Search-based Test-Case Generation by Monitoring Responsibility Safety Rules

Mohammad Hekmatnejad[1], Bardh Hoxha[2] and Georgios Fainekos[1]

Abstract— The safety of Automated Vehicles (AV) as Cyber-Physical Systems (CPS) depends on the safety of their consisting modules (software and hardware) and their rigorous integration. Deep Learning is one of the dominant techniques used for perception, prediction and decision making in AVs. The accuracy of predictions and decision-making is highly dependant on the tests used for training their underlying deep-learning. In this work, we propose a method for screening and classifying simulation-based driving test data to be used for training and testing controllers. Our method is based on monitoring and falsification techniques, which lead to a systematic automated procedure for generating and selecting qualified test data. We used Responsibility Sensitive Safety (RSS) rules as our qualifier specifications to filter out the random tests that do not satisfy the RSS assumptions. Therefore, the remaining tests cover driving scenarios that the controlled vehicle does not respond safely to its environment. Our framework is distributed with the publicly available S-TALIRO and Sim-ATAV tools.

I. INTRODUCTION

The development of safe, fully autonomous systems has become one of the seminal technological problems of our time. Their eventual development will have a significant impact on mobility, with solutions ranging from personal mobility, to mobility-on-demand systems. The potential of automation carries a plethora of environmental and financial benefits [1].

Major components of these systems are developed to tackle problems of perception, decision making, and control, often using machine learning techniques such as Deep Neural Networks [2]. Ensuring that these components are bug-free is a challenging problem. To make the problem more difficult, emergent behavior that arises as a result of the complex interaction between these components is extremely difficult to detect [3], [4].

Physical testing has shown to be expensive, time consuming, and generally inefficient. In addition, there is no consensus on what constitutes an adequate metric in this area. It is questionable whether metrics such as miles driven and miles per disengagement, which are typically reported to governmental agencies [5], are sufficient since most of the testing and miles driven are conducted in carefully chosen locations and scenarios. As a result, these metrics fail to capture a wide array of conditions and scenarios that are typical in day-to-day passenger driving.

Recent research [6], [7], [8], [9], [10], [11] supports simulation-based and virtual methods as a scalable alternative. Simulation enables inexpensive, quick testing for arbitrarily complex conditions and scenarios. It also speeds up development by enabling automakers to quickly iterate and support the rapid development of their systems. In addition, since simulation-based methods have been studied extensively in the testing and verification community (see [12] for an overview), it is only natural to attempt an extension to the autonomous vehicle domain.

In [13], the authors develop a probabilistic programming language SCENIC to define scenes, which are configurations of physical objects in traffic environments. The probabilistic component enables the assignment of behavior distributions to specific objects in the scene. In [14], [15], the authors present Sim-ATAV, part of the S-TALIRO [16] toolbox, for adversarial testing of autonomous vehicles with machine learning components. The tool enables the generation of test cases that falsify formal requirements expressed in Signal Temporal Logic (STL) [17]. Frameworks like [8] and [18] and the corresponding toolchains provide similar functionality to Sim-ATAV for search-based test generation. A challenge encountered by all these methods is how to properly define what constitutes a violation of safe driving behaviors.

In this paper, Responsibility Sensitive Safety (RSS) [19], [20], [21] rules are utilized for classifying and qualifying driving test data to determine ego-centric and meaningful driving scenarios. In order to achieve that, RSS specifications are formalized into STL to enable formal, algorithmic reasoning over them. In [22], RSS specifications were utilized to monitor naturalistic traffic data obtained from the CommonRoad library [23]. In this paper, the driving scenarios are generated through simulation. Given formal RSS specifications, a requirements-guided search through a stochastic optimization algorithm finds worst infractions of RSS specifications. The set of falsifying behaviors may be analyzed further by considering which part of the specification caused the falsification. This is beneficial since the root cause of failure may be determined and classified quickly. For example, using this approach, practitioners can return only the falsifying behaviors related to longitudinal distances between the ego car and other agents. To the best of the authors knowledge, this is the first time RSS specifications are formalized for testing vehicles in a simulated environment.

[1]Mohammad Hekmatnejad and Georgios Fainekos are with School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe AZ 85281, USA {mhekmatn,fainekos}@asu.edu

[2]Bardh Hoxha is with Toyota Research Institute of North America, Ann Arbor, MI, 48187, USA bardh.hoxha@toyota.com

A. Signal Temporal Logic

Signal Temporal Logic (STL) [17] was defined to express bounded time requirements over continuous-time (CT) signals. In this paper, we present the syntax and semantics of STL over discrete-time (DT) signals as resulting from simulations.

Definition II.1 (STL Syntax for DT signals). *Let x be a vector variable, i.e., $x = [x_1, \ldots, x_n]^T$, $p(x)$ be a function over the reals, and $\mathcal{I}$ be any non-empty interval of $\mathbb{R}_{\geq 0}$. The syntax for Signal Temporal Logic (STL) formulas is provided by the following grammar:*

$$\phi ::= \mathbf{T} \mid p(x) \geq 0 \mid \neg\phi \mid \phi \vee \phi \mid \bigcirc_{\mathcal{I}} \phi \mid \phi\mathcal{U}_{\mathcal{I}}\phi$$

where $\mathbf{T}$ is true, $\bigcirc_{\mathcal{I}}$ is the next sample operator, and $\mathcal{U}_{\mathcal{I}}$ is the until operator.

We utilize a quantitative interpretation for the STL semantics (see [17] for an overview). In order to define quantitative semantics with a topological interpretation over arbitrary predicates $p(x) \geq 0$, we will need to use a *generalized quasi-metric* $\mathbf{d}$ [24] (referred to simply as a metric for brevity in the following) to define a signed distance function:

Definition II.2 (Signed Distance). *Let $x \in X$ be a point, $S \subseteq X$ be a set and $\mathbf{d}$ be a metric. Then, we define the Signed Distance from x to S to be*

$$\mathbf{Dist}_{\mathbf{d}}(x, S) := \begin{cases} -\inf\{\mathbf{d}(x, x') \mid x' \in S\} & \text{if } x \notin S \\ \inf\{\mathbf{d}(x, x') \mid x' \notin S\} & \text{if } x \in S \end{cases}$$

Intuitively, the distance function returns positive values when x is in the set S and negative values when x is outside the set S. We should point out that we use the extended definition of supremum ($\sqcup$) and infimum ($\sqcap$). That is to say, the supremum of the empty set is defined to be the bottom element of the domain, while the infimum of the empty set is defined to be the top element of the domain. For example, when $\mathcal{V} = \overline{\mathbb{R}}_{\geq 0}$, then $\inf \emptyset := +\infty$.

We review STL semantics that map a formula φ and a trace σ to a value drawn from a lattice $\overline{\mathcal{V}} \triangleq \mathcal{V} \cup \{-v \mid v \in \mathcal{V}\}$. In this work, even though we need an arbitrary lattice of truth values $\mathcal{V}$ to treat Boolean signals, in most of the examples, we assume that $\mathcal{V} = \overline{\mathbb{R}}_{\geq 0}$ (with the usual negation (-) over the reals). We denote the robust valuation of the formula φ over the trace σ at sample i by $[\![\varphi]\!]_{\mathbf{d}}(\sigma, i)$. The semantics for a predicate $p(x) \geq 0$ evaluated at time i over trace σ is defined as the distance between $\sigma(i)$ and the set $[\![p(x) \geq 0]\!] \triangleq \{x \mid p(x) \geq 0\}$. Intuitively, this distance represents how robustly the point $\sigma(i)$ lies within (or is outside) the set $[\![p(x) \geq 0]\!]$. If this distance is zero, then the smallest perturbation of the point $\sigma(i)$ can affect the outcome of $\sigma(i) \in [\![p(x) \geq 0]\!]$.

Definition II.3 (Discrete-Time Robust Semantics). *Consider an extended generalized quasi-metric space $(\mathcal{Y}, \mathbf{d})$. Let $\sigma : N \to \mathcal{Y}$ be a trace, then the robust semantics of an STL formula φ with respect to σ at time sample i is defined as:*

$$\begin{aligned}
[\![\mathbf{T}]\!]_{\mathbf{d}}(\sigma, i) &:= \bigsqcup \mathcal{V} := \top \\
[\![p(x) \geq 0]\!]_{\mathbf{d}}(\sigma, i) &:= \mathbf{Dist}_{\mathbf{d}}(\sigma(i), [\![p(x) \geq 0]\!]) \\
[\![\neg\varphi_1]\!]_{\mathbf{d}}(\sigma, i) &:= -[\![\varphi_1]\!]_{\mathbf{d}}(\sigma, i) \\
[\![\varphi_1 \vee \varphi_2]\!]_{\mathbf{d}}(\sigma, i) &:= [\![\varphi_1]\!]_{\mathbf{d}}(\sigma, i) \sqcup [\![\varphi_2]\!]_{\mathbf{d}}(\sigma, i)
\end{aligned}$$

$[\![\bigcirc_{\mathcal{I}}\varphi_1]\!]_{\mathbf{d}}(\sigma, i) :=$
$$\begin{cases} [\![\varphi_1]\!]_{\mathbf{d}}(\sigma, i+1) & \text{if } i+1 \in N \text{ and } (i+1)\Delta t \in (i\Delta t + \mathcal{I}) \\ -\top \triangleq \bot & \text{otherwise (i.e., the bottom element)} \end{cases}$$

$[\![\varphi_1 \mathcal{U}_{\mathcal{I}} \varphi_2]\!]_{\mathbf{d}}(\sigma, i) :=$

$$\bigsqcup_{i' \in \{j \in N \mid j\Delta t \in (i\Delta t + \mathcal{I})\}} \left([\![\varphi_2]\!]_{\mathbf{d}}(\sigma, i') \sqcap \bigsqcap_{i \leq i'' < i'} [\![\varphi_1]\!]_{\mathbf{d}}(\sigma, i'') \right)$$

where $\top$ is the top element of the lattice, and $t + \mathcal{I} = \{t'' \mid \exists t' \in \mathcal{I} . t'' = t + t'\}$.

Intuitively, the requirement $\bigcirc_{\mathcal{I}}\varphi$ states that φ should be true at the next sample, which should occur some time in the physical time interval $\mathcal{I}$. For example, consider $\Delta t = 0.1$ and the formula $\psi = \bigcirc_{[0,0.1]}\mathbf{T}$, then ψ is true ($\top$) at sample i since $(i+1)\Delta t - i\Delta t = \Delta t \in [0, 0.1]$. However, for $\Delta t = 0.2$, ψ would evaluate to false ($\bot$). The operator $\varphi_1 \mathcal{U}_{\mathcal{I}} \varphi_2$ states that φ_2 should be satisfied at some time in the interval $\mathcal{I}$ and until then φ_1 should hold. The other common Boolean and temporal operators can be defined as syntactic abbreviations (see [17]). For example, $\Diamond_{\mathcal{I}}\phi \equiv \mathbf{T}\mathcal{U}_{\mathcal{I}}\phi$ stands for eventually at some time in the time interval $\mathcal{I}$, ϕ should be true, and $\Box_{\mathcal{I}}\phi \equiv \neg\Diamond_{\mathcal{I}}\neg\phi$ stands for always during the interval $\mathcal{I}$, ϕ should be true. When $\mathcal{I} = [0, \infty)$, we will be dropping $\mathcal{I}$ from the notation, e.g., $\Box_{[0,\infty)}\phi \equiv \Box\phi$.

An important operator that we will be using in this work is the *release* operator $\varphi_1 \mathcal{R}_{\mathcal{I}} \varphi_2 \equiv \neg(\neg\varphi_1 \mathcal{U}_{\mathcal{I}} \neg\varphi_2)$, which states that φ_2 should always hold during the time interval $\mathcal{I}$ up to (but not including) the time when φ_1 becomes true. In fact, we will need a slightly modified version of the release operator $\overline{\mathcal{R}}$ which does not require φ_2 to happen at all if φ_1 has happened in the past:

$[\![\varphi_1 \overline{\mathcal{R}}_{\mathcal{I}} \varphi_2]\!]_{\mathbf{d}}(\sigma, i) :=$

$$\bigsqcap_{i' \in \{j \in N \mid j\Delta t \in (i\Delta t + \mathcal{I})\}} \left([\![\varphi_2]\!]_{\mathbf{d}}(\sigma, i') \sqcup \bigsqcup_{i \leq i'' \leq i'} [\![\varphi_1]\!]_{\mathbf{d}}(\sigma, i'') \right)$$

We refer to the above operator as non-strict release operator. In fact, any non-strict release formula such as $\varphi_1 \overline{\mathcal{R}}_{\mathcal{I}} \varphi_2$ can be rewritten as $\varphi_1 \mathcal{R}_{\mathcal{I}}(\varphi_1 \vee \varphi_2)$ using the release operator.

B. Robustness-guided Falsification of STL Specifications

The problem of determining whether a CPS Σ satisfies a specification ϕ is an undecidable problem, i.e. there is no general algorithm that terminates and returns whether $\Sigma \models \phi$. Therefore, it is not possible to determine exactly the minimum robustness over all system behaviors. However, by repeatedly testing the system, we can check whether a behavior that does not satisfy the specification exists.

Falsification is the process of finding a counter-example or falsifying example that proves that the system does not satisfy the specification. The STL falsification problem is presented as follows:

Problem II.1 (STL Falsification). *Given an MTL formula ϕ and a system Σ, find initial conditions and input signals such that, when given to Σ, generate a falsifying trajectory for which $[\![\phi]\!](\mu) < 0$ (or with Boolean semantics $\mu \not\models \phi$).*

The robustness semantics of STL enable the development of an optimization framework that searches for falsifying behavior in a system. Specifically, the problem may be posed as a global non-linear optimization problem:

$$\boldsymbol{\sigma}^{\star} = \arg \min_{\boldsymbol{\sigma} \in \mathcal{L}(\Sigma)} [\![\varphi]\!]_{\mathbf{d}}(\boldsymbol{\sigma}). \quad (1)$$

Here, $\boldsymbol{\sigma}$ is a system trajectory and $\mathcal{L}(\Sigma)$ is the set of all system behaviors. If the global optimization algorithm converges to some local minimize $\tilde{\boldsymbol{\sigma}}$ such that $[\![\varphi]\!]_{\mathbf{d}}(\tilde{\boldsymbol{\sigma}}) < 0$, then a counterexample (adversarial sample) has been identified, which can be used for debugging (or for training).

In order to solve this non-linear non-convex optimization problem, a number of stochastic search optimization methods can be applied (e.g., [24] – for an overview see [12]). We leverage existing falsification methods to identify falsifying examples for an autonomous driving system.

III. RSS

In this paper, our focus is to use the RSS specifications for monitoring the safety of driving behavior of an ego car [19]. In our test environment, non-ego cars' behaviors are as unconstrained as possible. Hence, worst-case scenarios are not always meaningful. We utilize the RSS requirements to place constraints solely on the ego car's behavior and not other vehicles in the road (referred to as agent vehicles from now on).

Before diving into a formal definition, we first informally describe the RSS requirements for an ego car driving on the highway, and then formally represent them in STL. A situation is unsafe if and only if the longitudinal and lateral distances among the ego vehicle and agent vehicles are unsafe. By lateral and longitudinal safe distances, we refer to the minimum required distances, which are calculated based on the current velocities of the ego car, ego's maximum reaction time (ρ), and the maximum/minimum allowed accelerations stated in Table III.

We illustrate the concept of safe longitudinal distance and predicates needed in our definitions in Fig. 1. Similarly, in Fig. 2, we illustrate the concept of safe lateral distance and related predicates. The idea is that if an ego car continuously preserves the minimum required safe distances with a leading agent vehicle until the distances become unsafe, then it can react responsibly and responsively in order to reach a safe state again. There are two significant restrictions on the ego's driving behavior based on time. We use *hesitation* and *reaction* times to refer to times that fall in $[0, \rho)$ and

TABLE I: The RSS parameter description in the safe lateral and longitudinal definitions.

symbol	description
ρ	reaction time in $seconds$
μ	minimum lateral distance in $meter$
δ_t	sampling time in $seconds$
a_{minBr}^{lon}	minimum required longitudinal braking in m/s^2
a_{maxAcc}^{lon}	maximum allowed longitudinal accelerating in m/s^2
a_{minBr}^{lat}	minimum required lateral braking in m/s^2
a_{maxBr}^{lon}	maximum allowed longitudinal braking in m/s^2
a_{maxAcc}^{lat}	maximum allowed lateral accelerating in m/s^2

$[\rho, +\infty]$, respectively. For all the hesitation times, the ego car cannot exceed a maximum lateral/longitudinal acceleration bound depending on which distance changed from safe to unsafe. For all reaction times, the ego car has to react responsibly. That is, the ego car has to start decelerating with the minimum required deceleration depending on which distance changed from safe to unsafe. For the case that a situation becomes unsafe laterally, there is another restriction on the velocity of the ego car. That is, in the reaction time, the ego car has to control its lateral velocity so that it never deviates more than a constant $\mu/2$ from its original lateral position when the situation became unsafe laterally. All the restrictions are lifted when an unsafe situation becomes safe again. For more details see [19] and [22].
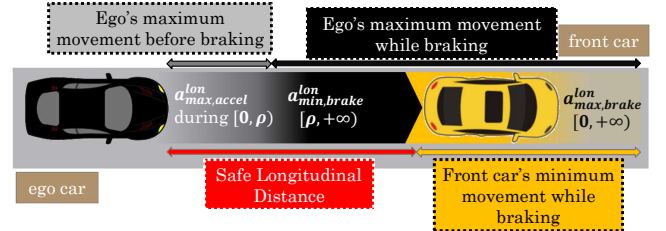

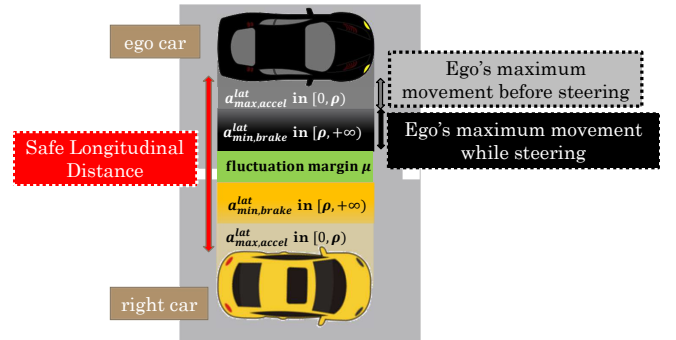Fig. 1: Longitudinal Safety. See Table I for the symbol definitions.

Fig. 2: Lateral Safety. See Table I for the symbol definitions.

In the rest of this section, we represent a slightly modified version of the formula $\varphi_{resp}^{lat,lon}$ from the Remark III.3 in

[22]. For the sake of space and readability, for the lateral requirements, we assumed that the ego car is driving on the left side of the other car. Also, we assume that the car behind the front car is the ego car.

Definition III.1 (RSS formalization in STL for a single ego car driving in highway along other cars). *The below formula $\varphi_{resp}^{lat,lon}$ enforces the required RSS restrictions on the velocity and accelerations of an ego car driving on a highway.*

$$\varphi_{resp}^{lat,lon} \equiv \varphi^{lon} \wedge \varphi^{lat} \wedge \varphi^{lat,lon} \wedge \varphi^{\neg lat,\neg lon}$$

$$\varphi^{lon} \equiv$$
$$\Box\left(\left(\neg S_{l,r}^{lat} \wedge S_{b,f}^{lon} \wedge \bigcirc(\neg S_{l,r}^{lat} \wedge \neg S_{b,f}^{lon}) \right) \to \bigcirc P^{lon} \right)$$

$$\varphi^{lat} \equiv$$
$$\Box\left(\left(S_{l,r}^{lat} \wedge \neg S_{b,f}^{lon} \wedge \bigcirc(\neg S_{l,r}^{lat} \wedge \neg S_{b,f}^{lon}) \right) \to \bigcirc P^{lat} \right)$$

$$\varphi^{lat,lon} \equiv$$
$$\Box\left(\left(S_{l,r}^{lat} \wedge S_{b,f}^{lon} \wedge \bigcirc(\neg S_{l,r}^{lat} \wedge \neg S_{b,f}^{lon}) \right) \to \right.$$
$$\left. \bigcirc(P^{lat} \vee P^{lon}) \right)$$

$$\varphi^{\neg lat,\neg lon} \equiv$$
$$\left((\neg S_{l,r}^{lat} \wedge \neg S_{b,f}^{lon}) \to \bigcirc(P^{lat} \vee P^{lon}) \right)$$

where P^{lon} and P^{lat} are defined as

$$P^{lon} \equiv \left((S_{l,r}^{lat} \vee S_{b,f}^{lon})\overline{\mathcal{R}}_{[0,\rho)}(A_{b,maxAcc}^{lon}) \wedge \right.$$
$$(S_{l,r}^{lat} \vee S_{b,f}^{lon})\overline{\mathcal{R}}_{[\rho,+\infty)}(A_{b,minBr}^{lon})$$
$$P^{lat} \equiv \left(P_{0,\rho}^{lat} \wedge P_{\rho,\infty}^{lat,1} \wedge P_{\rho,\infty}^{lat,2} \right)$$

and the subformulas $P_{0,\rho}^{lat}$, $P_{\rho,\infty}^{lat,1}$ and $P_{\rho,\infty}^{lat,2}$ are defined as

$$P_{0,\rho}^{lat} \equiv (S_{l,r}^{lat} \vee S_{b,f}^{lon})\overline{\mathcal{R}}_{[0,\rho)}(A_{l,maxAcc}^{lat})$$
$$P_{\rho,\infty}^{lat,1} \equiv \left(S_{l,r}^{lat} \vee S_{b,f}^{lon} \vee V_{l,stop}^{lat} \right)\overline{\mathcal{R}}_{[\rho,+\infty)}A_{l,minBr}^{lat}$$
$$P_{\rho,\infty}^{lat,2} \equiv (S_{l,r}^{lat} \vee S_{b,f}^{lon})\overline{\mathcal{R}}_{[\rho,+\infty)}\left(V_{l,stop}^{lat} \to \right.$$
$$\left. (S_{l,r}^{lat} \vee S_{b,f}^{lon})\overline{\mathcal{R}}(V_{l,neg}^{lat}) \right)$$

By $V_{l,neg}^{lat}$ we mean that the $\mu-$lateral velocity of the ego car is less than or equal to zero.

Note that all the predicates' definitions and assumptions can be found in [22]. The formula $\varphi_{resp}^{lat,lon}$ consists of four subformulas such as for each a unique trigger or response is defined. The subformula φ^{lon} gets triggered if a safe longitudinal and an unsafe lateral distances become unsafe all together. We call the time that both lateral and longitudinal distances become unsafe, a dangerous time. If time t_b is a new dangerous time and after that until the current time t all times in $[t_b, t]$ are dangerous times, then we call t_b the Dangerous Threshold time as in [19]. The desired response for this case enforces some longitudinal restrictions on the behavior of the ego car. Similarly, the subformula φ^{lat} gets

triggered if an unsafe longitudinal and a safe lateral distances become unsafe all together. The desired response for this case enforces some lateral restrictions on the behavior of the ego car.

Unlike the φ^{lon} and φ^{lat} subformulas, the next two subformulas $\varphi^{lat,lon}$ and $\varphi^{\neg lat,\neg lon}$ are designed to cover the cases that either a situation changes from completely safe to unsafe, or it starts completely unsafe. The desired response to both cases requires either or both responses of the two previous responses.

IV. RSS Application on Automated Test Generation and Classification

By using the formalized RSS requirements as STL formulas, we can collect more meaningful samples for testing purposes. More specifically, we are interested in screening test driving scenarios and classifying them based on safety conditions introduced by RSS. One advantage of automated and guided test generation and classification is to only generate automatically annotated tests for testing and training perception based systems in ADS.

In this section, we show how one can utilize the formalized RSS requirements to select useful scenarios among a set of arbitrarily generated driving scenarios for testing purposes. We used S-TaLiRo and Sim-ATAV in our experiments for monitoring and simulating trajectories, respectively [25], [16], [14].

To show the effectiveness of our proposed framework, we represent and use a basic collision avoidance specification (CAS from now on) concerning the distances of an ego vehicle and other agent vehicles. We compare the result of monitored scenarios by using the CAS and the RSS requirements separately. We formalized the CAS requirements in STL representation in the following definition.

Definition IV.1 (Collision Avoidance Specification). *We call two objects collided if and only if their shortest distance get closer than a δ. The STL version of the requirement is shown as below:*

$$\varphi_{cas} \equiv \Box\neg\left(dx_{ego}^{a1} < \delta_x \wedge dy_{ego}^{a1} < \delta_y \wedge \right.$$
$$\left. dx_{ego}^{a2} < \delta_x \wedge dy_{ego}^{a2} < \delta_y \right)$$

where Table II represents the symbol descriptions.

For the tests, we only considered highway traffic scenarios, which by design do not include intersections, and wherein all the self-driving vehicles are moving in the same direction. In the scenarios, there are two agent vehicles and one ego vehicle. We generated our test scenarios with some restrictions for the initial positions, orientations, and speeds of the vehicles. Also, we enable the two agent vehicles to drive along the highway and maneuver randomly. We used the information in Table III as our test generating configuration parameters.

Note that our approach is not restricted to the presented scenarios, and it applies to the more general traffic scenarios which are discussed in [19]. We used DP-TaLiRo [25], [26]

TABLE II: The CAS symbol description in the safe lateral and longitudinal definitions. All distances are in meter.

symbol	description
dx_{ego}^{a1}	longitudinal distance between the go vehicle and the adversarial vehicle 1
dx_{ego}^{a2}	longitudinal distance between the go vehicle and the adversarial vehicle 2
dy_{ego}^{a1}	lateral distance between the go vehicle and the adversarial vehicle 1
dy_{ego}^{a2}	lateral distance between the go vehicle and the adversarial vehicle 2
δ_x	a constant safe longitudinal distance
δ_y	a constant safe lateral distance

TABLE III: The parameter values in the case-study and experiments.

parameter	value range	parameter	value range
x_{init}^{ego}	[0,5] m	y_{init}^{ego}	[-1.5,4.5] m
θ_{init}^{ego}	[-π/8,π/8]	v_{init}^{ego}	[10,25] m/s
x_{init}^{a1}	[20,40] m	y_{init}^{a1}	[-3.5,-1.5] m
x_{init}^{a2}	[10,25] m	y_{init}^{a2}	[1,4] m
v_{init}^{a1}	[5,15] m/s	v_{init}^{a2}	[0,30] m/s
y^{a1}	[-5,0] m	v^{a1}	[10,30] m/s
y^{a2}	[0,5] m	v^{a2}	[4,20] m/s

as our offline monitoring tool for computing the robustness of the traffic scenarios.

A. Vehicle Controller

In our experiments, we used an existing Sim-ATAV vehicle controller. The controller receives (provided by simulator) camera, LIDAR, and Radar sensor information as to its input signals. There is a perception system as part of the controller that performs sensor data processing and sensor fusion. The controller is capable of detecting object types (vehicle, pedestrian) and their predicted trajectories. The control algorithm sends signals to the throttle and steering actuators. The controller's goal is to safely follow a predetermined path. For more information, we refer readers to [14]. We chose $25m/s$ as the max and target velocity for the ego vehicle.

B. Simulated Tests and Environment

In our experiments, the simulated environment consists of a straight highway with three lanes. The initial test conditions concern position, orientation, and velocity of the ego vehicle. The initial test conditions for the agent vehicles concern the same parameters as the ego vehicle, except that their orientations are fixed.

At each run of test generation, a uniform random sampler randomly chooses the initial conditions of all the vehicles.

TABLE IV: The RSS parameter values in the case-study and experiments.

parameter	value	parameter	value
ρ	0.5 sec	μ	0.4 m
δ_t	0.01 sec	a_{minBr}^{lon}	4 m/s^2
a_{maxAcc}^{lon}	4.5 m/s^2	a_{minBr}^{lat}	3 m/s^2
a_{maxBr}^{lon}	2.5 m/s^2	a_{maxAcc}^{lat}	3 m/s^2

It also generates random trajectories for agent vehicles. In this experiment, we predetermined a fixed trajectory for the ego vehicle. However, the ego vehicle's controller is free to do collision avoidance maneuvers. We ran a (hit-and-run) sampler to produce and store one thousand of sample scenarios for the test parameters shown in Table III. We chose the parameter values for demonstration purposes and they might not reflect realistic values. During the rest of the section, we refer to the code-base that controls test screening and classifying *test control unit*. The details of tasks in Fig. 3 are explained in the following:

- *(a)*: The very first step is to design test scenarios and assign range values for configurable parameters. For example, we used data in the Table III in our experiments. Also, we used Sim-ATAV to design driving scenarios.
- *(b)*: We use S-TaLiRo as our monitoring tool, and configure it based on the our test design. Two important tasks here are to formalize a desirable specification in STL form and define input signals. Here we use two STL formulas in Def. III.1 and Def. IV.1 for comparison purposes.
- *(c)*: The S-TaLiRo tool is equipped with random samplers and optimizers. We use the random-walk algorithm from the S-TaLiRo to generate uniformly random test data. When optimization is needed, we use Simulated Annealing with hit and run Monte Carlo sampling. The test data consists of all the initial values for the vehicles (as in Table III) that are needed to run the designed simulation. The test control unit passes the generated test to the Sim-ATAV through an interface. Sim-ATAV initializes the simulator (Webots) and makes it ready to run the test. When the simulation terminates, the Sim-ATAV passes the simulated trajectories of all the vehicles to the test control unit. Next, the test control unit translates the trajectories into the lane-based coordinate system that the RSS rules are defined based on. Lastly, using the translated trajectories, we compute the final input signals to the monitor. That is, each signal represents a corresponding predicate in the RSS formula such as safe distances or μ-lateral velocities.
- *(d)*: Our DP-TaLiRo monitor computes the robustness of the generated driving trajectories (input signal) for the ego vehicle given an STL specification formula. The positive or negative sign of the computed robustness value states if the specification is satisfiable or falsifiable, respectively. The magnitude of the robustness value determines the resistance of the signal against perturbations.
- *(e)*: In this experiment, we are interested in accumulating test scenarios that do not meet safety standards based on RSS rules. If the robustness of the RSS specification was negative, then the test driving scenario does not satisfy the RSS requirements, and, therefore, it is a useful sample.
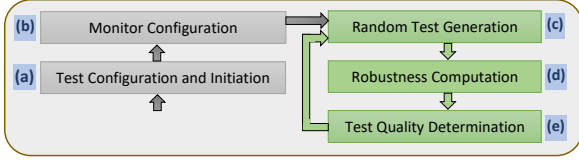
Fig. 3: The flowchart represents the executing order of major tasks in our framework.

(a) A sample trajectory of three vehicles. The ego vehicle's trajectory is shown in the middle lane and colored yellow. The red part of the vehicle's trajectory in the lowest lane highlights the close distance to the ego vehicle. The (b),(c),(d), and (e) refer to the times that below snapshots are taken.

(b) first snapshot.

(c) second snapshot.

(d) third snapshot.

(e) forth snapshot.

Fig. 4: A sample traffic scenario taken from Webots.

(a) A scenario starts in an unsafe longitudinal and safe lateral situation, and then it also becomes unsafe laterally when the cars change their lane.

(b) A scenario starts unsafe laterally and longitudinally. The blue agent vehicle turns in front of the ego vehicle and causes a virtual collision.
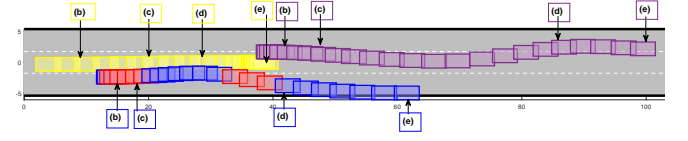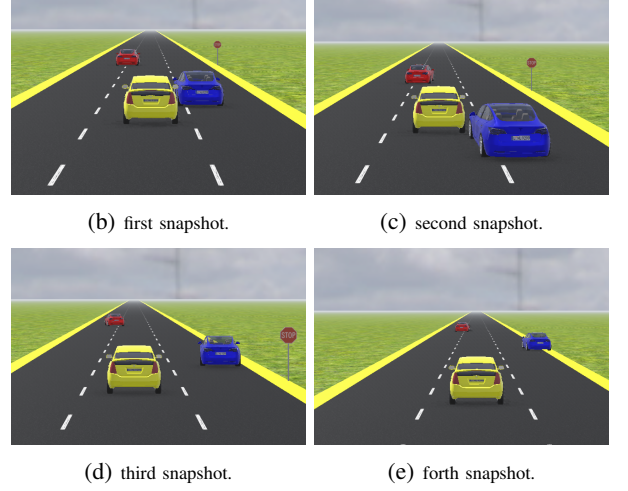
Fig. 5: Two common unsafe scenarios.

C. Case Study

We chose some interesting scenarios to explain and analyze here. In these scenarios we show how the RSS specification helps to select useful tests for verifying the vehicle's controller. The first scenario was test number 131 among 1000 uniformly generated tests. Figure 4a represents half of the trajectory in the test (from time $t = 0$ to time $t = 5$). The colors of boxes in each trace represents a vehicle in the simulation. The ego vehicle with yellow color drives in the middle lane. The two other vehicles in magenta and blue drive to the left and right of the ego vehicle. We highlighted the part of the trajectories that were close to the ego vehicle with red color. Also, under the trajectory diagram in Fig. 4, there are four snapshots taken from the simulation at different times.

We monitored the specification in Def. III.1 for the above scenario. The robustness result was negative, and the monitor reported the a_{maxAcc}^{lon} in subformula P^{lon} as the predicate that caused the maximum negative robustness value. This was due to high acceleration by the ego vehicle in the beginning of the scenario. The ego vehicle starts braking as soon as its controller detects the leading vehicle and assigns a high level of caution to it.

However, during the hesitation period, it did exceed the required maximum acceleration. Interestingly, this case did not violate the RSS rules for the blue vehicle driving closely to the right, but because of the red vehicle on the left of it.

In Figure 5, we demonstrated two common scenarios in our previous experiments. A test scenario illustrated in Fig. 5(a) starts for the ego vehicle in the right-most lane. The ego vehicle planned to change lane into the middle lane, and stay in that lane. The other agent vehicle in front of it started in the middle lane and then starts changing lane into the right-most lane. The vehicle control in the ego vehicle detects the front vehicle and tries to reduce its speed but not its direction. The RSS requirements demands for safe steering behavior which in this case was not respected, and therefore this is a useful test. In another situation demonstrated in Fig. 5(b), the ego initially is very close to the blue vehicle in front of it, and then the blue vehicle turns in front of it unsafely and collides with it. This scenario is initially unsafe, and the ego vehicle should respond according to either of the lateral or longitudinal safety requirements. If the initial speed of the ego vehicle is not too high, then it can brake on-time and satisfying the RSS requirements. Otherwise, the ego vehicle cannot react on-time and contributes to the collision, which qualifies it to be collected as a useful test. We emphasize that
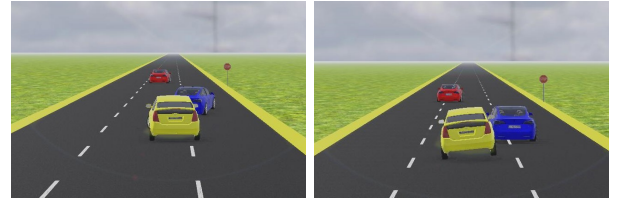
this class of test case results, which are hard for a human to detect, were one of the motivations for us to propose this approach for test generation.

In the next section, we discuss and present our experiments and their result.

D. The necessity of RSS in testing

For all the experiments in this section, we used the values in Table IV and Table III. Also, the length of each simulated driving scenario is 10 seconds. In our first experiment, we generated and stored one thousand uniformly random driving scenarios. We do not precisely follow the executing sequences shown in Fig. 3 as we are going to compare the result of monitoring the same test data against two different STL formulas. Next, we computed the robustness of the test data against CAS specifications.

TABLE V: Statistical result about robustness of monitoring RSS and CAS specifications. We used "pos" as an abbreviation of positive and "neg" as an abbreviation for negative.

description	values	description	values
Total pos (RSS)	771	Total pos (CAS)	406
Total neg (RSS)	229	Total neg (CAS)	594
Total zeros (RSS)	0	Total zeros (CAS)	0
one-by-one comparison between RSS and CAS results			
pos (RSS) pos (CAS)	387	neg (RSS) neg (CAS)	210
neg (RSS) pos (CAS)	19	pos (RSS) neg (CAS)	384

TABLE VI: The break down statistics of predicates that caused the highest negative robustness.

predicates	# of violations	description
$S_{b,f}^{lon}$	2	safe longitudinal distance
$S_{l,r}^{lat}$	23	safe lateral distance
$A_{l,maxAcc}^{lat}$	67	maximum allowed lateral acceleration
$A_{l,minBr}^{lat}$	0	minimum required lateral brake
$V_{l,stop}^{lat}$	68	zero $\mu-$lateral velocity
$V_{l,neg}^{lat}$	0	non-positive $\mu-$lateral velocity
$A_{b,maxAcc}^{lon}$	33	maximum allowed longitudinal acceleration
$A_{b,minBr}^{lon}$	36	minimum required longitudinal brake
Execution Statistics		
violation %	22.9%	falsified percentage using the RSS rules

The result is demonstrated in the third and fourth columns of the upper part of Table V. About 40% of the result did not violate their specification, and the other 60% did violate their specifications. The result of computed robustness of the test data against RSS specifications is shown in the first and second columns of the upper part of the same table. The 23% violated scenarios for the RSS case is noticeably less than the 60% for the CAS case. The lower part of Table V represents the statistics of all combinations of positive and negative results of each specification for each test. Note that the total number of tests for which both specifications falsified and satisfied them is 210 and 387, respectively. The result suggests that 384 falsified tests by the CAS specifications are not of our interest. Furthermore, 19 tests were falsified with the RSS requirements, but not with the CAS requirements. This implies that the test scenarios were potentially dangerous and hence worth further analysis, but they did not necessarily lead to a collision. In this experiment, using the RSS specifications, we can disregard 384 meaningless and add 19 meaningful tests (about 40% of the tests) that were misclassified by using a simple collision avoidance requirement (CAS). Table VI denotes the list of predicates

TABLE VII: Statistical result about robustness of monitoring RSS and CAS specifications in mostly unsafe situations. We used "pos" as an abbreviation of positive (satisfying specifications) and "neg" as an abbreviation for negative (violating specifications).

description	values	description	values
Total pos (RSS)	406	Total pos (CAS)	20
Total neg (RSS)	594	Total neg (CAS)	980
Total zeros (RSS)	0	Total zeros (CAS)	0
one-by-one comparison between RSS and CAS results			
pos (RSS) pos (CAS)	0	neg (RSS) neg (CAS)	574
neg (RSS) pos (CAS)	20	pos (RSS) neg (CAS)	406

in the RSS formula and the number of times that each one got falsified with the highest negative magnitude (most violating). A clustering algorithm can use the information about the falsified predicates to more tailor the test scenarios.

E. Improving search-based testing through RSS

In another experiment, we followed exactly the flow in Fig. 3. We engaged the optimizer and used the CAS specification as its cost function (s.t. optimality means lower robustness for an input signal) to find more dangerous test driving scenarios. We let the optimizer to find the worst falsifying behavior for the CAS specifications for one thousand iterations and stored the samples and their robustness result. The result is presented in the third and fourth columns of the upper part of Table VII. Exactly 2% of the result have positive robustness values, and the other 98% have negative values. The high percentage of negative result is because of the optimization. The result of computed robustness of the stored test data against RSS specifications is shown in the first and second columns of the upper part of the same table. The 60% negative robustness for the RSS case is substantially less than the 98% for the CAS case. The lower part of Table VII represents the statistics of all combinations of positive and negative results of each specification for each test. Note that the total number of tests for which both specifications falsified and satisfied them is 574 and 0, respectively. The result suggests that 406 falsified tests by the CAS specifications are not of our interest. Furthermore, 20 tests were falsified with the RSS requirements but not with the CAS requirements. In this experiment, using the RSS specifications, we can disregard 406 meaningless and add 20 meaningful tests (about 42% of the tests) that were misclassified by using a simple collision avoidance requirement.

F. Falsification by RSS

In the last experiment, according to the diagram in Fig. 3 we engaged the optimizer and used the RSS specification as its cost function to find more relevant test driving scenarios. We let the optimizer to try to optimally falsify the RSS specifications for 350 iterations and stored the samples and their robustness result. Similar to what we did in the last two experiments, we repeated the process for the CAS specifications. The result are depicted in Table VIII and Table IX. The optimizer is especially helpful when falsification is the priority. An alternative approach could be running a script that screens all the randomly generated tests and computes and classifies their safeties. However, this approach can take longer to falsify the specifications in comparison to using an optimizer.

The predicate statistics in Table VI and Table IX suggests that we can further classify test scenarios based on their violated constraints. For example, we might only be interested in unsafe test scenarios in which the maximum allowed lateral acceleration denoted by predicate a_{maxAcc}^{lat} was violated by the controller.

TABLE VIII: Statistical result about robustness of monitoring RSS and CAS specifications where we used the RSS rules for optimization. We used "pos" as an abbreviation of positive and "neg" as an abbreviation for negative.

description	values	description	values
Total pos (RSS)	292	Total pos (CAS)	50
Total neg (RSS)	58	Total neg (CAS)	300
Total zeros (RSS)	0	Total zeros (CAS)	0
one-by-one comparison between RSS and CAS results			
pos (RSS) pos (CAS)	49	neg (RSS) neg (CAS)	57
neg (RSS) pos (CAS)	1	pos (RSS) neg (CAS)	243

TABLE IX: The break down statistics of predicates that caused the highest negative robustness in the last experiment.

predicates	# of violations	description
$S_{b,f}^{lon}$	2	safe longitudinal distance
$S_{l,r}^{lat}$	3	safe lateral distance
$A_{l,maxAcc}^{lat}$	13	maximum allowed lateral acceleration
$A_{l,minBr}^{lat}$	0	minimum required lateral brake
$V_{l,stop}^{lat}$	0	zero $\mu-$lateral velocity
$V_{l,neg}^{lat}$	0	non-positive $\mu-$lateral velocity
$A_{b,maxAcc}^{lon}$	35	maximum allowed longitudinal acceleration
$A_{b,minBr}^{lon}$	5	minimum required longitudinal brake
Execution Statistics		
violation %	16.5%	falsified percentage using the RSS rules

V. CONCLUSION

In this paper, we present an automated and qualification based method for generating driving test scenarios. The generated tests could be used for discovering control software bugs in autonomous vehicles.

We integrated the RSS requirements into our Sim-ATAV framework, which is publicly available as part of S-TaLiRo tool. Now it is possible to test the control and perception system stack against the RSS model.

REFERENCES

[1] National Highway Traffic Safety Administration (NHTSA). Automated vehicles for safety. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[2] M. Siam, S. Elkerdawy, M. Jagersand, and S. Yogamani, "Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges," in *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

[3] T. B. Lee, "Report: Software bug led to death in Uber's self-driving crash," *Ars Technica*, 5 2018.

[4] T. T. Johnson, R. Gannamaraju, and S. Fischmeister, "A survey of electrical and electronic (e/e) notifications for motor vehicles," in *ESV'15*, 2015.

[5] D. o. M. V. State of California, *2019 Autonomous Vehicle Disengagement Reports*, 2019 (accessed March 3, 2020). [Online]. Available: https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/disengagement_report_2019

[6] C. E. Tuncali, T. P. Pavlic, and G. Fainekos, "Utilizing S-TaLiRo as an automatic test generation framework for autonomous vehicles," in *IEEE Intelligent Transportation Systems Conference*, 2016.

[7] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, "Adaptive stress testing with reward augmentation for autonomous vehicle validatio," in *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.

[8] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia, "Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems," in *International Conference on Computer Aided Verification*. Springer, 2019, pp. 432–442.

[9] R. Majumdar, A. Mathur, M. Pirron, L. Stegner, and D. Zufferey, "Paracosm: A language and tool for testing autonomous driving systems," *arXiv preprint arXiv:1902.01084*, 2019.

[10] H. Abbas, M. O'Kelly, A. Rodionova, and R. Mangharam, "Safe at any speed: A simulation-based test harness for autonomous vehicles," in 7th *International Workshop on Cyber-Physical Systems (CyPhy)*, 2017.

[11] M. Koschi, C. Pek, S. Maierhofer, and M. Althoff, "Computationally efficient safety falsification of adaptive cruise control systems," in *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.

[12] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts, "Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 45–64, 2016.

[13] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: a language for scenario specification and scene generation," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2019, pp. 63–78.

[14] C. E. Tuncali, G. Fainekos, D. Prokhorov, H. Ito, and J. Kapinski, "Requirements-driven test generation for autonomous vehicles with machine learning components," *IEEE Transactions on Intelligent Vehicles*, 2019.

[15] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[16] G. Fainekos, B. Hoxha, and S. Sankaranarayanan, "Robustness of specifications and its applications to falsification, parameter mining, and runtime monitoring with s-taliro," in *International Conference on Runtime Verification*. Springer, 2019, pp. 27–47.

[17] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Nickovic, and S. Sankaranarayanan, "Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications," in *Lectures on Runtime Verification*, ser. LNCS. Springer, 2018, vol. 10457, pp. 128–168.

[18] H. Abbas, M. E. O'Kelly, A. Rodionova, and R. Mangharam, "A driver's license test for driverless vehicles," *Mechanical Engineering*, vol. 139, pp. S13–S16, 2017.

[19] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv:1708.06374v6*, 2018.

[20] B. Gassmann, F. Oboril, C. Buerkle, S. Liu, S. Yan, M. S. Elli, I. Alvarez, N. Aerrabotu, S. Jaber, P. van Beek, D. Iyer, and J. Weast, "Towards standardization of AV safety: C++ library for responsibility sensitive safety," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[21] P. Koopman, B. Osyk, and J. Weast, "Autonomous vehicles meet the physical world: RSS, variability, uncertainty, and proving safety," in *International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, ser. LNCS, vol. 11698, 2019, pp. 245–253.

[22] M. Hekmatnejad, S. Yaghoubi, A. Dokhanchi, H. B. Amor, A. Shrivastava, L. Karam, and G. Fainekos, "Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic," in *17th ACM-IEEE International Conference on Formal Methods and Models for System Design*, 2019.

[23] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 719–726.

[24] H. Abbas, G. E. Fainekos, S. Sankaranarayanan, F. Ivancic, and A. Gupta, "Probabilistic temporal logic falsification of cyber-physical systems," *ACM Transactions on Embedded Computing Systems*, vol. 12, no. s2, May 2013.

[25] S-TaLiRo Tools, "https://sites.google.com/a/asu.edu/s-taliro/."

[26] G. E. Fainekos, S. Sankaranarayanan, K. Ueda, and H. Yazarel, "Verification of automotive control applications using s-taliro," in *2012 American Control Conference (ACC)*. IEEE, 2012, pp. 3567–3572.