Specification and Validation of Autonomous Driving Systems: A Multilevel Semantic Framework

Marius Bozga(✉) and Joseph Sifakis

Univ. Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, 38000 Grenoble, France
{Marius.Bozga,Joseph.Sifakis}@univ-grenoble-alpes.fr
http://www-verimag.imag.fr/

Abstract. Autonomous Driving Systems (ADS) are critical dynamic reconfigurable agent systems whose specification and validation raises extremely challenging problems. The paper presents a multilevel semantic framework for the specification of ADS and discusses associated validation problems. The framework relies on a formal definition of maps modeling the physical environment in which vehicles evolve. Maps are directed metric graphs whose nodes represent positions and edges represent segments of roads. We study basic properties of maps including their geometric consistency. Furthermore, we study position refinement and segment abstraction relations allowing multilevel representation from purely topological to detailed geometric. We progressively define first order logics for modeling families of maps and distributions of vehicles over maps. These are Configuration Logics, which in addition to the usual logical connectives are equipped with a coalescing operator to build configurations of models. We study their semantics and basic properties. We illustrate their use for the specification of traffic rules and scenarios characterizing sequences of scenes. We study various aspects of the validation problem including run-time verification and satisfiability of specifications. Finally, we show links of our framework with practical validation needs for ADS and advocate its adequacy for addressing the many facets of this challenge.

Keywords: Autonomous Driving System · Map modeling · Configuration logic · Traffic rule specification · Scene and scenario description · Runtime verification · Simulation and validation in the large

1 Introduction

The validation of ADS raises challenges far beyond the current state of the art because of their overwhelming complexity and the integration of non-explainable AI components. Providing sufficient evidence that these systems are safe enough

Institute of Engineering Univ. Grenoble Alpes.

J.-F. Raskin and K. Chatterjee (Eds.): Principles of Systems Design, LNCS 13660, pp. 85–106, 2022.
https://doi.org/10.1007/978-3-031-22337-2_5

is a hot and critical need, given the underlying economic and societal stakes. This objective mobilizes considerable investments and efforts by key players including big tech companies and car manufacturers. The efforts focus on the development of efficient simulation technology and common infrastructure for modelling the physical environment of ADS and their desired properties. They led in particular to the definition of common formats such as OpenDRIVE [1] for the description of road networks, and OpenSCENARIO [2] for the description of complex, synchronized maneuvers that involve multiple entities like vehicles, pedestrians and other traffic participants. Additionally, several open simulation environments such as CARLA [9] and LGSVL [23] are available for modelling and validation.

The paper proposes a semantic framework for the specification and validation of ADS. The framework provides a precise semantic model of the environment of ADS based on maps. It also includes logics for the specification and validation of properties of the semantic model and of the system dynamic behavior. Maps have been the object of numerous studies focusing on the formalization of the concept and its use for the analysis of ADS. A key research issue is to avoid monolithic representations and build maps by composition of components and heterogeneous data. This motivated formalizations using ontologies and logics with associated reasoning mechanisms to check consistency of descriptions and their correctness with respect to desired properties [3,5] or to generate scenarios [3,7]. Other works propose open source map frameworks for highly automated driving [1,19].

A different research line focuses on the validation of ADS either to verify satisfaction of safety and efficiency properties or even to check that vehicles respect given traffic rules. Many works deal with safety verification in a simple multilane setting. In [16] a dedicated Multi-Lane Spatial Logic inspired by interval temporal logic is used to specify safety and provide proofs for lane change controllers. The work in [21] presents a motion planner formally verified in Isabelle/HOL. The planner is based on manoeuver automata, a variant of hybrid automata, and properties are expressed in linear temporal logic.

Other works deal with scenarios for modeling the behavior of ADS. OpenSCENARIO [2] defines a data model and a derived file format for the description of scenarios used in driving and traffic simulators, as well as in automotive virtual development, testing and validation. The work in [8] proposes a visual formal specification language for capturing scenarios inspired from Message Charts and shows possible applications to specification and testing of autonomous vehicles. In [24] a scenario-based methodology for functional safety analysis is presented using the example of automated valet parking. The work in [14] presents an approach to automated scenario-based testing of the safety of autonomous vehicles, based on Metric Temporal Logic. Finally, the probabilistic language Scenic for the design and analysis of cyber physical systems allows the description of scenarios used to control and validate simulated systems of self-driving cars. The Scenic programming environment provides a big variety of constructs making possible modeling anywhere in the spectrum from concrete scenes to broad classes of abstract scenarios [13].

Other works focus on checking compliance of vehicles with traffic rules. A formalization of traffic rules in linear temporal logic is proposed in [11]. Runtime verification is applied to check that maneuvers of a high-level planner comply with the rules. Works in [20,22] formalize a set of traffic rules for highway scenarios in Isabelle/HOL; they show that traffic rules can be used as requirements to be met by autonomous vehicles and propose a verification procedure. A formalization of traffic rules for uncontrolled intersections is provided in [18] using the CLINGO logic programming language. Furthermore, the rules are applied by a simulator to safely control traffic across intersections. The work in [12] proposes a methodology for the formalization of traffic rules in Linear Temporal Logic; it is shown how evaluation of formalized rules on recorded drives of humans provides insight on what extent drivers respect the rules.

This work is an attempt to provide a minimal framework unifying the concepts for the specification of ADS and the associated validation problems. The proposed semantic framework clearly distinguishes between a static part consisting of the road network with its equipment and a dynamic part involving objects. We progressively introduce three logics to express properties of the semantic model at different levels. The *Metric Configuration Logic* (*MCL*) allows the compositional and parametric description of metric graphs. This is a first order logic with variables ranging over positions and segments. It uses in addition to logical connectives, a coalescing operator for the compositional construction of maps from segments. A *MCL* formula represents configurations of maps sharing a common set of locations. We discuss a specification methodology and show how various road patterns such as roundabouts, intersections, mergers of roads can be specified in *MCL*.

The *Mobile Metric Configuration Logic* (*M2CL*) is an extension of *MCL* with object variables and primitives for the specification of scenes as the distribution of objects over maps. *M2CL* formulas can be written as the conjunction of formulas describing: i) static map contexts; ii) dynamic relations between objects; iii) addressing relations between objects and maps. Last, we define *Temporal M2CL* (*TM2CL*), a linear temporal logic whose atomic propositions are formulas of *M2CL*. We illustrate the use of these logics for the specification of safety properties including traffic rules as well as the description of dynamic scenarios.

Additionally, we study the validation of properties expressed in the three logics and provide a classification of problems showing that validation of general dynamic properties boils down to constraint checking on metric graphs. Checking that a finite model satisfies a formula of *MCL* or *M2CL* amounts to eliminate quantifiers by adequate instantiation of variables. We argue that satisfiability of *M2CL* formulas can be reduced to satisfiability of *MCL* formulas which is an undecidable problem. We identify a reasonably expressive decidable subset of *MCL* and propose a decision procedure. Furthermore, we discuss the problem of runtime verification of *TM2CL* formulas and sketch a principle of solution inspired from a recent work with a similar configuration logic [10]. We complete the presentation on ADS validation with an analysis of practical needs for a rigorous validation methodology. We describe a general validation environment and show how the proposed framework provides insight into the different aspects of validation and related methodological issues.

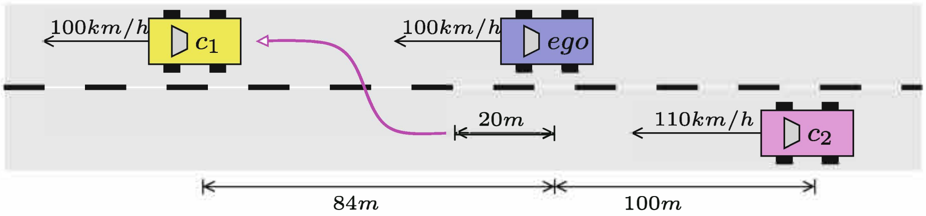

Fig. 1. A scenario example

To illustrate the specification and validation methodology based on the combined use of these three logics, let us consider a concrete example from [2] describing a scenario involving three cars moving on a two-lane road with their speeds and distances. We use *MCL* to describe the static environment in which the cars move. In this example, it is a two-lane road, but in the general case it can be a parametric map obtained by composing road segments. To describe a scene, such as the distribution of vehicles on a map, we use *M2CL* formulas. In this example, a scene is specified by the relative positions of the cars on the map and their speeds. Finally, to specify system properties, which are sequences of scenes, we use *TM2CL*. In this example, a scene sequence could be: car c_2 passes the *ego* car and moves to the right lane in front of it. The formulas in *TM2CL* can be used to specify traffic rules that must be satisfied by vehicle maneuvers.

The paper is structured as follows. In Sect. 2, we study metric graphs and their relevant properties for the representation of map models as well as the logic *MCL*, its main properties and application for map specification. Section 3 deals with the study of logics *M2CL* and *TM2CL* and their application to the specification of safety properties and the description of scenarios. Then, Sect. 4 discusses a classification of validations problems and approaches for their solution. Section 5 concludes with a summary of main results and a discussion about future developments. A long version of the paper is available in [6].

2 Metric Graphs and Metric Configuration Logic

2.1 Segments and Metric Graphs

Segments. We build contiguous road segments from a set $\mathcal{S}$ equipped with a partial concatenation operator $\cdot : \mathcal{S} \times \mathcal{S} \to \mathcal{S} \cup \{\bot\}$ and a length norm $\|.\| : \mathcal{S} \to \mathbb{R}_{\geq 0}$ satisfying the following properties:

(i) *associativity:* for any segments s_1, s_2, s_3 either both $(s_1 \cdot s_2) \cdot s_3$ and $s_1 \cdot (s_2 \cdot s_3)$ are defined and equal, or both undefined;
(ii) *length additivity wrt concatenation:* for any segments s_1, s_2 whenever $s_1 \cdot s_2$ defined it holds $\|s_1 \cdot s_2\| = \|s_1\| + \|s_2\|$;
(iii) *segment split:* for any segment s and non-negative a_1, a_2 such that $\|s\| = a_1 + a_2$ there exist unique s_1, s_2 such that $s = s_1 \cdot s_2$, $\|s_1\| = a_1$, $\|s_2\| = a_2$.

The last property allows us to define consistently a subsegment operation: $s[a_1, a_2]$ is the unique segment of length $a_2 - a_1$ satisfying $s = s_1 \cdot s[a_1, a_2] \cdot s_2$ where s_1, s_2 are such that $\|s_1\| = a_1$, $\|s_2\| = \|s\| - a_2$, for any $0 \le a_1 \le a_2 \le \|s\|$. For brevity, we use the shorthand notation $s[a, \text{-}]$ to denote the subsegment $s[a, \|s\|]$. Moreover, we define $s_1 \preccurlyeq s_2$ iff $s_1 = s_2[0, a]$ for some non-negative a.

Segments will be used to model building blocks of roads in maps considering three different interpretations. Interval segments simply define the length of a segment. Curve segments define the precise geometric form of the trajectory of a mobile object along the segment. Region segments are 2D-regions of given width around a center curve segment.

Interval Segments. Consider $\mathcal{S}_{interval} \stackrel{def}{=} \{[0, a] \mid a \in \mathbb{R}_{\ge 0}\}$, that is, the set of closed intervals on reals with lower bound 0, concatenation defined by $[0, a_1] \cdot [0, a_2] \stackrel{def}{=} [0, a_1 + a_2]$ and length $\|[0, a]\| \stackrel{def}{=} a$.

Curve Segments. Consider $\mathcal{S}_{curve} \stackrel{def}{=} \{c : [0, 1] \to \mathbb{R}^2 \mid c(0) = (0, 0),\ c \text{ curve}\} \cup \{\epsilon\}$ that is, the set of curves that are continuous smooth[1] and uniformly progressing[2] functions c, starting at the origin, plus a designated single point curve ϵ. The length is defined by taking respectively the length of the curve $\|c\| \stackrel{def}{=} \int_0^1 |\dot{c}(t)| dt$ and $\|\epsilon\| = 0$. The concatenation $c_1 \cdot c_2$ of two curves c_1 and c_2 is a partial operation that consists in joining the final endpoint of c_1 with the initial endpoint of c_2 provided the slopes at these points are equal. This condition preserves smoothness of the curve $c_1 \cdot c_2$ defined by $c_1 \cdot c_2 : [0, 1] \to \mathbb{R}^2$ where:

$$(c_1 \cdot c_2)(t) \stackrel{def}{=} \begin{cases} c_1(\frac{t}{\lambda}) \text{ if } t \in [0, \lambda] \\ c_1(1) + c_2(\frac{t-\lambda}{1-\lambda}) \text{ if } t \in [\lambda, 1] \end{cases} \text{ where } \lambda = \frac{\|c_1\|}{\|c_1\| + \|c_2\|}$$

Note that in this definition, c_1 and c_2 are scaled on sub-intervals of $[0, 1]$ respecting their length ratio. We additionally take $c \cdot \epsilon \stackrel{def}{=} \epsilon \cdot c \stackrel{def}{=} c$, for any c. For practical reasons, one can further restrict the set $\mathcal{S}_{curve}$ to curves of some form e.g., finite concatenation of parametric line segments and circle arcs. That is, for any $a, r \in \mathbb{R}^*_{\ge 0}$, $\varphi \in \mathbb{R}$, $\theta \in \mathbb{R}^*$ the curves $line[a, \varphi]$, $arc[r, \varphi, \theta]$ are defined as

$$line[a, \varphi](t) \stackrel{def}{=} (at \cos\varphi, at \sin\varphi) \quad \forall t \in [0, 1]$$
$$arc[r, \varphi, \theta](t) \stackrel{def}{=} (r(\sin(\varphi + t\theta) - \sin\varphi), r(-\cos(\varphi + t\theta) + \cos\varphi)) \quad \forall t \in [0, 1]$$

Note that a and r are respectively the length of the line and the radius of the arc, φ is the slope of the curve at the initial endpoint and θ is the degree of the arc. Figure 2 illustrates the composition of three segments of this parametric form.

---

[1] the derivative $\dot{c}$ exists and is continuous on $[0, 1]$.

[2] the instantaneous speed $|\dot{c}|$, that is, the Euclidean norm of the derivative is constant.

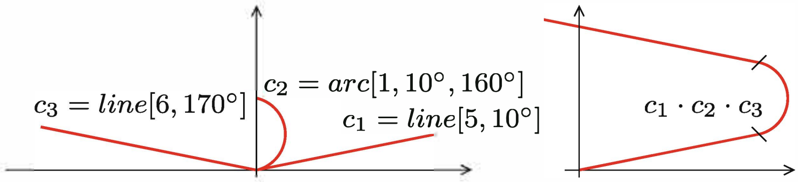

Fig. 2. Curve segments and their composition

Region Segments. Consider $\mathcal{S}_{region} \stackrel{def}{=} \mathcal{S}_{curve} \times \mathbb{R}^*_{\geq 0}$, that is, the set of pairs (c, w) where c is a curve and w a positive number, denoting respectively the region center curve and the region width. Region segments can be concatenated iff their curves can be concatenated and if their widths are equal, that is, $(c_1, w) \cdot (c_2, w) \stackrel{def}{=} (c_1 \cdot c_2, w)$ if $c_1 \cdot c_2 \neq \bot$. The length of a region segment is defined as the length of its center curve, $\|(c, w)\| \stackrel{def}{=} \|c\|$.

Region segments can be equally understood as sets of points in $\mathbb{R}^2$ defined by algebraic constraints. More precisely, for any curve c and width w the region segment (c, w) corresponds to the subset of $\mathbb{R}^2$ defined as $\{c(t) + \lambda \cdot \frac{ortho(\dot{c}(t))}{|\dot{c}(t)|} \mid t \in [0, 1],\ \lambda \in [-\frac{w}{2}, \frac{w}{2}]\}$ where *ortho* is the orthogonal operator on $\mathbb{R}^2$ defined as $ortho((a, b)) \stackrel{def}{=} (-b, a)$. In particular, the region generated by the curve $line[a, \varphi]$ is a rectangle containing the set of points $\{(at \cos\varphi - \lambda \sin\varphi, at \sin\varphi + \lambda \cos\varphi) \mid t \in [0, 1], \lambda \in [-\frac{w}{2}, \frac{w}{2}]\}$. The region generated by the curve $arc[r, \varphi, \theta]$ is a ring sector containing the set of points $\{((r + \lambda)(\sin(\varphi + t\theta) - r \sin\varphi, -(r + \lambda)\cos(\varphi + t\theta) + r\cos\varphi) \mid t \in [0, 1], \lambda \in [-\frac{w}{2}, \frac{w}{2}]\}$.

Metric Graphs. We use metric graphs $G \stackrel{def}{=} (V, \mathcal{S}, E)$ to represent maps, where V is a finite set of *vertices*, $\mathcal{S}$ is a set of segments and $E \subseteq V \times \mathcal{S}^\star \times V$ is a finite set of *edges* labeled by non-zero length segments in $\mathcal{S}^\star$. We also denote an edge $e = (v, s, v') \in E$ by $v \xrightarrow{s}_G v'$ and we define ${}^\bullet e \stackrel{def}{=} v$, $e^\bullet \stackrel{def}{=} v'$, $e.s \stackrel{def}{=} s$. For a vertex v, we define ${}^\bullet v \stackrel{def}{=} \{e | e^\bullet = v\}$ and $v^\bullet \stackrel{def}{=} \{e | {}^\bullet e = v\}$. We denote by E^+_{ac} the finite set of non-empty *acyclic*[3] directed paths with edges from E. We call a metric graph *strongly* (resp. *weakly*) connected if a *directed* (resp. *undirected*) path exists between any pair of vertices. A metric graph is called *acyclic* if at most one path, directed or undirected, exist between any pairs of vertices.

We consider the set $Pos_G \stackrel{def}{=} V \cup \{(e, a) \mid e \in E,\ 0 < a < \|e.s\|\}$ of *positions* defined by a metric graph. Note that $(e, 0)$ and $(e, \|e.s\|)$ are respectively the positions ${}^\bullet e$ and $e^\bullet$. Moreover, a s-labelled *ride* between positions (e, a) and (e', a') is an acyclic path denoted by $(e, a) \overset{s}{\rightsquigarrow}_G (e', a')$ and defined as follows:

(i) $e = e'$, $0 \leq a \leq a' \leq \|e.s\|$, $s = e.s[a, a']$
(ii) $e = e'$, $0 \leq a' \leq a \leq \|e.s\|$, $e^\bullet = {}^\bullet e$, $s = e.s[a, \text{-}] \cdot e.s[0, a'] \neq \bot$

[3] every edge occurs at most once in the path.

(iii) $e = e'$, $0 \le a' \le a \le \|e.s\|$, $w \in E_{ac}^{+}$, $e \notin w$, $e^{\bullet} = {}^{\bullet}w$, $w^{\bullet} = {}^{\bullet}e$, $s = e.s[a, \text{-}] \cdot w.s \cdot e.s[0, a'] \neq \bot$
(iv) $e \neq e'$, $e^{\bullet} = {}^{\bullet}e'$, $s = e.s[a, \text{-}] \cdot e'.s[0, a'] \neq \bot$
(v) $e \neq e'$, $w \in E_{ac}^{+}$, $e, e' \notin w$, $e^{\bullet} = {}^{\bullet}w$, $w^{\bullet} = {}^{\bullet}e'$, $s = e.s[a, \text{-}] \cdot w.s \cdot e'.s[0, a'] \neq \bot$

Figure 3 illustrates the five cases of the above definition for a simple graph with segments s_1, s_2 and s_3. Cases (i) and (ii) correspond to rides on the same segment. Case (iii) corresponds to rides originating and terminating in fragments of the same segment and also involving other segments between them. Finally cases (iv) and (v) are rides originating and terminating at different segments.

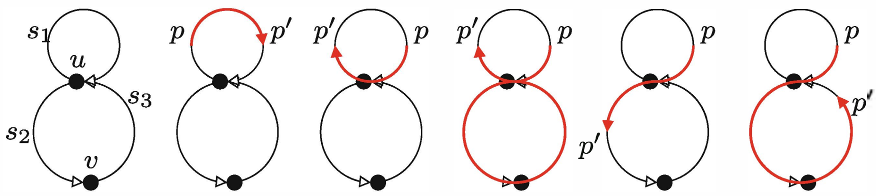

Fig. 3. Rides in metric graphs - cases (i)–(v) illustrated

We define the distance d_G between positions p, p' as 0 whenever $p = p'$ or the minimum length among all segments labeling rides from p to p' and otherwise $+\infty$ if no such ride exists. It can be checked that d_G is an *extended quasi-metric* on the set Pos_G and therefore, (Pos_G, d_G) is an extended quasi-metric space.

2.2 Properties of Metric Graphs

Contraction/Refinement. A metric graph $G' = (V', \mathcal{S}, E')$ is a *contraction* of a metric graph $G = (V, \mathcal{S}, E)$ (or dually, G is a *refinement* of G'), denoted by $G \sqsubseteq G'$, iff G is obtained from G' by transformations replacing some of its edges e by acyclic sequences of interconnected edges $e_1 e_2 \ldots e_n$ while preserving the segment labeling i.e., $e.s = e_1.s \cdot e_2.s \cdot \ldots \cdot e_n.s$. In Fig. 4, the graph on the right is a contraction of the one on the left iff $s_{12} = s_{14} \cdot s_{45} \cdot s_{52}$, $s'_{12} = s'_{16} \cdot s'_{62}$ and $s_{31} = s_{37} \cdot s_{78} \cdot s_{81}$.

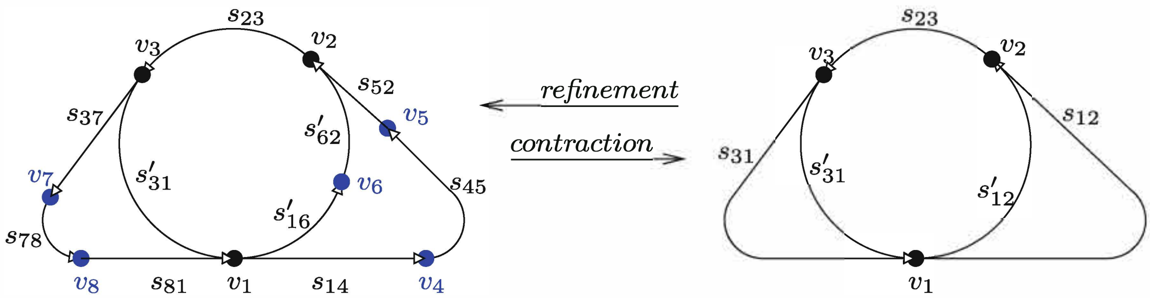

Fig. 4. Illustration of contraction/refinement on metric graphs

Note that metric graphs where all vertices have input or output degree greater than one cannot be contracted. Such vertices correspond to *junctions* (confluence of divergence of roads) when metric graphs represent maps. The following proposition states some key properties on contraction/refinement of metric graphs.

Proposition 1. *Let* $Con(G) \stackrel{def}{=} \{G' \mid G \sqsubseteq G'\}$, $Ref(G) \stackrel{def}{=} \{G' \mid G' \sqsubseteq G\}$ *be respectively the set of contractions, refinements of a metric graph* G.

(i) the refinement relation $\sqsubseteq$ *is a partial order on the set of metric graphs;*
(ii) for any metric graph G, *both* $(Con(G), \sqsubseteq)$ *and* $(Ref(G), \sqsubseteq)$ *are complete lattices, moreover,* $(Con(G), \sqsubseteq)$ *is finite;*
(iii) for any metric graphs G, G' *if* $G \sqsubseteq G'$ *then (1) the labelled transition systems* $(Pos_G, \mathcal{S}, \rightsquigarrow_G)$ *and* $(Pos_{G'}, \mathcal{S}, \rightsquigarrow_{G'})$ *are strongly bisimilar and (2) the quasi-metric spaces* (Pos_G, d_G) *and* $(Pos_{G'}, d_{G'})$ *are isometric;*

Abstraction/Concretization. Consider $\mathcal{S}$, $\mathcal{S}'$ as sets of segments associated with respectively concatenation $\cdot$, $\cdot'$, and length norm $\|.\|$, $\|.\|'$. A function $\alpha : \mathcal{S} \rightarrow \mathcal{S}'$ is a *segment abstraction* if it satisfies the following properties: (i) *length preservation*: $\|s\| = \|\alpha(s)\|'$, forall $s \in \mathcal{S}$ (ii) *homomorphism wrt concatenation*: $\alpha(s_1 \cdot s_2) = \alpha(s_1) \cdot' \alpha(s_2)$ for all $s_1, s_2 \in \mathcal{S}$ such that $s_1 \cdot s_2 \neq \bot$.

For example, the function $\alpha^{CI} : \mathcal{S}_{curve} \rightarrow \mathcal{S}_{interval}$ defined by $\alpha^{CI}(s) \stackrel{def}{=} [0, \|s\|]$ for all $s \in \mathcal{S}_{curve}$ is a an abstraction of curve segments as interval segments. Similarly, the function $\alpha^{RC} : \mathcal{S}_{region} \rightarrow \mathcal{S}_{curve}$ defined by $\alpha^{RC}((s, w)) \stackrel{def}{=} s$ for all $(s, w) \in \mathcal{S}_{region}$ is an abstraction of region segments as curve segments.

Dually, we can define concretization functions γ that go from intervals to curves, and from curves to regions. For example, for any angles φ, θ consider $\gamma^{IC}_{\varphi,\theta} : \mathcal{S}_{interval} \rightarrow \mathcal{S}_{curve}$ where respectively, $\gamma^{IC}_{\varphi,\theta}([0, a]) \stackrel{def}{=} arc[\frac{a}{\theta}, \varphi, \theta]$ if $\theta \neq 0$ or $\gamma^{IC}_{\varphi,\theta}([0, a]) \stackrel{def}{=} line[a, \varphi]$ if $\theta = 0$. Or, for any positive real w consider $\gamma^{CR}_w : \mathcal{S}_{curve} \rightarrow \mathcal{S}_{regions}$ where $\gamma^{CR}_w(s) \stackrel{def}{=} (s, w)$.

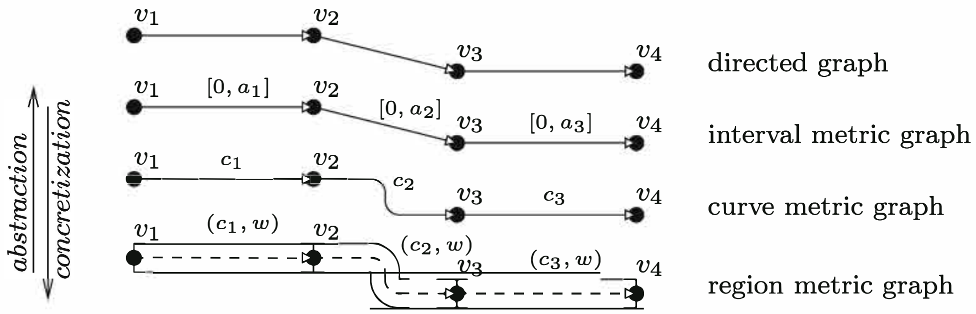

Fig. 5. Illustration of abstraction/concretization on metric graphs

Given a segment abstraction $\alpha : \mathcal{S} \rightarrow \mathcal{S}'$, a metric graph $G' = (V, \mathcal{S}', E')$ is an *α-abstraction* of a metric graph $G = (V, \mathcal{S}, E)$, denoted by $G' = \alpha(G)$, iff G' is obtained from G by replacing segments s by their abstractions $\alpha(s)$. That is, any edge $u \xrightarrow{s}_G v$ is transformed into an edge $u \xrightarrow{\alpha(s)}_{G'} v$. In a similar way, γ-concretization on metric graphs is defined for a segment concretization $\gamma : \mathcal{S}' \to \mathcal{S}$. Figure 5 illustrates the use of the three segment abstraction levels (respectively as intervals, curves, regions) and their associated metric graphs. Interval metric graphs are α^{CI}-abstractions of curve metric graphs, which in turn are α^{RC}-abstractions of region metric graphs. Propositions 2 and 3 state some key properties on abstraction on metric graphs.

Proposition 2. *For a segment abstraction $\alpha : \mathcal{S} \to \mathcal{S}'$ and metric graphs G, G' such that $G' = \alpha(G)$, the labelled transition system $(Pos_{G'}, \mathcal{S}', \rightsquigarrow_{G'})$ simulates the labelled transition system $(Pos_G, \mathcal{S}, \rightsquigarrow_G)$ renamed by α.*

Proposition 3. *Contraction and abstraction commute, that is, for any metric graphs G, G', for any segment abstraction α, if $G \sqsubseteq G'$ then $\alpha(G) \sqsubseteq \alpha(G')$.*

2.3 The Metric Configuration Logic

Syntax. Let consider a fixed set of segments $\mathcal{S}$ and assume there exists a finite set $\mathcal{S}^T$ of segment constructors s^T (or segment types), that is, partial functions $s^T : \mathbb{R}^m \to \mathcal{S}^\perp$ for some natural m. For example, we can take $\mathcal{S}^T_{curve} = \{line : \mathbb{R}^2 \to \mathcal{S}^\perp, arc : \mathbb{R}^3 \to \mathcal{S}^\perp\}$ as the set of constructor curve segments $\mathcal{S}_{curve}$.

Let K, Z, X be distinct finite sets of *variables* denoting respectively reals, segments and vertices of a metric graph. The syntax of the *metric configuration logic* (MCL) is defined in Table 1.

Table 1. *MCL* Syntax

$t ::= a \in \mathbb{R} \mid k \in K \mid t + t \mid t \cdot t$	*arithmetic terms*
$\psi_K ::= t \leq t'$	*arithmetic constraints*
$s ::= s^T(t_1, ..., t_m) \mid z \in Z \mid s \cdot s$	*segment terms*
$\psi_S ::= s = s' \mid s \preccurlyeq s' \mid \lVert s \rVert = t$	*segment constraints*
$p ::= x \in X \mid (x, s, t) \mid (t, s, x)$	*position terms*
$\psi_G ::= x \xrightarrow{s} x' \mid p = p' \mid p \overset{s}{\rightsquigarrow} p' \mid d(p, p') = t$	*position constraints*
$\phi ::= \psi_K \mid \psi_S \mid \psi_G$	*atomic formula*
$\mid\ \phi \oplus \phi \mid \phi \vee \phi \mid \neg\phi$	*non-atomic formula*
$\mid\ \exists k.\ \phi(k) \mid \exists z.\ \phi(z) \mid \exists x.\ \phi(x)$	*quantifiers*

Semantics. Let $G = (V, \mathcal{S}, E)$ be a metric graph fixed in the context, and let σ be an assignment of variables K, Z, X to respectively reals $\mathbb{R}$, segments $\mathcal{S}$, vertices V. As usual, we extend σ for evaluation of arithmetic terms (with variables from K) into reals. Moreover, we extend σ for the partial evaluation of segment terms (with variables from Z) and position terms (with variables from Z and X) into respectively segments $\mathcal{S}$ and positions Pos_G as defined by the rules in Table 2.

Table 2. Evaluation of *MCL* terms

$\sigma s^T(t_1, ..., t_m) \stackrel{def}{=} s^T(\sigma t_1, ..., \sigma t_m)$	$\sigma(x, s, t) \stackrel{def}{=} pos_G^{fwd}(\sigma x, \sigma s, \sigma t)$
$\sigma\ s \cdot s' \stackrel{def}{=} \sigma s \cdot \sigma s'$	$\sigma(t, s, x) \stackrel{def}{=} pos_G^{bwd}(\sigma x, \sigma s, \sigma t)$
where $pos_G^{fwd}, pos_G^{bwd} : V \times \mathcal{S} \times \mathbb{R} \to Pos_G^{\perp}$ are defined as	
$pos_G^{fwd}(v, s, a) \stackrel{def}{=} (e, a)$	only if $\exists!\ e = (v, s, v') \in E,\ 0 < a < \|s\|$
$pos_G^{bwd}(v, s, a) \stackrel{def}{=} (e, \|s\| - a)$	only if $\exists!\ e = (v', s, v) \in E,\ 0 < a < \|s\|$

We tacitly restrict to terms which evaluate successfully in their respective domains. The semantics of *MCL* is defined by the rules in Table 3. Note that a formula represents a configuration of metric graphs sharing common characteristics. Besides the logic connectives with the usual set-theoretic meaning, the coalescing operator $\oplus$ allows building graphs by grouping elementary constituents characterized by atomic formulas relating positions via segments. Hence, the formula $\phi_1 \oplus \phi_2$ represents the graph configurations obtained as the union of configurations satisfying ϕ_1 and ϕ_2 respectively. It differs from $\phi_1 \vee \phi_2$ in that this formula satisfies configurations that satisfy either ϕ_1 or ϕ_2.

Table 3. *MCL* Semantics

$\sigma, G \models t \leq t'$	iff	$\sigma t \leq \sigma t'$
$\sigma, G \models s = s'$	iff	$\sigma s = \sigma s'$
$\sigma, G \models s \preccurlyeq s'$	iff	$\sigma s \preccurlyeq \sigma s'$
$\sigma, G \models \|s\| = t$	iff	$\|\sigma s\| = \sigma t$
$\sigma, G \models x \xrightarrow{s} x'$	iff	$E = \{(\sigma x, \sigma s, \sigma x')\}$
$\sigma, G \models p = p'$	iff	$\sigma p = \sigma p'$
$\sigma, G \models p \overset{s}{\rightsquigarrow} p'$	iff	$\sigma p \overset{\sigma s}{\rightsquigarrow}_G \sigma p'$
$\sigma, G \models d(p, p') = t$	iff	$d_G(\sigma p, \sigma p') = \sigma t$
$\sigma, G \models \phi_1 \oplus \phi_2$	iff	$\sigma, (V, E_1) \models \phi_1$ and $\sigma, (V, E_2) \models \phi_2$ for some E_1, E_2 such that $E_1 \cup E_2 = E$
$\sigma, G \models \phi_1 \vee \phi_2$	iff	$\sigma, G \models \phi_1$ or $\sigma, G \models \phi_2$
$\sigma, G \models \neg\phi$	iff	$\sigma, G \not\models \phi$
$\sigma, G \models \exists k.\ \phi$	iff	$\sigma[k \mapsto a], G \models \phi$ for some $a \in \mathbb{R}$
$\sigma, G \models \exists z.\ \phi$	iff	$\sigma[z \mapsto s], G \models \phi$ for some $s \in \mathcal{S}$
$\sigma, G \models \exists x.\ \phi$	iff	$\sigma[x \mapsto v], G \models \phi$ for some $v \in V$

Properties. Table 4 provides a set of theorems giving insight into the characteristic properties of the logic. Theorems $(A.i)$–$(A.v)$ illustrate important properties of the $\oplus$ operator that is associative and commutative but not idempotent. As explained below, of particular interest for writing specifications are formulas of the form $\sim\phi \stackrel{def}{=} \phi \oplus true$. These are satisfied by configurations with graphs that contain a subgraph satisfying ϕ. Hence, while the formula $x \xrightarrow{s} x'$ characterizes the graphs with two vertices and a single edge labeled by s, the formula

Table 4. *MCL* Theorems

$(A.i)$	$(\phi_1 \oplus \phi_2) \oplus \phi_3$	$\equiv$	$\phi_1 \oplus (\phi_2 \oplus \phi_3)$
$(A.ii)$	$\phi_1 \oplus \phi_2$	$\equiv$	$\phi_2 \oplus \phi_1$
$(A.iii)$	$\phi \oplus \mathit{false}$	$\equiv$	false
$(A.iv)$	$\phi \oplus \phi$	$\not\equiv$	ϕ $(\mathit{in\ general})$
$(A.v)$	$\phi_1 \oplus (\phi_2 \vee \phi_3)$	$\equiv$	$(\phi_1 \oplus \phi_2) \vee (\phi_1 \oplus \phi_3)$
$(B.i)$	$\sim\sim\phi$	$\equiv$	$\sim\phi$
$(B.ii)$	ϕ	$\Longrightarrow$	$\sim\phi$
$(B.iii)$	$\sim(\phi_1 \vee \phi_2)$	$\equiv$	$\sim\phi_1 \vee \sim\phi_2$
$(B.iv)$	$\sim(\phi_1 \oplus \phi_2)$	$\equiv$	$\sim\phi_1 \oplus \sim\phi_2 \quad \equiv \quad \sim\phi_1 \wedge \sim\phi_2$
$(C.i)$	$x \xrightarrow{s} x' \wedge (\phi_1 \oplus \phi_2)$	$\equiv$	$(x \xrightarrow{s} x' \wedge \phi_1) \oplus (x \xrightarrow{s} x' \wedge \phi_2)$
$(C.ii)$	true	$\equiv$	$(x \xrightarrow{s} x' \oplus \neg(\sim x \xrightarrow{s} x')) \vee \neg(\sim x \xrightarrow{s} x')$
$(D.i)$	$d(p,p') = t \wedge p \overset{s}{\rightsquigarrow} p'$	$\Longrightarrow$	$t \leq \|s\|$
$(D.ii)$	$d(p,p') = t \wedge d(p',p'') = t'$	$\Longrightarrow$	$\exists k.\ d(p,p'') = k \wedge k \leq t + t'$

$\sim x \xrightarrow{s} x'$ characterizes the set of graphs containing such an edge. Thus $\sim$ is a closure operator which moreover satisfies theorems $(B.i)$–$(B.iv)$. Finally, theorems $(C.i)$–$(C.ii)$ relate the atomic formula $x \xrightarrow{s} x'$ to coalescing and the complement of their closure. The two last theorems $(D.i)$–$(D.ii)$ differ from the others in that they express specific properties of segment and position constraints.

Proposition 4. *Position constraints not involving edge constraints of the form $x \xrightarrow{s} x'$ are insensitive to metric graph contraction and refinement.*

Note that stronger preservation results for (even simple fragments of) *MCL* are hard to obtain because the domain of vertex variables is a fixed set of vertices. This makes *MCL* sensitive to both contraction and refinement. For example, the formula $\exists x.\ \exists y.\ x \overset{s}{\rightsquigarrow} y$ may not hold before and hold after refinement i.e., if a pair of vertices u, v satisfying the constraint is added by refinement.

We provide below abstraction preservation results for MCL formulas. Any segment abstraction $\alpha : \mathcal{S} \to \mathcal{S}'$ can be lifted to segment terms by taking respectively $\alpha(s^T(t_1, ..., t_m)) \overset{def}{=} (\alpha s^T)(t_1, ..., t_n)$, $\alpha(s_1 \cdot s_2) \overset{def}{=} \alpha(s_1) \cdot' \alpha(s_2)$, $\alpha(z) \overset{def}{=} z$. Moreover, α can be further lifted to *MCL* formulas on $\mathcal{S}$. We denote by $\alpha(\phi)$ the *MCL* formula on $\mathcal{S}'$ obtained by rewriting all the segment terms s occurring in ϕ by $\alpha(s)$. The following proposition relates abstractions on formulas to abstractions on metric graphs.

Proposition 5. *Let ϕ be an existential positive MCL formula. Then $G \models \phi$ implies $\alpha(G) \models \alpha(\phi)$ whenever:*

(i) ϕ does not contain distance constraints or
(ii) for any connected edges e_1, e_2 such that $e_1^\bullet = {}^\bullet e_2$ their segments compose, that is, $e_1.s \cdot e_2.s \neq \bot$.

3 ADS Specification

The results of the previous section provide a basis for the definition of both a dynamic model for ADS and of logics for the expression of their properties. The model is a timed transition system with states defined as the distribution of objects over of a metric graph representing a map. Objects may be mobile such as vehicles and pedestrians or static such as signaling equipment. The logics are two extensions of *MCL*, one for the specification of predicates representing sets of states and the other for the specification of its behavior.

We introduce first the concept of map and its properties. Then we define the dynamic model and the associated logics. Finally, we discuss the validation problem and its possible solutions.

3.1 Map Specification

A weakly connected metric graph $G = (V, \mathcal{S}, E)$ can be interpreted as a map with a set of roads R and a set of junctions J, defined in the following manner:

- a *road* r of G is a maximal directed path $r = v_0 \xrightarrow{s_1}_G v_1$, $v_1 \xrightarrow{s_2}_G v_2$, ..., $v_{n-1} \xrightarrow{s_n}_G v_n$ where all the vertices v_1, ..., v_{n-1} have indegree and outdegree equal to one. We say that v_0 is the *entrance* and v_n is the *exit* of r. Let $R = \{r_i\}_{i \in I}$ be the set of roads of G.
- a junction j of G is any maximal weakly connected sub-graph G' of G, obtained from G by removing from its roads all the vertices (and connecting edges) except their entrances and exits. Note that for a junction, its set of vertices of indegree (resp. outdegree) one are exits (resp. entrances) of some roads. Let $J = \{j_\ell\}_{\ell \in L}$ be the set of junctions of G.

Note that G is the union of the subgraphs representing its roads and junctions. For every junction, the strong connectivity of G implies that from any entrance there exists at least one path leading to an exit. Additionally, we assume that maps include information about features of roads, junctions that are relevant to traffic regulations:

- roads and junctions are *typed*: road types can be highway, built-up area roads, carriage roads, etc. Junctions types can be roundabouts, crossroads, highway exit, highway entrance, etc. We use standard notation associating a road or junction to its type e.g., $r : \mathit{highway}$, $j : \mathit{roundabout}$.
- roads, junctions and their segments have *attributes*. We use the dot notation $a.x$ and $a.X$ to denote respectively the attribute x or the set of attributes X of a. In particular, we denote by $r.en$ and $r.ex$ respectively the entrance and the exit of a road r and by $j.En$ and $j.Ex$ the sets of entrances and exits of a junction j. Similarly, $r.lanes$ is the number of lanes of the road r.

Note that contraction and refinement transform maps into maps. A road may be refined into a road while a junction may be decomposed into a set of roads and junctions. Furthermore, abstraction and concretization transform maps into maps as they preserve their connectivity.

Given a map with sets of roads and junctions R and J respectively, it is possible to derive compositionally its bottom-up and top-down specifications. We show first how we can get formulas ζ_j, ζ_r and ξ_j, ξ_r for the bottom-up and top-down specifications of j and r, respectively. Let us consider the junctions illustrated in Fig. 6:

- if ra is a roundabout with n entrances $ra.En = \{en_k\}_{k\in[1,n]}$ alternating with n exits $ra.Ex = \{ex_k\}_{k\in[1,n]}$ then its bottom-up specification is $\zeta_{ra} \stackrel{def}{=} \bigoplus_{k=1}^{n} \zeta_k \oplus \bigoplus_{k=1}^{n} \zeta_{k,k+1}$, where $\zeta_k \stackrel{def}{=} ex_k \xrightarrow{s_k} en_k$ and $\zeta_{k,k+1} \stackrel{def}{=} en_k \xrightarrow{s_{k,k+1}} ex_{k+1}$. The top-down specification is $\xi_{ra} \stackrel{def}{=} \bigwedge_{k=1}^{n} \xi_k \wedge \bigwedge_{k=1}^{n} \xi_{k,k+1}$ where $\xi_k \stackrel{def}{=} \sim\zeta_k$ and $\xi_{k,k+1} \stackrel{def}{=} \sim\zeta_{k,k+1}$.
- if in is an intersection with n entrances $in.En = \{en_k\}_{k=1,n}$ and n exits $in.Ex = \{ex_k\}_{k\in[1,n]}$ then its bottom-up specification is $\zeta_{in} \stackrel{def}{=} \bigoplus_{k=1}^{n} \zeta_k$ with $\zeta_k \stackrel{def}{=} \bigoplus_{j\in J_k} en_k \xrightarrow{s_{k,j}} ex_j$ and J_k is the set of indices of the exits of $j.Ex$ connected to the entrance en_k. Hence, the top-down specification is $\xi_{in} \stackrel{def}{=} \bigwedge_{k=1}^{n} \xi_k$ where $\xi_k \stackrel{def}{=} \sim\zeta_k$.
- the formulas for a merger mg and a fork fk with respectively n entrances and n exits and unique exit and entrance respectively, can be obtained as a particular case of an intersection.
- finally, for a road r the specifications are $\xi_r \stackrel{def}{=} \sim\zeta_r$ with $\zeta_r \stackrel{def}{=} r.en \xrightarrow{s_r} r.ex$.

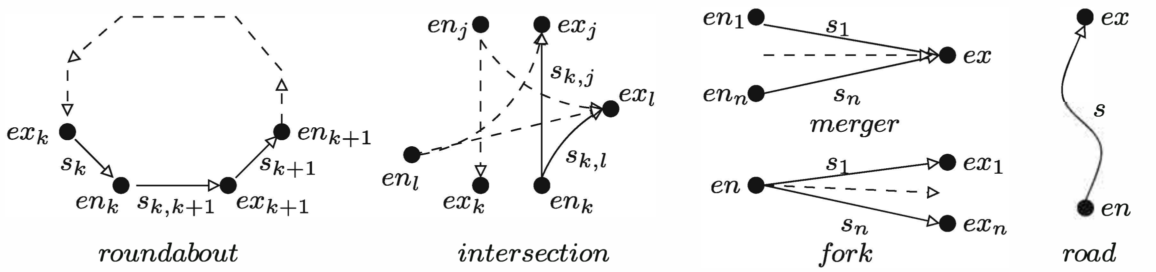

Fig. 6. Junctions and roads

3.2 Mobile *MCL* and Scenario Description for ADS

Mobile MCL (shorthand *M2CL*) is an extension of *MCL* for the specification of states of dynamic ADS models as distributions of objects over maps. Given a metric graph G representing a map, the state of an ADS is a tuple $\mathbf{s} \stackrel{def}{=} \langle \mathbf{s}_o \rangle_{o\in\mathcal{O}}$ representing the distribution of a finite set of objects $\mathcal{O}$ with their relevant dynamic attributes on the map G. The set of objects $\mathcal{O}$ includes a set of vehicles $\mathcal{C}$ and sets of immobile equipment such as lights, road signs, gates, etc.

For a vehicle c, its state $\mathbf{s}_c \stackrel{def}{=} \langle it, pos, sp, wt, ln, ...\rangle$ includes respectively its *itinerary* (from the set of segments $\mathcal{S}$), its *position* on the map (from Pos_G), its *speed* (from $\mathbb{R}_{\geq 0}$), the *waiting time* (from $\mathbb{R}_{\geq 0}$) which is the time elapsed since the speed of c became zero, the *lane* it is traveling (from $\mathbb{R}_{\geq 0}$), etc. For a traffic light lt, its state $\mathbf{s}_{lt} \stackrel{def}{=} \langle pos, cl, ...\rangle$ includes respectively its *position* on the map (from Pos_G), and its *color* (with values *red* and *green*), etc. For a map G and an initial state $\mathbf{s}^{(t_0)}$ we define a *run* as a sequence of consecutive states $[\mathbf{s}^{(t_i)}]_{i\geq 0}$ parameterized by an increasing sequence of time points $t_i \in \mathbb{R}_{\geq 0}$, equal to the sum of the time intervals elapsed for reaching the i-th state.

M2CL is equipped with object variables Y with attributes allowing to express constraints on object states. Object variables in Y are typed and denote objects from a finite set $\mathcal{O}$. Constraints are obtained by extending the syntax of *MCL* to include object attribute terms. For example, if y is a "vehicle" variable then $y.it$ is a segment term, $y.pos$ is a position term, and $y.ln$, $y.sp$, $y.wt$ are arithmetic terms of *M2CL*. Moreover, *M2CL* allows for equality $y = y'$ and existential quantification $\exists y$ of object variables.

The semantics of *M2CL* formulas is defined on distributions $\langle \sigma, G, \mathbf{s}\rangle$ where σ provides an interpretation of variables (including object variables) to their respective domains, G is a metric graph representing the map, and $\mathbf{s}$ is the system state vector for objects in $\mathcal{O}$. The evaluation of terms is extended to include object attributes, that is, for any object variable y with attribute *attr* we define $\sigma\ y.attr \stackrel{def}{=} \mathbf{s}_{\sigma y}(attr)$. Equality and existential elimination on objects variables are interpreted with the usual meaning, that is, $y = y'$ holds on $\langle \sigma, G, \mathbf{s}\rangle$ iff $\sigma y = \sigma y'$ and respectively $\exists y.\ \psi$ holds on $\langle \sigma, G, \mathbf{s}\rangle$ iff ψ holds on $\langle \sigma[y \mapsto o], G, \mathbf{s}\rangle$ for some object $o \in \mathcal{O}$.

From a methodological point of view, we restrict to *M2CL* formulas that can be written as boolean combinations of three categories of sub-formulas:

(i) ψ_{map} describing map specifications characterizing the static environment in which a dynamic system evolves,
(ii) ψ_{dyn} describing relations between distributions of the objects of a dynamic system,
(iii) ψ_{add} linking itinerary attributes of objects involved in ψ_{dyn} to position addresses of maps described by ψ_{map}.

The following set of primitives used respectively in sub-formulas of the above categories is needed to express ADS scenarios and specifications:

(i) for x, x' vertex variables, X set of vertex variables, $[x\ \textit{right-of}\ x'\ \textit{in}\ X]$, $[x\ \textit{opposite}\ x'\ \textit{in}\ X]$ express constraints on the positioning of x, x' with respect to the map restricted to vertices in X (typically a junction):
$$[x\ \textit{right-of}\ x'\ \textit{in}\ X] \stackrel{def}{=} \exists a.\exists r.\exists \varphi.\exists_{(0,\pi)}\theta.\ \bigvee_{x''\in X} x' \xrightarrow{line[a,\varphi]} x'' \wedge x \xrightarrow{arc[r,\varphi+\theta,-\theta]} x''$$
$$[x\ \textit{opposite}\ x'\ \textit{in}\ X] \stackrel{def}{=} \exists a.\exists \varphi.\ \bigvee_{x'',x'''\in X} x \xrightarrow{line[a,\varphi]} x'' \wedge x' \xrightarrow{line[a,\varphi+\pi]} x'''$$
(ii) for c, o respectively vehicle, object variables, d arithmetic term, $[c\ \textit{meets}(d)\ o]$ means that c reaches the position of o at distance d:
$$[c\ \textit{meets}(d)\ o] \stackrel{def}{=} \exists z.\ z \preccurlyeq c.it \wedge c.pos \stackrel{z}{\rightsquigarrow} o.pos \wedge \|z\| = d$$

(iii) a) for c a vehicle variable, X a set of vertex variables, [c *go-straight* X], [c *turn-right* X], [c *turn-left* X] express constraints on the itinerary of c within the map restricted to vertices in X (typically, a junction):

$$[c\ go\text{-}straight\ X] \stackrel{def}{=} \exists a.\exists\varphi.\ line[a,\varphi] \preccurlyeq c.it \wedge \bigvee_{x,x'\in X} c.pos = x \wedge x \xrightarrow{line[a,\varphi]} x'$$
$$[c\ turn\text{-}right\ X] \stackrel{def}{=} \exists r.\exists\varphi.\exists_{(-\pi,0)}\theta.\ arc[r,\varphi,\theta] \preccurlyeq c.it \wedge \bigvee_{x,x'\in X} c.pos = x \wedge x \xrightarrow{arc[r,\varphi,\theta]} x'$$
$$[c\ turn\text{-}left\ X] \stackrel{def}{=} \exists r.\exists\varphi.\exists_{(0,\pi)}\theta.\ arc[r,\varphi,\theta] \preccurlyeq c.it \wedge \bigvee_{x,x'\in X} c.pos = x \wedge x \xrightarrow{arc[r,\varphi,\theta]} x'$$

b) for o an object variable, X a set of vertex variables, l an optional arithmetic term, $[o@X,l]$ means that the position of o belongs to the map subgraph restricted to vertices in X and the lane of o is l:
$[o@X,l] \stackrel{def}{=} \Big(\exists d.\exists s. \bigvee_{x,x'\in X} x \xrightarrow{s} x' \wedge o.pos = (x,s,d) \vee o.pos = x\Big) \wedge o.ln = l$

Scenario Description for ADS. We define a scene as a triplet $\langle\psi_{map}, \psi_{add}, \psi_{dyn}\rangle$ of *M2CL* formulas without universal quantifiers where ψ_{add} defines the addresses of the objects involved in ψ_{dyn} in the map specified by ψ_{map}. As for maps, a scene can have a top-down and a bottom-up specification defined respectively by the formulas, $\sim\psi_{map} \Rightarrow \psi_{add} \wedge \psi_{dyn}$ and $\psi_{map} \wedge \psi_{add} \wedge \psi_{dyn}$.

A scenario is a sequence of scenes sharing a common map context and intended to describe relevant partial states of an ADS run. There are several proposals for scenario description languages [2,8,13]. Figure 1 presents a scenario of two scenes taken from [2]. The initial scene is defined by:

$$\begin{aligned}\psi_{map} &= [r : road(x,s,y)] \wedge [s.lanes = 2]\\ \psi_{add} &= [ego@r,1] \wedge [c_1@r,1] \wedge [c_2@r,2]\\ \psi_{dyn} &= [ego\ meets(84)\ c_1] \wedge [c_2\ meets(100)\ ego] \wedge [ego.sp = c_1.sp = 100 \wedge c_2.sp = 110]\end{aligned}$$

The second scene after the vehicle c_2 passes the *ego* vehicle is:

$$\begin{aligned}\psi'_{map} &= [r : road(x,s,y)] \wedge [s.lanes = 2]\\ \psi'_{add} &= [ego@r,1] \wedge [c_1@r,1] \wedge [c_2@r,1]\\ \psi'_{dyn} &= [ego\ meets(20)\ c_2] \wedge [c_2\ meets(64)\ c_1] \wedge [ego.sp = c_1.sp = 100 \wedge c_2.sp = 110]\end{aligned}$$

Note that from a semantic point of view, a scene is characterized by minimal models of *M2CL* $\langle\sigma, G, \mathbf{s}\rangle$ that satisfy the formula and where all irrelevant components of $\mathbf{s}$ are omitted. For instance, in the minimal models of the two scenes only the components of $\mathbf{s}$ corresponding to c_1, c_2 and *ego* are taken.

3.3 Temporal *M2CL* and Specification of ADS

Temporal M2CL (shorthand *TM2CL*) is defined as the linear time temporal extension of *M2CL*. The syntax is as follows:

$$\Phi ::= \phi \mid \mathbf{N}\ \Phi \mid \Phi\ \mathbf{U}\ \Phi \mid \Phi \wedge \Phi \mid \exists c.\ \Phi \mid \neg\Phi$$

where ϕ is *M2CL* formula. We consider moreover the *eventually* operator $\Diamond\Phi \stackrel{def}{=} true\ \mathbf{U}\ \Phi$, and *always* operator $\Box\Phi \stackrel{def}{=} \neg\Diamond\neg\Phi$. The semantics of *TM2CL* is defined on triples $(\sigma,\ G,\ [\mathbf{s}^{(t_i)}]_{i\geq 0})$ containing respectively an assignment σ of

Table 5. Semantics of *TM2CL*

$\sigma, G, [\mathbf{s}^{(t_i)}]_{i\geq 0} \models \phi$	iff $\sigma, G, \mathbf{s}^{(t_0)}) \models \phi$
$\sigma, G, [\mathbf{s}^{(t_i)}]_{i\geq 0} \models \mathbf{N}\ \Phi$	iff $\sigma, G, [\mathbf{s}^{(t_i)}]_{i\geq 1} \models \Phi$
$\sigma, G, [\mathbf{s}^{(t_i)}]_{i\geq 0} \models \Phi_1\ \mathbf{U}\ \Phi_2$	iff $\exists k \geq 0.\ \forall j \in [0, k-1].\ \sigma, G, [\mathbf{s}^{(t_i)}]_{i\geq j} \models \Phi_1$ and $\sigma, G, [\mathbf{s}^{(t_i)}]_{i\geq k} \models \Phi_2$
$\sigma, G, [\mathbf{s}^{(t_i)}]_{i\geq 0} \models \Phi_1 \wedge \Phi_2$	iff $\sigma, G, [\mathbf{s}^{(t_i)}]_{i\geq 0} \models \Phi_1$ and $\sigma, G, [\mathbf{s}^{(t_i)}]_{i\geq 0} \models \Phi_2$
$\sigma, G, [\mathbf{s}^{(t_i)}]_{i\geq 0} \models \exists o.\ \Phi$	iff $\sigma[o \mapsto u], G, [\mathbf{s}^{(t_i)}]_{i\geq 0} \models \Phi,$ for some $u \in \mathcal{O}$
$\sigma, G, [\mathbf{s}^{(t_i)}]_{i\geq 0} \models \neg\Phi$	iff $\sigma, G, [\mathbf{s}^{(t_i)}]_{i\geq 0} \not\models \Phi$

vehicle variables defined in the *TM2CL* context, a map G and a run $[\mathbf{s}^{(t_i)}]_{i\geq 0}$ on G for a finite set of objects $\mathcal{O}$. The semantic rules are defined in Table 5.

We use *TM2CL* for both the specification of system properties and traffic rules. The difference between the two concepts is not clear-cut although it is implicit in many works. System properties characterize the desired ADS behavior in terms of relations between speeds and distances taking into account relevant dynamic characteristics. These include properties such as keeping safe distance or keeping acceleration and deceleration rates between some bounds.

Traffic rules are higher-level specifications for enhanced safety and efficiency that usually depend on the driving context. They deal not only with obligations such as yielding right of way and traffic control at junctions but also advice on how to drive sensibly and safely in situations disrupting traffic flow such as congestion, accidents and works in progress. We provide below a formalization of system properties and traffic rules showing the expressiveness of our modeling framework. We formalize a set of traffic rules for an intersection j with all-way stop provided in [26]. The rules are implications of the form $\sim\zeta(j) \Rightarrow \Phi(j)$ where $\zeta(j)$ is the *MCL* formula characterizing j and $\Phi(j)$ is a *TM2CL* formula describing contraints on the driver behavior. We provide below the constraints in English and the corresponding *TM2CL* formulas:

(i) "If a driver arrives at the intersection and no other vehicles are present, then the driver can proceed":
$\forall c.\forall st.\ \Box\ [st@j.en] \wedge [c@j.en] \wedge [\neg\exists c'.\ c' \neq c \wedge [c'@j]] \Rightarrow \Diamond[c@j]$

(ii) "If, on approach of the intersection, there are one or more cars already there, let them proceed, then proceed yourself":
$\forall c.\forall st.\ \Box\ [st@j.en] \wedge [c\ meets(d)\ st] \wedge [d \leq d_{min}] \Rightarrow [\neg[c@j]]\ \mathbf{U}\ [\neg\exists c'.c' \neq c \wedge [c'@j]]$

(iii) "If a driver arrives at the same time as another vehicle, the vehicle on the right has the right-of-way":
$\forall c.\forall c'.\ \Box\ [c@j.en] \wedge [c'@j.en'] \wedge [c.wt = c'.wt] \wedge [j.en\ right\text{-}of\ j.en'\ in\ j] \Rightarrow [c'@j.en']\ \mathbf{U}\ [c@j]$

(iv) "(a) If two vehicles arrive opposite each other at the same time, and no vehicles are on the right, then they may proceed at the same time if they are going straight ahead. (b) If one vehicle is turning and one is going straight, the right-of-way goes to the car going straight:"

$$\forall c.\forall c'.\ \Box\ [c@j.en] \wedge [c'@j.en'] \wedge [c.wt = c'.wt = 0] \wedge [j.en\ \mathit{opposite}\ j.en'\ \mathit{in}\ j] \wedge \neg[\exists c''.\ [c''@j.en''] \wedge [j.en''\ \mathit{right\text{-}of}\ j.en\ \mathit{in}\ j] \vee [j.en''\ \mathit{right\text{-}of}\ j.en'\ \mathit{in}\ j]] \wedge [c\ \mathit{go\text{-}straight}\ j] \wedge [c'\ \mathit{go\text{-}straight}\ j]\ \Rightarrow\ \Diamond[c@j] \wedge [c'@j]$$

$$\forall c.\forall c'.\ \Box\ [c@j.en] \wedge [c'@j.en'] \wedge [c.wt = c'.wt = 0] \wedge [j.en\ \mathit{opposite}\ j.en'\ \mathit{in}\ j] \wedge \neg[\exists c''.\ [c''@j.en''] \wedge [j.en''\ \mathit{right\text{-}of}\ j.en] \vee [j.en''\ \mathit{right\text{-}of}\ j.en']] \wedge [c\ \mathit{go\text{-}straight}\ j] \wedge \neg[c'\ \mathit{go\text{-}straight}\ j]\ \Rightarrow\ \Diamond[c@j]$$

(v) "If two vehicles arrive opposite each other at the same time and one is turning right and one is turning left, the right-of-way goes to the vehicle turning right. Since they are both trying to turn into the same road, priority should be given to the vehicle turning right as they are closest to the lane":

$$\forall c.\forall c'.\ \Box\ [c@j.en] \wedge [c'@j.en'] \wedge [c.wt = c'.wt = 0] \wedge [j.en\ \mathit{opposite}\ j.en'\ \mathit{in}\ j] \wedge [c\ \mathit{turn\text{-}right}\ j] \wedge [c'\ \mathit{turn\text{-}left}\ j]\ \Rightarrow\ \Diamond[c@j]$$

4 ADS Validation

4.1 Classification of Validation Problems

The following categories of validation problems can arise in our framework:

MCL **and** *M2CL* **Model-Checking:** (i) Given a map specification ϕ as a closed *MCL* formula and a metric graph G decide if G is a model of ϕ. The problem boils down to checking satisfiability of a *segment logic* (*SL*) formula obtained by quantifier elimination of vertex variables and partial evaluation of graph constraints in ϕ according to G. We present later in this section a decision procedure for *SL*. (ii) Similarly, given a distribution specification ϕ as a closed *M2CL* formula, a map G and a state $\mathbf{s}$ for a *finite* set of objects $\mathcal{O}$, decide if $\langle G, \mathbf{s}\rangle$ is a model of ϕ. Again, the problem boils down to checking satisfiability of a *SL* formula obtained by quantifier elimination of vertex and object variables an and partial evaluation of attribute terms.

TM2CL **Runtime Verification:** Given a temporal specification Φ as a *TM2CL* formula, a map G and a run $[\mathbf{s}^{(t_i)}]_{i\geq 0}$ of an ADS, check if $G, [\mathbf{s}^{(t_i)}]_{i\geq 0}$ is a model of Φ. This problem boils down to evaluating the semantics of Φ on the run. In [10] we consider a similar runtime verification problem for temporal configuration logic and runs of dynamic reconfigurable systems. We have shown that the evaluation of linear-time temporal operators and the model-checking of state/configuration specifications can be dealt separately. The same idea can be applied here: on one hand, the temporal formulas can be handled by LamaConv [17] to generate FSM monitors and on the other hand, the model-checking of distribution specifications can be handled by a SMT solver (such as Z3) by using an encoding into a decidable theory.

MCL **and** *M2CL* **Satisfiability Checking:** (i) Given a map specification ϕ as a closed *MCL* formula decide if ϕ is satisfiable, that is, it has at least one model.

We then show in this section that the problem can be effectively solved for a significant fragment of *MCL* including a restricted form of bottom-up map specifications. Notice that entailment checking, that is, deciding validity of $\forall \mathbf{x}.\ \phi_1 \Rightarrow \phi_2$ for map specifications ϕ_1, ϕ_2 where $fv(\phi_1) = fv(\phi_2) = \mathbf{x}$, boils down to checking satisfiability of $\exists \mathbf{x}.\ \phi_1 \wedge \neg\phi_2$, and can be solved under the same restrictions. (ii) Similarly, given a distribution specification ϕ as a closed *M2CL* formula decide if ϕ is satisfiable, that is, it has at least one model. The problem can be reduced to the satisfiability checking of *MCL* specifications whenever ϕ is of the restricted form $\exists y_1 ... \exists y_k.\ \phi'$ where y_1, ... y_k are the only object variables occurring in ϕ'. In this case, every object variable y can be *substituted* by a finite number of *MCL* variables y_{attr} encoding its identity and attributes. As example, for a vehicle variable y consider an identity (real) variable y_{id}, a segment variable y_{it}, a position variable y_{pos}, real variables y_{ln}, y_{sp}, y_{wt}, etc. After replacement, we obtain an equisatisfiable *MCL* formula by enforcing the additional constraints that state attributes are consistently assigned (e.g., $(y_{id} = y'_{id}) \Rightarrow y_{it} = y'_{it})$ for all pairs y, y' of vehicle variables among y_1, ..., y_k. Finally, notice also that entailment checking between distributed specifications can be solved as well, by reduction to satisfiability checking as explained above.

4.2 Satisfiability Checking

Satisfiability Checking of *MCL*. The satisfiability checking for *MCL* formula is undecidable in general. Actually, the combined use of edge constraints $x \xrightarrow{s} x'$, equalities on vertex positions $x = x'$, boolean operators and quantifiers leads to undecidability, as it allows the embedding of first order logic on directed graphs.

Nevertheless, for a significant class of *MCL* formulas, their satisfiability checking can be reduced to satisfiability checking of *segment logic* (*SL*), that is, the fragment of *MCL* without vertex variables, which is a first order logic combining only arithmetic and segment constraints.

A *complete metric graph specification* ψ^* is a *MCL* formula of the form:

$$(\wedge_{1\le i<j\le n}\ x_i \ne x_j) \wedge (\forall y.\ \vee_{i=1}^{n}\ \ y = x_i) \wedge$$
$$(\bigoplus_{i=1}^{n} \bigoplus_{j=1}^{n} \bigoplus_{h=1}^{m_{ij}}\ x_i \xrightarrow{s_{ijh}} x_j) \wedge (\wedge_{i=1}^{n} \wedge_{j=1}^{n} \wedge_{1\le h<h'\le m_{ij}}\ s_{ijh} \ne s_{ijh'})$$

that is, where the set of free vertex variables is $\mathbf{x} = \{x_1, ..., x_n\}$. Note that a complete metric specification characterizes a metric graph with precisely n vertices (in correspondence with vertex variables $x_1, \ldots, x_n$) and, with precisely m_{ij} distinct edges (that is, defined by the constraints $x_i \xrightarrow{s_{ijh}} x_j$ for $h = 1, m_{ij}$), for every pair of vertices x_i, x_j.

Theorem 1. *Let ψ^* be a complete metric graph specification with free variables $\mathbf{x} \uplus \mathbf{z} \uplus \mathbf{k}$. For any MCL formula ϕ with $fv(\phi) \subseteq \mathbf{x} \uplus \mathbf{z} \uplus \mathbf{k}$ holds*

1. *the closed MCL formula $\exists \mathbf{x}.\ \exists \mathbf{z}.\ \exists \mathbf{k}.\ \psi^* \wedge \phi$ is satisfiable iff*
2. *the closed SL formula $\exists \mathbf{z}.\ \exists \mathbf{k}.\ (\wedge_{i=1}^{n} \wedge_{j=1}^{n} \wedge_{1\le h<h'\le m_{ij}}\ s_{ijh} \ne s_{ijh'}) \wedge (\wedge_{i=1}^{n} \wedge_{j=1}^{n} \wedge_{h=1}^{m_{ij}}\ \|s_{ijh}\| > 0) \wedge tr(n, E^*, \mu^*, \phi)$ is satisfiable, where $n = card\ \mathbf{x}$, $E^* = \cup_{i=1}^{n} \cup_{j=1}^{n} \{(i, s_{ijh}, j)\}_{h=1,m_{ij}}$, $\mu^* = \{x_i \mapsto i\}_{i=1,n}$ and the translation $tr(n, E, \mu, \phi)$ is defined in Table 6.*

Table 6. Translation rules for Theorem 1. (complete definition in [6])

$$
\begin{aligned}
tr(n,E,\mu,\psi_K) &\stackrel{def}{=} \psi_K \\
tr(n,E,\mu,\psi_S) &\stackrel{def}{=} \psi_S \\
tr(n,E,\mu,x \xrightarrow{s} y) &\stackrel{def}{=} \begin{cases} s = s_{ijh} \text{ if } E = \{(i, s_{ijh}, j)\},\ \mu x = i,\ \mu y = j \\ \mathit{false} \quad \text{otherwise} \end{cases} \\
tr(n,E,\mu,p = p') &\stackrel{def}{=} \textbf{eq-pos}(n,E,\mu,p,p') \\
tr(n,E,\mu,p \stackrel{s}{\rightsquigarrow} p') &\stackrel{def}{=} \textbf{acyclic-path}(n,E,\mu,p,s,p') \\
tr(n,E,\mu,\phi_1 \oplus \phi_2) &\stackrel{def}{=} \bigvee_{E_1 \cup E_2 = E} tr(n,E_1,\mu,\phi_1) \wedge tr(n,E_2,\mu,\phi_2) \\
tr(n,E,\mu,\phi_1 \vee \phi_2) &\stackrel{def}{=} tr(n,E,\mu,\phi_1) \vee tr(n,E,\mu,\phi_2) \\
tr(n,E,\mu,\neg\phi) &\stackrel{def}{=} \neg tr(n,E,\mu,\phi) \\
tr(n,E,\mu,\exists k.\ \phi) &\stackrel{def}{=} \exists k.\ tr(n,E,\mu,\phi) \\
tr(n,E,\mu,\exists z.\ \phi) &\stackrel{def}{=} \exists z.\ tr(n,E,\mu,\phi) \\
tr(n,E,\mu,\exists x.\ \phi) &\stackrel{def}{=} \bigvee_{i=1}^{n} tr(n,E,\mu[x \mapsto i],\phi)
\end{aligned}
$$

Proof. $(1 \Rightarrow 2)$ If the formula $\psi^* \wedge \phi$ is satisfiable, then it has a metric graph model isomorphic to the (unique up to edge labeling) metric graph G_{ψ^*} specified by ψ^*. The translated formula $tr(n, E^*, \mu^*, \phi)$ represents the evaluation of the semantics of ϕ on the metric graph G_{ψ^*} according to the rules defined in Table 3. It must be therefore satisfiable as well, as initially $\psi^* \wedge \phi$ is satisfiable. $(2 \Rightarrow 1)$ If the conjunction of the translated formula $tr(n, E^*, \mu^*, \phi)$ and the additional constraints has a model, ones can use it to build a metric graph, isomorphic to G_{ψ^*}, satisfying both ψ^* and ϕ. In particular the additional constraints ensure that the metric graph is well formed, that is, all edges are labeled by non-zero length segments, and there are no replicated edges between any pairs of vertices.

Satisfiability Checking of ***SL*****.** If segments $\mathcal{S}$ are restricted to particular interpretations, the satisfiability checking of formula of *SL* can be further reduced to satisfiability checking of formulas of extended arithmetic on reals.

Theorem 2. *If segments $\mathcal{S}$ are defined as intervals*

1. *the closed SL formula ϕ is satisfiable iff*
2. *the closed real arithmetic formula $tr_1(\phi)$ is satisfiable, where the translation $tr_1(\phi)$ is defined in Table 7.*

Proof. With interval interpretation, segments are precisely determined by their length and all segment operations and constraints boil down to operations and constraints on reals. Moreover, we remark that the transformation does not require multiplication[4] on real terms, henceforth, the translated formula $tr_1(\phi)$ belongs to linear arithmetic iff all arithmetic constraints ψ_K within ϕ were linear.

[4] Except if needed for encoding the length of segment types.

Table 7. Translation rules for Theorem 2

$tr_1(\\|s^T(t_1,...,t_m)\\|) \stackrel{def}{=} len\text{-}s^T(t_1,...,t_m)$	$tr_1(s = s') \stackrel{def}{=} tr_1(\\|s\\|) = tr_1(\\|s'\\|)$
$tr_1(\\|z\\|) \stackrel{def}{=} k_z$	$tr_1(s \preccurlyeq s') \stackrel{def}{=} tr_1(\\|s\\|) \leq tr_1(\\|s'\\|)$
$tr_1(\\|s \cdot s'\\|) \stackrel{def}{=} tr_1(\\|s\\|) + tr_1(\\|s'\\|))$	$tr_1(\\|s\\| = t) \stackrel{def}{=} tr_1(\\|s\\|) = t$

5 Discussion

The proposed framework relies on a minimal set of semantically integrated concepts. It is expressive and modular as it introduces progressively the basic concepts and carefully separates concerns. It supports a well-defined specification and validation methodology without semantic gaps as discussed in [6]. Using configuration logic allows the specification of behavioral properties taking into account map contexts. This is a main difference from approaches relying on temporal logics that cannot account for map configurations and where formulas characterize sets of runs in some implicit map environment, usually a simple multi-lane setting. Configuration logic specifies scenes as conjunctions of formulas describing map configurations and vehicle distributions linked by an addressing relation. It enables enhanced expressiveness by combining static and dynamic aspects while retaining the possibility to consider them separately. It considers maps as the central concept of the semantic model and emphasizes the needs for multilevel representation depending on the type of goals to be met including long-term mission goals, mid-term maneuver goals and short-term safety and trajectory tracking goals. Among the three abstraction levels, curve segment models play a central role. Interval segment models can account for simple properties depending only on relative distances between the involved mobiles. For properties depending on topological and geometric relations, curve segment models are needed. The expression of such properties involves primitives such as go-straight, turn-right, turn-left, right-of and opposite. Region segment models are needed for low level properties taking into account the dimensions of the objects and their movement in the 2D space.

The paper is the culmination of work developed over the past three years both on foundations of autonomous systems [15,25] and on modelling and validation of reconfigurable dynamic systems using the DR-BIP component framework [4,10]. We plan to extend this work in two directions. The first is to leverage on the DR-BIP execution semantics and formalize ADS dynamics as the composition of object behavior acting on maps. The second is to extend our work on runtime verification of dynamic reconfigurable systems [10] by developing adaptive validation techniques driven by adequate model coverage criteria. These techniques should provide model-based evidence that a good deal of the many and diverse driving situations are covered (e.g. different types of roads, of junctions, of traffic conditions, etc.). Finally, we will investigate diagnostics generation techniques linking failures to their causes emerging from risk factors such as violations of traffic regulations and unpredictable events.

References

1. ASAM OpenDRIVE® - open dynamic road information for vehicle environment. Technical report, V 1.6.0, ASAM e.V., Mar 2020. https://www.asam.net/standards/detail/opendrive
2. ASAM OpenScenario® - dynamic content in driving simulation, UML modeling rules. Technical report, V 1.0.0, ASAM e.V., Mar 2020. https://www.asam.net/standards/detail/openscenario
3. Bagschik, G., Menzel, T., Maurer, M.: Ontology based scene creation for the development of automated vehicles. In: Intelligent Vehicles Symposium, pp. 1813–1820. IEEE (2018)
4. El Ballouli, R., Bensalem, S., Bozga, M., Sifakis, J.: Four exercises in programming dynamic reconfigurable systems: methodology and solution in DR-BIP. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11246, pp. 304–320. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03424-5_20
5. Beetz, J., Borrmann, A.: Benefits and limitations of linked data approaches for road modeling and data exchange. In: Smith, I., Domer, B. (eds.) EG-ICE 2018. LNCS, vol. 10864, pp. 245–261. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91638-5_13
6. Bozga, M., Sifakis, J.: Specification and validation of autonomous driving systems: a multilevel semantic framework. CoRR abs/2109.06478 (2021). https://arxiv.org/abs/2109.06478
7. Chen, W., Kloul, L.: An ontology-based approach to generate the advanced driver assistance use cases of highway traffic. In: KEOD, pp. 73–81. SciTePress (2018)
8. Damm, W., Kemper, S., Möhlmann, E., Peikenkamp, T., Rakow, A.: Using traffic sequence charts for the development of HAVs. In: ERTS 2018, Proceedings (2018)
9. Dosovitskiy, A., Ros, G., Codevilla, F., López, A.M., Koltun, V.: CARLA: an open urban driving simulator. In: Proceedings of Machine Learning Research, vol. 78, pp. 1–16. PMLR (2017). CoRL
10. El-Hokayem, A., Bozga, M., Sifakis, J.: A temporal configuration logic for dynamic reconfigurable systems. In: SAC, pp. 1419–1428. ACM (2021)
11. Esterle, K., Aravantinos, V., Knoll, A.C.: From specifications to behavior: maneuver verification in a semantic state space. In: IV, pp. 2140–2147. IEEE (2019)
12. Esterle, K., Gressenbuch, L., Knoll, A.C.: Formalizing traffic rules for machine interpretability. In: CAVS, pp. 1–7. IEEE (2020)
13. Fremont, D.J., et al.: Scenic: a language for scenario specification and data generation. CoRR abs/2010.06580 (2020). https://arxiv.org/abs/2010.06580
14. Fremont, D.J., et al.: Formal scenario-based testing of autonomous vehicles: from simulation to the real world. In: ITSC, pp. 1–8. IEEE (2020)
15. Harel, D., Marron, A., Sifakis, J.: Autonomics: in search of a foundation for next-generation autonomous systems. Proc. Natl. Acad. Sci. USA **117**(30), 17491–17498 (2020)
16. Hilscher, M., Linker, S., Olderog, E.-R., Ravn, A.P.: An abstract model for proving safety of multi-lane traffic manoeuvres. In: Qin, S., Qiu, Z. (eds.) ICFEM 2011. LNCS, vol. 6991, pp. 404–419. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24559-6_28
17. Institute for Software Engineering and Programming Languages, University of Lübeck: LamaConv - Logics and Automata Converter Library (2020). https://www.isp.uni-luebeck.de/lamaconv

18. Karimi, A., Duggirala, P.S.: Formalizing traffic rules for uncontrolled intersections. In: ICCPS, pp. 41–50. IEEE (2020)
19. Poggenhans, F., et al.: Lanelet2: a high-definition map framework for the future of automated driving. In: ITSC, pp. 1672–1679. IEEE (2018)
20. Rizaldi, A., Althoff, M.: Formalising traffic rules for accountability of autonomous vehicles. In: ITSC, pp. 1658–1665. IEEE (2015)
21. Rizaldi, A., Immler, F., Schürmann, B., Althoff, M.: A formally verified motion planner for autonomous vehicles. In: Lahiri, S.K., Wang, C. (eds.) ATVA 2018. LNCS, vol. 11138, pp. 75–90. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01090-4_5
22. Rizaldi, A., et al.: Formalising and monitoring traffic rules for autonomous vehicles in Isabelle/HOL. In: Polikarpova, N., Schneider, S. (eds.) IFM 2017. LNCS, vol. 10510, pp. 50–66. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66845-1_4
23. Rong, G., et al.: LGSVL simulator: a high fidelity simulator for autonomous driving. CoRR abs/2005.03778 (2020). https://arxiv.org/abs/2005.03778
24. Schönemann, V., et al.: Scenario-based functional safety for automated driving on the example of valet parking. In: Arai, K., Kapoor, S., Bhatia, R. (eds.) FICC 2018. AISC, vol. 886, pp. 53–64. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-03402-3_5
25. Sifakis, J.: Autonomous systems – an architectural characterization. In: Boreale, M., Corradini, F., Loreti, M., Pugliese, R. (eds.) Models, Languages, and Tools for Concurrent and Distributed Programming. LNCS, vol. 11665, pp. 388–410. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21485-2_21
26. Wikipedia. https://en.wikipedia.org/wiki/All-way_stop