Extending Urban Multi-Lane Spatial Logic to Formalise Road Junction Rules*

Maike Schwammberger
University of Oldenburg
Oldenburg, Germany
schwammberger@informatik.uni-oldenburg.de

Gleifer Vaz Alves
Federal Univeristy of Technology - Parana
Ponta Grossa, Brazil
gleifer@utfpr.edu.br

During the design of autonomous vehicles (AVs), several stages should include a verification process to guarantee that the AV is driving safely on the roads. One of these stages is to assure the AVs abide by the road traffic rules. To include road traffic rules in the design of an AV, a precise and unambiguous formalisation of these rules is needed. However, only recently this has been pointed out as an issue for the design of AVs and the few works on this only capture the temporal aspects of the rules, leaving behind the spatial aspects. Here, we extend the spatial traffic logic, Urban Multi-lane Spatial Logic, to formalise a subset of the UK road junction rules, where both temporal and spatial aspects of the rules are captured. Our approach has an abstraction level for urban road junctions that could easily promote the formalisation of the whole set of road junction rules and we exemplarily formalise three of the UK road junction rules. Once we have the whole set formalised, we will model, implement, and formally verify the behaviour of an AV against road traffic rules so that guidelines for the creation of a Digital Highway Code for AVs can be established.

1 Introduction

Even though autonomous vehicles (AVs) are not yet thoroughly used on our roads [19], we are aware that sooner or later we shall see AVs driving on the roads [15]. We consider autonomous vehicles that comply with SAE levels 4 or 5 [18], meaning that the vehicle is either completely driverless, or it manages specific manoeuvres driverless, without a human driver intervening at any point.

So, there is a need to face many challenges. Especially, those issues related to the safety of AVs, i.e. how to assure that the AV behaves safely on the roads? For that, several issues, like obstacle avoidance, sensing the environment, speed control, object detection and recognition and the proper use of traffic rules, need to be addressed.

So far, the issue of traffic rules has not been a major concern for the design of an AV in the research community, as discussed by Prakken [26] and Alves et al. [3]. However, some recent work like the references [5], [25] and [21] have started to draw attention to the challenge of transforming a Highway Code into a Digital Highway Code. Notice that a set of traffic rules composes the rule book or precisely the Highway Code, while a Digital Highway Code is the version of the Highway Code supposed to comprehend those traffic rules designed for AVs. There is a clear trade-off on how to wrap the traffic road rules into a digital format in a way that the fewest possible changes are made considering the existent Highway Codes [5]. At the same time, this digital version of the Highway Code should be understandable for the AVs [21].

*This research was supported by the German Research Council (DFG) in the PIRE Projects SD-SSCPS and ISCE-ACPS under grants no. FR 2715/4-1, FR 2715/5-1.

M. Farrell & M. Luckcuck (Eds.):
Third Workshop on Formal Methods
for Autonomous Systems (FMAS2021)
EPTCS 348, 2021, pp. 1–19, doi:10.4204/EPTCS.348.1

However, for a Digital Highway Code that works for AVs, it is necessary to tackle the challenges of translating road traffic rules (written in natural language) into a language understandable for autonomous systems. Such language needs to be precise and unambiguous since these rules are involved in the process of safety assurance of road users. Once these rules are formalised and deployed into an AV, the AV behaviour can be properly checked against road traffic scenarios to assure that safety road requirements are being followed by the AV (NB: here safety requirements are only those related to the road traffic rules).

In this paper, we follow ideas related to previous work (see ref. [4]), where the UK Highway Code (specifically the section of Road Junction rules [11]) has been used as a basis for the proof-of-concept presented in ref. [4]. For the road traffic rules from the UK Highway Code, temporal and spatial aspects can be identified. For instance, "look all around **before** entering the junction"; "do not cross a road **until** there is a **safe gap**" ("before" and "until" reveal a temporal aspect, while "safe gap" reveals a spatial aspect). As a consequence, we need a formalism suitable to abstract not only the temporal aspects but also the spatial elements of the road traffic rules. *Linear temporal logic (LTL)* is a clear answer to capture the temporal aspects, and was used in references [3, 4], to represent the temporal elements of the road junction rules. Ref. [4] presents an architecture for modelling, implementing, and formal verifying the behaviour of an agent (representing an AV) against three road traffic rules (from the UK Highway Code).

As a proper candidate to represent spatial elements of traffic rules we identify *Urban Multi-lane Spatial Logic (UMLSL)*, which is used to formalise traffic situations at intersections in [29]. UMLSL is an interval logic that bases on Interval Temporal Logic (ITL) from [22] and is thus dedicated to capture spatial aspects of traffic. Also, *automotive-controlling timed automaton (ACTA)* are presented as a formal semantics for a crossing controller for turn manoeuvres. We aim to extend the logic UMLSL from [29] so that the road junction rules from the UK Highway Code can be formalised and analysed with it.

Our key goal is to enrich the approach from [3,4] so that not only the temporal order between events, but also spatial aspects, e.g. a safe gap, can be formalised. For this, we introduce formalisations for non-autonomous traffic participants and road side units (e.g. a traffic sign) and dedicated traffic rule controllers to the approach from [29].

Our contribution is organised as follows. As a background, we present an overview over the specification of temporal aspects of traffic rules from [4] and give an overview over the spatial traffic logic UMLSL from [29] in Sect. 2. We motivate and define our UMLSL extension for traffic rules in Sect. 3 and exemplarily formalise some of the UK traffic rules in Sect. 4. We present related work to our approach in Sect. 5 and conclude our work in Sect. 6 with a summary and some insights into future work possibilities.

2 Background

We give preliminary information about the approach on formalising traffic rules using temporal logic from [3] in Sect. 2.1 and in Sect. 2.2, we present details on the abstract model and logic *Urban Multi-lane Spatial Logic UMLSL* from [29].

2.1 The Road Junction Rules

In the UK Highway Code there are different sections which handle the road traffic rules for Overtaking, Roundabouts, Road Junctions, among others [11]. Here we are concerned with the

section of Road Junction rules, which is composed by 14 rules, from rule 170 to 183. The road junction rules describe how the driver is supposed to behave when entering a road junction, turning to right or left, waiting for a traffic light, etc. As it follows we show the first three rules (170, 171, and 172) that we have been previously formalised in LTL [3] and subsequently used in our agent-based architecture [4]. Observe that LTL can be used for specifying temporal properties and it uses basic propositional operators ($\wedge$, $\vee$, $\rightarrow$, $\neg$) and temporal modalities ($\Box$, $\Diamond$, $\bigcirc$, $\cup$, representing resp. always, eventually, next, and until).

Rule 170 (UK Highway Code): You should watch out for road users (RU). Watch out for pedestrians crossing a road junction (JC) into which you are turning. If they have started to cross they have priority, so give way. Look all around before emerging (NB: For the sake of clarity, we choose to use the term enter as an action which represents not only a driver entering a road junction, but also emerging from a road junction to another road). Do not cross or join a road until there is a safe gap (SG) large enough for you to do so safely.

Rule 170, represented in LTL, describes when the autonomous vehicle (AV) may enter the junction (JC):

$$\Box\ (\texttt{(watch(AV, JC, RU)} \wedge (\neg\ \texttt{cross(RU, JC)} \wedge \texttt{(exists(SG, JC)))} \rightarrow (\texttt{(exists(SG, JC)} \wedge \neg\ \texttt{cross(RU, JC))} \cup \texttt{enter(AV, JC))))}$$

Informal Description: it is always the case that the AV is supposed to watch for any road users (RU) at the junction (JC) and there are no road users crossing the junction and there is a safe gap (SG). Then, no road users crossing the junction and the existence of a safe gap should remain true, until the AV may enter the junction.

Rule 170 represented in LTL, when the autonomous vehicle (AV) should give way at the junction (JC):

$$\Box\ \texttt{(watch(AV,JC,RU)} \wedge (\texttt{cross(RU,JC))} \rightarrow \texttt{give-way(AV,JC))}$$

Informal Description: it is always necessary to watch out for road users (RU) and check if there is a road user crossing the junction. Then, the AV should give way to traffic.

Rule 171 (UK Highway Code): You MUST stop behind the line at a junction with a 'Stop' sign (ST) and a solid white line across the road. Wait for a safe gap (SG) in the traffic before you move off.

Rule 171 represented in LTL:

$$\texttt{exists(ST,JC)} \rightarrow \Box\ \texttt{(stop(AV,JC)} \cup \texttt{(exists(SG,JC)} \wedge \texttt{(exists(SG,JC)} \cup \texttt{enter(AV,JC))))}$$

Informal Description: when there is a stop sign (ST), then it is always the case the AV should stop at the junction until there is a safe gap (SG). And the safe gap must remain true until the AV enter at the junction.

Rule 172 (UK Highway Code): The approach to a junction may have a 'Give Way' sign (GW) or a triangle marked on the road (RO). You MUST give way to traffic on the main road (MR) when emerging from a junction with broken white lines (BWL) across the road.

Rule 172 represented in LTL:

$$\Box\ \texttt{((exists(AV,RO)} \wedge \texttt{enter(AV,JC))} \wedge \texttt{((exists(BWL,JC)} \vee \texttt{exists(GW,JC))} \rightarrow \texttt{give-way(AV,MR)))}$$

Informal Description: It is always the case that when there is an AV driving on a Road (RO) and the AV enters the junction and there is a Broken White Line (BWL) or a Give Way sign (GW), then the AV should give way to the traffic on the Main Road (MR).

2.2 An Abstract Model for Urban Traffic Scenarios

We introduce the *Urban Multi-lane Spatial Logic (UMLSL)* of [29] which allows for the formalisation of traffic manoeuvres at intersections. The term *intersection* is equal to the term *road junction* that is used in the UK Highway Code and in [3] (cf.Sect. 2.1). Hitherto, no traffic rules have been considered using UMLSL. Nonetheless, some road junction rules are already expressible with it "by accident". E.g. safety in the sense of collision freedom has been formally proven in [8, 29, 30] through mathematical proofs and UPPAAL model-checking [6].

Formulae of UMLSL are evaluated over an abstract representation of real-world intersections. Thus, we first introduce details about this *abstract model* before giving details on the logic UMLSL itself. We focus on those concepts from [29] that we actually extend in Sect. 3 and we leave out formal definitions for most of the concepts in this section. We refer the interested reader to [29] for more formal and in-depth details for our basis. As a running example, we use the traffic situation that is depicted in Fig. 1.

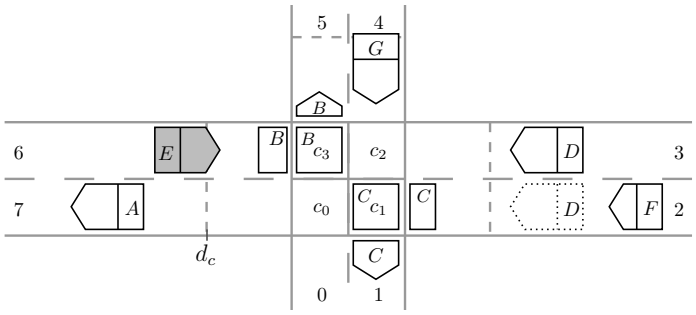

Figure 1: Example for the abstract model from [29].

The abstract model contains a set $\mathbb{CS}$ of *crossing segments* $c_0, c_1, \ldots$ and a set $\mathbb{L}$ of *lane segments* $0, 1, \ldots$ that connect different crossings. Each crossing segment and each lane segment has a finite length. Each car is assigned a unique *car identifier* $A, B, \ldots \in \mathbb{I}$ and a real value for the *position pos* of its rear on a lane or crossing segment. For our example from Fig. 1, we use car E as the ego *car* with a valuation $\nu(\text{ego}) = E$ to refer to this car. We distinguish between the *reservation* of a car that formalises the space a car is actually occupying (cf. $res(D) = \{3\}$) and the *claim* of a car, indicating the space a car plans to drive on in the future (cf. $clm(D) = \{2\}$, where car D plans to change back to lane 2 after it finished overtaking the slower car F). A claim is thus comparable to setting the turn signal. We also differentiate between claims and reservations on lane segments (*clm*, *res*) and on crossing segments (*cclm*, *cres*).

Urban road network. Connections of lane and crossing segments are formalised by a directed graph structure called *urban road network* $\mathcal{N}$ with the set of nodes $\mathcal{V} = \mathbb{L} \cup \mathbb{CS}$. The directed edges between lane and crossing segments specify the driving direction for continuous lane segments. For instance, while a car is allowed to drive from lane 6 onto crossing segment c_3, this is not allowed the other way around. Each car $C \in \mathbb{I}$ follows an infinite path $pth(C)$ with $pth : \mathbb{I} \to (\mathbb{Z} \to \mathcal{V})$, resembling its travelling route through the urban road network. E.g. in Fig. 1, the path of car E for turning right at the depicted crossing is given by $pth(E) = \langle \ldots 6, c_3, c_2, c_1, 1, \ldots \rangle$.

Traffic snapshot. Information like the road network $\mathcal{N}$, reservations, claims, positions and paths of all cars are collected in a global *traffic snapshot TS*. For the example from Fig. 1, we

have $clm(E) = cclm(E) = \emptyset$, as no space on a lane or crossing segment is claimed for car E (only car D has an active claim $clm(D) = \{2\}$). Further on, we observe $cres(E) = \emptyset$ and $res(E) = \{6\}$ as car E does not occupy a crossing segment but has some space reserved on lane 6. Car B, currently turning at the intersection, has reserved lanes $res(B) = \{5,6\}$ and a crossing reservation $cres(B) = \{c_3\}$.

One traffic snapshot can be compared to one snapshot of the overall traffic at an intersection at one moment. Whenever, e.g., time passes or a car claims or reserves a new lane or crossing segment, the traffic snapshot changes with respective traffic snapshot evolution transitions. For instance, with a time transition $TS_0 \xrightarrow{t} TS_1$ a traffic snapshot TS_0 evolves to a traffic snapshot TS_1, meaning that new positions are determined for all cars $C \in \mathbb{I}$ after t time units passed and cars moved along their paths with respect to their speed and acceleration values. Other traffic snapshot evolution transitions are triggered by the cars themselves. E.g., with a transition $TS_0 \xrightarrow{\mathsf{cc}(E)} TS_1$, crossing segments are claimed for car E along its path through the intersection.

Virtual view. For reasoning about traffic manoeuvres with the two-dimensional logic *Urban Multi-lane Spatial Logic (UMLSL)*, it is unrealistic and moreover unnecessary to consider an arbitrarily large traffic snapshot TS. Instead, we consider only a finite excerpt of TS called *Virtual View* (cf. [32]). A virtual view $V(E) = (L, X, E)$ is built around the ego car E and contains a sequence of parallel *virtual lanes* L and an extension interval X that determines how far "ahead" and "back" car E looks. For the example from Fig. 1 and for a right-turn view $V(E)$ for car E, we have virtual lanes $L = \langle\langle 6, c_3, c_2, c_1, 1\rangle, \langle 7, c_0, 0\rangle\rangle$.

Urban Multi-lane Spatial Logic. Formulae of UMLSL are built from (spatial) atoms, Boolean connectors and first-order quantifiers. Further on, spatial concepts that are inspired by *Interval Temporal Logic* (ITL) [22] are used. UMLSL introduces four different types of spatial atoms; The atom $re(C)$ (resp. $cl(C)$) formalises the reservation (resp. claim) of an arbitrary car C on some lane or crossing segment. With the atom $free$, free space on a lane or crossing segment is formalised and cs represents crossing segments. Note that no differentiation between between a crossing claim or reservation and a lane claim or reservation is done on the syntactical level of UMLSL. Also note that the lane number of a reserved lane is not available on the syntactical level of the atom $re(C)$. This is as the goal of this atom is neither to specify the identifier of a reserved lane nor the exact position of a car C on that lane, but rather to formalise whether a lane exists on which car C has a reservation. By combining these atoms with Boolean connectors, we can, e.g., state that car E occupies a crossing segment ($cs \wedge re(E)$) or that a crossing segment is free ($cs \wedge free$).

With the spatial connector $\frown$, UMLSL uses a variation of the chop operator ; from ITL. With a spatial formula $re(E) \frown free$, we can, e.g., state that there is free space in front of the reservation of our ego car E. Note that "in front of" or "right of" are informal descriptions for the adjacency of the two space intervals that are formalised by the atoms $re(E)$ and $free$.

Beside the horizontal chop operator $\frown$, UMLSL also introduces a vertical chop operator which is used by arranging two UMLSL formulae ϕ_1 and ϕ_2 one above the other. With this, elements that are located on two neighbouring lane segments can be formalised. E.g., the formula $\begin{smallmatrix} re(D) \\ cl(D) \end{smallmatrix}$ describes the situation where car D has a reservation on lane 3 and a claim on the neighbouring lane 2.

UMLSL introduces a comparison $u = v$ of variables $u, v \in \mathsf{Var}$ to, e.g., compare two car identifiers and a comparison $\ell = r$, to reason about the length ℓ of a spatial interval. This is, e.g., used for checking the distance of a car to an upcoming intersection.

Definition 1 (Syntax of UMLSL)**.** *Consider a car variable* $c \in \mathrm{CVar}$*, a real variable* $r \in \mathrm{RVar}$ *and general variables* $u, v \in \mathrm{Var}$*. The syntax of atomic* UMLSL formulae *is defined by* $\mathtt{a} ::= cs \mid true \mid u = v \mid \ell = r \mid free \mid re(c) \mid cl(c)$*, whereas an arbitrary* UMLSL *formula* ϕ_U *is formalised as follows:*

$$\phi_U ::= \mathtt{a} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \exists c\colon \phi_1 \mid \phi_1 \frown \phi_2 \mid \begin{smallmatrix}\phi_2\\ \phi_1\end{smallmatrix}$$

We denote the set of all UMLSL formulae by $\Phi_{\mathbb{U}}$*.*

In the following, we frequently use the abbreviation $\langle\phi\rangle$ to state that an arbitrary formula $\phi \in \Phi_{\mathbb{U}}$ holds *somewhere* in a view $V(E)$ of car E. This modality is used to abstract from exact positions in UMLSL formulae.

Example 1 (Syntax of UMLSL). For the example from Fig. 1, the UMLSL formula

$$ca(E) \equiv \langle re(E) \frown (free \wedge \ell < d_c \wedge \neg cs) \frown cs\rangle \tag{1}$$

formalises the "crossing ahead check" for car E, meaning that in front of (in Fig. 1: "right of") the reservation $re(E)$ of car E there is some free space, that is not on an intersection, with a length smaller than d_c and in front of (in Fig. 1: "right of") that there is a crossing space. △

The logic UMLSL is given a semantics that defines when a traffic snapshot satisfies a given formula. For this, the semantics of a UMLSL formula is evaluated over a traffic snapshot TS, a virtual view $V(E)$ and a variable valuation ν. The variable valuation ν respects types of variables, so that $\nu\colon \mathrm{CVar} \to \mathbb{I}$ and $\nu\colon \mathrm{RVar} \to \mathbb{R}$. Giving formal definitions for the semantics of the basic logic UMLSL would go beyond the scope of formalising traffic rules. However, we explain the semantics of our extension in Sect. 3.

3 UMLSL for Traffic Rules (USL-TR)

To formalise traffic rules (cf. [11]), we need to be able to reason about traffic signs and non-autonomous traffic participants (e.g. pedestrians, cyclists, human-driven cars, …), which is not yet possible using UMLSL.

Our goal is to keep the necessary UMLSL extension as minimal and elegant as possible. At the same time, we aim for a versatile UMLSL extension that is not tailored around the three traffic rules from Sect. 2.1 that we exemplarily formalise in the following Sect. 4 with our extension. This is as we want to keep the extension as general as possible so that a wider variety of traffic rules is formalisable. This includes that our extension is not limited to UK traffic.

To avoid a cumbersomly long abbreviation like "UMLSL-TR", we name the extended logic by *Urban Spatial Logic for Traffic Rules (USL-TR)*. USL-TR contains all elements of UMLSL and extends its abstract model and logic by two elements:

- A formalism for **static objects** (i.e. pedestrians, road-side units like traffic signs, traffic lights, …), and
- a formalism for non-autonomous **road users** (e.g. cyclists, human-driven cars...).

While it may seem unusual to capture pedestrians within the term "static objects", this is a reasonable design decision for the scope of this paper as we explain in the following; One of the main features of the basic logic UMLSL is that formulae of UMLSL are evaluated over a cut-out of an abstract model, which again is built upon a directed graph topology called urban road network. This urban road network contains lane and crossing segments that are connected

via (un-) directed edges and does not contain sidewalks or a roadside in general in its current version. A semantical introduction of such aspects is non-trivial but seems interesting for future work (cf. Sect. 6). Due to this, we cannot formalise the movement of a pedestrian, e.g. that a pedestrian on a sidewalk "is about to cross a road". Thus, for now, we capture a pedestrian that is about to cross a road at a crosswalk or already does so with an abstract static object which is either present or not in one traffic snapshot TS. Informally, this can be compared to a virtual cross-walk that appears whenever a pedestrian wishes to cross a road and that reserves the whole width of the road for crossing pedestrians. With this, we can, e.g., formalise the fragment "Watch out for pedestrians crossing a road junction" from Rule 170 of the UK Highway Code (cf. Sect.2.1). Note that this paper's goal is to formalise traffic rules and that we do not to reason about collision avoidance strategies with pedestrians. For the latter, e.g., movement directions of pedestrians would need to be considered in future work.

Also note that the described design decision implies that we deviate from the term *road user* that was used in the UK traffic rule book [11] and in the approach that we enrich [3]. Thus, from now on, the term road user comprises non-autonomous and autonomous entities that are not only crossing a road, but are actually able to drive on lane and crossing segments, e.g. cyclists, (non-) autonomous cars, motorcyclists, ... We frequently abbreviate the term "autonomous road user" to AV. In the following, we first describe two approaches that inspire our work in Sect. 3.1. After that, we define the necessary extensions to the abstract model for urban traffic in Sect.3.2 and then introduce syntax and semantics for the new logic USL-TR in Sect. 3.3.

3.1 From Hazards to Road-Side Units and Road Users

The extension USL-TR is inspired by two previous approaches that are presented in [9, 23]. Both approaches introduce moving or stationary hazards to UMLSL's predecessor logic MLSL from [17] to allow for hazard warning protocols. MLSL focuses solely on highway traffic, i.e. one-way traffic and no road intersections. Thus, our contribution is to adapt the ideas from [9,23] to the urban traffic case. The term "stationary hazards" from [23] comprises, e.g., a road accident, dense fog or a damaged road and the term *"moving hazards"* is used in [9] for non-autonomous, human-driven, cars. The main goal of both works is to show that a car receives a hazard warning message early enough and that no collisions with a hazard occur. Basically, we broaden the term "hazard" to a larger variety of objects to formalise traffic rules. We describe key differences and adaptation ideas in the following for both works [9, 23].

In [23], the authors introduce an object, namely a single *stationary hazard*, to the highway logic MLSL. The key difference from our approach is that [23] is tailored to cope with multi-lane highway scenarios and not with urban intersections. The second difference is that only one single hazard is allowed for the entire world and that this single hazard is hard-coded into the traffic snapshot TS. To formalise traffic rules, we allow for an arbitrary number of road-side units in one traffic snapshot TS and we add a possibility to add new and delete outdated static objects to a traffic snapshot TS. E.g., the need to install a warning sign for a damaged road might exist after an accident occurred but becomes obsolete after the damage was repaired.

In [9], the author proposes adaptations of [23] that allow for multiple stationary and moving hazards on a highway. We adopt a function from [9] that allows for an AV to turn into a moving hazard and vice versa. This is motivated by reality, as it allows for a take-over by a driver, e.g. if the AV has a malfunction that blocks an autonomy function or simply because only some types of manoeuvres can be handled autonomously by the AV (cf. SAE level 4).

Besides the domain "urban traffic", a key difference of our approach from [9] is that we do not give stationary objects a positive extension on a lane or crossing segment. This is because we assume road-side units to be positioned beside the road, not on it and because collisions with road-side units are not a topic of this paper. Further on, we are not restricted to "hazards", but instead consider a larger variety of static objects and non-autonomous road users.

3.2 Changes to the Abstract Model for Urban Traffic

We explain how to integrate static objects and road users into the existing abstract model for urban traffic from [29]. Throughout the remainder of this paper, new concepts are explained using the traffic situation that is depicted in Fig. 2.

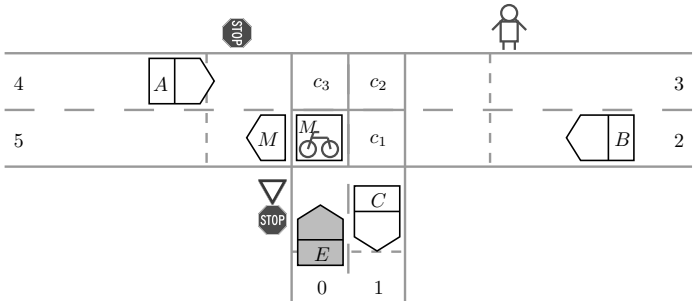

Figure 2: Example for an abstract model with both static objects (stop and give-way sign, a pedestrian), a non-autonomous road user (cyclist M) and autonomous road users A, B and E.

We introduce the set $\mathbb{O}$ containing identifiers for *static objects* like roadside units into our model. E.g., a stop sign could be identified with $stop \in \mathbb{O}$. To include static objects $o \in \mathbb{O}$ into the traffic snapshot TS, we introduce a function obj, which assigns a set of tuples containing a position and a lane or a crossing segment to each object o. With this, the same type of object can exist more than only once in the overall traffic snapshot TS (cf. the two priority signs in Fig. 2). Note that, as motivated in the previous section, we do not assign a positive extension ("size") to static objects. This means that in our case an object is a dot with a position on the road. This also holds for objects that would have a positive extension in reality, like the pedestrian depicted in Fig. 2. The intuition is that some of the traffic rules, e.g. rule 170 from the UK Highway Code (cf. Sect. 2.1), demand that a car should "watch out for pedestrians crossing a road junction" and that for this, it is sufficient that an autonomous car realises that there exists a pedestrian at a position at the roadside. However, for future work it might be of interest to add objects with an extension to our model (cf. Sect. 6).

We include identifiers for road users into the existing set $\mathbb{I}$of car identifiers and name $\mathbb{I}$ by *set of identifiers* in the following. With this we follow the intuition of [9], which is to allow for an AV C to turn into a non-autonomous road user and vice versa.

Note that, in our abstract model, we do not formally distinguish between different types of non-autonomous road users and that the visualisation of a bicycle for road user M is only depicted in Fig. 2 as a visual reminder that not all road users are AVs as before in Sect. 2.2 in Fig. 1. This differentiation is not necessary as those three UK traffic rules that were introduced in Sect. 2.1 and that we formalise in Sect. 4 also do not differentiate between different road users. E.g., rule 170 says to "watch out for road users" in general. However, for future work it

is of interest to distinguish between different road users. For instance, a cyclist may move with a slower velocity than a motorcyclist.

We do not repeat the lengthy definition of traffic snapshot elements that was introduced for UMLSL in [29]. Instead, we only define our object and road user extensions to the traffic snapshot TS and abbreviate other traffic snapshot elements with $\star$. Such other elements include, e.g., (crossing) reservations, the urban road network $\mathscr{N}$, and positions of cars (cf. Sect. 2.2).

Definition 2 (Traffic Snapshot Extensions)**.** *We extend the Definition of a traffic snapshot from [29] and abbreviate the extension with $TS = (\star, obj, aut)$, where $\star$ summarises those elements of TS which are defined in [29] and which are not altered by this definition. Given an arbitrary road user identifier $C \in \mathbb{I}$ and a static object $O \in \mathbb{O}$ the new elements in TS are defined as follows:*

- $obj\colon \mathbb{O} \to \mathscr{P}((\mathbb{L} \cup \mathbb{CS}) \times \mathbb{R})$ *such that $obj(O)$ yields a set of 2-tuples of each a lane resp. crossing segment $s \in \mathbb{L} \cup \mathbb{CS}$ together with a real position of O on the respective segment s and*
- $aut\colon \mathbb{I} \to \mathbb{B}$ *indicates whether an element $C \in \mathbb{I}$ is an AV or a non-autonomous road user.*

Example 2 (Extended Traffic Snapshot). Let us consider the traffic situation that is visualised in Fig. 2 and let us assume that the give way sign $gw \in \mathbb{O}$ is placed close to the intersection on lane 0 at position 98 (exemplarily assuming that lane 0 is, e.g., 100 units long and that thus 98 is indeed "close to the intersection"). We then have $obj(gw) = \{(0,98)\}$. For the stop signs, which are placed at lanes 0 and 4 respectively, we set $obj(stop) = \{(0,98),(4,198)\}$, exemplarily assuming that lane 4 is longer than lane 0 and that the stop sign at lane 0 is installed at the same position as the give way sign. Note that no positions for stop signs on the neighbouring lanes 1 and 5 are provided, as these are both lanes leaving away from the intersection. For the pedestrian $ped \in \mathbb{O}$, we set $obj(ped) = \{(2,90),(3,90)\}$ as we assume that for crossing the road, both lane segments 2 and 3 are reserved for her as a virtual cross-walk.

For the road users, we have $aut(M) = 0$ for the cyclist M and $aut(A) = aut(B) = aut(E) = 1$ for the autonomous cars A, B and E. The reserved spaces of all road users A, B, E and M are assigned according to the definition of a traffic snapshot TS from [29]: We have lane reservations $res(A) = \{4\}$, $res(B) = \{2\}$, $res(E) = \{0\}$, $res(M) = \{5\}$ and a crossing reservation $cres(M) = \{c_0\}$ for road user M on crossing segment c_0. △

With Def. 2, we define the extended structure of one single traffic snapshot with objects and road users at one distinct moment. As introduced before, a traffic snapshot changes, e.g. when a car $C \in \mathbb{I}$ reserves some crossing segments or when time passes and new positions for all road users are determined.

For static objects $O \in \mathbb{O}$, we introduce a function place assigning a new tuple containing a lane or crossing segments and a position to O. Reversely, a previously placed object O can be removed from the traffic snapshot TS through a function rm. Note that we use the overriding notation $\oplus$ of the specification language Z for function updates [33].

Definition 3 (Placing and removing static objects)**.** *Consider a current traffic snapshot $TS = (\star, obj, aut)$, where $\star$ again marks those traffic snapshot elements that were introduced in [29] and that are not of concern for this definition. For all $O \in \mathbb{O}$, $s \in \mathbb{L} \cup \mathbb{CS}$ and $p \in \mathbb{R}$ the following transitions hold:*

$$TS \xrightarrow{\mathsf{place}(O,s,p)} TS' \quad \Leftrightarrow \quad TS' = (\star, obj', aut) \ \wedge \ obj' = obj \ \oplus \ \{O \mapsto obj(O) \cup (s,p)\}\}$$

$$TS \xrightarrow{\mathsf{rm}(O,s,p)} TS' \quad \Leftrightarrow \quad TS' = (\star, obj', aut) \ \wedge \ obj' = obj \ \oplus \ \{O \mapsto obj(O) \setminus \{(s,p)\}\}$$

Example 3 (Placing and removing static objects). Again consider the example from Fig. 2. Through a function call $\mathsf{place}(stop, 2, 50)$, the visualised traffic snapshot evolves as a new instance of the stop sign *stop* is placed at lane segment 2 at position 50. Note that the set union operator ensures that existing placements of $stop \in \mathbb{O}$ are not altered. Alternatively, with a function call $\mathsf{rm}(stop, 0, 98)$, the instance of the stop sign at lane 0 at position 98 is removed, where the set difference operator ensures that only the one instance of *stop* is removed from $obj(stop)$. △

For non-autonomous road users $ru \in \mathbb{RU}$, we follow [9] and introduce a switching function that can be used to switch an autonomous car $C \in \mathbb{I}$ to a non-autonomous road user and vice versa.

Definition 4 (Switching the status of road users). *Consider a current traffic snapshot* $TS = (\star, obj, aut)$. *For all* $C \in \mathbb{I}$ *the following transition holds.*

$$TS \xrightarrow{\mathsf{switch}(C)} TS' \quad \Leftrightarrow \quad TS' = (\star, obj', aut) \ \wedge \ aut' = aut \ \oplus \ \{C \mapsto \neg aut(C)\}$$

Example 4 (Switching the status of road users). On calling $\mathsf{switch}(A)$, the status $aut(A) = 1$ of the AV A i changed to $aut(A) = 0$. With this, A is considered a non-autonomous road user. △

3.3 Syntax and Semantics of USL-TR

We extend the syntax and semantics of UMLSL by atoms for static objects and non-autonomous road users. For this sets of variables CVar and OVar ranging over identifiers from the set $\mathbb{I}$ and over the set of static objects $\mathbb{O}$ are used. Def. 1 on the syntax of UMLSL is extended as follows for the syntax of the traffic rule logic USL-TR.

Definition 5 (Syntax of new USL-TR concepts). *For an object variable* $o \in \mathrm{OVar}$ *and a car variable* $c \in \mathrm{CVar}$, *we extend the atomic* UMLSL formulae a *from Def. 1 as follows:* $\mathsf{a}' ::= \mathsf{a} \mid ob(o) \mid ru(c)$, *All definitions of binary and spatial connectors, as well as first-order logic quantifiers, to build USL-TR formulae* ϕ_T *remain as of Def. 1. We denote the set of all USL-TR formulae by* $\Phi_{\mathbb{T}}$.

Example 5 (Syntax of new USL-TR concepts). Consider car A and the stop sign that it approaches on lane 4 in the traffic situation that is depicted in Fig. 2. This situation can be formalised by $\phi_1 \equiv re(A) \frown free \frown ob(Stop)$, which informally reads as "there exists a reservation for car A, a part of free space in front of A and after that there exists a stop sign". The existence of the road user M on the intersection in front of car E can be formalised with $\phi_2 \equiv \langle re(E) \frown free \frown (ru(M) \wedge cs) \rangle$, which reads as "there exists a reservation for car E, a part of free space in front of E and after that there exists a road user M that is on a crossing segment". △

For the semantics of a USL-TR formula ϕ_T, a variable valuation $\nu \colon \mathrm{OVar} \to \mathbb{O}$ is used for objects and a valuation $\nu \colon \mathrm{CVar} \to \mathbb{I}$ is used for autonomous and non-autonomous road users. For variables $c, d \in \mathrm{CVar}$ the semantic difference of the atoms $re(c)$ for a reservation of an autonomous car $\nu(c)$ and $ru(d)$ for the reservation of a road user $\nu(d)$ is the autonomy flag $aut(c)$ (resp. $aut(d)$). Thus, together with the semantics of the new atoms for static objects $ob(o)$ and road users $ru(c)$, we give the changed semantics of the atom $re(c)$. Note that the autonomy flag is the only change of $re(c)$ compared to [29] and that we explain the mathematical concepts of the definition in detail subsequently. We again use the Z specification language.

Definition 6 (Semantics of new USL-TR concepts)**.** *With respect to a traffic snapshot TS, a virtual view $V = (L,X,E)$ and a valuation of variables ν, with $c \in \mathrm{CVar}$ and $o \in \mathrm{OVar}$, the* satisfaction *of the spatial USL-TR atoms $re(c)$, $ru(c)$ and $ob(o)$ is defined as follows:*

$$TS,V,\nu \models re(c) \Leftrightarrow \#L = 1 \text{ and } |X| > 0 \text{ and } aut(c) = true \text{ and } \forall s_i \colon L(1); \exists X_i \subseteq X \bullet$$
$$s_i \in cres(\nu(c)) \cup res(\nu(c)) \text{ and } (s_i, X_i) \in seg_V(\nu(c)) \text{ and } X \subseteq \bigcup_{i=1}^{\#L(1)} X_i \quad (2)$$

$$TS,V,\nu \models ru(c) \Leftrightarrow \#L = 1 \text{ and } |X| > 0 \text{ and } aut(c) = false \text{ and } \forall s_i \colon L(1); \exists X_i \subseteq X \bullet$$
$$s_i \in cres(\nu(c)) \cup res(\nu(c)) \text{ and } (s_i, X_i) \in seg_V(\nu(c)) \text{ and } X \subseteq \bigcup_{i=1}^{\#L(1)} X_i \quad (3)$$

$$TS,V,\nu \models ob(o) \Leftrightarrow \#L = 1 \text{ and } \#L(1) = 1 \text{ and } |X| = 0 \text{ and } \exists s \colon L(1); \exists p \colon \mathbb{R} \bullet$$
$$X = [p,p] \text{ and } (s,p) \in obj(o) \quad (4)$$

To satisfy the atomic formulae $re(c)$ and $ru(c)$, the view V has to be occupied completely by the respective element. This holds if V consists of only one virtual lane ($\#L = 1$) and has a positive extension ($|X| > 0$). As a quick reminder for the reader: Basically, one virtual lane complies with one possible path through an intersection (cf. exemplary virtual lanes for the traffic situation from Fig. 1 on p. 5). Then it is checked if for all lane or crossing segments $s_i \in \mathbb{CS} \cup \mathbb{L}$ that are contained in the one virtual lane $L(1)$, a (crossing) reservation for $\nu(c)$ exists ($s_i \in cres(\nu(c)) \cup res(\nu(c))$). With the abstract function $seg_V(\nu(c))$, it is checked that the considered extension interval X_i on segment s_i is visible in V and that all segments in V are completely occupied by $\nu(c)$ (cf. last part of formulae (2) and (3)). Definition (4) restricts the view V even further: As a static object is only a dot on a segment, the virtual lane $L(1)$ may only contain one single segment $s \in \mathbb{CS} \cup \mathbb{L}$ and the interval extension is limited to $X = [p,p]$, where p is the position value that is assigned to s in $obj(o)$. Note that as single USL-TR atoms are always required to completely fill a view V_i, the larger view $V(E)$ that is considered around one ego car E to reason about traffic rules generally consists of several smaller views V_i.

Example 6 (Evaluation of USL-TR formulae over views)*.* We continue the previous example 5 and again consider formula $\phi_2 \equiv re(E) \frown free \frown (ru(M) \wedge cs)$, which is parted into three parts using the chop operator $\frown$. As each sub-formula ϕ_2^i is evaluated over one sub-view V_i, the view satisfying ϕ_2 can be, e.g., parted into these three sub-views. △

4 Formalisation of UK Road Junction Rules using USL-TR

With USL-TR, we propose a means to formalise spatial aspects of traffic rules, like e.g. that a stop sign is ahead or that a safe gap is large enough for an AV. As motivated earlier, traffic rules also contain temporal aspects, which were the focus of the previous work [4] that we briefly outline in Sect. 2.1. Here, our means to formalise temporal aspects of traffic manoeuvres are extended timed automata controllers that use formulae of USL-TR in guards and invariants. With this, we follow [29], where *automotive-controlling timed automata (ACTA)* were introduced as an extension of the original timed automata from [1] to specify and verify a crossing controller for turn manoeuvres at intersections.

Our overall endeavour is to fully integrate USL-TR into the agent-based approach from [4]. For this, it is of interest to consider a combination of the agent-based UPPAAL implementation

from [4] with the UPPAAL implementation that was done in [8,30] to verify the system properties of the crossing controller from [29]. However, this paper focuses on the extension USL-TR. We exemplarily show-case the usability of USL-TR by formalising the UK road junction rules 170 to 172 that were introduced in Sect. 2.1. We sketch connections between the approaches [4] and USL-TR in the following paragraph but refer to future work for the actual integration of USL-TR into [4] (cf. Sect.6).

Besides spatial and temporal aspects, the nature of a *rule* generally requests a certain behaviour, i.e. *action*. E.g., the need to stop the car on encountering a stop sign. For this, [4] uses abstract actions like `enter(AV, JC)` or `stop(AV,JC)` (cf. Sect. 2.1). As a counterpart, ACTA come with certain *controller actions* which allow to, e.g., set a turn signal at an intersection or accelerate/decelerate an AV. In this section, we introduce USL-TR guards and invariants for the detection part of a rule (e.g. "there is a stop sign"). In [4], abstract actions like `watch(AV,JC,RU)` were used for this. This paper is about steps towards a Digital Highway Code, thus encoding traffic rules for AVs. Nonetheless, the rules also hold for non-autonomous vehicles as they are from the UK Highway Code.

A non-trivial problem that we face in this section, and that was also sketched in [4], is the problem of accurately translating natural language sentences into an accurate machine-readable and -understandable language. E.g., in the part "watch out for road users" of rule 170, it is not specified where and when an AV should do this. As the rules are from the road junction part of the UK Highway Code, we assume that they should hold **at intersections** and, if not specified differently, do so **always**. The natural language translation problem is outlined in detail in [27]. One of the difficulties is the often imprecise wording and the ambiguous semantics of some natural language phrases. For now, we explain our formalisation choices as detailed as possible. However, automatic means to extract formal specifications of traffic rules from their natural language counterparts are of interest for future work. E.g., in [14], the authors automatically extract requirements specifications from semi-formal natural language requirements.

Safe gap. We start with the formalisation of a "safe gap" as this is a feature frequently required by several of the considered UK traffic rules. Safe gaps on crossing segments are needed if an AV wants to enter an intersection and safe gaps on lane segments are needed when the AV leaves the intersection. For the specification of safe gaps on lane segments for overtaking manoeuvres, we also refer to [16], where a predecessor logic of UMLSL for two-way country roads is introduced. In our urban traffic case, safe gaps can be formalised for an ego car E by using the basic UMLSL version from [29]. We assume that the size of a safe gap is relative to the size $size_E$ of the ego car. This size is retrieved in [29] through a sensor function. A safe gap of free space for ego car E anywhere, i.e. not necessarily on an intersection, can be formalised with the formula

$$sg(E) \equiv free \wedge \ell >= size_E. \tag{5}$$

Consequently, a safe gap on an intersection (resp. on a lane) can be specified by adding "$\wedge cs$" ("$\wedge \neg cs$" resp.) to formula (5). However, for road junction rules, we need to specify that the safe gap is free on the intersection in front of E and not on any arbitrary intersection. Thus, formula (5), needs to be embedded into a formula that is specified from the perspective of and relatively to the ego car E. This is done via the formula

$$sg_I(E) \equiv \langle (re(E) \wedge \neg cs) \frown (free \wedge \neg cs) \frown (sg(E) \wedge cs) \rangle, \tag{6}$$

which is an adaptation of the crossing ahead check $ca(E)$ from formula (1), p. 6. Formula (6) states that the reservation of the ego car E is on a lane segment before an intersection, that

there might be some free space between the AV and the intersection (the car will not stand exactly in front of the intersection when it checks for a safe gap) and then there is a free safe gap (cf. formula (5)) on the intersection.

Rule 170. As seen in Sect. 2.1, rule 170 contains the following four parts:

1. You should watch out for road users.
2. Watch out for pedestrians crossing a road junction into which you are turning. If they have started to cross they have priority, so give way.
3. Look all around before emerging.
4. Do not cross or join a road until there is a safe gap large enough for you to do so safely.

As we do not identify any dependencies between these parts, we formalise and describe them separately. Each part is required to hold, which could be, e.g., achieved by a conjunction or by modelling four controllers that run in parallel and ensure that each part holds invariantly.

Part 1. The first part of rule 170 demands to *"watch out for road users"* (we assume "always", "at intersections", cf. previous remark on p. 4). For this, we refer to a concept from [29], where such a feature is already included in the behaviour of the crossing controller: On approaching an intersection, ego's controller always checks for *potential collisions* using the UMLSL formula

$$pc(c) \equiv c \neq \text{ego} \wedge \langle cl(\text{ego}) \wedge (re(c) \vee cl(c)) \rangle. \tag{7}$$

For a car c different than ego, formula (7) checks for path intersections of ego's own claim ("path through an intersection") and the claims or reservations of other road users. If a potential collision with any road user c exists, ego withdraws its claim and only enters the intersection later if no potential collisions are detected. Note that in [29] such road users $c \in \mathbb{I}$ only include AVs. However, as in this work non-autonomous road users are included into the set of identifiers $\mathbb{I}$ (cf. Sect. 3.2, Def.2), the potential collision check also applies to non-autonomous road users.

An extension of this part of rule 170, namely rule 221, has been formulated in the UK Highway Code and requires to "watch out for long vehicles which may be turning at a junction ahead", as those might need the whole intersection for their manoeuvre. Note that this extension is already included in the above description, as a long vehicle would simply demand all crossing segments for its path through the intersection. With this, ego would also watch out for long vehicles with $pc(c)$. However the urban road network $\mathscr{N}$ (cf. Sect. 2.2, p. 4) needs to be built accordingly to allow for long vehicles.

Part 2. The second part of rule 170 focuses solely on pedestrians: *"Watch out for pedestrians crossing a road junction ... so give way."*. To formalise this rule with USL-TR, we again manipulate the crossing-ahead check from formula (1). Specifically, we formalise a "pedestrian ahead" check $pa(E)$ for the ego car E:

$$pa(E) \equiv \langle re(E) \frown (free \wedge \ell < d_p) \frown ob(Ped) \rangle. \tag{8}$$

Formula (8) holds if a pedestrian $Ped \in \mathbb{O}$ is ahead of the ego car E within a given safety distance d_p. Next, the rule demands that the AV should give way to the pedestrian. Basically, *"give way"* is a request that an AV should decelerate or stop to let a pedestrian cross. This is implemented through the simplified traffic rule controller $ACTA_{170}$ that we depict in Fig. 3: Consider the transition from the initial location q_0 to location q_1. If the guard (8) holds, ego decelerates using the controller action "accelerate" $\texttt{acc}(-a)$ with the negation of an abstract acceleration

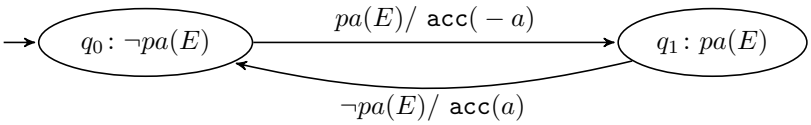

Figure 3: $ACTA_{170}$ implementing the part of rule 170 of the UK highway code for road junctions where an AV waits for a pedestrian to cross a road.

value $a \in \mathbb{R}^+$ and changes to location q_1, where $pa(E)$ holds invariantly. If the pedestrian is not crossing anymore and $\neg pa(E)$ holds, the controller accelerates again with a positive acceleration value a. We assume that a is large enough so that the car can decelerate timely and can come to a stillstand before reaching the crossing pedestrian.

Note that we use $ACTA_{170}$ as a tool to show how our formalisations can be used as guards and invariants in a controller that is built to follow traffic rules. However, it is a very abstract model for decelerating and accelerating an AV. So, instead of a controller action $acc(a)$ (resp. $acc(-a)$), it is more realistic that $ACTA_{170}$ would rather communicate the need for an acceleration (resp. deceleration) to a speed control unit within the AV.

Part 3. We interpret that the third part of rule 170, "Look all around before emerging", requires the AV to check for path intersections with other AVs, non-autonomous road users or with pedestrians before leaving ("emerging from") an intersection. This can be formalised by

$$look(E) \equiv \langle (re(E) \wedge cs) \frown (\neg cs \wedge sg(E)) \rangle \wedge \neg \exists c \colon pc(c) \wedge \neg pa(E). \tag{9}$$

Basically, the three conjugated fragments of formula (9) formalise more than the informal and imprecise phrase "Look all around before emerging" comprises:

- it is checked that car E is on an intersection ($re(E) \wedge cs$) and that after the intersection, there is a free safe gap for car E available for E to emerge into ($\neg cs \wedge sg(E)$, cf. formula (5)),
- it is checked that no potential collisions with other road users exist ($\neg \exists c \colon pc(c)$), and
- we ensure that no pedestrian is ahead (cf. formula (8)).

If formula (9) holds as a transition guard in $ACTA_{170}$, ego is allowed to leave the intersection, which again means adjusting E's speed. We do not explicitly depict this transition in Fig. 3.

Part 4. This part demands to "not cross or join a road until there is a safe gap". We understand that this rule demands i. that an AV does not enter an intersection ("do not cross") until there is a fitting safe gap available, and ii. that an AV only then leaves an intersection ("join a road"), if a safe gap is available on the road after an intersection. We already included part ii. into the first fragment of our formalisation (9). For part i., we refer back to formula (6), where we formalised a free safe gap on an intersection that lays ahead of E.

Rule 171. This rule demands that the AV "must stop behind the line at a junction with a stop sign and a solid white line across the road". A stop sign or a "solid white line across the road" that is ahead can be identified with a USL-TR formula equivalent to the pedestrian ahead check from formula (8). We rewrite the "pedestrian ahead" check $pa(E)$ from formula (8) to a "stop sign ahead" check $sta(E)$ for the ego car E:

$$sta(E) \equiv \langle re(E) \frown (free \wedge \ell < d_{st}) \frown ob(Stop) \rangle. \tag{10}$$

Note that we require $ob(Stop) \in \mathbb{O}$ and assume that d_{st} is a distance within which a stop sign is considered close enough for the AV to act. Here, the action again is a deceleration of the AV so that it comes to a timely standstill. Additionally, rule 171 demands to wait for a safe gap before moving onto an intersection (cf. Sect. 2.1), which we discussed before in detail for rule 170.

Rule 172. Instead of a stop sign, a give way sign is in the focus of this rule: "The approach to a junction may have a 'Give Way' sign or a triangle marked on the road. You must give way to traffic on the main road when emerging from a junction with broken white lines across the road". The identification of a give way sign (or any other optical identification of it) is again similar to the pedestrian ahead check (8). We name this check by "give way ahead", $gwa(E)$, but do not rewrite formula (8) here for reasons of brevity.

If $gwa(E)$ holds, the AV E must give way to the traffic on the main road. For this, we can use a feature from [8], where fairness was introduced to the crossing controller from [29]. For this, AVs are assigned priorities on arriving at an intersection. The longer an AV waits in front of the intersection, the more its priority increases. For our implementation of rule 172 this means that whenever the "give way ahead" formula $gwa(E)$ holds for an AV E, E receives a priority penalty. On the other hand, another AV C on the main road receives a priority bonus on approaching a crossing where it has the right of way. With this, using the crossing controller from [8], cars on the main road will always get the right of way.

5 Related Work

There are some approaches which aim to formalise road traffic rules. Pek et al. [24] formalise the safety of lane change manoeuvres to avoid collisions. The authors use as reference the Vienna Convention on traffic rules to formalise a single rule on the safe distance. They use algebraic equation to formalise this road traffic rule. Rizaldi et al. [28] formalise and codify part of the German Highway Code on the overtaking traffic rules in LTL (three rules are formalised). They show how the LTL formalisation can be properly used to abstract concepts from the traffic rules and obtain unambiguous and precise specification for the rules. In addition, they formally verify the traffic rules using Isabelle/HOL theorem prover and also monitor an AV applying a given traffic rule, which has been previously formalised using LTL. Bhuiyan et al. [7] assess driving behaviour against traffic rules, specifically the overtaking rules from the Queensland Highway Code. Two types of rules are specified: overtaking to the left and to the right. Moreover, they intend to deal with rules exceptions and conflicts in traffic rules (this is solved by setting priorities among the rules). Using DDL (Defeasible Deontic Logic) they assess the driving behaviour telling if the driver has permission or it is prohibited to apply a given rule for overtaking. The results basically show if the proposed methodology has recommended (or not) the proper behaviour for the driver (permission or prohibition). Besides, Esterle et al. [13] present a fomalisation of traffic rules for two-lane roads ("dual carriageways") in LTL to specify temporal behaviour. A set of formalised traffic rules is presented and evaluated on a public dataset.

In [10], Traffic Sequence Charts (TSCs) have been introduced as a visual language for describing first-order logic predicates for traffic situations. TSCs allow to introduce arbitrary objects and traffic rules like, e.g., a lane-change rule are exemplarily formalised in [10]. However, TSCs abstract from several aspects. Also, we aim at the formalisation of traffic rules at road junctions, while, to our best knowledge, TSCs are limited to multi-lane highway traffic scenarios. Nonetheless, a combination of TSCs with our approach is of interest for future work.

The authors of [20] focus on the translation of traffic rules from the California's DMV driver handbook from natural language to formal language (i.e. first order logic representations) and simulate their approach for exemplary four way and three way uncontrolled intersections using the CARLA urban driving simulator for autonomous vehicles [12]. They show that the behaviour of autonomous vehicles under their controller are more realistic compared to CARLA's default FIFO controller. However, neither of the aforementioned works formalise spatial aspects of traffic rules and only one of them [20] formalises road junction rules.

6 Conclusion

We have presented an extension for the logic UMLSL to handle a subset of road junction rules of the UK Highway Code. We capture not only the temporal but also spatial aspects of traffic rules, e.g. a safe gap.

Despite having shown the formalisation of only three road junction rules, the extension of the formalisation for the set of all 14 road junction rules of the UK Highway Code should not pose to many difficulties. This is as the remaining rules outline similar elements such as traffic lights, dual carriageways, other uses of safe gap situations or manoeuvres like turning at a road junction. With static objects and road users, USL-TR already contains the necessary concepts to abstract and formalise the aforementioned elements from the road junction rules.

Our vision is that the formalisation can provide some guidelines for the deployment of a Digital Highway Code for AVs. Two important guidelines are i. the spatial abstraction to represent a safe gap, which is largely used throughout the UK road junction rules (cf. p. 12); and ii. the effortlessly switching of status (autonomous or non-autonomous) for a given road user (cf. Def. 4), which helps to represent special emergency scenarios, where the AV control is given back to the human driver (cf. [2]).

As future work, we see two research directions; Firstly, we will examine the need for further extensions of USL-TR. E.g., static objects currently have a position on a segment, but no positive extension ("size"). Such a size could be of interest to properly model pedestrian crosswalks or to integrate hazardous situations into the USL-TR world (cf. Sect. 3.1). Also, it would be of interest to consider an integration of a roadside into the abstract model. With this, e.g., a pedestrian approaching a crosswalk on a sidewalk could be specified. Note that with this, we would also be able to consider the movement of pedestrians on sidewalks and could model pedestrians as a special type of road user instead of the static abstraction that we currently use (cf. p. 7).

Secondly, and most importantly, we shall model, implement, and verify the road junction rules following the agent-based architecture defined by Alves et al. [4]. For that, we will need to change the model and implementation in a way the spatial elements represented by the USL-TR are properly described. The agent implementation will have to consider an extension in the agent's environment to represent the lanes and crossing segments. For this, the UPPAAL implementation of the abstract model for UMLSL from [8,31] might be of help. Also, the status switching function for an AV can be easily captured by changing the agent's belief (since the agent is implemented using a BDI language, cf. [4]). Besides, we intend to use simulation tools like, e.g., CARLA [12] to evaluate our approach in a setting that is closer to reality.

References

[1] Rajeev Alur & David L. Dill (1994): *A Theory of Timed Automata*. *Theoretical Computer Science* 126(2), pp. 183–235, doi:10.1016/0304-3975(94)90010-8.

[2] Gleifer Vaz Alves, Louise Dennis, Lucas Fernandes & Michael Fisher (2020): *Reliable Decision-Making in Autonomous Vehicles*. In Andrea Leitner, Daniel Watzenig & Javier Ibanez-Guzman, editors: *Validation and Verification of Automated Systems: Results of the ENABLE-S3 Project*, Springer International Publishing, Cham, pp. 105–117, doi:10.1007/978-3-030-14628-3_10.

[3] Gleifer Vaz Alves, Louise Dennis & Michael Fisher (2020): *Formalisation and Implementation of Road Junction Rules on an Autonomous Vehicle Modelled as an Agent*. In Emil Sekerinski, Nelma Moreira, José N. Oliveira, Daniel Ratiu, Riccardo Guidotti, Marie Farrell, Matt Luckcuck, Diego Marmsoler, José Campos, Troy Astarte, Laure Gonnord, Antonio Cerone, Luis Couto, Brijesh Dongol, Martin Kutrib, Pedro Monteiro & David Delmas, editors: *Formal Methods. FM 2019 International Workshops*, Lecture Notes in Computer Science, Springer International Publishing, Cham, pp. 217–232, doi:10.1007/978-3-030-54994-7_16.

[4] Gleifer Vaz Alves, Louise Dennis & Michael Fisher (2021): *A Double-Level Model Checking Approach for an Agent-Based Autonomous Vehicle and Road Junction Regulations*. *Journal of Sensor and Actuator Networks* 10(3), doi:10.3390/jsan10030041. Available at `https://www.mdpi.com/2224-2708/10/3/41`.

[5] Michelle Avary & Tim Dawkins (2020): *Safe Drive Initiative: Creating safe autonomous vehicle policy (World Economic Forum)*. Available at `https://www.weforum.org/reports/safe-drive-initiative-creating-safe-autonomous-vehicle-policy/`.

[6] Gerd Behrmann, Alexandre David & Kim G. Larsen (2004): *A Tutorial on Uppaal*. In Marco Bernardo & Flavio Corradini, editors: *Formal Methods for the Design of Real-Time Systems*, Springer, pp. 200–236, doi:10.1007/978-3-540-30080-9_7.

[7] Hanif Bhuiyan, Guido Governatori, Andy Bond, Sébastien Demmel, Mohammad Badiul Islam & Andry Rakotonirainy (2020): *Traffic Rules Encoding Using Defeasible Deontic Logic*. In Villata Serena, Jakub Harasta & Petr Kremen, editors: *Legal Knowledge and Information Systems - JURIX 2020: The Thirty-third Annual Conference, Brno, Czech Republic, December 9-11, 2020*, *Frontiers in Artificial Intelligence and Applications* 334, IOS Press, pp. 3–12, doi:10.3233/FAIA200844.

[8] Christopher Bischopink & Maike Schwammberger (2019): *Verification of Fair Controllers for Urban Traffic Manoeuvres at Intersections*. In Emil Sekerinski, Nelma Moreira, José N. Oliveira, Daniel Ratiu, Riccardo Guidotti, Marie Farrell, Matt Luckcuck, Diego Marmsoler, José Campos, Troy Astarte, Laure Gonnord, Antonio Cerone, Luis Couto, Brijesh Dongol, Martin Kutrib, Pedro Monteiro & David Delmas, editors: *Formal Methods. FM 2019 International Workshops - Porto, Portugal, October 7-11, 2019, Revised Selected Papers, Part I*, *Lecture Notes in Computer Science* 12232, Springer, pp. 249–264, doi:10.1007/978-3-030-54994-7_18.

[9] Cristopher Bischopink (2018): *Moving Hazards - Reasoning about humand drivers in autonomous traffic*. Master's thesis, University of Oldenburg.

[10] Werner Damm, Eike Möhlmann, Thomas Peikenkamp & Astrid Rakow (2018): *A Formal Semantics for Traffic Sequence Charts*. In Marten Lohstroh, Patricia Derler & Marjan Sirjani, editors: *Principles of Modeling - Essays Dedicated to Edward A. Lee on the Occasion of His 60th Birthday*, Springer, pp. 182–205, doi:10.1007/978-3-319-95246-8_11.

[11] Department for Transport (2017): *Using the road (159 to 203) - The Highway Code - Guidance - GOV.UK*. Available at `https://www.gov.uk/guidance/the-highway-code/using-the-road-159-to-203`.

[12] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López & Vladlen Koltun (2017): *CARLA: An Open Urban Driving Simulator*. In: *1st Annual Conference on Robot Learning, CoRL*

2017, Mountain View, California, USA, November 13-15, 2017, Proceedings, *Proceedings of Machine Learning Research* 78, PMLR, pp. 1–16. Available at `http://proceedings.mlr.press/v78/dosovitskiy17a.html`.

[13] Klemens Esterle, Luis Gressenbuch & Alois C. Knoll (2020): *Formalizing Traffic Rules for Machine Interpretability.* In: *3rd IEEE Connected and Automated Vehicles Symposium, CAVS 2020, Victoria, BC, Canada, November 18 - December 16, 2020*, IEEE, pp. 1–7, doi:10.1109/CAVS51000.2020.9334599.

[14] Shalini Ghosh, Daniel Elenius, Wenchao Li, Patrick Lincoln, Natarajan Shankar & Wilfried Steiner (2016): *ARSENAL: Automatic Requirements Specification Extraction from Natural Language.* In Sanjai Rayadurgam & Oksana Tkachuk, editors: *NASA Formal Methods*, Springer International Publishing, Cham, pp. 41–46, doi:10.1007/978-3-319-40648-0_4.

[15] Andrew J. Hawkins & Richard Lawler (2021): *Tesla finally begins shipping 'Full Self-Driving' beta version 9 after a long delay.* Available at `https://www.theverge.com/2021/7/10/22570081/tesla-fsd-v9-beta-autopilot-update`.

[16] Martin Hilscher, Sven Linker & Ernst-Rüdiger Olderog (2013): *Proving Safety of Traffic Manoeuvres on Country Roads.* In Zhiming Liu, Jim Woodcock & Huibiao Zhu, editors: *Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday*, *Lecture Notes in Computer Science* 8051, Springer, pp. 196–212, doi:10.1007/978-3-642-39698-4_12.

[17] Martin Hilscher, Sven Linker, Ernst-Rüdiger Olderog & Anders P. Ravn (2011): *An Abstract Model for Proving Safety of Multi-lane Traffic Manoeuvres.* In Shengchao Qin & Zongyan Qiu, editors: *13th Int. Conference on Formal Engineering Methods, ICFEM, Proc.*, Springer, pp. 404–419, doi:10.1007/978-3-642-24559-6_28.

[18] SAE International (2018): *J 3016: Surface vehicle recommended practice – (R) Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles.*

[19] Hyunjoo Jin (2021): *Tesla tells regulator that full self-driving cars may not be achieved by year-end. Reuters.* Available at `https://www.reuters.com/business/autos-transportation/tesla-tells-regulator-that-full-self-driving-cars-may-not-be-achieved-by-year-//2021-05-07/`.

[20] Abolfazl Karimi & Parasara Sridhar Duggirala (2020): *Formalizing traffic rules for uncontrolled intersections.* In: *11th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2020, Sydney, Australia, April 21-25, 2020*, IEEE, pp. 41–50, doi:10.1109/ICCPS48487.2020.00012.

[21] UK Law Commission (2020): *Automated Vehicles: Summary of the Analysis of Responses to Consultation Paper 2 on Passenger Services and Public Transport.* Available at `https://www.lawcom.gov.uk/project/automated-vehicles/`.

[22] Ben Moszkowski (1985): *A Temporal Logic for Multilevel Reasoning About Hardware. Computer* 18(2), pp. 10–19, doi:10.1109/MC.1985.1662795.

[23] Ernst-Rüdiger Olderog & Maike Schwammberger (2017): *Formalising a Hazard Warning Communication Protocol with Timed Automata.* In Luca Aceto, Giorgio Bacci, Giovanni Bacci, Anna Ingólfsdóttir, Axel Legay & Radu Mardare, editors: *Models, Algorithms, Logics and Tools - Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday*, *Lecture Notes in Computer Science* 10460, Springer, pp. 640–660, doi:10.1007/978-3-319-63121-9_32.

[24] C. Pek, P. Zahn & M. Althoff (2017): *Verifying the safety of lane change maneuvers of self-driving vehicles based on formalized traffic rules.* In: *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1477–1483, doi:10.1109/IVS.2017.7995918.

[25] Robbel Philipp, David Wittmann, Christian Knobel, Jack Weast, Neil Garbacik, Philipp Schnetter et al. (2019): *Safety First for Automated Driving.* Available at `https://newsroom.intel.com/news/intel-auto-industry-leaders-publish-new-automated-driving-safety-framework/`.

[26] Henry Prakken (2017): *On the problem of making autonomous vehicles conform to traffic law.* *Artificial Intelligence and Law* 25(3), pp. 341–363, doi:10.1007/s10506-017-9210-0. Available at `https://link.springer.com/article/10.1007/s10506-017-9210-0`.

[27] Aarne Ranta (2011): *Translating between Language and Logic: What Is Easy and What Is Difficult.* In Nikolaj Bjørner & Viorica Sofronie-Stokkermans, editors: *Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction, Wroclaw, Poland, July 31 - August 5, 2011. Proceedings*, *Lecture Notes in Computer Science* 6803, Springer, pp. 5–25, doi:10.1007/978-3-642-22438-6_3.

[28] Albert Rizaldi, Jonas Keinholz, Monika Huber, Jochen Feldle, Fabian Immler, Matthias Althoff, Eric Hilgendorf & Tobias Nipkow (2017): *Formalising and Monitoring Traffic Rules for Autonomous Vehicles in Isabelle/HOL.* In Nadia Polikarpova & Steve A. Schneider, editors: *Integrated Formal Methods - 13th International Conference, IFM 2017, Turin, Italy, September 20-22, 2017, Proceedings*, *Lecture Notes in Computer Science* 10510, Springer, pp. 50–66, doi:10.1007/978-3-319-66845-1_4.

[29] Maike Schwammberger (2018): *An abstract model for proving safety of autonomous urban traffic.* *Theoretical Computer Science* 744, pp. 143–169, doi:10.1016/j.tcs.2018.05.028.

[30] Maike Schwammberger (2018): *Introducing Liveness into Multi-lane Spatial Logic lane change controllers using UPPAAL.* *Electronic Proceedings in Theoretical Computer Science* 269, pp. 17–31, doi:10.4204/EPTCS.269.3.

[31] Maike Schwammberger (2020): *Distributed Controllers for Provably Safe, Live and Fair Autonomous Car Manoeuvres in Urban Traffic.* Ph.D. thesis, University of Oldenburg. Available at `https://oops.uni-oldenburg.de/id/eprint/4961`.

[32] Björn Wachter & Bernd Westphal (2007): *The Spotlight Principle.* In Byron Cook & Andreas Podelski, editors: *Verification, Model Checking, and Abstract Interpretation, 8th International Conference, VMCAI 2007, Nice, France, January 14-16, 2007, Proceedings*, *Lecture Notes in Computer Science* 4349, Springer, pp. 182–198, doi:10.1007/978-3-540-69738-1_13.

[33] Jim Woodcock & Jim Davies (1996): *Using Z – Specification, Refinement, and Proof.* Prentice Hall.