Specifying and Monitoring Safe Driving Properties with Scene Graphs

Felipe Toledo*[1], Trey Woodlief*[1], Sebastian Elbaum[1], and Matthew B. Dwyer[1]

***Abstract*— With the proliferation of autonomous vehicles (AVs) comes the need to ensure they abide by safe driving properties. Specifying and monitoring such properties, however, is challenging because of the mismatch between the semantic space over which typical driving properties are asserted (e.g., vehicles, pedestrians, intersections) and the sensed inputs of AVs. Existing efforts either assume for such semantic data to be available or develop bespoke methods for capturing it. Instead, this work introduces a framework that can extract scene graphs (SGs) from sensor inputs to capture the entities related to the AV, and a domain-specific language that enables building propositions over those graphs and composing them through temporal logic. We implemented the framework to monitor for specification violations of 3 top AVs from the CARLA Autonomous Driving Leaderboard, and found that the AVs violated 71% of properties during at least one test. Artifact available at `https://github.com/less-lab-uva/SGSM`.**

I. INTRODUCTION

Autonomous vehicles (AVs) are quickly approaching wide-spread public-road deployment, with several companies already leveraging fleets of AV taxis in multiple US cities [1], [2]. However, deployments of full AV systems have led to multiple human and animal fatalities [3], [4], [5], [6], [7] and have shown failures while interacting with emergency vehicles [6], even when the system correctly identifies the emergency vehicle [8]. While some analysis from the companies deploying AVs suggests that AVs are involved in fewer collisions that pose risk of injury compared to human drivers [9], [10], we continue to see AVs violate required driving behavior with grave consequences.

Ideally, AVs would be deployed without latent faults due to extensive validation and verification [11], [12]. However, the inherent complexities of these systems and the long-tail of potential scenarios make it infeasible to provide complete and strong guarantees [13], [14]. These limitations have motivated the use of runtime monitors that can evaluate compliance of safety specifications during deployment [13], [14], [15], [16], [17]. However, current monitoring mechanisms are inadequate for checking driving behavior as they cannot account for the spatiotemporal distribution of entities (e.g., other vehicles, pedestrians, traffic signals) that may influence the AV driving behavior, and which can only be obtained from complex multi-dimensional sensors like camera and LiDAR. Alternatively, approaches that do account for driving behaviors do it through bespoke, handcrafted translation between the monitor's input, e.g. sensor input, or internal system state, and the semantics of the safety specifications, limiting generalizability (as per related work in Section V).

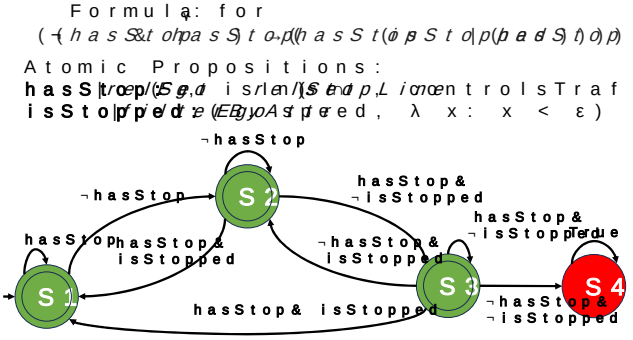


Fig. 1: LTL$_f$ for safe driving property, Atomic Propositions over the image sensor data, and DFA for property ψ_9

A key challenge with developing monitors for the driving behavior of AVs is the mismatch between the semantic space over which typical road properties are asserted (e.g., cars, stop lights, intersections) and the input space of AVs which are typically in the form of sensed data (e.g., images, radar, point clouds). As a motivating example, consider the following rule (ψ_9 in Table I) from the Virginia Driving Code § 46.2-821 "The driver of a vehicle approaching an intersection on a highway controlled by a stop sign shall, immediately before entering such intersection, stop at a clearly marked stop line [...]" [18]. Evaluating this property requires extracting information about road lanes, stopping signals, e.g. stop signs, painted markers, etc., which lanes the signals affect, and if the vehicle occupies those lanes.

To address these limitations, and building on our previous work on SGs [19], we propose a framework for SG Safety Monitoring, SGSM, that enables the specification of road properties for AVs and their automated synthesis as part of a system monitor. The approach builds on two key domain-specific components: 1) a spatial scene graph generator (SGG) that can extract rich scene representations from sensor inputs for the AV domain into SGs that abstract the entities related to the AV, and 2) a domain-specific language (DSL) that enables a developer to define programmable queries over the SG and compose the output of those queries as part of discrete metric temporal logic properties that can be monitored at runtime. Together, the SG and DSL offer a rich space to express common road properties relevant to AVs that can be automatically encoded as a runtime monitor.

Returning to the motivating example, Fig. 1 shows the safety specification described in linear temporal logic over finite traces (LTL$_f$), the atomic propositions (APs) expressed in our DSL, and the deterministic finite automaton (DFA) automatically synthesized from the LTL$_f$ formula. Fig. 2

*Equal contribution [1]University of Virginia, USA {ft8bn, adw8dm, selbaum, matthewbdwyer}@virginia.edu

This work was funded in part by NSF through grant #2312487 and U.S. Army Research Office under grant number W911NF-24-1-0089. Trey Woodlief was supported by a University of Virginia SEAS Fellowship.

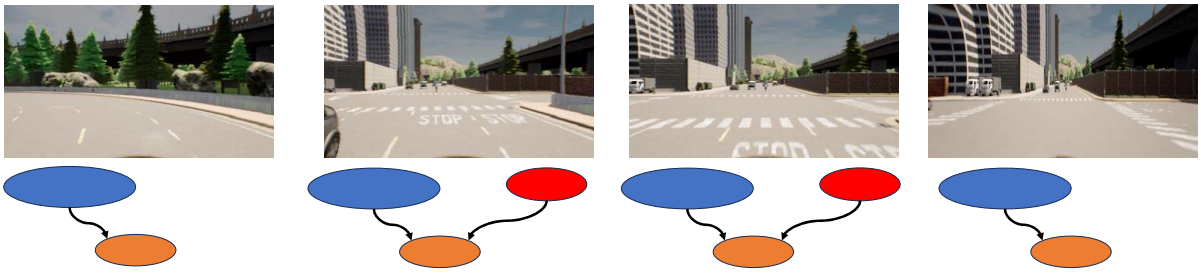


Fig. 2: AV (TCP) running an intersection without stopping. Top: AV Camera Images. Middle: sub-SG for checking safety property. Bottom: Atomic Propositions evaluated from SG and updated state of the DFA shown in Fig. 1 leading to violation.

shows a snippet of a time sequence in which an AV passes an intersection controlled by a stop line without stopping. The top row shows the AV's camera input, while the second row shows the subgraph of the SG relevant to the property extracted from each input image. As the AV approaches the intersection, the stop line appears in the graph with the relationship that it "controlsTrafficOf" the lane that the ego vehicle (ego from now on) is in; yet, ego's velocity remains consistent. The APs and states shown at the bottom correspond to the transitions and state of the DFA at that time; note that as the input sequence progresses, and the stop line is included in the SG but ignored by the AV, the DFA moves toward and finally enters the failure state, S4, indicating a violation.

We introduce the first domain-cognizant, general, and extendable approach for the specification of AV safety driving properties that can be encoded for automatic monitoring during runtime. The approach is domain-cognizant in that it bridges the gap from raw sensor data to primitive propositions that capture domain concepts. It is general in that it is independent of the AV implementation, only requiring access to external inputs and outputs, e.g. sensor data and AV control commands, from which SGs can be derived. It is extendable in that the DSL building blocks can be combined to encode properties beyond the ones we study. We implemented the approach in CARLA [20] to explore its capabilities in simulation for 3 AVs from the CARLA leaderboard competition. We find that these AV systems, though highly performant under the existing competition metrics, consistently violate driving rules—in 50% of executions the AV crossed into opposing traffic (ψ_1) and in 73% of executions the AV ignored a stop sign (ψ_9).

II. BACKGROUND

We briefly survey work that is foundational to our approach, including SGG to extract scene semantics, formulations of propositions over graphs, and temporal logic to specify sequences of proposition values.

A. Scene Graph Generation (SGG)

SGG is an emerging area of research focused on extracting relationships between objects from sensor data, e.g., from an image input inferring a pedestrian is on a crosswalk. SGs are directed graphs [21], with a vertex set V that represents the set of entities captured by a sensor, e.g., camera or LiDAR, and a set of directed edges $(u, v) \in E$ describing their relationships. More formally, an SG,

$$G = (V, E : V \mapsto V, Ego \in V, \\ kind : V \mapsto K, rel : E \mapsto R, att : V \cup E \mapsto M)$$

has a distinguished Ego vertex and functions to access the entity $kind$ of a vertex, the relation encoded by an edge, and attribute values of vertices and edges. A map M is used to associate attribute values with each type of attribute.

SGGs can be configured to work with different sets and parameterizations of entities and relationships, and can capture additional attribute information, e.g., the color of the traffic light. This allows SGs to be tailored to different domains. In recent years, many SGG techniques have been developed [22] leveraging object detection systems (e.g. [23], [24]) to detect different entities, and then extract relationships between them. In the realm of AVs, more tailored SGGs have been proposed [25], [26], [27], that leverage domain-specific semantics like road types, vehicle types, and static or dynamic entities. Our framework uses an SGG to extract graph based abstractions of sensor data from the world, and our study builds on an SGG that operates in CARLA.

B. Graph Properties

There is a rich literature on methods for specifying properties of graphs. Given the relational nature of graphs, properties could be specified as queries in relational algebra [28] or in more specialized graph query languages built on relational algebra primitives [29], [30]. Using such methods one can formulate a wide range of property specifications. For example, one can express that "a graph contains a stop signal that controls the lane ego is in" by combining primitives relational *join* and set *intersection* as follows:

$$join(Ego, \texttt{isIn}) \cap join(\texttt{stopsignal}, \texttt{controls}) \neq \emptyset$$

where $\texttt{stopsignal} = \{v : v \in V \wedge kind(v) = \text{stop signal}\}$.

In this work, we focus on core primitives that can be used to specify properties like the one described above. In addition to standard set operations, those primitives include join (relSet) and a primitive that allows selecting a subset of vertices based on properties of their attributes (filterByAttr). More complicated properties can be expressed over paths by

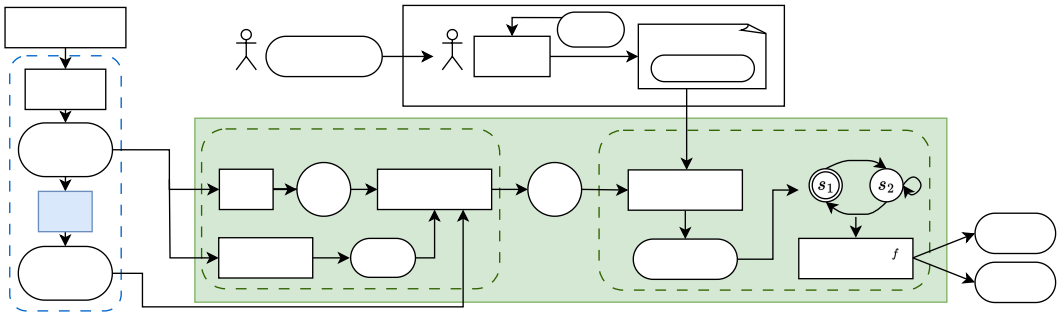


Fig. 3: SGSM Framework overview. SGSM offline phase at the top, and online phase at the bottom.

composition and iteration using these primitives. Executable specifications built in this way are appropriate for runtime monitoring, in contrast to more declarative approaches [30].

C. Linear Temporal Logic

Linear Temporal Logic (LTL) is a formal language that has been widely used for modeling and analyzing systems with temporal aspects, including embedded and cyber-physical systems [31], [32], [33]. An LTL formula ψ, is satisfied by an infinite sequence of truth valuations of APs [34]. There are **logic operators**: *And* ($\wedge$), *Or* ($\vee$), *Not* ($\neg$), etc., and **temporal operators**: *Next* ($\mathcal{X}$), *Until* ($\mathcal{U}$), *Always* ($\mathcal{G}$), *Eventually* ($\mathcal{F}$). By leveraging these range of operators, LTL allows for the precise specification of the system's behavior over time. For runtime monitoring, we use LTL over discrete, finite traces (LTL$_f$) [35]. An LTL$_f$ formula can be automatically converted to a DFA that validates whether a finite trace satisfies the property [36], [37] as shown in Fig. 1. LTL$_f$ does not provide a language for specifying the APs themselves; rather, the APs must be evaluated before being consumed by the LTL$_f$ formula.

III. FRAMEWORK

SGSM (Fig. 3) has two phases to enable the specification and runtime monitoring of driving properties for AVs.

A. Overview

1) Offline: SGSM assists developers in formally specifying driving properties prepared by domain experts, such as those found in a driving manual, in a computable format that is checkable at runtime. The assistance comes in the form of a DSL, called Scene Graph Language (SGL), that allows the developer to define LTL$_f$ formulas and evaluate propositions on the SG and ego's state. The developers begin by converting the specification into an SGL function, and define the sampling rate at which the property needs to be checked. Once the specification is written as an SGL function, it is incorporated into a system monitor during the online phase to check for property violations.

2) Online: As ego senses the environment, it provides data to the system under test (SUT) to produce a decision output for the vehicle. SGSM provides a runtime monitor (green box) to check for property violations by processing sensor data and appending the decision output to create SG*. It then evaluates the SGL function over SG*, updates the LTL$_f$ DFA, and outputs if the property holds or is violated.

B. DSL: Scene Graph Language (SGL)

We introduce SGL, which combines LTL$_f$ and a set of SG querying functions, to facilitate the specification of driving properties. The functions are:

relSet computes the join of a set of vertices and a relation.

$$relSet : (V_1 \subseteq V, r \in R) \mapsto V_2 \subseteq V$$
$$V_2 = \{v_2 : v_1 \in V_1 \wedge (v_1, v_2) \in E \wedge rel(v_1, v_2) = r\}$$

For example, the set of lanes controlled by a stop sign is $relSet(stopSigns, \text{controlsTrafficOf})$. SGL also supports **relSetR** – the join of the transpose of the given relation.

filterByAttr selects a subset of vertices, V_1, whose attribute, m, satisfies a given predicate, f.

$$filterByAttr : (V_1 \subseteq V, m \in M, f : T \mapsto bool) \mapsto V_2 \subseteq V$$
$$V_2 = \{v : v \in V_1 \wedge type(att(v)[m]) = T \wedge f(att(v)[m])\}$$

For example, $filterByAttr(trafficLights, \text{lightState}, \lambda x : x = \text{Red})$ yields the red traffic lights. See Table III in the online appendix for SGL encodings of the properties studied.

Other Operations. SGL includes standard operators for numeric comparison, boolean logic, and set manipulation which are used to convert from vertex sets to APs. For example, whether ego has a throttle attribute below a given threshold, ϵ, is specified by $|filterByAttr(Ego, \text{throttle}, \lambda x : x < \epsilon)| = 1$. SGL also defines a discrete metric operator, \$[N][AP], to ease the LTL specification over repeated APs by unrolling the AP N times using the $\mathcal{X}$ operator. The APs are the building blocks for specifying different aspects of the AV's environment and can be combined with temporal operators through LTL to express temporal relationships in the AV's behavior, enabling a precise characterization of its actions and responses in dynamic environments.

C. Monitor

The monitor has two main modules:

1) Representation creation: This module consumes sensor data to estimate the state of ego and to produce an SG through the SGG component. The resulting SG is enriched by the SG annotator component with information about the SUT's output, and the state of ego to produce SG*.

2) Property evaluation: This module takes in SG* and an SGL function containing the LTL$_f$ property. It first evaluates each AP by querying SG*, and then uses the AP values

to update the DFA state. Depending on the DFA state, the monitor returns whether the property holds or is violated.

Table IV in the online appendix gives a complete list of APs computed to evaluate the properties studied. These APs yield an understanding of the spatial and temporal distribution of entities related to ego. For example, ψ_9 checks if ego responds to stop signs by evaluating the $hasStop$ and $isStopped$ APs. $hasStop$ is true iff the set of lanes controlled by stop signs and lines ($stopSignLanes$) intersect with the set of lanes ego is in ($egoLanes$) is non-empty, which would indicate that ego is being directed to stop. $isStopped$ is true iff the set of ego with speed $< \epsilon$ ($egoStopped$) is non-empty, indicating that ego is stopped.

IV. STUDY

We aim to answer the following research questions:

- RQ#1: What driving properties can SGSM express?
- RQ#2: Can SGSM find safety violations in AV systems?

A. Setup

To evaluate SGSM's ability to act as an automatic safety monitor, we need a common execution environment on which to run several AV systems to monitor.

1) Common Execution Platform: For running the study, we used the CARLA simulator for urban driving [20], which is widely-targeted for AV development due to its realistic environments, complex traffic simulation, and ability to model a variety of relevant road scenarios. CARLA holds a competition called the Autonomous Driving Leaderboard, which provides preconfigured scenarios to challenge the community to create systems that can drive autonomously. The challenge includes a variety of towns, 10 different scenarios, each one of them defining a different traffic situation, and a set of routes. We evaluated the 3 top-ranked systems [38] as of June 2022, using the provided evaluation routes for Town05, that includes 2-lane roads and 3-lane highways; 4-lane and T intersections; traffic lights, stop signs, crossing lanes; and pedestrians, cyclists, cars and trucks.

We developed an SGG in the form of a Python module that interfaces with the CARLA API to extract the relevant entities, their attributes, and compute their relationships with each other and the road structure. The SGG uses ground truth information from CARLA to include all entities within a $50m$ by $50m$ area horizontally centered on ego and vertically offset to include $45m$ ahead of ego. We adopt the default entity and relationship scheme from prior work on SGs for AVs [25], [19], enriched with additional information about which roads and junctions lanes belong to. Our unoptimized SGG and annotator take on average 288 ms to create a single SG* and our monitor takes 67 ms to evaluate all properties on it using an Intel Xeon Silver 4216 CPU @ 2.10GHz, 128 GB of RAM, and one Nvidia Titan RTX. While our simulation-based SGG uses ground truth information to eliminate the effects of sensor noise in our study, the current trajectory of SGG research in conjunction with the availability of HD maps for AV systems is promising for implementation of SGSM on real-world systems.

2) AV Systems Evaluated: Each AV takes in a list of waypoints from the route and produces at each frame a control for steering, throttle, and brake; each system has different sensors and software. **Interfuser [39]** consists of a Deep Neural Network (DNN) with a transformer [40] architecture, and a controller that generates a set of actions for ego. It takes 3 images from 3 RGB cameras and a cropped center image to focus on distant traffic lights, a LiDAR point cloud, and the GPS coordinates and computes a set of waypoints, an object density map, traffic light state, stop sign presence, and if the vehicle is in a junction. These are fed into the controller to produce the output. **TCP [41]** takes in 1 image from an RGB camera, ego's speed, and the GPS coordinates and uses a DNN composed of a CNN-based image encoder using ResNet34 [42], and two GRU [43] branches for trajectory and control predictions. **LAV [44]** consists of a perception DNN, motion planner, and controller. The DNN consumes 3 images from 3 RGB cameras and a LiDAR point cloud, and outputs a BEV map which is fed to the planner along with the next waypoint coordinates to produce the next 10 future waypoints. The waypoints are passed to the controller along with a braking signal from a binary DNN classifier to compute the output.

B. RQ#1. Properties Evaluated

To evaluate SGSM's ability to encode safe driving properties relevant to AV systems, we selected 9 properties from the laws and best practices of the Virginia Driving Code [18]. Laws were selected to yield a set of properties within scope of current AV systems and diverse in both temporal aspects required to analyze the property compliance and richness of the SG structure required to evaluate the APs.

Table I shows the successful encoding of those properties, with their relevant statute, a short English summary, and their encoding using the APs over the SG* composed through the LTL$_f$ formula. Additionally, the number of states in the DFA is shown as a measure of temporal complexity. We note that precisely encoding the semantics of the law is challenging. Returning to the stop sign example, the APs are evaluated over the LTL$_f$ formula to track if $isStopped$ is true at least once between $hasStop$ becoming true and later becoming false, indicating that ego stopped while being controlled by the stop sign. This is a necessary but insufficient specification to meet the criteria under the law; notably, this does not check that the vehicle stopped *at the stop line* rather than before, nor does it enforce separate stops for successive stop signs along the same lane.

As ψ_4, ψ_7, and ψ_8 contain a threshold parameter, we instantiate 3 versions of each, for a total of 15 monitors. For ψ_4, $S \in \{5, 10, 15\}\frac{m}{s}$ was chosen to represent parking-lot, urban, and suburban driving speeds. For ψ_7, empirical studies found that lane changes take $4.6s$ on average with a std dev of $2.3s$ and max of $13.3s$ [45]; thus we select $T \in \{5, 10, 15\}s$ to represent the average, 2 std dev, and beyond max. For ψ_8, we select $T \in \{5, 10, 15\}s$ as the time to clear the intersection as a left turn across a 4 lane road at $10mph$ takes $5s$, and we allow for a buffer factor of $1-3\times$.

TABLE I: Properties implemented in SGL to address RQ#1

ψ	VA Code	English Summary of Property	LTLf Formula over SG propositions	# DFA States
ψ_1	§ 46.2-804	Ego vehicle cannot be in the opposing lane	$\mathcal{G}(\neg isOppLane)$	2
ψ_2	§ 46.2-802	Ego vehicle cannot be out of the road.	$\mathcal{G}(\neg isOffRoad)$	2
ψ_3	§ 46.2-802	If ego vehicle is in the rightmost lane, then ego vehicle should not steer to the right.	$\mathcal{G}(isInRightLane \wedge \neg isJunction \rightarrow isNotSteerRight)$	2
ψ_4	§ 46.2-816	Ego vehicle should not be behind another entity in the same lane whithin 4 meters while travelling at a speed $> S$.	$\mathcal{G}(isNearColl \rightarrow \neg isFasterThanS)$	2
ψ_5	§ 46.2-816	If ego vehicle is between 4 and 7 meters of the closest vehicle in the same lane and then comes within 4 meters of a vehicle in the same lane, throttle must not be positive.	$\mathcal{G}((isSuperNear \wedge \neg isNearColl) \wedge \mathcal{X}(isNearColl) \rightarrow \mathcal{X}(isNoThrottle))$	3
ψ_6	§ 46.2-888	If the ego vehicle is moving and there is no entity in the same lane as the ego vehicle within 7 meters, and there is no red traffic light or stop sign controlling the ego vehicle's lane, then the ego vehicle should not stop.	$\mathcal{G}(\neg isStopped \wedge \neg(isSuperNear \vee isNearColl) \wedge \neg hasRed \wedge \neg hasStop \wedge \mathcal{X}(\neg(isSuperNear \vee isNearColl) \wedge \neg hasRed \wedge \neg hasStop) \rightarrow \mathcal{X}(\neg isStopped))$	2
ψ_7	§ 46.2-804	If ego vehicle is not in a junction, then ego vehicle cannot be in more than one lane for more than T seconds (N samples).	$\neg\mathcal{F}\$[N][isMultipleLanes \wedge \neg isJunction]$	N+1
ψ_8	§ 46.2-833	Ego vehicle must exit junctions within T seconds (N samples).	$\neg\mathcal{F}\$[N][isOnlyJunction]$	N+1
ψ_9	§ 46.2-821	Once the ego vehicle detects a new stop signal controlling its lane, it must stop before passing the stop signal.	$\mathcal{G}((\neg hasStop \wedge \mathcal{X}(hasStop)) \rightarrow (\mathcal{X}(hasStop\ \mathcal{U}\ (isStopped \vee \mathcal{G}(hasStop)))))$	4

We note that while some parameters can be expressed in SGL, others are reliant on the parameterization of the underlying SGG. In ψ_4, ψ_5, and ψ_6, we use 4 and 7 meters as the distance thresholds because these correspond to the 'near collision' and 'super near' relationship used by prior AV SGGs [25], [19]. Further, the underlying laws do not provide concrete values, e.g. the law from ψ_5 says "[...] a motor vehicle shall not follow [a vehicle] **more closely than is reasonable and prudent** [...]" (emphasis added) [18].

C. RQ#2. Violations Observed

As described in RQ#1, we derive 15 properties from the Virginia Driving Code and use SGSM to implement a monitor for each property. We ran each AV system through the 10 evaluation scenarios of the CARLA leaderboard and separately evaluated all 15 properties at a rate of $2Hz$.

Table II shows how many of the 10 routes resulted in a violation for each AV system for each of the 15 properties. Note that since the properties are defined as global properties, we track only the first violation, for a maximum of 10 possible violations per AV system per property. We find that the number of violations ranges from 51 for TCP to 72 for Interfuser (over 150 possible violations). Fig. 4a shows an instance of Interfuser violating ψ_1; as the road curved to the right, Interfuser did not steer right enough and drifted into the opposing lane. Fig. 4b shows LAV violating ψ_2, turning left through a junction too sharply, exiting the junction into the median between two lanes. While this is not off of the road bed, the SGG denotes it as off road because it is not part of a defined lane of traffic. Fig. 2 shows TCP violating ψ_9 over a series of frames. TCP approaches a junction with a marked stop line, but it does not stop and enters the junction.

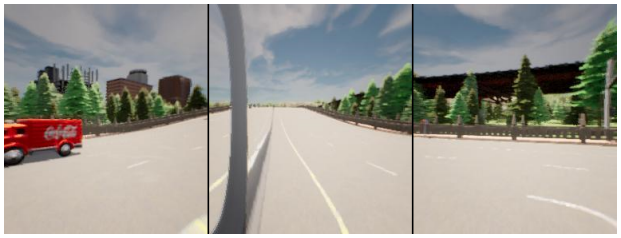
The property violation statistics also give insights into the driving style of the AVs. None of the AVs violated ψ_4, meaning that they maintained sufficient follow distance from lead vehicles. However, we also see that Interfuser and TCP violated ψ_6 over more than half the routes, i.e., they stopped in the middle of the roadway. While we do observe 9 cases where this stoppage is unjustifiable, in 4 other cases we observe that the AV is stopping due to a stopped vehicle ahead of it but farther than the 7 meters prescribed in ψ_6, and in the remaining 4 cases there is a traffic light that is transitioning out of red. This highlights the difficulty in concretizing the parameters used in the specification given the imprecise definitions in the driving manual; 7 meters may be acceptable depending on circumstances. This is further shown in the performance across the parameterizations of ψ_7 and ψ_8. As T increases, the specification is more relaxed which leads to fewer violations; e.g. TCP reduces from 8 violations to 0 under ψ_8 when T is increased from 5 to 10. Although TCP eliminates all violations, Interfuser and LAV do not improve as rapidly. This may point to different AV's optimizations; they likely did not optimize for junction crossing times, and instead may have prioritized moving cautiously through a junction leading to slower transits.

Overall, the study highlights three features of SGSM. First, it showcases how it enabled the specification and monitoring of driving properties that included entities like lanes, vehicles, and traffic signals; their attributes like speed and color; and their relations like is in, controls, and opposes. Second, it shows how SGSM can be parameterized to support a rich set of property types, from stateless to temporal, over propositions that are easily accessible through the scene graph. Third, it provides evidence of SGSM's generality as per its direct application to monitor three distinct systems.

D. Threats of validity

In this work we showed the feasibilty of implementing an SG-based monitor and its utility for checking safety property specifications based on driving rules. The external validity of our results, however, is bounded by our use of simulation to create the SGs using ground truth data. Working in simulation enabled us to construct an SGG module that generates accurate SG representations of the world to judge the cost-effectiveness of the framework as a whole, but we recognize that it will be necessary to consider SGGs using various sensor types and in the wild. Moreover, CARLA suffers from the simulation-reality gap [46], so deploying the approach in the real world will be necessary to assess its generalizability. Similarly, more complex operational domain

(a) Interfuser violates ψ_1. Missed road curve, crossed into opp. lane.



(b) LAV violates ψ_2. Left turn missed lane and drove into median.

Fig. 4: Interfuser and LAV safety violations.

TABLE II: Count of Routes (10 total) with Property Violations by AV System to address RQ#2

Property / System	ψ_1	ψ_2	ψ_3	ψ_4 S=5	ψ_4 S=10	ψ_4 S=15	ψ_5	ψ_6	ψ_7 T=5	ψ_7 T=10	ψ_7 T=15	ψ_8 T=5	ψ_8 T=10	ψ_8 T=15	ψ_9	Total
InterFuser	3	0	10	0	0	0	3	9	10	5	5	10	5	5	7	72
TCP	6	0	10	0	0	0	2	6	5	3	3	8	0	0	8	51
LAV	6	1	10	0	0	0	3	2	8	6	5	10	6	1	7	65
Total	15	1	30	0	0	0	8	17	23	14	13	28	11	6	22	188

properties should be specified and checked. The internal validity of our results are related to our implementation of SGSM, and to overcome this threat we released an artifact.

V. RELATED WORK

Prior work has built specialized monitors for AV software subcomponents such as trajectory prediction [47], collision avoidance [48], or interfaces between AV components such as the CAN bus [15], [49] or through ROS topics [50]. These efforts include propositions over simple types, e.g., "disengaged cruise control" or "traveled for 2 seconds". Prior work has also examined monitoring end-to-end systems. Desai et al. propose using observable trajectories to monitor path following and safety buffers using signal temporal logic (STL) [51]. Stamenkovich et al. use system-independent runtime monitors that observe only the external inputs and outputs to check properties specified in LTL [14]. Morse et al. [52] characterize spatial relationships between sensed objects and robot behaviors, by using graph representations and First Order Logic (FOL), that can be used for runtime monitoring. However, these techniques assume there is a mapping from the sensed inputs to the semantics of the APs.

The introduction of machine-learned components to process multi-dimensional sensors complicate the design of monitors due to the black-box nature of such components and their ability to perform previously separate subtasks end-to-end. Grieser et al. provide a mechanism for monitoring a limited set of safety properties based on LiDAR point clouds [53], however, it is not generalizable to other sensors or properties. Work in shielded reinforcement learning aims to learn [54] or enforce [55] safety properties for agents specified in temporal logic and has shown to increase robustness of learned behaviors. But again, extracting the APs used in the specification requires either limiting the propositions to those already consumed by the agent or additional effort to extract the relevant semantics, both of which limit generalizability. Anderson et al. try to overcome this issue by introducing Spatial Regular Expressions for pattern matching over perception streams containing spatial and temporal data, leveraging object detection networks [56]. Nonetheless, it can only reason about relationships given by bounding box overlap, and misses richer types of relationships like proximity between entities or traffic semantics.

Another line of related research has explored different ontologies in the AV domain [57] for scenario-based testing [58], [59] and for situation assessment and decision making [60], [61], [62]. The main limitation of these approaches, however, is that the ontologies are completely tied to the SUT, thus only encoding the information needed by the system and making them nongeneralizable. Our previous work on SGs for AV testing [19] demonstrated the utility of SGs as a basis for measuring coverage of nontemporal properties, but does not provide a mechanism to express and automatically check the rich properties studied here. Closer to our abstraction, Majzik et al. envisioned using a graph-based ontology of the environment with STL to monitor system performance [63], but defines no properties for self-driving cars. Our work extends and formalizes this notion with: a spatial-relation graph that can be computed from external, system-independent inputs, a graph-semantics logic DSL and LTL$_f$ that can specify safety-critical properties; and we demonstrate that this approach can automatically find property violations at runtime for AV driving systems.

VI. CONCLUSION

Ensuring that AVs' behaviors abide by safe driving properties is key to their successful deployment. Specifying and monitoring such properties, however, is challenging as they depend not just on the AV but also on related entities that influence its behavior but are not readily accessible. This paper introduces SGSM, a framework to support the specification of safe driving properties and their automatic synthesis into an AV runtime monitor. It provides a general mechanism based on scene graphs to abstract relevant entities from sensor inputs and a domain-specific language to enable property specification over those graphs. The study shows the expressiveness of the DSL for specifying real driving properties and the generality of the monitoring mechanism through its application to 3 off-the-shelf AV systems where it uncovers various driving violations.

REFERENCES

[1] R. Bellan, "Cruise inches into waymo's territory in the phoenix area," Aug 2023, accessed on 02.07.2024. [Online]. Available: https://techcrunch.com/2023/08/08/cruise-inches-into-waymos-territory-in-the-phoenix-area/

[2] ——, "Cruise and waymo win robotaxi expansions in san francisco," Aug 2023, accessed on 02.07.2024. [Online]. Available: https://techcrunch.com/2023/08/10/cruise-and-waymo-win-robotaxi-expansions-in-san-francisco/

[3] A. Marshall, "Uber video shows the kind of crash self-driving cars are made to avoid," Mar 2018, accessed on 02.07.2024. [Online]. Available: https://www.wired.com/story/uber-self-driving-crash-video-arizona/

[4] N. Board, "Collision between vehicle controlled by developmental automated driving system and pedestrian. nat. transpot. saf. board, washington, dc," USA, Tech. Rep. HAR-19-03, 2019. URL https://www.ntsb. gov/investigations . . . , Tech. Rep., 2019.

[5] B. Templeton, "Tesla in taiwan crashes directly into overturned truck, ignores pedestrian, with autopilot on," *Forbes*, Jun 2020, accessed on 02.07.2024. [Online]. Available: https://www.forbes.com/sites/bradtempleton/2020/06/02/tesla-in-taiwan-crashes-directly-into-overturned-truck-ignores-pedestrian-with-autopilot-on/?sh=20a7458f58e5link

[6] N. E. Boudette and N. Chokshi, "U.s. will investigate tesla's autopilot system over crashes with emergency vehicles," *New York Times*, Aug 2021, accessed on 02.07.2024. [Online]. Available: https://www.nytimes.com/2021/08/16/business/tesla-autopilot-nhtsa.html

[7] R. Bellan, "A waymo self-driving car killed a dog in 'unavoidable' accident," Jun 2023, accessed on 02.07.2024. [Online]. Available: https://techcrunch.com/2023/06/06/a-waymo-self-driving-car-killed-a-dog-in-unavoidable-accident/

[8] A. Roy and H. Jin, "California regulator probes crashes involving gm's cruise robotaxis," Aug 2023, accessed on 02.07.2024. [Online]. Available: https://www.reuters.com/business/autos-transportation/gms-cruise-robotaxi-collides-with-fire-truck-san-francisco-2023-08-19/

[9] T. Victor, K. Kusano, T. Gode, R. Chen, and M. Schwall, "Safety performance of the waymo rider-only automated driving system at one million miles," Tech. Rep., February 2023, accessed on 02.07.2024. [Online]. Available: https://storage.googleapis.com/sdc-prod/v1/safety-report/Waymo-Safety-Methodologies-and-Readiness-Determinations.pdf

[10] L. Zhang, "Cruise's safety record over 1 million driverless miles," Apr 2023, accessed on 02.07.2024. [Online]. Available: https://getcruise.com/news/blog/2023/cruises-safety-record-over-one-million-driverless-miles/

[11] H. Araujo, M. R. Mousavi, and M. Varshosaz, "Testing, validation, and verification of robotic and autonomous systems: A systematic review," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 2, mar 2023. [Online]. Available: https://doi.org/10.1145/3542945

[12] N. Mehdipour, M. Althoff, R. D. Tebbens, and C. Belta, "Formal methods to comply with rules of the road in autonomous driving: State of the art and grand challenges," *Automatica*, vol. 152, p. 110692, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109822005568

[13] K. Watanabe, E. Kang, C.-W. Lin, and S. Shiraishi, "Runtime monitoring for safety of intelligent vehicles," in *Proceedings of the 55th annual design automation conference*, 2018, pp. 1–6.

[14] J. Stamenkovich, L. Maalolan, and C. Patterson, "Formal assurances for autonomous systems without verifying application software," in *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*. IEEE, 2019, pp. 60–69.

[15] A. Kane, O. Chowdhury, A. Datta, and P. Koopman, "A case study on runtime monitoring of an autonomous research vehicle (arv) system," in *Runtime Verification: 6th International Conference, RV 2015, Vienna, Austria, September 22-25, 2015. Proceedings*. Springer, 2015, pp. 102–117.

[16] M. Mauritz, F. Howar, and A. Rausch, "Assuring the safety of advanced driver assistance systems through a combination of simulation and runtime monitoring," in *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications: 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part II 7*. Springer, 2016, pp. 672–687.

[17] K. Leach, C. S. Timperley, K. Angstadt, A. Nguyen-Tuong, J. Hiser, A. Paulos, P. Pal, P. Hurley, C. Thomas, J. W. Davidson, *et al.*, "Start: A framework for trusted and resilient autonomous vehicles (practical experience report)," in *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2022, pp. 73–84.

[18] "Virginia code title 46.2 chapter 8 - motor vehicles, regulation of traffic."

[19] T. Woodlief, F. Toledo, S. Elbaum, and M. B. Dwyer, "S3c: Spatial semantic scene coverage for autonomous vehicles," in *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)*. ACM, 2024.

[20] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[21] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei, "Image retrieval using scene graphs," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3668–3678.

[22] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. G. Hauptmann, "A comprehensive survey of scene graphs: Generation and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

[23] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[24] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[25] A. V. Malawade, S.-Y. Yu, B. Hsu, H. Kaeley, A. Karra, and M. A. Al Faruque, "roadscene2vec: A tool for extracting and embedding road scene-graphs," *Knowledge-Based Systems*, vol. 242, p. 108245, 2022.

[26] J. Li, H. Gang, H. Ma, M. Tomizuka, and C. Choi, "Important object identification with semi-supervised learning for autonomous driving," pp. 2913–2919, 2022.

[27] A. Prakash, S. Debnath, J. Lafleche, E. Cameracci, G. State, S. Birchfield, and M. T. Law, "Self-supervised real-to-sim scene generation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, oct 2021, pp. 16 024–16 034. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.01574

[28] A. Silberschatz, H. Korth, and S. Sudarshan, *Database systems concepts*. McGraw-Hill, Inc., 2005.

[29] R. Angles, M. Arenas, P. Barceló, A. Hogan, J. Reutter, and D. Vrgoč, "Foundations of modern query languages for graph databases," *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, pp. 1–40, 2017.

[30] D. Jackson, "Alloy: a lightweight object modelling notation," *ACM Transactions on software engineering and methodology (TOSEM)*, vol. 11, no. 2, pp. 256–290, 2002.

[31] T. Reinbacher, M. Függer, and J. Brauer, "Runtime verification of embedded real-time systems," *Formal methods in system design*, vol. 44, pp. 203–239, 2014.

[32] S. Pinisetty, P. S. Roop, S. Smyth, N. Allen, S. Tripakis, and R. V. Hanxleden, "Runtime enforcement of cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, pp. 1–25, 2017.

[33] H. Jiang, S. Elbaum, and C. Detweiler, "Reducing failure rates of robotic systems though inferred invariants monitoring," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1899–1906.

[34] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. ieee, 1977, pp. 46–57.

[35] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *IJCAI'13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. Association for Computing Machinery, 2013, pp. 854–860.

[36] S. Zhu, G. Pu, and M. Y. Vardi, "First-order vs. second-order encodings for ltlf-to-automata."

[37] F. Fuggitti, "Ltlf2dfa," March 2019.

[38] E. A. F. CARLA Team, Intel Autonomous Agents Lab and AlphaDrive, "Autonomous driving on carla leaderboard," accessed on 02.07.2024. [Online]. Available: https://paperswithcode.com/sota/autonomous-driving-on-carla-leaderboard

[39] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu, "Safety-enhanced autonomous driving using interpretable sensor fusion transformer," in *Conference on Robot Learning*. PMLR, 2023, pp. 726–737.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[41] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao, "Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline," *Advances in Neural Information Processing Systems*, vol. 35, pp. 6119–6132, 2022.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[43] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[44] D. Chen and P. Krähenbühl, "Learning from all vehicles," in *CVPR*, 2022.

[45] T. Toledo and D. Zohar, "Modeling duration of lane changes," *Transportation Research Record*, vol. 1999, no. 1, pp. 71–78, 2007.

[46] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *European Conference on Artificial Life*. Springer, 1995, pp. 704–720.

[47] A. Farid, S. Veer, B. Ivanovic, K. Leung, and M. Pavone, "Task-relevant failure detection for trajectory predictors in autonomous vehicles," in *Conference on Robot Learning*. PMLR, 2023, pp. 1959–1969.

[48] C. Luo, R. Wang, Y. Jiang, K. Yang, Y. Guan, X. Li, and Z. Shi, "Runtime verification of robots collision avoidance case study," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1. IEEE, 2018, pp. 204–212.

[49] R. Wang, Y. Wei, H. Song, Y. Jiang, Y. Guan, X. Song, and X. Li, "From offline towards real-time verification for robot systems," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1712–1721, 2018.

[50] J. Huang, C. Erdogan, Y. Zhang, B. Moore, Q. Luo, A. Sundaresan, and G. Rosu, "Rosrv: Runtime verification for robots," in *Runtime Verification: 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings 5*. Springer, 2014, pp. 247–254.

[51] A. Desai, T. Dreossi, and S. A. Seshia, "Combining model checking and runtime verification for safe robotics," in *International Conference on Runtime Verification*. Springer, 2017, pp. 172–189.

[52] C. Morse, L. Feng, M. Dwyer, and S. Elbaum, "A framework for the unsupervised inference of relations between sensed object spatial distributions and robot behaviors," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 901–908.

[53] J. Grieser, M. Zhang, T. Warnecke, and A. Rausch, "Assuring the safety of end-to-end learning-based autonomous driving through runtime monitoring," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*. IEEE, 2020, pp. 476–483.

[54] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[55] B. Könighofer, J. Rudolf, A. Palmisano, M. Tappler, and R. Bloem, "Online shielding for reinforcement learning," *Innovations in Systems and Software Engineering*, pp. 1–16, 2022.

[56] J. Anderson, G. Fainekos, B. Hoxha, H. Okamoto, and D. Prokhorov, "Pattern matching for perception streams," in *International Conference on Runtime Verification*. Springer, 2023, pp. 251–270.

[57] M. Zipfl, N. Koch, and J. M. Zöllner, "A comprehensive review on ontologies for scenario-based testing in the context of autonomous driving," in *2023 IEEE Intelligent Vehicles Symposium (IV)*, 2023, pp. 1–7.

[58] F. Klueck, Y. Li, M. Nica, J. Tao, and F. Wotawa, "Using ontologies for test suites generation for automated and autonomous driving functions," in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2018, pp. 118–123.

[59] F. Wotawa, J. Bozic, and Y. Li, "Ontology-based testing: An emerging paradigm for modeling and testing systems and software," in *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2020, pp. 14–17.

[60] S. Ulbrich, T. Nothdurft, M. Maurer, and P. Hecker, "Graph-based context representation, environment modeling and information aggregation for automated driving," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 541–547.

[61] M. Hülsen, J. M. Zöllner, and C. Weiss, "Traffic intersection situation description ontology for advanced driver assistance," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 993–999.

[62] M. Buechel, G. Hinz, F. Ruehl, H. Schroth, C. Gyoeri, and A. Knoll, "Ontology-based traffic scene modeling, traffic regulations dependent situational awareness and decision-making for automated vehicles," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1471–1476.

[63] I. Majzik, O. Semeráth, C. Hajdu, K. Marussy, Z. Szatmári, Z. Micskei, A. Vörös, A. A. Babikian, and D. Varró, "Towards system-level testing with coverage guarantees for autonomous vehicles," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 2019, pp. 89–94.

APPENDIX

TABLE III: Intermediate variables used in APs

Name	SGL expression
$egoLanes$	$relSet(Ego,$ isIn$)$
$egoRoads$	$relSet(egoLanes,$ isIn$)$
$egoJunctions$	$relSet(egoRoads,$ isIn$)$
$oppLanes$	$relSet(egoLanes,$ opposes$)$
$offRoad$	$filterByAttr(egoLanes,$ kind, $\lambda x : x =$ offRoad$)$
$rightLanes$	$relSet(egoLanes,$ toRightOf$)$
$steerRight$	$filterByAttr(Ego,$ steer, $\lambda x : x > 0)$
$inEgoLane$	$relSetR(egoLanes,$ isIn$) \setminus \{Ego\}$
$nearColl$	$relSet(inEgoLane,$ near_coll$)$
$superNear$	$relSet(inEgoLane,$ super_near$)$
$egoFasterS$	$filterByAttr(Ego,$ speed, $\lambda x : x > S)$
$noThrottle$	$filterByAttr(Ego,$ throttle, $\lambda x : x < \epsilon)$
$tLights$	$filterByAttr(G,$ kind, $\lambda x : x =$ trafficLight$)$
$redLights$	$filterByAttr(tLights,$ lightState, $\lambda x : x =$ Red$)$
$trafLightLns$	$relSet(redLights,$ controlsTrafficOf$)$
$stopSigns$	$filterByAttr(G,$ kind, $\lambda x : x =$ stopSign$)$
$stopSignLanes$	$relSet(stopSigns,$ controlsTrafficOf$)$
$egoStopped$	$filterByAttr(Ego,$ speed, $\lambda x : x < \epsilon)$
$juncRoads$	$relSetR(egoJunctions,$ *isIn*$)$

TABLE IV: Atomic Propositions

Atomic Prop.	SGL expression
$isJunction$	$\lvert egoJunctions\rvert > 0$
$isOppLane$	$\lvert oppLanes\rvert > 0$
$isOffRoad$	$\lvert offRoad\rvert > 0$
$isInRightLane$	$\lvert rightLanes\rvert = 0$
$isNotSteerRight$	$\lvert steerRight\rvert = 0$
$isNearColl$	$\lvert nearColl\rvert > 0$
$isFasterThanS$	$\lvert egoFasterS\rvert = 1$
$isSuperNear$	$\lvert superNear\rvert > 0$
$isNoThrottle$	$\lvert noThottle\rvert = 1$
$isMultipleLanes$	$\lvert egoLanes\rvert > 1$
$hasRed$	$\lvert trafLightLns \cap egoLanes\rvert > 0$
$hasStop$	$\lvert stopSignLanes \cap egoLanes\rvert > 0$
$isStopped$	$\lvert egoStopped\rvert = 1$
$isOnlyJunction$	$\lvert egoRoads \setminus juncRoads\rvert = 0$