International Journal on Software Tools for Technology Transfer
https://doi.org/10.1007/s10009-023-00723-0

FOUNDATIONS FOR MASTERING CHANGE

Special Section: Rigorous Engineering of Collective Adaptive Systems

Correct by design coordination of autonomous driving systems

Marius Bozga[1] · Joseph Sifakis[1]

Accepted: 10 October 2023

© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2023

Abstract

The paper proposes a method for the correct by design coordination of autonomous driving systems (*ADS*). It builds on previous results on collision avoidance policies and the modeling of *ADS* by combining descriptions of their static environment in the form of maps, and the dynamic behavior of their vehicles. An *ADS* is modeled as a dynamic system involving a set of vehicles coordinated by a *Runtime* that based on vehicle positions on a map and their kinetic attributes, computes free spaces for each vehicle. Vehicles are bounded to move within the corresponding allocated free spaces. We provide a correct by design safe control policy for an *ADS*, if its vehicles and the *Runtime* respect corresponding assume-guarantee contracts. The result is established by showing that the composition of assume-guarantee contracts is an inductive invariant that entails *ADS* safety. We show that it is practically possible to define speed control policies for vehicles that comply with their contracts. Furthermore, we show that traffic rules can be specified in a linear-time temporal logic as a class of formulas that constrain vehicle speeds. The main result is that, given a set of traffic rules, it is possible to derive free-space policies of the *Runtime* such that the resulting system behavior is safe by design with respect to the rules.

Keywords autonomous driving systems · traffic rule specification · map specification · collision avoidance policy · assume-guarantee contract · correctness by design

1 Introduction

Autonomous driving systems (*ADS*) are probably the most difficult systems to design and validate, because the behavior of their agents is subject to temporal and spatial dynamics. They are real-time distributed systems involving components with partial knowledge of their environment, pursuing specific goals, while the collective behavior must meet given global goals.

Development of trustworthy *ADS* is an urgent and critical need. It poses challenges that go well beyond the current state of the art due to their overwhelming complexity. These challenges include, on the one hand, modeling the system and specifying its properties, usually expressed as traffic rules, on the other hand, building the system and verifying its correctness with respect to the desired system properties.

Modeling involves a variety of issues related to the inherent temporal and spatial dynamics, as well as to the need for an accurate representation of the physical environment in which vehicles operate. Many studies focus on formalizing and standardizing a concept of map that is central to semantic awareness and decision making. These studies often use ontologies and logics with associated reasoning mechanisms to check the consistency of descriptions and their accuracy with respect to desired properties [2, 3]. Other works propose open source mapping frameworks for highly automated driving [1, 16]. Finally, the SOCA method [7] proposes an abstraction of maps called zone graph, and uses this abstraction for a morphological behavior analysis.

There is an extensive literature on *ADS* validation that involves two interrelated problems: the specification of system properties and the application of validation techniques. The specification of properties requires first-order temporal logics because parameterization and genericity are essential for the description of situations involving a varying numbers of vehicles and types of traffic patterns. The work in [17, 18] formalizes a set of traffic rules for highway scenarios in Isabelle/HOL. It shows that traffic rules can be used as requirements for autonomous vehicles and proposes a verifi-

✉ M. Bozga
marius.bozga@univ-grenoble-alpes.fr

J. Sifakis
joseph.sifakis@univ-grenoble-alpes.fr

[1] VERIMAG, Univ. Grenoble Alpes, CNRS, Grenoble INP, 38000 Grenoble, France

Published online: 02 November 2023

cation procedure. A formalization of traffic rules for uncontrolled intersections is provided in [12], which shows how the rules can be used by a simulator to safely control traffic at intersections. The work in [10] proposes a methodology for formalizing traffic rules in linear temporal logic; it shows how the evaluation of formalized rules on recorded human behaviors provides insight into how well drivers follow the rules.

Many works deal with the formal verification of controllers that perform specific maneuvers. For example, in [11], a dedicated multi-way spatial logic inspired by interval temporal logic is used to specify safety and provide proofs for lane change controllers. The work in [19] presents a formally verified motion planner in Isabelle/HOL. The planner uses maneuver automata, a variant of hybrid automata, and linear temporal logic to express properties. In [10], runtime verification is applied to check that the maneuvers of a high-level planner conform to traffic rules expressed in linear temporal logic.

Of particular interest to this work are correct by construction techniques, where system construction is guided by a set of properties that the system is guaranteed to satisfy. They involve either the application of monolithic synthesis techniques or compositional reasoning throughout a component-based system design process. There is considerable work on controller synthesis from a set of system properties, usually expressed in linear temporal logic, see for example [13, 21, 26–28]. These are algorithmic techniques that have been extensively studied in the field of control. They consist of restricting the controllable behavior of a system interacting with its environment so that a set of properties are satisfied. Nonetheless, their application is limited due to their high computational cost, which depends in particular on the type of properties and the complexity of the system behavior.

An alternative to synthesis is to achieve correctness by design as a result of composing the properties of the system components. Component properties are usually “assume-guarantee” contracts characterizing a causal relationship between a component and its environment: if the environment satisfies the “assume” part of the contract, the state of the component will satisfy the “guarantee” part, e.g., [4, 8, 15]. The use of contracts in system design involves a decomposition of overall system requirements into contracts that provide a basis for more efficient analysis and validation. In addition, contract-based design is advocated as a method for achieving correctness by design, provided that satisfactory implementations of the system can be found [23]. There are a number of theoretical frameworks that apply mainly to continuous or synchronous systems, especially for analysis and verification purposes [14, 20, 22]. They suffer computational limitations because, in the general case, they involve the symbolic solution of fixed-point equations, which restricts the expressiveness of the contracts [14]. Furthermore, they are only applicable to systems with a static architecture, which excludes dynamic reconfigurable systems, such as autonomous systems.

The paper builds on previous results [6] on a logical framework for parametric specification of *ADS* combining models of the system’s static environment in the form of maps and the dynamic properties of its vehicles. Maps are metric graphs whose vertices represent locations and edges are labeled with segments that can represent roads at different levels of abstraction, with characteristics such as length or geometric features characterizing their shape and size.

An *ADS* model is a dynamic system consisting of a map and a set of vehicles moving along specific routes. Its state can be conceived as the distribution of vehicles on a map with their positions, speeds, and other kinematic attributes. For its movement, each vehicle has a safe estimate of the free space in its neighborhood, according to predefined visibility rules. We assume that vehicle coordination is performed by a *Runtime* that, for given vehicle positions and speeds on the map, can compute the free spaces on each vehicle’s itinerary in which it can safely move.

We consider without loss of generality, *ADS* with a discretized execution time step Δt. Knowing its free space, each vehicle can move by adapting its speed in order to stay in this space, braking if necessary in case of emergency. At the end of each cycle, taking into account the movements of the vehicles, the *Runtime* updates their positions on the map. The cycle iterates by calculating the free spaces from the new state.

We study a safe control policy for *ADS*, which is correct by design. It results from the combination of two types of assume-guarantee contracts: one contract for each vehicle and another contract for the *Runtime* taking into account the positions of the vehicles on the map. The contract for a vehicle states that, assuming that initially the dynamics of the vehicle allow it to stay in the allocated free space, it will stay in this free space. Note that the details of the contract implementation are irrelevant; only the I/O relationship between free space and vehicle speed matters. The *Runtime* contract asserts that if the free spaces allocated to vehicles at the beginning of a cycle are disjoint, they can be allocated new disjoint free spaces provided they have fulfilled their contract. The combination of these two contracts leads to a control policy that satisfies an inductive invariant, implying system safety.

We build on this general result by specializing its application in two directions. First, we show that it is possible to define speed policies for vehicles that satisfy their assume-guarantee contract. Second, we show that it is possible to define free-space policies for the *Runtime* enforcing safety constraints of a given set of traffic rules. We formalize traffic rules as a class of properties of a linear temporal logic. We provide a method that derives from a given set of traffic rules, constraints on the free spaces chosen by the *Runtime* such that the resulting system behavior is safe with respect to these rules. This is the main result of the paper establishing correctness by design of general *ADS*, provided that their components comply with their respective contracts.

The paper is structured as follows. In Sect. 2, we establish the general framework by introducing the basic models and concepts for the representation of maps. In Sect. 3, we introduce the dynamic model of *ADS* involving a set of vehicles and a *Runtime* for their coordination. We show how a correct by design safe control policy is obtained by combining assume-guarantee contracts for the vehicles and the *Runtime*. In Sect. 4, we study the principle of speed policies respecting the vehicle contract and show its application through an example. In Sect. 5, we formalize traffic rules as a class of formulas of a linear temporal logic, and show how it is possible to generate from a set of traffic rules free-space policies such that the system is safe by design. In Sect. 6, we briefly describe the implementation of the approach and experiments underway. Section 7 concludes with a discussion of the significance of the results, future developments, and applications. A short version of the paper is available in [5].

2 Map representation

Following the idea presented in [6], we build contiguous road segments from a set $\mathcal{S}$ equipped with a partial concatenation operator $\cdot : \mathcal{S} \times \mathcal{S} \to \mathcal{S} \cup \{\bot\}$, a length norm $\|.\| : \mathcal{S} \to \mathbb{R}_{\geq 0}$ and a partial subsegment extraction operator $.[.,.] : \mathcal{S} \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathcal{S} \cup \{\bot\}$. Thus, given a segment s, $\|s\|$ represents its length and $s[a,b]$. for $0 \leq a < b \leq \|s\|$, represents the sub-segment starting at length a from its origin and ending at length b. Segments can be used to represent roads at different levels of abstraction, from intervals to regions. In this paper, we consider $\mathcal{S}$ as the set of curves obtained by concatenation of line segments and circle arcs, for representing roads of a map. More precisely, for any $a, r \in \mathbb{R}^*_{\geq 0}$, $\varphi \in \mathbb{R}$, $\theta \in \mathbb{R}^*$ the curves $line[a,\varphi]$, $arc[r,\varphi,\theta]$ are defined as

$$line[a,\varphi](t) \overset{def}{=} (at\cos\varphi, at\sin\varphi) \ \forall t \in [0,1]$$

$$\begin{aligned} arc[r,\varphi,\theta](t) \overset{def}{=} (&r(\sin(\varphi + t\theta) - \sin\varphi), \\ &r(-\cos(\varphi + t\theta) + \cos\varphi)) \ \forall t \in [0,1] \end{aligned}$$

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 1 illustrates the composition of three curves of this parametric form.

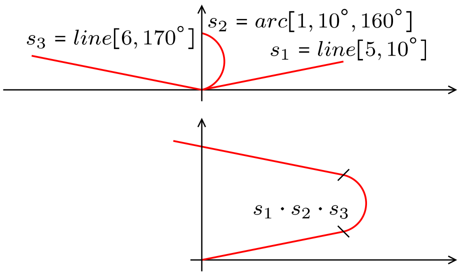
$s_3 = line[6, 170^\circ]$
$s_2 = arc[1, 10^\circ, 160^\circ]$
$s_1 = line[5, 10^\circ]$
$s_1 \cdot s_2 \cdot s_3$

Fig. 1 Curve segments and their composition

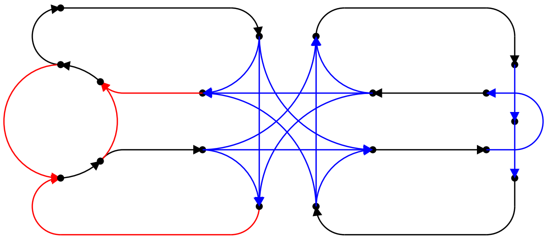

Fig. 2 A map with junctions (blue edges) and merger vertices (red edges) (Color figure online)

We use metric graphs $G \overset{def}{=} (V, \mathcal{S}, E)$ 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 (denoted $\mathcal{S}^\star$). For an edge $e = (v, s, v') \in E$ we denote ${}^\bullet e \overset{def}{=} v$, $e^\bullet \overset{def}{=} v'$, $e.seg \overset{def}{=} s$. For a vertex v, we define ${}^\bullet v \overset{def}{=} \{e \mid e^\bullet = v\}$ and $v^\bullet \overset{def}{=} \{e \mid {}^\bullet e = v\}$. We call a metric graph *connected* (resp. *weakly connected*) if a path (resp. an undirected path) exists between any pair of vertices.

We consider the set $Pos_G \overset{def}{=} V \cup \{(e,a) \mid e \in E,\ 0 \leq a \leq \|e.seg\|\}$ of *positions* defined by a metric graph. Note that positions $(e,0)$ and $(e, \|e.seg\|)$ are considered equal respectively to positions ${}^\bullet e$ and $e^\bullet$. We denote by $p \xrightarrow{s}_G p'$ the existence of an s-labelled *edge ride* between succeeding positions $p = (e,a)$ and $p' = (e,a')$ in the same edge e whenever $0 \leq a < a' \leq \|e.seg\|$ and $s = e.seg[a,a']$.. Moreover, we denote by $p \overset{s}{\rightsquigarrow}_G p'$ the existence of an s-labelled *ride* between arbitrary positions p, p', that is, $\rightsquigarrow_G \overset{def}{=} (\to_G)^+$ the transitive closure of edge rides. Finally, we define the distance d_G from position p to position 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. Whenever G is fixed in the context, we omit the subscript G for positions Pos_G, distance d_G, and rides $\to_G$ or $\rightsquigarrow_G$.

A connected metric graph $G = (V, \mathcal{S}, E)$ can be interpreted as a map, structured into roads and junctions, as depicted in Fig. 2, subject to additional assumptions:

– We restrict to metric graphs that are 2D-consistent [6], meaning intuitively they can be drawn in the 2D-plane such that the geometric properties of the segments are compatible with the topological properties of the graph. In particular, if two distinct paths starting from the same vertex v meet at another vertex v', the coordinates of v' calculated from each path are identical. For the sake of simplicity, we further restrict to graphs where distinct vertices are located at distinct points in the plane, and moreover, where no edge is self-crossing (meaning actually that distinct positions (e,a) of the same edge e are located at distinct points).

- The map is equipped with a symmetric *junction* relationship $\times$ on edges E, which abstracts the geometric crossing (or the proximity) between edges at positions other than the edge endpoints. This relationship is used to define the *junctions* of the map, that is, as any non-trivial equivalence class in the transitive closure of $\times$. Actually, junctions need additional signalization to regulate the traffic at their edges (e.g., traffic lights, stop signs, etc.). In addition, we assume a partial ordering $\prec_j$ on the set of vertices to reflect their static priorities as junction entries.
- To resolve conflicts at merger vertices, i.e., vertices with two or more incident segments that do not belong to a junction, we assume that the map is equipped with a static priority relationship. Specifically, for a vertex v, there is a total priority order $\prec_v$ on the set of edges ${}^{\bullet}v$. This order reflects an abstraction of the static priority rules associated with each of the merging edges (e.g., right-of-way, yield-priority, etc.).
- Each edge e is associated with a maximal speed limit $e.v \in \mathbb{R}_{\geq 0}$.

In the remainder of the paper, we consider a fixed metric graph $G = (V, \mathcal{S}, E)$ altogether with the junction relationship $\times$, static priorities $\prec_v$ and edge speed limits as discussed above. Also, we extend the junction and priority relationships from edges to their associated positions, that is, consider $(e_1,a_1) \sim (e_2,a_2) \overset{def}{=} e_1 \sim e_2$ for any relation $\sim \in \{\times, (\prec_v)_{v\in V}\}$. Finally, we denote by $r_1 \uplus r_2$ the property that rides r_1, r_2 in G are *non-crossing*, that is, their sets of positions are disjoint and, moreover, not belonging to the same junction(s), except for endpoints.

3 The *ADS* dynamic model

3.1 General *ADS* architecture

Given a metric graph G representing a map, the state of an *ADS* is a tuple $\langle st_o \rangle_{o \in \mathcal{O}}$ 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 fixed equipment such as lights, road signs, gates, etc. For a vehicle c, its state $st_c \overset{def}{=} \langle c.p, c.\delta, c.v, c.wt, c.it \dots \rangle$ includes respectively its *position* on the map (from *Pos*), its *displacement* traveled since $c.p$ (from $\mathbb{R}_{\geq 0}$), 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, its *itinerary* (from the set of segments $\mathcal{S}$), which labels a ride starting at $c.p$, etc. For a traffic light lt, its state $st_{lt} \overset{def}{=} \langle lt.p, lt.cl, \dots \rangle$ includes respectively its *position* on the map (from *Pos*), its *color* (with values *red* and *green*), etc.

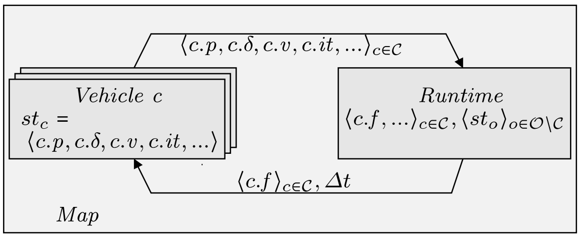
$\langle c.p, c.\delta, c.v, c.it, \dots \rangle_{c\in\mathcal{C}}$
Vehicle c
$st_c =$
$\langle c.p, c.\delta, c.v, c.it, \dots \rangle$
Runtime
$\langle c.f, \dots \rangle_{c\in\mathcal{C}}, \langle st_o \rangle_{o\in\mathcal{O}\setminus\mathcal{C}}$
$\langle c.f \rangle_{c\in\mathcal{C}}, \Delta t$
Map

Fig. 3 General *ADS* architecture

The general *ADS* model is illustrated in Fig. 3 and consists of a set of vehicle models $\mathcal{C}$ and a *Runtime* that interact cyclically with period Δt. The *Runtime* calculates free space values for each vehicle c which are lengths $c.f$ of initial rides on their itineraries $c.it$ whose positions are free of obstacles. In turn, the vehicles adapt their speed to stay within the allocated free space. Specifically, the interaction proceeds as follows:

- Each vehicle c applies a *speed policy* for period Δt respecting its free space $c.f$ received from the *Runtime*. During Δt, it travels a distance $c.\delta'$ to some new position $c.p'$, and at the end of the period its speed is $c.v'$, its itinerary $c.it'$, etc. The new state is then communicated to the *Runtime*.
- The *Runtime* updates the system state on the map taking into account the new vehicle states and time-dependent object attributes. Then it applies a *free space policy* computing the tuple $\langle c.f' \rangle_{c\in\mathcal{C}}$, the new free space for all vehicles based on the current system state. The corresponding free spaces are then communicated to vehicles and the next cycle starts.

Note that the coordination principle described is independent of the type of segments used in the map, e.g., intervals, curves or regions. For simplicity, we take the free spaces to measure the length of an initial ride without obstacles on the vehicle's itinerary. This abstraction is sufficient to state the basic results. We will discuss later how they can be generalized for richer interpretations of the map.

3.2 Assume-guarantee for safe control policies

We give below the principle of a safe control policy for vehicles, which respects their allocated free space, applying assume-guarantee reasoning.

We consider the following hypothesis: for a vehicle c, there exists a function $B_c : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ that gives the minimum braking distance c needs to stop from speed v, in case of emergency. Furthermore, for a position p, a segment s labeling a ride starting at p and non-negative distance f, we denote by $Ahead(p,s,f)$ the ride consisting of the positions reachable from p following the segment s within distance f, formally $Ahead(p,s,f) \overset{def}{=} \{p' \in Pos \mid \exists \delta \leq f .\ p \overset{s[0,\delta].}{\rightsquigarrow} p'\}$.

The following definition specifies a safe control policy using assume-guarantee reasoning on the components of the *ADS* architecture. We consider assume-guarantee contracts on components defined as pairs of properties A/G specifying respectively the input-output component behavior for a cycle, i.e., respectively, what the component guarantees (G) provided its environment conforms to given assumption (A).

Definition 1 (safe control policy)
A control policy is safe if

– Each vehicle $c \in \mathcal{C}$ respects the A/G contract:

$$0 \leq c.v \ \wedge \ B_c(c.v) \leq c.f \ /$$
$$0 \leq c.v' \ \wedge \ 0 \leq c.\delta' \ \wedge \ c.\delta' + B_c(c.v') \leq c.f \ \wedge$$
$$c.p \overset{c.it[0,c.\delta'].}{\rightsquigarrow} c.p' \ \wedge \ c.it' = c.it[c.\delta', -]$$

– The *Runtime* respects the A/G contract:

$$\bigwedge\nolimits_{c \in \mathcal{C}} 0 \leq c.\delta \leq c.f$$
$$\wedge \ \biguplus\nolimits_{c \in \mathcal{C}} Ahead(c.p, c.it, c.f - c.\delta) \ /$$
$$\bigwedge\nolimits_{c \in \mathcal{C}} c.f - c.\delta \leq c.f'$$
$$\wedge \ \biguplus\nolimits_{c \in \mathcal{C}} Ahead(c.p, c.it, c.f')$$

The policy is the joint enforcement of safe speed policies for vehicles and safe free space policies for the *Runtime*. Vehicle safe speed policies require that if a vehicle can brake safely by moving forward within its allocated free space at the beginning of a cycle, then it can adapt its speed moving forward within this space. *Runtime* safe free-space policies require that if the free spaces of the vehicles are non-crossing at the beginning of a cycle, it is possible to find new non-crossing free spaces for the vehicles, provided they move forward in their allocated free space.

Theorem 1
Safe control policies preserve the following invariants:

– *the speed is positive and compliant to the free space, for all vehicles,* $\bigwedge_{c \in \mathcal{C}} 0 \leq c.v \wedge B_c(c.v) \leq c.f$,
– *the free spaces are non-crossing,* $\biguplus_{c \in \mathcal{C}} Ahead(c.p, c.it, c.f)$.

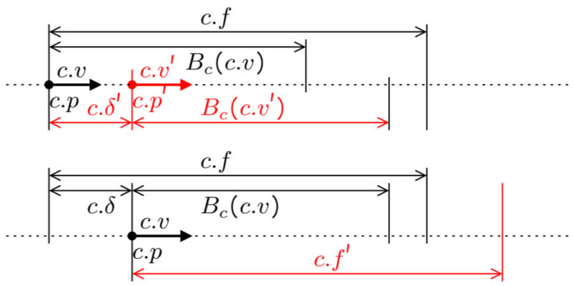

Fig. 4 Illustration of the Hoare triples for vehicle c

Proof
Consider the usual notation $\{\phi\}P\{\psi\}$ for Hoare triples, denoting that whenever the precondition ϕ is met, executing the program P establishes the postcondition ψ. Let $A_P(X)/G_P(X,X')$ be a contract for a program P whose initial state X satisfies the assumption $A_P(X)$ and which, if terminates guarantees the relation $G_P(X,X')$ between X and the final state X', then Hoare triples are established by the following proof rule:

$$\frac{\phi(X) \implies A_P(X) \qquad \exists X.\ \phi(X) \wedge G_P(X,X') \implies \psi(X')}{\{\phi\}P\{\psi\}}$$

We now prove that the conjunction of the two assertions in the theorem is an inductive invariant, holding at the beginning of every cycle. First, using the rule above for the assume-guarantee contract on *all* vehicles, we establish the following Hoare triple, where $\|_{c \in \mathcal{C}}\, c$ represents the program executed by the vehicle controllers in one cycle:

$$\{\bigwedge\nolimits_{c \in \mathcal{C}} 0 \leq c.v \wedge B_c(c.v) \leq c.f \wedge$$
$$\biguplus\nolimits_{c \in \mathcal{C}} Ahead(c.p, c.it, c.f)\}$$
$$\|_{c \in \mathcal{C}} \ c$$
$$\{\bigwedge\nolimits_{c \in \mathcal{C}} 0 \leq c.v' \wedge 0 \leq c.\delta' \wedge c.\delta' + B_c(c.v') \leq c.f \wedge$$
$$\biguplus\nolimits_{c \in \mathcal{C}} Ahead(c.p', c.it', c.f - c.\delta')\}$$

The arithmetic constraints on the speed, distance traveled, and free space are implied from the guarantee. The constraint on the free space takes into account the update of the vehicle positions, that is, moving ahead into their free space by the distance traveled (see Fig. 4, top). Second, using the assume-guarantee contract on the *Runtime* we establish the Hoare triple:

$$\{\bigwedge\nolimits_{c \in \mathcal{C}} 0 \leq c.v \wedge 0 \leq c.\delta \wedge c.\delta + B_c(c.v) \leq c.f \wedge$$
$$\biguplus\nolimits_{c \in \mathcal{C}} Ahead(c.p, c.it, c.f - c.\delta)\}$$
$$Runtime$$
$$\{\bigwedge\nolimits_{c \in \mathcal{C}} 0 \leq c.v \wedge B_c(c.v) \leq c.f' \wedge$$
$$\biguplus\nolimits_{c \in \mathcal{C}} Ahead(c.p, c.it, c.f')\}$$

That is, the *Runtime* re-establishes the invariant essentially by providing at least the same free space as in the previous cycle (see Fig. 4, bottom). □

Note that this theorem guarantees the safety of the coordination, since the vehicles respecting their contracts remain in their allocated free spaces, which are non-crossing by construction. Nevertheless, the result leaves a lot of freedom to vehicles and the *Runtime* to choose speeds and non-crossing free spaces. In particular, two questions arise concerning these choices. The first question is wether the system can reach states where no progress is possible. One can imagine traffic jam situations, for example when vehicles do not have enough space to move. The second question is whether free space choices can be determined by traffic rules that actually enforce fairness in resolving conflicts between vehicles. This question is discussed in detail in Sect. 5.

We show below that it is possible to compute non-blocking control policies by strengthening the contracts satisfied by the vehicles and the *Runtime* with additional conditions. For vehicles, we require that they move in a cycle if their free space is greater than a minimum free space f_{min}. This constant should take into account the dimensions of the vehicles and their dynamic characteristics, e.g., the minimum space needed to safely reach a non-negative speed from a stop state. Additional conditions for the contract of the *Runtime* are that, when all vehicles are stopped, it can find at least one free space greater than f_{min}.

Definition 2 (non-blocking control policy)
A control policy is non-blocking if there exists non-negative f_{min} such that:

- each vehicle $c \in C$ respects the A/G contract:

$$c.f \geq f_{min} \;/\; c.v' > 0$$

- the *Runtime* respects the A/G contract:

$$\bigwedge_{c \in C} c.v = 0 \;/\; \max_{c \in C} c.f' \geq f_{min}.$$

Theorem 2
Non-blocking control policies ensure progress, i.e., there is always a vehicle whose speed is positive.

Proof
The proof is an immediate consequence of the two contracts of Def. 2. If all vehicles stop moving during a cycle, the *Runtime* will necessarily find at least $f_{\min}$ free space for at least one of them. Then, during the next cycle, at least one vehicle will move again with positive speed, which concludes the proof. □

4 Speed policies abiding by the vehicle contract

In this section, we show that it is possible for vehicles to compute speed policies in accordance with their contract. The behavior of each vehicle is defined by a controller, which, given its current speed and its free space, computes the displacement for Δt so that it can safely move in the free space. Such safe speed policies have been studied in [24, 25].

We illustrate the principle of safe speed policy with respect to f considering that each vehicle is equipped with a controller that receives a free space value and adjusts its speed adequately. For the sake of simplicity, assume the controller can select among three different constant acceleration values $\{-b_{max}, 0, a_{max}\} \in \mathbb{R}$ respectively, the negative value $-b_{max}$ for decreasing, the zero value for maintaining and the positive value a_{max} for increasing the speed. At every cycle, the controller selects the highest acceleration value to which the vehicle guarantee holds, as defined by its contract in Def. 1. Nonetheless, an exception applies for the particular case where the vehicle stops within the cycle, which cannot be actually handled with constant acceleration.

Definition 3 (region-based speed policy)
The region-based speed policy defines the new speed v' and displacement δ' using a region decomposition of the safe $v \times f$ space (that is, where $v \geq 0$ and $f \geq B(v)$) as follows:

$$v', \delta' \overset{def}{=} \begin{cases} 0,\ f & \text{if } f \geq B(v), f - v\Delta t < B(v), \\ & \quad v - b_{max}\Delta t < 0 \\ v - b_{max}\Delta t,\ v\Delta t - b_{max}\Delta t^2/2 & \\ & \text{if } f \geq B(v), f - v\Delta t < B(v), \\ & \quad v - b_{max}\Delta t \geq 0 \\ v,\ v\Delta t & \text{if } f - v\Delta t \geq B(v), \\ & \quad f - v\Delta t - a_{max}\Delta t^2/2 < \\ & \quad B(v + a_{max}\Delta t) \\ v + a_{max}\Delta t,\ v\Delta t + a_{max}\Delta t^2/2 & \\ & \text{if } f - v\Delta t - a_{max}\Delta t^2/2 \geq \\ & \quad B(v + a_{max}\Delta t) \end{cases} \tag{1}$$

Intuitively, the regions are defined such that, when the corresponding acceleration is constantly applied for Δt time units, the guarantee on the vehicle is provable given the assumptions and the region boundary conditions. For illustration, the regions are depicted in Fig. 5 for some concrete values of $\Delta t = 1\ sec$, $a_{max} = 2.5 m/s^2$ and $b_{max} = -3.4 m/s^2$.

Moreover, the vehicle position and the itinerary are updated according to the travelled distance by taking $c.p'$ such that $c.p \overset{c.it[0,c.\delta'].}{\rightsquigarrow} c.p'$ and $c.it' = c.it[c.\delta', -]..$ Furthermore, the waiting time $c.wt$ is updated by taking $c.wt' \overset{def}{=} c.wt + \Delta t$ if $c.v = c.v' = 0$ and $c.wt' \overset{def}{=} 0$ otherwise.

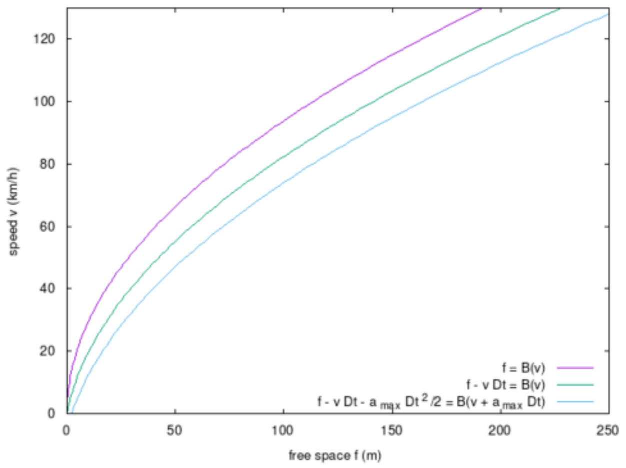

Fig. 5 Region decomposition for safe speed policy

Proposition 1

The region-based speed policy respects the safety contract for vehicles if the braking function is $B(v) = v^2/2b_{max}$.

Proof

According to Def. 1, assume that $0 \leq v$ and $B(v) \leq f$. The policy must guarantee that $0 \leq v'$, $0 \leq \delta'$, $\delta' + B(v') \leq f$, $c.p \overset{c.it[0,c.\delta'].}{\rightsquigarrow} c.p'$, $c.it' = c.it[c.\delta', -]$.. Obviously, the last two constraints are explicitely enforced. For the remaining constraints, the proof is made on a case by case basis for the four regions:

(i) Immediate as $v' = 0$, $\delta' = f$ in this case.

(ii) The constraint $v' \geq 0$ is equivalent to $v - b_{max}\Delta t \geq 0$, which is one of the region boundaries. The constraint $\delta' \geq 0$ holds as $\delta' = v\Delta t - b_{max}\Delta t^2/2$. Consider the constraint $\delta' + B(v') \leq f$. The term $\delta' + B(v')$ can be successively rewritten as follows:

$$v\Delta t - b_{max}\Delta t^2/2 + B(v - b_{max}\Delta t) =$$

$$v\Delta t - b_{max}\Delta t^2/2 + (v - b_{max}\Delta t)^2/2b_{max} =$$

$$v^2/2b_{max} = B(v)$$

Henceforth, $\delta' + B(v') \leq f$ is equivalent to $B(v) \leq f$ and holds from the assumption.

(iii) The constraint $v' \geq 0$ holds because $v' = v$. The constraint $\delta' \geq 0$ holds because $\delta' = v \cdot \Delta t$. The constraint $\delta' + B(v') \leq f$ is equivalent to $v\Delta t + B(v) \leq f$, which is one of the region boundaries.

(iv) The constraint $v' \geq 0$ holds because $v' = v + a_{max}\Delta t$ and v, a_{max}, Δt are all positive. Similarly, the constraint $\delta' \geq 0$ holds, as $\delta' = v\Delta t + a_{max}\Delta t^2/2$. The constraint $\delta' + B(v') \leq f$ is equivalent to $v\Delta t + a_{max}\Delta t^2/2 + B(v + a_{max}\Delta t) \leq f$ and is the region boundary. □

Proposition 2

The region-based speed policy respects the non-blocking contract for vehicles for any $f_{min} \geq B(a_{max}\Delta t) + a_{max}\Delta t^2/2$.

Proof

Actually, the value $B(a_{max}\Delta t) + a_{max}\Delta t^2/2$ represents the minimal amount of space for which the policy will select acceleration a_{max} when the speed is zero, as defined by the condition of the 4th region. □

Note that the speed policy works independently of the value of the parameter Δt, which is subject only to implementation constraints, e.g., it must be large enough to allow the controlled electromechanical system to realize the desired effect. A large Δt may imply low responsiveness to changes and jerky motion, but will never compromise the safety of the system.

The proposed implementation of the speed policy is "greedy" in the sense that it applies maximum acceleration to move as fast as possible in the available space. We could have "lazy" policies that do not move as fast as possible and simply extend the travel time. We have shown in [24] that the region-based speed policy approaches the optimal safety policy, i.e., the one that gives the shortest travel time when we refine the choice of acceleration and deceleration rates in the interval $[-b_{max}, a_{max}]$.

5 Free space policies implied by traffic rules

In this section, we study free space safety policies for a given set of global system properties describing traffic rules. We formalize traffic rules as a class of linear temporal logic formulas and provide a method for computing free-space values for vehicles that allow them to meet a given set of traffic rules.

5.1 Writing specifications of traffic rules

Traffic rules are a special class of properties that can be reliably applied by people. They involve the responsibility of a driver who can control the speed and direction of a vehicle on the basis of an approximate knowledge of its kinetic state. They do not include conditions that are difficult to assess by the subjective judgment of human drivers, whereas they could be verified by properly instrumented computers. Thus, traffic rules are based on topological considerations rather than quantitative information, such as an accurate comparison of vehicle speeds. Furthermore, they can be formulated as implications, where the implicant is a condition that can be easily checked by a driver, and the conclusion is a constraint

Table 1 Location and itinerary predicates

	$c@x$
$x = o:$	$c.p = o.p$
$x = u:$	$c.p = u$
$x = e:$	$\exists a.\ c.p = (e, a)$
	$c \rightarrow x$
$x = o:$	$\exists \delta.\ c.p \xrightarrow{c.it[0,\delta].} o.p$
$x = u:$	$\exists \delta.\ c.p \xrightarrow{c.it[0,\delta].} u$
$x = e:$	$\exists \delta.\ \exists a > 0.\ c.p \xrightarrow{c.it[0,\delta].} {}^{\bullet}e \wedge c.p \overset{c.it[0,\delta+a].}{\rightsquigarrow} (e, a)$
	$c \rightsquigarrow x$
$x = o:$	$\exists \delta.\ c.p \overset{c.it[0,\delta].}{\rightsquigarrow} o.p$
$x = u:$	$\exists \delta.\ c.p \overset{c.it[0,\delta].}{\rightsquigarrow} u$
$x = e:$	$\exists \delta.\ \exists a > 0.\ c.p \overset{c.it[0,\delta].}{\rightsquigarrow} {}^{\bullet}e \wedge c.p \overset{c.it[0,\delta+a].}{\rightsquigarrow} (e, a)$

on controllable variables that call for a possible corrective action by the driver.

Given a map G and a set of objects $\mathcal{O}$, we specify traffic rules as formulas of a linear time logic of the following form, where $\Box$ is the *always* time modality and N is the *next* time modality:

$$\Box\ \forall c_1.\ \forall o_2.\ \dots\ \forall o_k.$$
$$\phi(c_1, o_2, \dots, o_k) \implies \mathsf{N}\, \psi(c_1, o_2, \dots, o_k) \qquad (2)$$

A rule says that for any run of the system, the satisfaction of the precondition ϕ implies that the postcondition ψ holds in the next state. Both ϕ and ψ are boolean combinations of state predicates, as defined below. Furthermore, we assume that ψ constrains the speed of a single vehicle c_1 for which the property is applicable, and which we call for convenience the *ego* vehicle.

The rules involve state predicates ϕ in the form of first-order assertions built from variables and object attributes (denoting map positions, segments, reals, etc.) using available primitives on map positions (e.g., rides $\rightsquigarrow$, edge rides $\rightarrow$, distance d, equality $=$), on segments (e.g., concatenation and subsegment extraction), in addition to real arithmetic and Boolean operators.

Moreover, we define auxiliary non-primitive *location* and *itinerary* predicates, which prove useful for the expression of traffic rules. For a vehicle $c \in \mathcal{C}$ and x either an object $o \in \mathcal{O}$, a vertex u or an edge e of the map, we define the predicates $c@x$ (c *is at* x), $c \rightarrow x$ (c *meets* x *along the same edge*), $c \rightsquigarrow x$ (c *meets* x), as in Table 1. Furthermore, for a vehicle $c \in \mathcal{C}$ and non-negative δ let $c.p \oplus_c \delta$ denote the future position of c after traveling distance δ, that is, either $c.p$ if $\delta = 0$ or the position p' such that $c.p \overset{c.it[0,\delta].}{\rightsquigarrow} p'$. We extend $\oplus_c$ to arbitrary future positions of c by taking $(c.p \oplus_c \delta) \oplus_c \delta' \overset{def}{=} c.p \oplus_c (\delta + \delta')$ and we consider the total ordering $\leq_c$ defined as $c.p \oplus_c \delta \leq_c c.p \oplus_c \delta'$ if and only if $\delta \leq \delta'$.

We define the semantics of state predicates ϕ in the usual way, by providing a satisfaction relation σ, $st \vdash \phi$, where σ is an assignment of free variables of ϕ and st is a system state. A complete formal definition can be found in [6]. The semantics of rules is defined on pairs σ, $[st^{(t_i)}]_{i \geq 0}$ consisting of a function σ assigning objects instances to object variables of the formulas, and a run $[st^{(t_i)}]_{i \geq 0}$ for a finite set of objects $\mathcal{O}$. For initial state $st^{(t_0)}$ we define *runs* as sequences of consecutive states $[st^{(t_i)}]_{i \geq 0}$ obtained along the cyclic *ADS* execution as described in Sect. 3.1 and parameterized by the sequence of time points $t_i \overset{def}{=} t_0 + i \cdot \Delta t$, that is, equal to the time for reaching the ith system state.

We provide examples of traffic rules in Table 2. We restrict ourselves to safety rules that characterize boundary conditions that should not be violated by the driver controlling the vehicle speed. Therefore, the preconditions characterize potential conflict situations occurring at intersections, as well as other constraints implied by the presence of obstacles or speed rules, e.g., traffic lights or speed limit signals. The preconditions may involve various itinerary and location predicates and constraints on the speed of the ego vehicle. Moreover, the latter are limited to constraints maintained by the vehicle and involving braking functions in the form $B_c(c.v)\ \#\ k$ where k is a distance with respect to a reference position on the map and $\#$ is a relational symbol $\# \in \{<, \leq, =, \geq, >\}$. Furthermore, the postconditions involve two types of constraints on the speed of the ego vehicle: either speed regulation constraints that limit the distance to full stop, that is $B_{c_1}(c_1.v)$, or speed limitation constraints, which require that the speed $c_1.v$ does not exceed a given limit value.

Note the difference with other approaches using unrestricted linear temporal logic, with “eventually” and “until” operators, to express traffic rules, e.g. [6]. We have adopted the above restrictions because they closely characterize the vehicle safety obligations in the proposed model. Furthermore, as we will show below, traffic rules of this form can be translated into free-space rules that can reinforce the policy managed by the *Runtime*.

5.2 Deriving free-space rules from traffic rules

We show that we can derive from traffic rules limiting the speed of vehicles, rules on free space variables controlled by the *Runtime* such that both the traffic rules and the free-space contract hold.

To express constraints on the free-space variables $c.f$, we use, for vehicles c, auxiliary *limit position* variables $\langle c.\pi \rangle_{c \in \mathcal{C}}$ such that $c.\pi = c.p \oplus_c c.f$. In other words, the limit position $c.\pi$ defines the position beyond which a vehicle should not

Table 2 Traffic rules

1	Enforcing safety distance between following vehicles c_1 and c_2.	$\Box\ \forall c_1.\ \forall c_2.\ c_1 \rightsquigarrow c_2 \implies \mathsf{N}\, B_{c_1}(c_1.v) \leq d(c_1.p, c_2.p)$
2	Coordination within all-way-stop junctions	
(i)	Safe braking of vehicle c_1 approaching a stop so_1.	$\Box\ \forall c_1.\ \forall so_1.\ c_1 \rightarrow so_1 \implies \mathsf{N}\, B_{c_1}(c_1.v) \leq d(c_1.p, so_1.p)$
(ii)	Vehicle c_1 obeys a stop sign when another vehicle c_2 crosses the junction.	$\Box\ \forall c_1.\ \forall so_1.\ \forall c_2.\ c_1@so_1 \wedge c_1.v = 0 \wedge c_2.v > 0 \wedge c_1.p \rtimes c_2.p \implies \mathsf{N}\, c_1.v = 0$
(iii)	If two vehicles c_1, c_2 are waiting before the respective stops so_1, so_2 and c_2 waited longer than c_1, then c_1 has to stay stopped.	$\Box\ \forall c_1.\ \forall so_1.\ \forall c_2.\ \forall so_2.\ c_1@so_1 \wedge c_1.v = 0 \wedge c_2@so_2 \wedge c_2.v = 0\ \wedge c_1.p \rtimes c_2.p \wedge c_1.wt < c_2.wt \implies \mathsf{N}\, c_1.v = 0$
(iv)	If two vehicles c_1, c_2 are waiting before the respective stops so_1, so_2 the same amount of time and c_2 is at an entry with higher priority, then c_1 has to stay stopped.	$\Box\ \forall c_1.\ \forall so_1.\ \forall c_2.\ \forall so_2.\ c_1@so_1 \wedge c_1.v = 0 \wedge c_2@so_2 \wedge c_2.v = 0\ \wedge\ c_1.p \rtimes c_2.p \wedge c_1.wt = c_2.wt \wedge so_1.p \prec_j so_2.p \implies \mathsf{N}\, c_1.v = 0$
3	Coordination using traffic-lights: if vehicle c_1 meets a red traffic light lt_1, it will remain in safe distance.	$\Box\ \forall c_1.\ \forall lt_1.\ c_1 \rightarrow lt_1 \wedge lt_1.color = red \wedge B_{c_1}(c_1.v) \leq d(c_1.p, lt_1.p) \implies \mathsf{N}\, B_{c_1}(c_1.v) \leq d(c_1.p, lt_1.p)$
4	Priority-based coordination of two vehicles c_1 and c_2, whose itineraries meet at merger vertex u.	
(i)	If c_2 cannot stop at u, then c_1 must give way	$\Box\ \forall c_1.\ \forall c_2.\ \forall u.\ c_1 \rightarrow u \wedge B_{c_1}(c_1.v) \leq d(c_1.p, u)\ \wedge c_2 \rightarrow u \wedge B_{c_2}(c_2.v) > d(c_2.p, u) \implies \mathsf{N}\, B_{c_1}(c_1.v) \leq d(c_1.p, u)$
(ii)	If c_1, c_2 are reaching u and c_1 has less priority than c_2, then c_1 must give way.	$\Box\ \forall c_1.\ \forall c_2.\ \forall u.\ c_1 \rightarrow u \wedge B_{c_1}(c_1.v) = d(c_1.p, u) \wedge c_1.p \prec_u c_2.p\ \wedge\ c_2 \rightarrow u \wedge B_{c_2}(c_2.v) = d(c_2.p, u) \implies \mathsf{N}\, B_{c_1}(c_1.v) \leq d(c_1.p, u)$
5	Enforcing speed limits for vehicle c_1	
(i)	If c_1 is traveling in an edge e, then its speed should be lower than the speed limit.	$\Box\ \forall c_1.\ \forall e.\ c_1@e \implies \mathsf{N}\, c_1.v \leq e.v$
(ii)	If c_1 is approaching an edge e, then it controls its speed so that it complies with the speed limit at the entrance of e.	$\Box\ \forall c_1.\ \forall e.\ c_1 \rightarrow e \implies \mathsf{N}\, B_{c_1}(c_1.v) \leq d(c_1.p, {}^{\bullet}e) + B_{c_1}(e.v)$

be according to its contract. It is clear that for given $c.\pi$ and $c.p$, $c.f$ is defined as the distance from $c.p$ to $c.\pi$.

Using the limit position variables $\langle c.\pi \rangle_{c \in \mathcal{C}}$ we can transform structurally any state formula ϕ into a free space formula ϕ_π by replacing constraints on speeds by induced constraints on limit positions as follows, for relational symbol # and t a non-negative real constant:

$$\begin{array}{ll} B_c(c.v) \# d(c.p,x) + t & \mapsto\ c.\pi \#_c\ x \oplus_c t \\ c.v \# t & \mapsto\ c.\pi \#_c\ c.p \oplus_c B_c(t) \end{array}$$

The first case concerns speed regulation constraints bounding the limit position $c.\pi$ relatively to the position x of a fixed or moving obstacle ahead of c, that is, a stop or traffic light sign, a vehicle, etc. The second case concerns speed limitation constraints bounding $c.\pi$ relatively to the current vehicle position $c.p$ and the allowed speed.

Given a state formula ϕ, the following theorem guarantees preservation between properties involving speed constraints and properties involving limit positions, in relation to the vehicle speed contracts.

Theorem 3
The following equivalences hold:

$$\begin{array}{ll} (i) & \phi \Longleftrightarrow (\exists\ c.\pi)_{c \in \mathcal{C}}\ \phi_\pi \wedge \bigwedge_{c \in \mathcal{C}} B_c(c.v) = d(c.p,c.\pi) \\ (ii) & \swarrow\!\phi \Longleftrightarrow (\exists\ c.\pi)_{c \in \mathcal{C}}\ \phi_\pi \wedge \bigwedge_{c \in \mathcal{C}} B_c(c.v) \leq d(c.p,c.\pi) \end{array}$$

where $\swarrow\!\phi$ is the speed-lower closure of ϕ, that is, ϕ where speed constraints of the form $c.v$ # t and $B_c(c.v)$ # $d(c.p,x) + t$ for # $\in \{\geq, >\}$ are removed. (In the above, we used the notation $(\exists\ c.\pi)_{c \in \mathcal{C}}$ to denote the quantifier prefix $\exists c_1.\pi \ldots \exists c_n.\pi$ when $\mathcal{C} = \{c_1, \ldots, c_n\}$).

Proof
(i) Assuming $B_c(c.v) = d(c.p,c.\pi)$, the following equivalences between constraints on speed $c.v$ and derived constraints on limit position $c.\pi$ hold trivially, for any # $\in \{<, \leq, =, \geq, >\}$:

$$\begin{array}{rr} & B_c(c.v) \# d(c.p,x) + t \wedge B_c(c.v) = d(c.p,c.\pi) \\ \Longleftrightarrow & d(c.p,c.\pi) \# d(c.p,x) + t \wedge B_c(c.v) = d(c.p,c.\pi) \\ \Longleftrightarrow & c.\pi \# x \oplus_c t \wedge B_c(c.v) = d(c.p,c.\pi) \end{array}$$

$$\begin{array}{rr} & c.v \# t \wedge B_c(c.v) = d(c.p,c.\pi) \\ \Longleftrightarrow & B_c(c.v) \# B_c(t) \wedge B_c(c.v) = d(c.p,c.\pi) \\ \Longleftrightarrow & d(c.p,c.\pi) \# B_c(t) \wedge B_c(c.v) = d(c.p,c.\pi) \\ \Longleftrightarrow & c.\pi \# c.p \oplus_c B_c(t) \wedge B_c(c.v) = d(c.p,c.\pi) \end{array}$$

This implies, assuming $\bigwedge_c B_c(c.v) = d(c.p,c.\pi)$, that any state formula ϕ is equivalent to the derived constraint ϕ_π on limit positions, that is:

$$\begin{array}{rl} & \phi \wedge \bigwedge_{c \in \mathcal{C}} B_c(c.v) = d(c.p,c.\pi) \\ \Longleftrightarrow & \phi_\pi \wedge \bigwedge_{c \in \mathcal{C}} B_c(c.v) = d(c.p,c.\pi) \end{array}$$

Table 3 Free space rules derived from traffic rules

1	$\Box\ \forall c_1.\ \forall c_2.\ c_1 \rightsquigarrow c_2 \implies \mathsf{N}\, c_1.\pi \leq_{c_1} c_2.p$
2 (i)	$\Box\ \forall c_1.\ \forall so_1.\ c_1 \to so_1 \implies \mathsf{N}\, c_1.\pi \leq_{c_1} so_1.p$
2 (ii)	$\Box\ \forall c_1.\ \forall so_1.\ \forall c_2.\ c_1 @ so_1 \wedge c_1.\pi =_{c_1} c_1.p \wedge c_2.\pi >_{c_2} c_2.p \wedge c_1.p \asymp c_2.p \implies \mathsf{N}\, c_1.\pi =_{c_1} c_1.p$
2 (iii)	$\Box\ \forall c_1.\ \forall so_1.\ \forall c_2.\ \forall so_2.\ c_1 @ so_1 \wedge c_1.\pi =_{c_1} c_1.p \wedge c_2 @ so_2 \wedge c_2.\pi =_{c_2} c_2.p \ \wedge c_1.p \asymp c_2.p \wedge c_1.wt < c_2.wt \implies \mathsf{N}\, c_1.\pi =_{c_1} c_1.p$
2 (iv)	$\Box\ \forall c_1.\ \forall so_1.\ \forall c_2.\ \forall so_2.\ c_1 @ so_1 \wedge c_1.\pi =_{c_1} c_1.p \wedge c_2 @ so_2 \wedge c_2.\pi =_{c_2} c_2.p \ \wedge\ c_1.p \asymp c_2.p \wedge c_1.wt = c_2.wt \wedge so_1.p \prec_j so_2.p \implies \mathsf{N}\, c_1.\pi =_{c_1} c_1.p$
3	$\Box\ \forall c_1.\ \forall lt_1.\ c_1 \to lt_1 \wedge lt_1.color = red \wedge c_1.\pi \leq_{c_1} lt_1.p \implies \mathsf{N}\, c_1.\pi \leq_{c_1} lt_1.p$
4 (i)	$\Box\ \forall c_1.\ \forall c_2.\ \forall u.\ c_1 \to u \wedge c_1.\pi \leq_{c_1} u \ \wedge c_2 \to u \wedge c_2.\pi >_{c_2} u \implies \mathsf{N}\, c_1.\pi \leq_{c_1} u$
4 (ii)	$\Box\ \forall c_1.\ \forall c_2.\ \forall u.\ c_1 \to u \wedge c_1.\pi =_{c_1} u \wedge c_1.p \prec_u c_2.p \ \wedge\ c_2 \to u \wedge c_2.\pi =_{c_2} u \implies \mathsf{N}\, c_1.\pi \leq_{c_1} u$
5 (i)	$\Box\ \forall c_1.\ \forall e.\ c_1 @ e \implies \mathsf{N}\, c_1.\pi \leq_{c_1} c_1.p \oplus_{c_1} B_{c_1}(e.v)$
5 (ii)	$\Box\ \forall c_1.\ \forall e.\ c_1 \to e \implies \mathsf{N}\, c_1.\pi \leq_{c_1} {}^{\bullet}e \oplus_{c_1} B_{c_1}(e.v)$

Then, as ϕ does not involve limit positions, we have:

$$\begin{aligned} \phi &\iff \phi \wedge (\exists c.\pi)_c \bigwedge_{c \in \mathcal{C}} B_c(c.v) = d(c.p, c.\pi) \\ &\iff (\exists c.\pi)_c\ \phi \wedge \bigwedge_{c \in \mathcal{C}} B_c(c.v) = d(c.p, c.\pi) \\ &\iff (\exists c.\pi)_c\ \phi_\pi \wedge \bigwedge_{c \in \mathcal{C}} B_c(c.v) = d(c.p, c.\pi) \end{aligned}$$

(ii) Immediate consequence of (i) by applying $\swarrow$ on both sides. □

Recalling Thm. 1, notice that $B_c(c.v) \leq d(c.p, c.\pi)$ is enforced by safe control policies as $d(c.p, c.\pi) = c.f$. Therefore, any property ϕ is preserved through equivalence only when all the vehicles run with the maximal allowed speed by the distance to their limit positions. Otherwise, the speed-lower closure $\swarrow\phi$ is preserved through equivalence, that is, only the upper bounds on speeds as derived from corresponding bounds on limit positions.

Therefore, all traffic rules of form (2) which, for states satisfying the precondition ϕ, constrain the speed of vehicle c_1 at the next cycle according to constraint ψ, are transformed into free-space rules on limit positions of the form:

$$\begin{aligned} &\Box\ \forall c_1.\ \forall o_2.\ \ldots \forall o_k. \\ &\quad \phi_\pi(c_1, o_2, \ldots, o_k) \implies \mathsf{N}\, \psi_\pi(c_1, o_2, \ldots, o_k) \end{aligned} \tag{3}$$

Notice that the postcondition ψ_π is of the form $c_1.\pi \leq_{c_1} b_\psi(c_1, o_2, \ldots, o_k)$ for a position term b_ψ obtained by the transformation of ψ. For illustration, in Table 3 we provide the corresponding free-space rules derived from the traffic rules in Table 2.

We are now ready to define the *Runtime* free space policy based on traffic rules.

Definition 4 (free-space policy based on traffic rules) Let $\mathcal{R}$ denotes the set of traffic rules of interest, e.g., the ones defined in Table 2. For a current *ADS* state *st* and current limit positions and free spaces $\langle c.\pi, c.f \rangle_{c \in \mathcal{C}}$, the policy computes new limit positions and new free spaces $\langle c.\pi', c.f' \rangle_{c \in \mathcal{C}}$ as follows:

$$\begin{aligned} c.\pi' \overset{def}{=} \min_{\leq c} \{\, \sigma b_\psi \mid [\Box \forall c_1. \forall o_2 \ldots \forall o_k.\ \phi &\implies \\ \mathsf{N}\, \psi] \in \mathcal{R},\ \sigma[c/c_1],\ st &\vdash \phi_\pi \} \\ \cup \{\, e^{\bullet} \mid \exists a < \|e\|,\ c.\pi = (e, a), c &\rightsquigarrow e \} \end{aligned} \tag{4}$$

$$c.f' \overset{def}{=} \delta \text{ such that } c.p \oplus_c \delta = c.\pi' \tag{5}$$

Actually, that means computing for every vehicle c the new limit position $c.\pi'$ as the nearest position with respect to $\leq_c$ from two sets of bounds. The first set contains the bounds σb_ψ computed for all the free space rules derived from the traffic rules in $\mathcal{R}$ and applicable for c at the given state *st*. The second set contains the endpoint $e^{\bullet}$ of the edge e where the current limit position $c.\pi$ is located. It is needed to avoid "jumping" over $e^{\bullet}$, even though this is allowed by application of the rules, as $e^{\bullet}$ may be a merger node and should be considered for solving potential conflicts. Then, we define the new free space $c.f'$ as the distance δ from the current position $c.p$ to the new limit position $c.\pi'$, measured along the itinerary of c.

Note that if the free-space policy respects the assume-guarantee contract of the *Runtime* from Def. 1, it further guarantees the satisfaction of all traffic rules from $\mathcal{R}$ where both the pre- and the postcondition ϕ and ψ are speed-lower closed formulas. First, conformance with respect to the contract is needed to obtain the invariants $B_c(c.v) \leq c.f = d(c.p, c.\pi)$ according to Thm. 1. Second, these invariants ensure preservation through equivalence between speed-lower closed formula and derived formula on limit positions, according to Thm. 3. Third, the free-space policy ensures the satisfaction of the derived free-space rules, that is, by construction, it chooses limit positions ensuring postconditions ψ_π hold whenever preconditions ϕ_π hold. As these formulas are preserved through equivalence, it leads to the satisfaction of the original traffic rule.

5.3 Correctness with respect to the free-space contract

We prove correctness, that is, conformance with the assume-guarantee contract of Def. 1, of the free-space policy obtained by the application of the traffic rules from Table 2 excluding the one concerning traffic lights. For this rule, we need additional assumptions taking into account the light functioning and the behavior of the crossing vehicles.

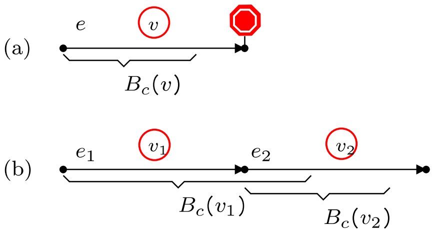

Fig. 6 Explaining restrictions on speed limits

First, we assume that the vehicle braking dynamics are compatible with the speed limits associated with the map segments, that means:

- For any edge e leading to a junction (and henceforth a stop sign) or a merger, vertex holds $B_c(e.v) \le \|e\|$, for any vehicle $c \in \mathcal{C}$ (see Fig. 6(a)),
- For any consecutive edges e_1, e_2 holds $B_c(e_1.v) \le \|e_1\| + B_c(e_2.v)$, for any vehicle $c \in \mathcal{C}$ (see Fig. 6(b)) i.e., between two consecutive speed limit changes, there is sufficient space to adapt the speed.

Second, we call an *ADS* state $\langle st_o \rangle_{o \in \mathcal{O}}$ *consistent with limit positions* $\langle c.\pi \rangle_{c \in \mathcal{C}}$ iff for every vehicle $c \in \mathcal{C}$:

- The limit position is ahead of the current vehicle position, that is, $c.p \le_c c.\pi$,
- There is no stop sign located strictly between the current vehicle position and the limit position, that is, $c.p <_c so.p <_c c.\pi$ does not hold for any stop so,
- The limit position conforms to the speed limits of the current edge (e_1) and next edge (e_2) on the itinerary of c, that is, $d(c.p, c.\pi) \le B_c(e_1.v)$ and $d(c.p, c.\pi) \le d(c.p, {}^\bullet e_2) + B_c(e_2.v)$.

For vehicle c and position p located ahead of c on its itinerary, we denote by $ahead_c(p) \overset{def}{=} Ahead(c.p, c.it, f)$ for $f = d(c.p, p)$, that is, the space ahead of c until position p. In particular, $ahead_c(c.\pi) = Ahead(c.p, c.it, c.f)$ holds as $c.\pi = c.p \oplus_c c.f$. The following lemma provides the basic conditions that guarantee the correctness of the free space policy.

Lemma 1

Let st be an ADS *state and* $\langle c.\pi \rangle_{c \in \mathcal{C}}$ *be limit positions such that:*

- *the state st is consistent with the limit positions* $\langle c.\pi \rangle_{c \in \mathcal{C}}$,
- *the spaces ahead up to the limit positions are non-crossing, that is,* $\biguplus_{c \in \mathcal{C}} ahead_c(c.\pi)$.

Let $\langle c.\pi' \rangle_{c \in \mathcal{C}}$ *be the new limit positions computed for state st and* $\langle c.\pi \rangle_{c \in \mathcal{C}}$ *according to the free space policy. Then, the following hold:*

(a) $c.\pi \le_c c.\pi'$, *for every vehicle* $c \in \mathcal{C}$,

(b) *the state st is consistent with the new limit positions* $\langle c.\pi' \rangle_{c \in \mathcal{C}}$,

(c) *the spaces ahead up to the new limit positions are non-crossing, that is,* $\biguplus_{c \in \mathcal{C}} ahead_c(c.\pi')$.

Proof

(a) Let us fix an arbitrary vehicle c. According to the free-space policy the limit position $c.\pi'$ is defined as the nearest position among several bounds on the itinerary of c. We will show that, in all situations, these bounds b are at least as far as the current limit position $c.\pi$, that is $c.\pi \le_c b$ and hence the result. First, consider all applicable traffic rules and their associated bounds, as presented in Table 3:

- Rule 1: The bound b is defined as the position $c_2.p$ of the heading vehicle c_2. The constraint $c.\pi \le_c c_2.p$ holds because the spaces ahead to the limit positions are assumed non-crossing.
- Rule $2(i)$: The bound b is defined as the position $so.p$ of the stop sign located ahead of c. The constraint $c.\pi \le_c so.p$ holds because the state st is assumed consistent, that is, no stop sign between the vehicle and the current limit position.
- Rule $3(ii, iii, iv)$: The bound b is defined as the current position $c.p$, which is equal to the current limit $c.\pi$ due to the constraint $c.v = 0$ in their respective preconditions, therefore, obviously $c.\pi \le_c c.\pi$.
- Rule $4(i, ii)$: The bound b is defined as the merger node u and the preconditions of the rules contain respectively $c.\pi \le_c u$ and $c.\pi = u$.
- Rule $5(i, ii)$: Assume c is located on some edge e_1 and the next edge is e_2. As st is consistent with current limit positions, this implies $d(c.p, c.\pi) \le B(e_1.v)$ and $d(c.p, c.\pi) \le d(c.p, {}^\bullet e_2) + B_c(e_2.v)$. This is equivalent to $c.\pi \le_c c.p \oplus_c B_c(e_1.v)$ and $c.\pi \le_c {}^\bullet e_2 \oplus_c B_c(e_2.v)$, which are the constraints for speed limit rules.

Second, we consider the position $e^\bullet$, where the edge e contains the limit position $c.\pi$. That is, $c.\pi = (e, a)$ for some $a < \|e\|$ and hence $c.\pi \le_c (e, \|e\|) = e^\bullet$.

(b) We know that $c.p \le_c c.\pi$ for all vehicles c. Then, from (a) above we obtain immediately that $c.p \le_c c.\pi'$ for all vehicles c. Moreover, according to traffic rule $2(i)$, new limit positions $c.\pi'$ cannot move over a stop sign so unless the vehicle c is at so. That means, stop signs cannot be inserted between a vehicle and their limit positions. Finally, according to traffic rules $5(i, ii)$, the new limit positions are at most as far as the bounds for the current and next edge speed limits. The restriction on speed limits from Fig. 6(b) is needed whenever vehicles are located at vertices, because of the transition from the current to the next edge.

(c) We have seen that limit positions can either stay unchanged or move forward on the itineraries of their respective vehicles. In order to show that spaces until the new limit positions are non-crossing, we need to focus on moving limit

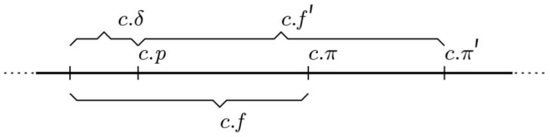

Fig. 7 Update of limit position and free space along the itinerary of c

positions. We prove the following facts, which guarantee that these spaces remain indeed non-crossing:

- No limit position exceeds the current position of a vehicle. Actually, this is explicitly excluded by the traffic rule 1. Therefore, no space is extended so as to overlap with an existing space.
- Two limit positions never move simultaneously so that the corresponding spaces cross each other. First, crossing may happen at junctions i.e., if two limit positions simultaneously enter the same junction. Such situations are excluded by traffic rules $2(i,ii,iii,iv)$, which force vehicles to stop and then solve conflicts between them in a deterministic manner. Second, crossing may happen at merger nodes i.e., if two limit positions will simultaneously move over a merger node. These situations are explicitly excluded by traffic rule $4(i,ii)$, which solve conflicts at merger nodes based on priorities, plus the extra rule forbidding limit positions to jump over map vertices. □

The next proposition states the correctness of the free-space policy constructed from traffic rules.

Proposition 3
The free space policy respects the safety contract for the Runtime *provided the initial* ADS *state is consistent with initial limit positions.*

Proof
Consider a state *st* consistent with limit positions $\langle c.\pi\rangle_{c\in\mathcal{C}}$ and satisfying the assumptions stated in Def. 1. We are therefore satisfying the premises of Lemma 1 and then the new limit positions $\langle c.\pi'\rangle_{c\in\mathcal{C}}$ satisfy the conclusions (a)–(c), as stated by the Lemma 1.

Then, for any vehicle c, using (a), we know that the new limit position satisfies $c.\pi \leq_c c.\pi'$. Consequently, the new free space $c.f'$, which is the distance from the current vehicle position to the next limit position, satisfies $c.f' \geq c.f - c.\delta$, that is, the first assertion of the *Runtime* guarantee in Def. 1. This inequality can be understood from Fig. 7, which depicts the generic situation for a vehicle c.

Moreover, using (c), we obtain the second assertion of the *Runtime* guarantee, that is, $\biguplus_{c\in\mathcal{C}} Ahead(c,c.f',)$.

Finally, using (b) we know that the state *st* is consistent with new limit positions $\langle c.\pi'\rangle_{c\in\mathcal{C}}$. Then, as long as the vehicles move forward into their free spaces according to their contract, the system state st' is consistent with respect to these new limit positions. So, eventually, at the beginning of the next *Runtime* cycle we are back to the initial situation considered for *st* and limit positions $\langle c.\pi\rangle_{c\in\mathcal{C}}$. That means essentially that state consistency with respect to limit positions is an inductive invariant in the system, so it can be safely assumed to hold at any time, provided it holds initially. □

5.4 Non-blocking free-space policies

The free-space policy based on traffic rules is not non-blocking. While continuously meeting traffic rules, an *ADS* can potentially evolve into a blocking state. For example, when a subset of vehicles is blocked in a roundabout so that none of them can move further and eventually exit the roundabout and all other vehicles are waiting to enter the same roundabout. The non-blocking contract can be however fulfilled if the *Runtime* monitors the traffic from a global point of view and prevents situations as described above.

Consider that there exists a constant f_{min} such that for all map edges e, $\|e\| > f_{min}$ and $e.v > B^{-1}(f_{min})$. Furthermore, let $E_\times \subseteq E$ be the subset of edges belonging to junctions. An elementary directed path $\gamma = e_1 e_2 \ldots e_m \in E^*$ is *critical* either if (i) it is a circuit visiting at most once every junction, or (ii) it ends and returns at the same junction, while visiting at most once every other junction. Let $\#_j(\gamma)$ be the number of distinct junctions of a critical path γ. We define the capacity $w(\gamma)$ of a critical path γ as the least number of vehicles that could "block" the critical path γ minus one:

$$w(\gamma) \overset{def}{=} \#_j(\gamma) + \left(\sum\nolimits_{e\in\gamma\setminus E_\times} \lfloor \|e\| \;/\; f_{min} \rfloor\right) - 1 \qquad (6)$$

That is, we assume that a junction can be blocked by one vehicle, and a non-junction edge e can be blocked by $\lfloor \|e\| \;/\; f_{min} \rfloor$ vehicles.

Proposition 4
The free-space policy derived from traffic rules respects the non-blocking contract for the Runtime *as long as the number of vehicles on every critical path γ is lower than the path capacity $w(\gamma)$.*

Proof
Consider a state where all vehicles are stopped. Consider an arbitrary vehicle c_0. Two situations can happen, respectively (i) c_0 is waiting at an entry of a junction since another vehicle is already in, or (ii) c_0 is waiting behind another stopped vehicle. This means that in both cases, some other vehicle c_1 is actually blocking c_0. We continue the same reasoning for c_1, and can find another vehicle c_2 blocking it, and so on.

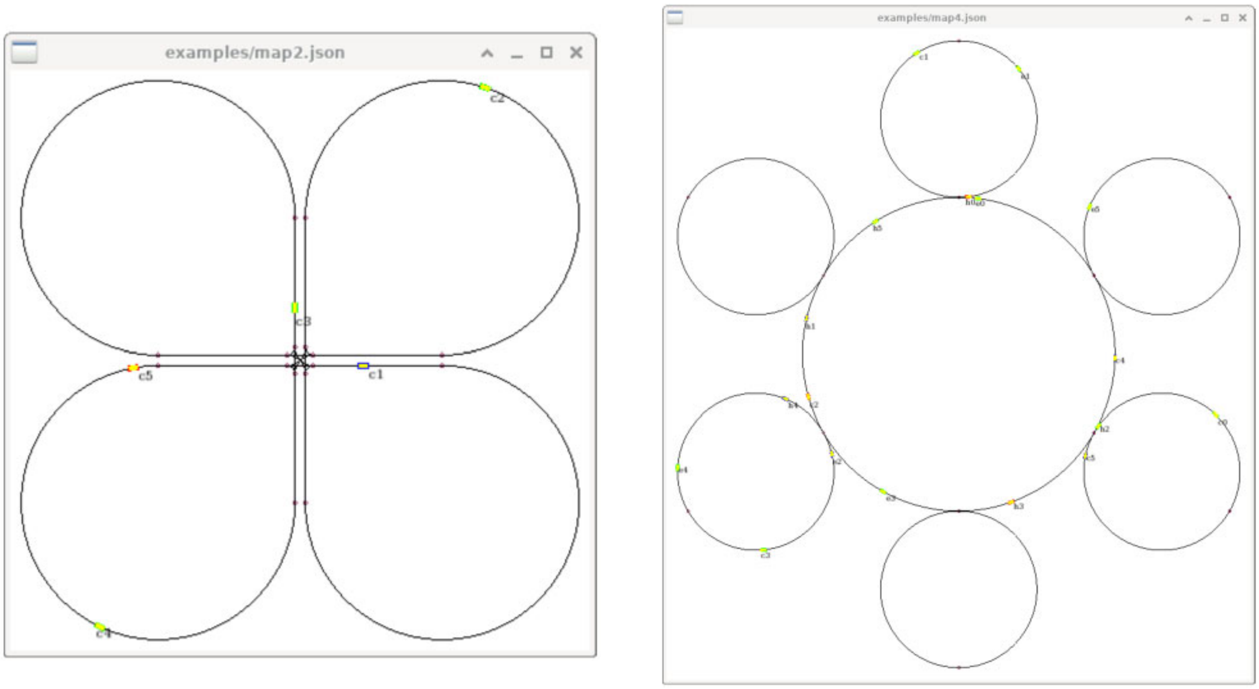

Fig. 8 Snapshots of *ADS* simulation

As the number of vehicles is finite, we finally find a circular chain $c_i, c_{i+1}, \ldots, c_n, c_i$ of vehicles that block each other successively.

Any chain of vehicles as above is located on some critical path γ, that is, either a circuit or a path going twice through the same junction of the map. Assume the number of vehicles on the path γ is lower than the path capacity $w(\gamma)$. We distinguish two situations:

– *The critical path contains only junction edges.* Then, as the path capacity is $\#_j(\gamma) - 1$, at least one of the junctions must be clear of vehicles. Therefore, at least one of the vehicles waiting at that junction entries will obtain the free space to proceed. That is, the vehicle in the preceding junction or another one waiting at some other entry, depending on the applicable traffic rules.

– *The critical path contains both junction and non-junction edges.* If at least one of the junctions is not empty, we can reason as in the previous case and find a vehicle that can proceed. Otherwise, as every junction contains one vehicle, the number of vehicles on the remaining non-junction edges is at most $\sum_{e \in \gamma \setminus E_\times} \lfloor \|e\| / f_{min} \rfloor - 1$. That means, one can find a non-junction edge with more than f_{min} unused space. That space is eventually allocated to either one of the vehicles waiting on the edge, or to a vehicle waiting to enter the edge (that is, exiting from a junction, entering through a merger node, etc.), depending on applicable rules. □

6 Experiments

We are currently developing a prototype for *ADS* simulation implementing the speed and free space policies introduced. It takes as inputs (i) a map defined as an annotated metric graph with segments that are parametric lines or arcs, and (ii) an initial state containing the positions, the initial speeds, and the itineraries of a number of vehicles. The simulation proceeds then as explained in Sect. 3 by, using the specific control policies from Sects. 4 and 5. The prototype uses the SFML[1] library for graphical rendering of states. Figure 8 presents simulation snapshots for two simple examples running respectively 5 and 18 vehicles. Note that performance scales up smoothly as the number of vehicles increases because each rule is applied on a linear finite horizon structure. Furthermore, compared to simulators that particularize an *ego* vehicle, the *Runtime* treats all vehicles the same and ignores their speed control policy as long as they fulfill their contract.

7 Discussion

The paper studies results for the correct by design coordination of *ADS* based on assume-guarantee contacts. The coordination follows a two-step synchronous interaction protocol

[1] Simple and Fast Multimedia Library, https://www.sfml-dev.org/.

between vehicles and a *Runtime* that, based on the distribution of vehicles on a map, computes the corresponding free spaces. A first result characterizes safe control policies as the combination of assume-guarantee contracts for vehicles and the *Runtime*. This result is then specialized by showing how policies consistent with their respective contracts can be defined for vehicles and the *Runtime*. In particular, for vehicles, we provide a principle for defining speed policies and, for the *Runtime*, we compute free-space policies that conform to a set of traffic rules. The results are general and overcome the limitations of a posteriori verification. They can be applied to *ADS* involving a dynamically changing number of vehicles. In addition, they rely on a general map-based environment model, which has been extensively studied in [6]. Control policies for vehicles and the *Runtime* can be implemented efficiently. In particular, the speed policy has been tested in various implementations [24, 25] and found to be not only safe, but also closer to the optimum when refining the space of possible accelerations.

Note that the results can be extended, with slight modifications, to maps where the segments are curves or regions to express traffic rules involving properties of two-dimensional space, for example for passing maneuvers. For example, if we consider region maps, their segments are regions of constant width centered on curves. Itineraries, free spaces, and $B(v)$ will be regions. The relationship $B(v) \leq f$ becomes $B(v) \subseteq f$ and the addition of lengths of segments should be replaced by the disjoint union of the regions they represent. The speed control policy remains unchanged in principle, but requires a function computing the distance traveled in a region. Finally, the runtime verification of the disjointness of free spaces may incur a computational cost, depending on the accuracy of the region representation.

The presented results provide a basis for promising developments in several directions. One direction is to extend the results to achieve correctness by design for general properties. We have shown that traffic rules, which are declarative properties of vehicles, can be abstracted into safety constraints on free spaces. In this way, we solved a simple synthesis problem by transforming a "static" constraint on vehicle speed into a "dynamic" constraint on shared resources.

An interesting question that should be further investigated is whether the method can be extended to more general properties involving the joint obligation of many vehicles. For example, we can require that for any pair of vehicles c_1 and c_2 that are sufficiently close, the absolute value of the difference between their speeds is less than a constant k, i.e., $abs(c_1.v - c_2.v) \leq k$. This can be achieved by a free space constraint that gives more free space to the vehicle with the lower speed, assuming that vehicle speed policies are not "lazy" and use as soon as possible the available space.

For general properties involving more than one vehicle, it seems realistic to translate them directly into free-space constraints that will enforce the constraints processed by the *Runtime* to ensure the safe control policy. In particular, in addition to safety properties, we could devise free-space policies that optimize criteria such as road occupancy and uniform separation for a given group of vehicles e.g. platoon systems studied in [9]. Note that achieving non-blocking control is such a property that involves the application of occupancy criteria.

Another direction is to move from centralized to distributed coordination with many runtimes. It seems possible to partition traffic rules according to the geometric scope of their application, e.g., a specific runtime could control access to each junction. Finally, the *Runtime* can be used as a monitor to verify that the vehicle speed policies of an *ADS* are safe and respect the given traffic rules.

References

1. ASAM OpenDRIVE® open dynamic road information for vehicle environment. Tech. Rep. V 1.6.0, ASAM e.V, (2020) https://www.asam.net/standards/detail/opendrive
2. Bagschik, G., Menzel, T., Maurer, M.: Ontology based scene creation for the development of automated vehicles. In: Intelligent Vehicles Symposium, pp. 1813–1820. IEEE, Los Alamitos (2018)
3. Beetz, J., Borrmann, A.: Benefits and limitations of linked data approaches for road modeling and data exchange. In: EG-ICE, Lecture Notes in Computer Science, vol. 10864, pp. 245–261. Springer, Berlin (2018)
4. Benveniste, A., Caillaud, B., Nickovic, D., Passerone, R., Raclet, J., Reinkemeier, P., Sangiovanni-Vincentelli, A.L., Damm, W., Henzinger, T.A., Larsen, K.G.: Contracts for system design. Found. Trends Electron. Des. Autom. **12**(2–3), 124–400 (2018)
5. Bozga, M., Sifakis, J.: Correct by design coordination of autonomous driving systems. In: ISoLA (3). Lecture Notes in Computer Science, vol. 13703, pp. 13–29. Springer, Berlin (2022)
6. Bozga, M., Sifakis, J.: Specification and validation of autonomous driving systems: a multilevel semantic framework. In: Principles of Systems Design. Lecture Notes in Computer Science, vol. 13660, pp. 85–106. Springer, Berlin (2022)
7. Butz, M., Heinzemann, C., Herrmann, M., Oehlerking, J., Rittel, M., Schalm, N., Ziegenbein, D.: SOCA: domain analysis for highly automated driving systems. In: ITSC, pp. 1–6. IEEE, Los Alamitos (2020)
8. Chatterjee, K., Henzinger, T.A.: Assume-guarantee synthesis. In: TACAS, Lecture Notes in Computer Science, vol. 4424, pp. 261–275. Springer, Berlin (2007)
9. El-Hokayem, A., Bensalem, S., Bozga, M., Sifakis, J.: A layered implementation of DR-BIP supporting run-time monitoring and analysis. In: SEFM, Lecture Notes in Computer Science, vol. 12310, pp. 284–302. Springer, Berlin (2020)
10. Esterle, K., Gressenbuch, L., Knoll, A.C.: Formalizing traffic rules for machine interpretability. In: CAVS, pp. 1–7. IEEE, Los Alamitos (2020)
11. Hilscher, M., Linker, S., Olderog, E., Ravn, A.P.: An abstract model for proving safety of multi-lane traffic manoeuvres. In: ICFEM. Lecture Notes in Computer Science, vol. 6991, pp. 404–419. Springer, Berlin (2011)

12. Karimi, A., Duggirala, P.S.: Formalizing traffic rules for uncontrolled intersections. In: ICCPS, pp. 41–50. IEEE, Los Alamitos (2020)
13. Kress-Gazit, H., Pappas, G.J.: Automatically synthesizing a planning and control subsystem for the DARPA urban challenge. In: CASE, pp. 766–771. IEEE, Los Alamitos (2008)
14. Mavridou, A., Katis, A., Giannakopoulou, D., Kooi, D., Pressburger, T., Whalen, M.W.: From partial to global assume-guarantee contracts: compositional realizability analysis in FRET. In: FM. Lecture Notes in Computer Science, vol. 13047, pp. 503–523. Springer, Berlin (2021)
15. Meyer, B.: Applying "design by contract". Computer **25**(10), 40–51 (1992)
16. Poggenhans, F., Pauls, J., Janosovits, J., Orf, S., Naumann, M., Kuhnt, F., Mayr, M.: Lanelet2: a high-definition map framework for the future of automated driving. In: ITSC, pp. 1672–1679. IEEE, Los Alamitos (2018)
17. Rizaldi, A., Althoff, M.: Formalising traffic rules for accountability of autonomous vehicles. In: ITSC, pp. 1658–1665. IEEE, Los Alamitos (2015)
18. Rizaldi, A., Keinholz, J., Huber, M., Feldle, J., Immler, F., Althoff, M., Hilgendorf, E., Nipkow, T.: Formalising and monitoring traffic rules for autonomous vehicles in isabelle/hol. In: IFM. Lecture Notes in Computer Science, vol. 10510, pp. 50–66. Springer, Berlin (2017)
19. Rizaldi, A., Immler, F., Schürmann, B., Althoff, M.: A formally verified motion planner for autonomous vehicles. In: ATVA, Lecture Notes in Computer Science, vol. 11138, pp. 75–90. Springer, Berlin (2018)
20. Saoud, A., Girard, A., Fribourg, L.: Assume-guarantee contracts for continuous-time systems. Automa **134**, 109910 (2021)
21. Schwarting, W., Alonso-Mora, J., Rus, D.: Planning and decision-making for autonomous vehicles. Annu. Rev. Control Robot. Auton. Syst. **1**, 187–210 (2018). Https://doi.org/10.1146/annurev-control-060117-105157
22. Sharf, M., Besselink, B., Molin, A., Zhao, Q., Johansson, K.H.: Assume/guarantee contracts for dynamical systems: Theory and computational tools CoRR (2020). arXiv:2012.12657
23. Sun, M., Bakirtzis, G., Jafarzadeh, H., Fleming, C.: Correct-by-construction: a contract-based semi-automated requirement decomposition process. CoRR (2019). arXiv:1909.02070
24. Wang, Q., Li, D., Sifakis, J.: Safe and efficient collision avoidance control for autonomous vehicles. In: MEMOCODE, pp. 1–6. IEEE, Los Alamitos (2020)
25. Wang, Q., Zheng, X., Zhang, J., Sifakis, J.: A hybrid controller for safe and efficient collision avoidance control CoRR (2021). https://arxiv.org/abs/2103.15484. arXiv:2103.15484
26. Waqas, M., Murtaza, M.A., Nuzzo, P., Ioannou, P.: Correct-by-construction design of adaptive cruise control with control barrier functions under safety and regulatory constraints (2022). https://arxiv.org/abs/2203.14110
27. Wongpiromsarn, T., Karaman, S., Frazzoli, E.: Synthesis of provably correct controllers for autonomous vehicles in urban environments. In: ITSC, pp. 1168–1173. IEEE, Los Alamitos (2011)
28. Wongpiromsarn, T., Topcu, U., Murray, R.M.: Receding horizon temporal logic planning. IEEE Trans. Autom. Control **57**(11), 2817–2830 (2012)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.