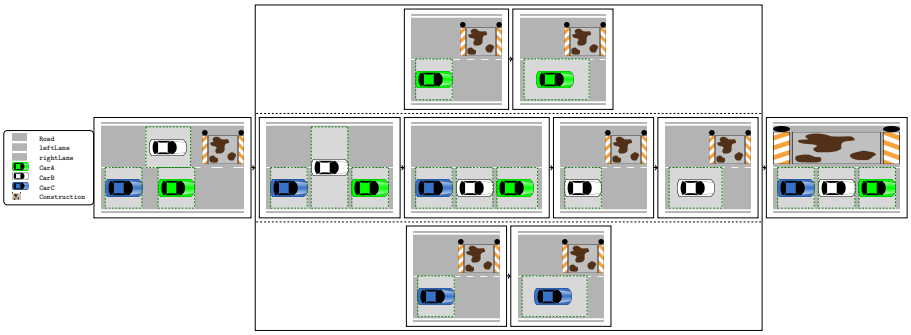
3.3 Knowledge about Physical Possibilities and Necessities

This type of knowledge covers the possible dynamics of the traffic participants like cars, bikes and other objects moving in a traffic scenario. For cars one can have complex models with differential equations or simple models like the linear bicycle model of vehicles. The knowledge about the physical-dynamic properties describes the (in-)capabilities of an entity. Colloquially, they can often be recognized by the modal verb *can* or the adverbs *possible* and *necessary*, such as in "cars cannot fly". Physical necessities are often stated as laws. Newton's second law $F = ma$ thus expresses a necessary connection between force, mass and acceleration. At the same time, this limits the space of possible outcomes of these quantities.

Knowledge about the physical evolution belongs to the world model of TSCs. This knowledge defines the possible temporal evolution of the objects involved. Suitable formalisms are differential equations or hybrid automata [8]. A trajectory of the overall system then results as one possible composition of the individual trajectories and in this respect depends on the concrete object instantiation, see the explanations on ontology knowledge above. The simulation approach in [5] is currently limited to the generation of Bézier curves.



a Zipper rule as a requirement



b A detailed scenario that meets the requirement

Fig. 2: Zipper rule

3.4 Script Knowledge

Script knowledge describes the sequence of typical traffic maneuvers. Basic charts are exactly the means of choice to formalize this form of knowledge. For example, in Figure 1 we have already seen how a sequence of invariant nodes can be used to formalize a typical overtaking maneuver. In script knowledge, we conceptualize our ideas about typical traffic sequences. The overtaking TSC of Figure 1 hence defines what an overtaking maneuver *is*. In this way, script knowledge is related to ontological knowledge, except that script knowledge goes beyond static entity relations as it describes the temporal changes of the (spatial) relations of entities.

3.5 Deontic Knowledge

This form of knowledge builds on top of script knowledge. Deontic knowledge includes all obligations and permissions that should apply in a traffic scenario. This includes, in particular, the knowledge about traffic rules. While script knowledge allows us to explain in general terms what an overtaking maneuver is, deontic knowledge imposes certain requirements on such a maneuver. For example, an overtaking maneuver *may* only be carried out if it is not *prohibited* by a no overtaking sign.

Deontic knowledge can be captured in the TSCs through premise-consequence charts, as premise-consequence charts state requirements on the described scenario that have to be met.

As an example we consider the zipper rule. The TSC in Figure 2a formalizes the knowledge about the zipper rule as a general requirement. Whenever there are three cars A, B, and C that approach a construction area blocking the left

lane as depicted in the premise of the TSC, then it is required that the cars arrange themselves after a while as depicted in the consequence. Hence, this TSCs allows freedom on how the cars perform the rearrangement, especially how Car B is performing the lane change as long as all of them are queued correctly in the end.

In contrast, Figure 2b depicts exemplary script knowledge that satisfies the requirement above. The TSC consists of a sequence of several invariant nodes that starts the premise of Figure 2a and ends with the very last invariant node of the consequence. In between there is a concurrency node with three subsequences as elements. The upper and lower sequence are sequences formalize that Car A and Car C pass the left border of the construction area. The middle sequence describes the maneuver of Car B in more detail: First the Car B performs a lane change between the other vehicles, then it is on the right lane together with the other ones in a queue. After that, there is still a phase, where Car B is in front of the construction area, hence the lane change maneuver has to be finished before the construction site. Finally there is a phase, where Car B will pass the border of the construction area.

3.6 Epistemic knowledge

As a last type, we consider epistemic knowledge. This type includes any knowledge that traffic participants have access to, whether by individual prior knowledge, perception, or communication. This form includes the knowledge about relative distance to other traffic participants via sensors measurements, or prior knowledge like knowing that an active left blinking lights indicates a lane change.

In general, TSCs offer little to no systematic way to formalize the epistemic knowledge of agents. In [8], however, the combination of TSCs with life sequence charts (LSCs) has been proposed to formalize the communication behaviour of individual traffic agents. On the one hand, it is conceivable that knowledge about the perceptual capabilities of traffic agents could be encoded in the world model in the form of physical knowledge about occlusion properties or sensor models. For modeling of individual (possibly false) beliefs about the traffic situation, TSCs do not seem to offer any support now, since the viewpoints of individual agents are missing and only the global viewpoint can be specified.

Epistemic knowledge can be found, for example, in the German StVO, §5.2 “They may overtake only if they can see that, during the entire overtaking manoeuvre, they will in no way impede oncoming traffic. [...].” [10] One possible interpretation is that the overtaking person must not have any epistemic uncertainties with regard to oncoming traffic. However, this cannot be expressed with the existing TSC language, because of the following reasons: First, there is no separate distinction of the acting subject, such as the overtaking person in this case. Possibly, one could use the “ego” designator, which has, on a semantic level, no special role yet. Second, the epistemic state of the agent cannot be expressed. With reference to the possible-world semantics of epistemic modal logics, a conceivable formalization would be a visual representation of all future developments of the scenario that the agent considers possible. An agent only

has epistemic certainties about those relations that apply in any of these scenarios. Thus, as long as the agent still considers a future development possible in which there is a danger to the oncoming traffic, he must not overtake.

4 TSC2RL Application

In Section 3 we introduced different types to cover and classify many aspects of traffic relevant knowledge. Considering the TSC as Knowledge Building Block to be applied in a traffic situation (e.g. a vehicle intends to overtake another vehicle as in Figure 1 and needs knowledge for execution), via the TSCs it is known *what* (Ontology) kind of objects *should* or *must* (Script, Deontic) do under *which* (Physics) physical possibilities. Hence, we consider TSCs to be particularly suitable to formalize declarative knowledge in traffic scenarios. In addition, the original intention of TSC leaves open *how* the scenario is realized and which trajectories are to be driven by the actors. Thus, the leading question of this section is whether descriptive knowledge stored in TSCs can be used such that an agent learns *how* to perform maneuvers that are encoded in TSCs.

4.1 Related Work

One approach to find TSC realizations is the transformation into a satisfiability problem as in [5]. The idea is to translate the TSC into a formula over the theory of real linear arithmetic, whose satisfying solutions correspond to admissible trajectories, and generate them by using an SMT-solver[4]. While this approach works quite well to find first scenarios, the trajectories are restricted to Bézier splines with a fixed amount of control points leading to less freedom for the exploration of different scenarios.

Apart from our TSCs, another opportunity to describe scenarios is signal temporal logic (see [23]). It is a natural way to quantify the violation or satisfaction of temporal logic predicates. [11] proposed a robustness metric, initially for metric temporal logic.

The integration of constraints into AI models has become popular in the last decade. Such constraints not only correspond to allowed ranges for individual variables but can represent physical laws, domain-specific knowledge or other types of prior knowledge. The idea is to let an AI model not learn already existing knowledge but to directly integrate this knowledge into the training process in order to facilitate this process. We refer to surveys considering the integration of such knowledge such as [6], [30], or [25], for a broad overview of knowledge integration into AI with applications in autonomous driving, see [31]. In particular, for knowledge integration into AI, a standard way is reward shaping. Given the strengths of TSCs as formalizing language, one could ask

[4] Satisfiability modulo theories (SMT) is the problem of determining whether a Boolean combination of theory-specific relations, such as inequality constraints in linear real arithmetic, is satisfiable.

whether we can apply the techniques of RL training using STL formulae for our purposes.

Training RL agents using STL has been first proposed by [1]. Here, they either aim at maximizing the probability of satisfying the specified requirements or at maximizing the expected robustness. However, the STL requirements correspond to trajectories of a specified size, so for all shorter trajectories, no reward can be defined, implying at least a delayed reward. [1] re-define the objective on a τ-MDP, which incorporates τ-horizon paths instead of states, allowing for defining rewards for each element of this space of such paths. The clear disadvantage is that the complexity exponentially increases with the horizon length τ. [27] point out that for the approach from [1], that tractability in continuous spaces is not given. [4] propose to define the STL robustness over partial traces so that the reward functions are shaped locally, in contrast to the global approaches from [1] and [20]. [22] point out that the STL robustness is often approximated model-free, without including system dynamics, making it inaccurate. They propose a model-based variant, where a Gaussian process regression model is trained. [13] consider the case of inexact measurements and propose a robust variant of STL.

According to [9], the STL robustness values are often computed offline when the whole trajectory is available and propose an online algorithm. The advantage of STL robustness metrics is, for example, that they can act locally. In the cart pole example of [4], there is a strict angle in which the pole must be at the end. Global reward functions cannot represent this, and using this strict angle globally would lead to too many negative rewards, preventing from effective learning. The STL formula in contrast can represent the requirement that this strict angle only has to be satisfied eventually, which however is also possible with TSCs.

Nevertheless, the requirements described by TSCs can be interpreted as relaxations of STL requirements in the sense that TSCs do not necessarily fix the time intervals in which a certain property must hold. In STL however, properties that have to hold globally or eventually are accompanied with a fixed time interval which would have to be fixed in advance. Therefore, such STL predicates are unsuitable for TSCs: Since although each TSC invariant node may be accompanied with a time interval as well in the sense that the agent has a limited time that it can spend in a particular node, there is no further restriction when to move to the next TSC node, in which other requirements must hold. In other words, the concrete time interval in which a certain family of properties must hold cannot be fixed in advance but depends on the agent. The predicate that a property must be satisfied eventually in a given time interval could be incorporated by extracting the maximum time the agent has in order to reach the final TSC node, however, this requirement to somewhen reach the goal would be overloaded by all the other requirements. Nevertheless, due to successful applications of RL in autonomous driving and the data efficiency of RL training that allows to train an AI model without data, whose preparation is notoriously expensive, we want to close the gap and allow for TSC formulae to enter RL through suitable reward shaping. Therefore, we intend to use node-

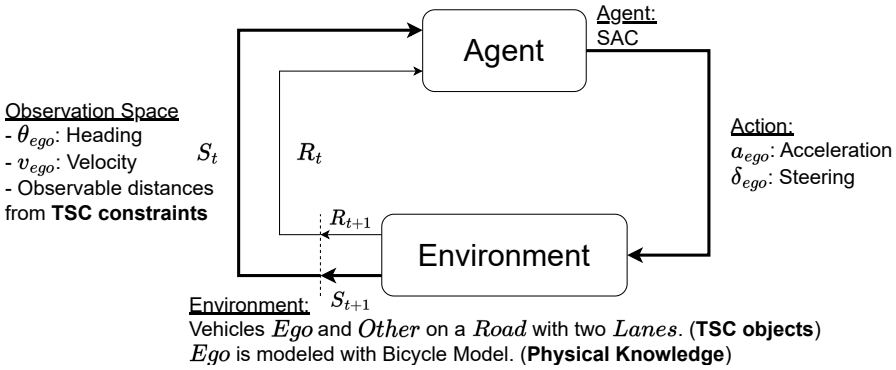


Fig. 3: TSC2RL environment

specific reward terms in order to encourage the agent to satisfy the node-specific requirements, accompanied with a reward term that pushes the agent forward through the whole TSC. With this technique, it should potentially be possible to go beyond existing work in the sense that an autonomous agent can be trained even w.r.t. the rather coarse specifications of a TSC, which causes established approaches such as STL to be difficult to apply.

4.2 Training Environment

Basically a reinforcement learning agent learns to execute optimized actions by observing the resulting state and reward from the environment where it acts in. In Figure 3, we extend this basic interaction flow with the implementation details for a TSC based learning. From the bulletin board, we know that there are two vehicles and a road with two lanes. Here, the ontological knowledge from the world model is applied: The lanes have a constant width of 4 m each and fixed positions for their borders: $y^{top}_{lane1} = 4\,\mathrm{m}$, $y^{bot}_{lane1} = 0\,\mathrm{m}$ and $y^{top}_{lane2} = 0\,\mathrm{m}$, $y^{bot}_{lane2} = -4\,\mathrm{m}$. *Ego* and *Other* are two cars with a two-dimensional position (x_{ego}, y_{ego}), a yaw heading θ_{ego} and a velocity v_{ego} (x_{other}, y_{other}, θ_{other} and v_{other} respectively). Here, (x_{ego}, y_{ego}) and (x_{other}, y_{other}) are the positions of the center point of cars' rear axle. Thus, we consider a training environment with two cars on a two lane road.

For vehicle's dynamics, knowledge on physical evolution is used. As the goal is to train an agent for *Ego*, only *Ego*'s dynamics are modelled in more detail. We use the following differential equations from the bicycle model:

$$\begin{aligned} \dot{x}_{ego} &= v_{ego}\cos(\theta_{ego}), \quad \dot{y}_{ego} = v_{ego}\sin(\theta_{ego}), \\ \dot{\theta}_{ego} &= v_{ego}\frac{\tan(\delta_{ego})}{L}, \quad \dot{v}_{ego} = a_{ego}, \end{aligned} \tag{1}$$

where $L = 2.8\,\mathrm{m}$ expresses the ontological knowledge about *Ego*'s wheelbase and the input variables a_{ego}, δ_{ego} denote *Ego*'s acceleration and steering angle. As

there are no further derivatives that are independent from others, a_{ego} and δ_{ego} are the only dynamic input variables and thus define the agent's action space.

The vehicle *Other* is modelled in a more simple way. During the training phase, the velocity and the x-position has been fixed with $v_{other} = 0\,\frac{\mathrm{m}}{\mathrm{s}}$ and $x_{other} = 100\,\mathrm{m}$. Later, the agent is applied in an environment with a constant non-zero velocity $v_{other} > 0$. In any case, *Other* stays on the right lane and cannot violate the TSC, because the training environment always guarantees that at least $y^{bot}_{lane2} < y_{other} < y^{top}_{lane2}$ holds.

4.3 Reward shaping

Notably, recent literature has already explored reward shaping in various contexts, including driving maneuvers such as overtaking. For instance, the work in [3] combines a Deep Q Neural Network with reinforcement learning, creating a simulative training environment. Here, the reward function includes terms to evaluate collisions, hazardous actions, lane changes, and overtaking maneuvers.

The reward function in [18] contains terms to evaluate, whether driving progress is made to counteract a high reward for simply standing. Furthermore, there are terms to penalize a drift out of the lane. The authors in [24] shaped the reward to encompass overtaking training, employing a more intricate function incorporating spatial relationships between vehicles and road topology. Moreover, these approaches incorporate components addressing constant speed and steering angle.

The work in [21] also deals with the training of an automated overtaking, where a navigation term evaluates the distance covered during a maneuver. In contrast to the prior works, we try to use the formalized knowledge in TSCs and thus mainly process knowledge to train an agent. Here, the focus is on the individual process steps such as a correct approaching, swerving, passing, and rejoining during an overtaking maneuver. In the work [17], an ACC was trained with an optimized distance measure. With this work, it was aimed to obtain script knowledge and reward terms using the TSC language, which can then be integrated into AI training using reward shaping.

For *Ego* who has to be controlled through the overtaking TSC, the agent has to select appropriate actions such that the TSC invariants $A, \ldots, E$ in Figure 1 are satisfied. Hence, we need a reward function that gives feedback regarding the satisfaction of the invariants which can be checked via the following formulae:

$$\begin{aligned}
\varphi_A &\iff x_{other} - x_{ego} > 50 \quad \wedge y^{bot}_{lane2} < y_{ego} \wedge y_{ego} < y^{top}_{lane2},\\
\varphi_B &\iff x_{other} - x_{ego} > 0 \quad \wedge y^{bot}_{lane2} < y_{ego} \wedge y_{ego} < y^{top}_{lane1},\\
\varphi_C &\iff \qquad\qquad\quad y^{bot}_{lane1} < y_{ego} \wedge y_{ego} < y^{top}_{lane1},\\
\varphi_D &\iff x_{other} - x_{ego} < 0 \quad \wedge y^{bot}_{lane2} < y_{ego} \wedge y_{ego} < y^{top}_{lane1},\\
\varphi_E &\iff x_{other} - x_{ego} < -50 \wedge y^{bot}_{lane2} < y_{ego} \wedge y_{ego} < y^{top}_{lane2}.
\end{aligned} \tag{2}$$

Those formulae can derived from the spatial view constraints in each invariant node. Here, the fundamental knowledge about space and time from Subsection 3.1 is applied, where relative distances between objects are transferred to

inequalities. If an object O is located left from an object P, their horizontal positions $O.x$ and $P.x$ are transferred to the constraint $O.x < P.x$.

If the agent violates all invariants at once, the whole TSC is violated and the agent misses it's tasks. We consider this as the worst case behaviour and provide in this case for the minimum penalty of -1. If the agent has fulfilled the TSC completely, the agent has completed it's task and gets the maximum positive reward of $+1$. First experiments with such a simple reward concept resulted in agents that are standing still, driving backwards or driving a circle on the road. Hence, the agent needs to be forced to approach and we used the TSC sequence as guideline. The more invariant nodes in a TSC sequence are satisfied, the closer the agent already reached it's goal. Therefore, we give fractions of the maximum reward $+1$ for partial TSC satisfaction as follows: Let k be the number of invariant nodes in a sequence. As the agent always starts with the first invariant (A in our overtaking example), there is a neutral reward, if only the first invariant node is satisfied. For each additional invariant satisfaction, the agent gets more reward in $\frac{1}{k-1}$ steps, as the agent has $k-1$ remaining invariant nodes to satisfy. Hence, the agent gets a higher reward the further it proceeds in the TSC sequence, in more detail if it already satisfies an invariant node close to the end.

To test the TSC2RL concept in general first, the experiments in this paper focus on a reward function for the simple sequence of five invariant nodes in case of the overtaking example from Figure 1. The reward shaping process above results to the following function:[5]

$$R = \begin{cases} 1 & \text{if } \varphi_E, \\ 0.75 & \text{if } \varphi_D \wedge \neg\varphi_E, \\ 0.5 & \text{if } \varphi_C \wedge \neg\varphi_D \wedge \neg\varphi_E, \\ 0.25 & \text{if } \varphi_B \wedge \neg\varphi_C \wedge \neg\varphi_D \wedge \neg\varphi_E, \\ 0 & \text{if } \varphi_A \wedge \neg\varphi_B \wedge \neg\varphi_C \wedge \neg\varphi_D \wedge \neg\varphi_E, \\ -1 & \text{otherwise.} \end{cases} \tag{3}$$

4.4 Observation Space

The observation space should give the reinforcement learning agent an update of the environment where it acts in. Two aspects have to be considered: On the one hand, the observation function must provide sufficient information in order to make different state-action pairs distinguishable from the reward perspective, i.e., the reward function must not be just constant over the observation space. Otherwise, the observation and reward procedure would be orthogonal in the sense that there is no chance to learn based on the reward function. On the other hand, the observation space should reflect a realistic vehicle perception. Here, epistemic knowledge applies and the observation space should be consistent with vehicle's sensors and measuring instruments.

[5] Note that $\varphi_A \wedge \neg\varphi_B \wedge \neg\varphi_C \wedge \neg\varphi_D \wedge \neg\varphi_E$ never applies as φ_A implies φ_B. Hence, the minimum attainable nonnegative reward is 0.25.

To address the first aspect, we start designing the observation space with the invariant formulae $\varphi_A, \ldots, \varphi_E$ in Equation 2. Each of them is a boolean formula of linear inequalities based on the distances:

$$x_{ego} - x_{other}, \qquad y^{bot}_{lane2} - y_{ego},$$
$$y_{ego} - y^{top}_{lane2}, \qquad y_{ego} - y^{top}_{lane1}, \qquad y^{bot}_{lane1} - y_{ego}.$$

These distance values in longitudinal or lateral dimension can be mapped to vehicle's distance sensors. Because the distances are related to the center point of vehicle's rear axle, the vehicle's heading around this center point is not captured in this vector. Hence, it would be very hard, as the agent has to learn the vehicle's dynamic model implicitly for steering and acceleration. To close this gap, the vehicle's heading angle θ_{ego} is an additional part of the observation space. Furthermore, also the current velocity v_{ego} is used. Although this value may be given implicitly in the observations of distances over time, we add it explicitly to the observation space as it is important for the bicycle model. Because $y^{top}_{lane2} = y^{bot}_{lane1} = 0$, which is the position of the middle line, the terms $y_{ego} - y^{top}_{lane2}$ and $y^{bot}_{lane1} - y_{ego}$ refer to the same distances and we only use the first representation. Finally, we have the following observation space:

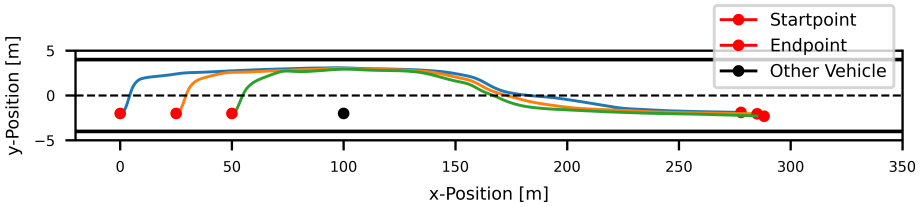
$$obs = \begin{pmatrix} x_{ego} - x_{other}, & y^{bot}_{lane2} - y_{ego}, \\ y_{ego} - y^{top}_{lane1}, & y^{bot}_{lane1} - y_{ego}, \\ \theta_{ego}, & v_{ego} \end{pmatrix}.$$

4.5 Experiments

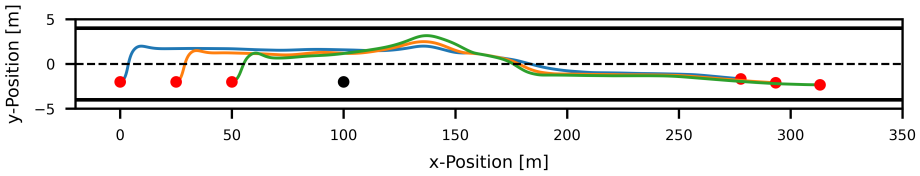
As we have described how to derive the training environment, the reward and the observation space from the TSC in the previous sections, we can apply them in a training process. In our experiments, we trained a Soft Actor Critic (SAC) [16] agent with 500.000 environment steps in total. In each environment step the next state is calculated via the bicycle model equations from Equation 1 with stepsize $dt = 0.02\,\mathrm{s}$. If a maximum episode time of $40\,\mathrm{s}$ is reached or the agent leaves the observation space, the agent is reset to a starting position behind the vehicle *Other* on the right lane. The agent gets a reward according to Equation 3 every five environment steps.

Figure 4 contains the application of four agents in 500 timesteps in case that *Other* is parking with $v_{other} = 0\,\frac{\mathrm{m}}{\mathrm{s}}$. Each agent has been trained separately. The subfigures Figure 4a - Figure 4d show the resulting path of each agent, when it controls *Ego* from a starting position in front of *Other*. In each path, the vehicle successfully completes an overtaking maneuver: *Ego* starts with a distance of at least $50\,\mathrm{m}$ to *Other* on the right lane, then a lane change is performed, after that, *Ego* overtakes *Other*, performs another lane change and finally reaches the right lane again. Hence, the TSC sequence from Figure 1 is satisfied.

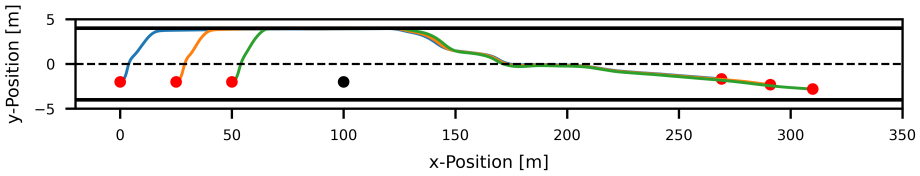
Some trajectories have a curvy and oscillating shape. From a perspective of comfortable or goal-oriented trajectories, this can be judged as undesired behaviour. However, nothing with regard to those trajectory requirements has been



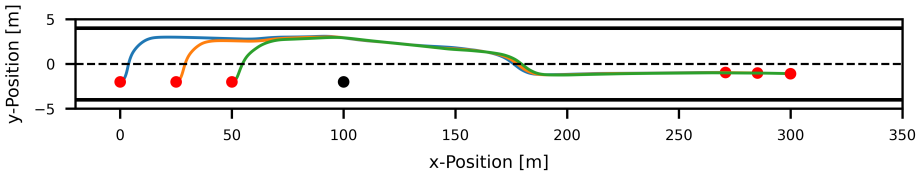
a Application of the first trained agent.



b Application of the second trained agent.



c Application of the third trained agent.

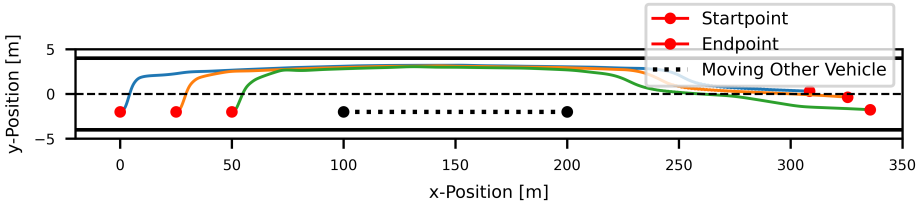


d Application of the fourth trained agent.

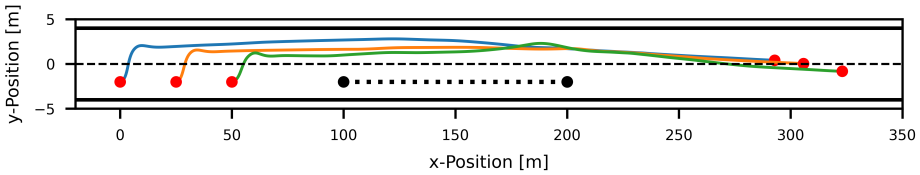
Fig. 4: Application of TSC2RL agents with $v_{other} = 0\,\frac{\mathrm{m}}{\mathrm{s}}$ and $x_{other} = 100\,\mathrm{m}$.

formalized in the TSCs. As the TSCs are an abstract description language, the detailed scenario execution shall remain open and a curvy and oscillating trajectory flow is valid behaviour. Hence, each agent can drive waveringly without being forced to reduce the angular velocity. Furthermore, on this development stage of TSC2RL, we focus to train the script knowledge of the given TSC formalization. Knowledge on comfortable trajectories might be added into future reward functions. Here, we are working on other experiments with reward functions for optimized TSC constraint satisfaction and small angular velocities.

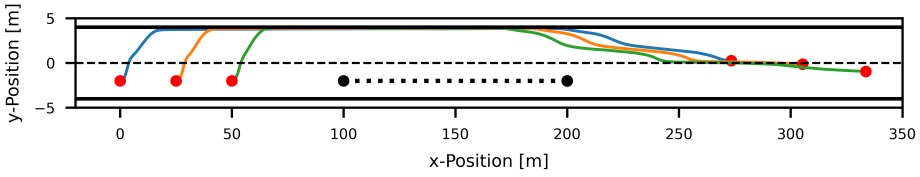
The agents in Figure 4 are trained while *Other* is parking with $v_{other} = 0\,\frac{\mathrm{m}}{\mathrm{s}}$. Looking at the TSC in Figure 1, one may note, that no restriction for v_{other} is made. In fact *Other* is nearly unrestricted and can move, accelerate and steer as long as it stays on the right lane. To check agents' transferability, we apply



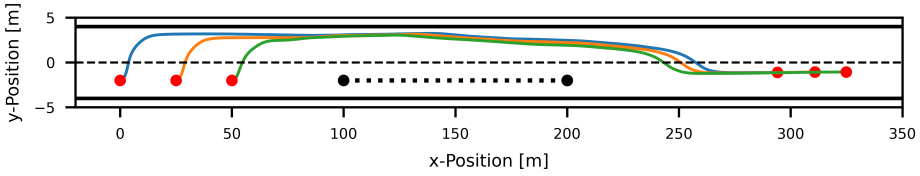
a Application of the first trained agent.



b Application of the second trained agent.



c Application of the third trained agent.



d Application of the fourth trained agent.

Fig. 5: Application of TSC2RL agents with $v_{other} = 10\,\frac{\mathrm{m}}{\mathrm{s}}$.

each agent in case that *Other* is moving with $v_{other} = 10\,\frac{\mathrm{m}}{\mathrm{s}}$ in Figure 5. This is possible, because the agent was not trained to absolute road positions but on relative distances to *Other*. In contrast to Figure 4, the agents in Figure 5 overtake the vehicle *Other* but did not perform the final lane change in any case. Because each trajectory was generated with the same amount of time steps, the trajectories differ in their velocity profile. In fact, also the agents from Figure 5a and Figure 5c would have managed a complete overtaking maneuver, if enough time steps would have been performed.

4.6 Future Work

The experiments in Figure 5 show, that an agent trained on simple conditions could be applied to more complex scenarios. This enables new capabilities in

scenario playout and testing, in particular, it goes beyond existing methods, which would be difficult to be applied in this setting since TSC specifications are in general rather coarse. For example, if RL agents for vehicle A and C from Figure 2b are trained, a reactive test environment grows up. If requirements are specified with TSCs and RL agents are trained for traffic participants, a controller for vehicle B can be tested for zipper rule compliance. In contrast to a static playout that may come from [5], the vehicles A and C can react to vehicle B's behaviour. As a first next step, we will apply the TSC2RL concept to more complex TSCs like the zipper rule, whereby at least two vehicles are controlled by trained RL agents.

In general, a TSC does not restrict the occurrence of further traffic as long as the depicted requirements are satisfied. In case of the overtaking maneuver in Figure 1, it would be possible that there is another car behind the black one or approaching on the oncoming lane. Currently, we have only trained an agent to overtake, who has not learned to react to oncoming traffic. Such additional important knowledge has to be formalized in TSCs as well. This leads to the research question of how and whether an agent can be trained with another TSC further on or how multiple agent trained from different TSCs can be combined.

With respect to the node operations *Choice* and *Concurrency* from Table 1 one may argue that the reward shaping process is restricted to sequential TSCs only. One procedure to overcome this is to consider the node-individual reward as a new maximum reward for any possible subsequences. If, i.e, a choice node is in a sequence $\boxed{I} \rightarrow \boxed{Choice} \rightarrow \boxed{J}$ and the choice node has two subsequences of the form $\boxed{K_1} \rightarrow \boxed{K_2} \rightarrow \ldots \rightarrow \boxed{K_k}$, we assign $\frac{1}{3}$ for $\boxed{I}$, $\frac{2}{3}$ for the complete choice node, $+1$ for $\boxed{J}$ and $\frac{1}{3} + \frac{1}{k} \cdot \frac{2}{3} \ldots \frac{1}{3} + \frac{k}{k} \cdot \frac{2}{3}$ for the nodes $\boxed{K_1} \ldots \boxed{K_k}$. However it is not clear, whether this approach prefers one of the subsequences indirectly, if they have a different length. Hence, we plan to evaluate this approach in future work together with further reward functions that include the satisfaction level of TSC constraints, for example. Furthermore, the timed automaton from [15] could be used for reward functions to check simulation runs for their TSC satisfaction. Then, the reward calculation can profit from a more detailed analysis of the current and previous invariant satisfaction and from the extent to which the agent has already achieved the complete TSC scenario.

5 Conclusion

In this paper, we presented the Traffic Sequence Charts language as a formalization tool for knowledge as well as being the basis for knowledge infused reinforcement learning. We, discussed the applicability to different types of knowledge from the automotive domain. It turns out that ontological knowledge and knowledge about physics could be anchored in the world model. With knowledge on space and time as fundamental prerequisite, script knowledge can be formalized directly via sequences of invariant nodes and, building upon this, deontic knowledge as premise-consequence charts. Currently, TSCs lacking in formalizing epistemic knowledge. However, we give two examples formalizing traffic

related knowledge that can act as *Knowledge Building Block* (KBB) to describe *what* (Ontology) kind of objects *should* or *must* (Script, Deontic) act under *which* (Physics) physical possibilities. We have shown that the TSC2RL approach profits from the KBBs as it fills the gap on *how* to realize a TSC scenario. With our experiments, we demonstrated that it is possible to train reinforcement learning agents based on the TSC language. More precisely, the declarative script knowledge in the TSC sequence is transferred into performative knowledge in the RL agent. In future, a more precise reward function could support the agent to learn improved behaviour. Furthermore, a stepwise transfer to more complex scenarios is intended for more TSC comprehensive agents and a reactive test environment.

References

1. Aksaray, D., Jones, A., Kong, Z., Schwager, M., Belta, C.: Q-learning for robust satisfaction of signal temporal logic specifications. In: 2016 IEEE 55th Conference on Decision and Control (CDC), pp. 6565–6570 (2016). https://doi.org/https://doi.org/10.1109/CDC.2016.7799279
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2007)
3. Bai, Z., Shangguan, W., Cai, B., Chai, L.: Deep Reinforcement Learning Based High-level Driving Behavior Decision-making Model in Heterogeneous Traffic. In: 2019 Chinese Control Conference (CCC), pp. 8600–8605 (2019). https://doi.org/10.23919/ChiCC.2019.8866005
4. Balakrishnan, A., Deshmukh, J.V.: Structured reward shaping using signal temporal logic specifications. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3481–3486 (2019). https://doi.org/https://doi.org/10.1109/IROS40897.2019.8968254
5. Becker, J., Koopmann, T., Neurohr, B., Neurohr, C., Westhofen, L., Wirtz, B., Böde, E., Damm, W.: Simulation of Abstract Scenarios: Towards Automated Tooling in Criticality Analysis. In: Autonomes Fahren. Ein Treiber zukünftiger Mobilität, pp. 42–51. SATW (Schweizerische Akademie der Technischen Wissenschaften) (2022). https://doi.org/10.5281/zenodo.5907154
6. Borghesi, A., Baldo, F., Milano, M.: Improving deep learning models via constraint-based domain knowledge: a brief survey. (2020). https://doi.org/https://doi.org/10.48550/arXiv.2005.10691
7. Damm, W., Kemper, S., Möhlmann, E., Peikenkamp, T., Rakow, A.: Traffic sequence charts – From visualization to semantics. Tech. rep., TR 14 AVACS 117, SFB/TR 14 AVACS, 10 2017 (2017)
8. Damm, W., Möhlmann, E., Peikenkamp, T., Rakow, A.: A Formal Semantics for Traffic Sequence Charts. In: Lohstroh, M., Derler, P., Sirjani, M. (eds.) Principles of Modeling: Essays Dedicated to Edward A. Lee on the Occasion of His 60th Birthday, pp. 182–205. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-95246-8_11
9. Deshmukh, J.V., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., Seshia, S.A.: Robust online monitoring of signal temporal logic. Formal Methods in System Design **51**, 5–30 (2017). https://doi.org/https://doi.org/10.1007/s10703-017-0286-7
10. for Digital, F.M., Transport, German Road Traffic Regulations (Translation),

11. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theoretical Computer Science **410**(42), 4262–4291 (2009). https://doi.org/https://doi.org/10.1016/j.tcs.2009.06.021
12. Filipe, J.: An Organizational Semiotics Model for Multi-agent Systems Design. In: Dieng, R., Corby, O. (eds.) Knowledge Engineering and Knowledge Management Methods, Models, and Tools, pp. 449–456. Springer Berlin Heidelberg, Berlin, Heidelberg (2000). https://doi.org/https://doi.org/10.1007/3-540-39967-4_34
13. Finkbeiner, B., Fränzle, M., Kohn, F., Kröger, P.: A truly robust signal temporal logic: Monitoring safety properties of interacting cyber-physical systems under uncertain observation. Algorithms **15**(4), 126 (2022). https://doi.org/https://doi.org/10.3390/a15040126
14. foretellix, Measurable Scenario Description Language Reference, https://www.foretellix.com/wp-content/uploads/2020/07/M-SDL_LRM_OS.pdf (2020). Accessed: 2024-01-26.
15. Grundt, D., Köhne, A., Saxena, I., Stemmer, R., Westphal, B., Möhlmann, E.: Towards Runtime Monitoring of Complex System Requirements for Autonomous Driving Functions. In: Luckcuck, M., Farrell, M. (eds.) Proc. Int. Workshop on Formal Methods for Autonomous Systems (FMAS) 2022. EPTCS, pp. 53–61 (2022). https://doi.org/10.4204/EPTCS.371.4
16. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, pp. 1861–1870. PMLR (2018). https://proceedings.mlr.press/v80/haarnoja18b.html
17. Jurj, S.L., Grundt, D., Werner, T., Borchers, P., Rothemann, K., Möhl-mann, E.: Increasing the Safety of Adaptive Cruise Control Using Physics-Guided Reinforcement Learning. Energies **14**(22) (2021). https://doi.org/10.3390/en14227572. https://www.mdpi.com/1996-1073/14/22/7572
18. Kaushik, M., Prasad, V., Krishna, K.M., Ravindran, B.: Overtaking Maneuvers in Simulated Highway Driving using Deep Reinforcement Learning. In: 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 1885–1890 (2018). https://doi.org/10.1109/IVS.2018.8500718
19. Kratzer, A.: "Modality". In: Semantik / Semantics. Ed. by A. von Stechow and D. Wunderlich. De Gruyter Mouton, 1991, pp. 639–650. ISBN: 9783110203363. https://doi.org/doi:10.1515/9783110126969.7.639. https://doi.org/10.1515/9783110126969.7.639.
20. Li, X., Vasile, C.-I., Belta, C.: Reinforcement learning with temporal logic rewards. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3834–3839 (2017). https://doi.org/https://doi.org/10.1109/IROS.2017.8206234
21. Li, X., Qiu, X., Wang, J., Shen, Y.: A Deep Reinforcement Learning Based Approach for Autonomous Overtaking. In: 2020 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–5 (2020). https://doi.org/10.1109/ICCWorkshops49005.2020.9145279
22. Lin, Y., Li, H., Althoff, M.: Model predictive robustness of signal temporal logic predicates. IEEE Robotics and Automation Letters (2023). https://doi.org/https://doi.org/10.1109/LRA.2023.3324582
23. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: International Symposium on Formal Techniques in Real-Time and Fault-Tolerant

Systems, pp. 152–166 (2004). https://doi.org/https://doi.org/10.1007/978-3-540-30206-3_12
24. Ngai, D.C.K., Yung, N.H.C.: A Multiple-Goal Reinforcement Learning Method for Complex Vehicle Overtaking Maneuvers. IEEE Transactions on Intelligent Transportation Systems **12**(2), 509–522 (2011). https://doi.org/10.1109/TITS.2011.2106158
25. Rai, R., Sahu, C.K.: Driven by data or derived through physics? a review of hybrid physics guided machine learning techniques with cyber-physical system (cps) focus. IEEE Access **8**, 71050–71073 (2020)
26. von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Pfrommer, J., Pick, A., Ramamurthy, R., Walczak, M., Garcke, J., Bauckhage, C., Schuecker, J.: Informed Machine Learning – A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems. IEEE Transactions on Knowledge and Data Engineering **35**(1), 614–633 (2023). https://doi.org/10.1109/TKDE.2021.3079836
27. Saxena, N., Gorantla, S., Jagtap, P.: Funnel-Based Reward Shaping for Signal Temporal Logic Tasks in Reinforcement Learning. IEEE Robotics and Automation Letters **9**, 1373–1379 (2022). https://api.semanticscholar.org/CorpusID:259950729
28. Schwammberger, M.: Distributed Controllers for Provably Safe, Live and Fair Autonomous Car Manoeuvres in Urban Traffic. PhD thesis, Oldenburg University, Germany (2021). https://oops.uni-oldenburg.de/4961/.
29. Weizi, L., Liu, K., Shuzhang, L., Hongqiao, Y.: An Agent Based Approach for Customized Clinical Pathway. Information Technology and Applications, International Forum on **2**, 468–472 (2009). https://doi.org/10.1109/IFITA.2009.325
30. Willard, J., Jia, X., Xu, S., Steinbach, M., Kumar, V.: Integrating physics-based modeling with machine learning: A survey. (2020)
31. Wörmann, J., Bogdoll, D., Bührle, E., Chen, H., Chuo, E.F., Cvejoski, K., van Elst, L., Gleißner, T., Gottschall, P., Griesche, S., *et al.*: Knowledge Augmented Machine Learning with Applications in Autonomous Driving: A Survey. arXiv preprint arXiv:2205.04712 (2022)