The Supplementary Materials

I. Proof Of the calculation of Θ

Proof. We first list the following axioms that are assumed to be true.

- $\pi \vDash \neg\mu \implies \pi \nvDash \mu$
- if $\Phi = x \wedge y, \pi \vDash \neg x \implies \pi \nvDash \Phi, \pi \vDash \neg y \implies \pi \nvDash \Phi$
- if $\Phi = x \vee y, \pi \vDash (\neg x \wedge \neg y) \implies \pi \nvDash \Phi$
- $N(\neg\theta_1) = \Theta(\theta_1), \Theta(\neg\theta_1) = N(\theta_1)$

If Φ is μ, we have $\pi \vDash \neg\mu \implies \pi \nvDash \mu$, so $\Theta(\mu) = \{\neg\mu\}$ is reasonable.

If Φ is $x \wedge y$, we have $\pi \vDash \neg x \implies \pi \nvDash \Phi, \pi \vDash \neg y \implies \pi \nvDash \Phi$, so $\Theta(x \wedge y) = \Theta(x) \cup \Theta(y)$ is reasonable.

If Φ is $x \vee y$, we have $\pi \vDash (\neg x \wedge \neg y) \implies \pi \nvDash \Phi$, so $\Theta(x \vee y) = \{x \wedge y \mid x \in \Theta(a) \wedge y \in \Theta(b)\}$ is reasonable.

If Φ is $\bigcirc x$, we have $(\pi, t) \vDash \bigcirc\Phi \iff (\pi, t+1) \vDash \Phi$. Given $\pi \vDash \neg\mu \implies \pi \nvDash \mu$, we can get $(\pi, t) \vDash \bigcirc\neg\Phi \iff (\pi, t+1) \nvDash \Phi$. Hence, $\Theta(\bigcirc x) = \{\bigcirc x' \mid x' \in \Theta(x)\}$ is reasonable.

If Φ is $x \ \mathcal{U}_\mathcal{I} \ y$, in our definition, there are two parts of $\Theta(x \, \mathcal{U}_\mathcal{I} \, y)$. The first part is set Θ_1: $\{x' \wedge y' \mid x' \in \Theta(x) \wedge y' \in \Theta(y)\}$ which implies these equations are satisfies: $\pi \vDash x' \implies \pi \nvDash x, \pi \vDash y' \implies \pi \nvDash y$. Given the definition of $\mathcal{U}_\mathcal{I}$: $(\pi, t) \vDash x \, \mathcal{U}_\mathcal{I} \, y \iff \exists t' \in t + \mathcal{I}$ such that $(\pi, t') \vDash y \wedge \forall t'' \in [t, t'], (x, t'') \vDash x$, we can easily get:

$$\forall \xi \in \Theta_1.\ \pi \vDash \xi \implies \pi \nvDash x \, \mathcal{U}_\mathcal{I} \, y$$

The second part is set Θ_2: $\{x' \, \mathcal{U}_\mathcal{I} \, y' \mid x' \in \Theta(\neg x \vee y) \wedge y' \in \Theta(x \vee y)\}$. In order to obtain a contradiction, assume that there is an element ξ of set Θ_2 that satisfies $\pi \vDash \xi \implies \pi \vDash \Phi$. Then, for $\pi \vDash \Phi$, we get $\exists t' \in t + \mathcal{I}$ such that $(\pi, t') \vDash y \wedge \forall t'' \in [t, t'], (x, t'') \vDash x$, which means $(x \wedge \neg y)$ is satisfied until y is satisfied. For $\pi \vDash \xi$, we get $\exists t' \in t + \mathcal{I}$ such that $(\pi, t') \vDash (\neg x \wedge \neg y) \wedge \forall t'' \in [t, t'], (x, t'') \vDash (x \wedge \neg y)$, which means $(x \wedge \neg y)$ is satisfied until $(\neg x \wedge \neg y)$ is satisfied. Hence, at time step t, if x is satisfied in $[t, t']$ and violated at time step t'' after t', we should get that y is satisfied before t'' and y is violated before t'' at the same time. This is a contradiction, and so the assumption that there is an element ξ of set Θ_2 satisfies $\pi \vDash \xi \implies \pi \vDash \Phi$ must be false. We can get:

$$\forall \xi \in \Theta_2.\ \pi \vDash \xi \implies \pi \nvDash x \, \mathcal{U}_\mathcal{I} \, y$$

Hence, $\Theta(x \ \mathcal{U}_\mathcal{I} \ y) = \{x' \ \mathcal{U}_\mathcal{I} \ y' \mid x' \in \Theta(\neg x \vee y) \wedge y' \in \Theta(x \vee y)\} \cup \{x' \wedge y' \mid x' \in \Theta(x) \wedge y' \in \Theta(y)\}$ is reasonable.

Since the temporal operators $\mathcal{U}_\mathcal{I}$ and $\bigcirc$ are functionally complete, we omit the proof of the remaining temporal operators. $\square$

II. Basic Operation of Genetic Algorithm

Our law-breaking fuzzing algorithm must be customized for every DSL language and is based on the Genetic Algorithm. As shown in *Genetic Algorithm* [1], the population encoding, mutation and crossover are basic operations of the genetic algorithm. We customize the population encoding, mutation and crossover for the design of law-breaking fuzzing algorithm.

Population encoding. We describe a test case by depicting the ego vehicle, NPC vehicles, pedestrians, static obstacles, and the environment. Each object has different operable parameters. For the ego vehicle, the operable parameter is its starting point. For NPC vehicles and pedestrians, all the parameters except the speed at the destination are operable, since the speed at the destination is always zero. For static obstacles and the environment, all the parameters in the DSL are operable.

These operable parameters are divided into six divisions: *position, speed, type, color, daytime, and weather*. The division of position can be divided into three sub-divisions: vehicle position(i.e, ego position and NPC vehicles' positions), pedestrian position, and obstacle position. Note that the vehicle position is always within the lane or junction area, while pedestrians may move around the empty regions of the map, and obstacles can be placed at anywhere of the map (e.g, a basketball on the road, a meteorite on the roof, or a kite stuck in a tree). The division of speed can also be classified into vehicle speed and pedestrian speed due to different speed ranges. Similarly, the division of type can be divided into vehicle type, pedestrian type, and obstacle type.

Among the above divisions, the value of position and speed are continuous, which leads to real-value encoding for vehicle position, pedestrian position, obstacle position, vehicle speed, and pedestrian speed. The trajectory of NPC vehicles and pedestrians is defined by a sequence of states, including the initial and target state. Every state contains information of position and speed, we call every state among the trajectory a waypoint and. Because vehicles are moving along lanes, we can translate the position of a vehicle in a waypoint into a lane position, allowing the encoding of each position to be described by its offset in the lane position format. Then, for every waypoint, the lane ID remains the same and the position and speed are encoded into a pair to form the chromosomes of vehicle position and vehicle speed. The encoding for pedestrian(or obstacle) is the same except for the different regions of position. The divisions of type, color, and daytime are discrete values. They are encoded as discrete vectors. The division of weather is a dictionary variable and the value of every key is encoded as a continuous value.

Crossover. For each pair of test cases, the crossover can be

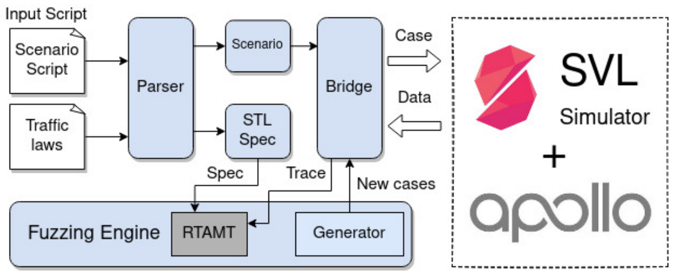

Fig. 1: The implementation with the LGSVL+Apollo simulation environment.

done only in the same category. In theory, we can perform crossover for each category. In this work, to avoid the problem of infeasible scenarios, we abandon the crossover on the chromosomes of vehicle and pedestrian positions.

Mutation. The mutation is necessary for every division. For division with continuous value, i.e., position, speed, and weather, we apply Gaussian mutation. We also apply the clipping function to avoid invalid values.

III. Implementation

We implemented `LawBreaker` based on the LGSVL simulation framework and evaluate it with two versions of Baidu Apollo. Note that our approach can also be implemented for other platforms (e.g. Carla+Autoware, LGSVL+Autoware) since our specification language and fuzzing algorithms are independent of the underlying simulation environment.

The workflow of our implementation is shown in Figure 1. The input consists of the scenario description in AVUnit and traffic law specifications in `LawBreaker`, whereas the output consists of a set of test cases that violate those traffic laws in different ways. In the following, we describe the various modules we implemented to connect the specification parser, the fuzzing engine, and the simulator with ADS frameworks.

Parser. In the parser module, the script is divided into two parts: the scenario description and the traffic laws. These two parts are parsed separately since the traffic laws can be translated to an independent abstract syntax tree (AST) for the simulator, i.e. without the scenario constructs. Our parser for `LawBreaker` is implemented using *ANTLR4* [2]. The parser checks the grammar of the specifications and translates our language to standard STL formulas.

Bridge. Since customisation is necessary for the simulators and ADSs, we need a bridge that adapts the scenarios to a format recognised by the simulator, and also collects and extracts the variables from the messages published by the ADS. For instance, we can get the position of the ego vehicle and map information. Therefore, we can get the area “ahead” of the ego vehicle. Any NPC vehicle within this “ahead” area is recognized as “NPCAhead”. Overall, the bridge can spawn scenarios for the simulator and generate a trace using the collected data.

Fuzzing Engine. Our implementation supports the mutation of six categories, i.e. position, speed, time, weather, NPC vehicle type, and pedestrian type. Furthermore, we embedded the tool *RTAMT* [3] to compute the robustness of the specifications with respect to the trace obtained from the bridge.

LGSVL+Apollo. The maps we use are all from the map store [4] of LGSVL. For the Car model, we choose *Lincoln-MKZ2017*—the detailed model information can be found online [5]. The available sensor modules (e.g. Planning, Control, Prediction, PerceptionObstacle) of Apollo are all enabled to ensure that the ADS can run normally. Note that since our goal is to test the ADS and the perception part of Apollo 6.0 is still under development, we follow the recommendation of the vendor and use ground truth as the input of the module `PerceptionObstacle`.

References

[1] S. Mirjalili, “Genetic algorithm,” in *Evolutionary algorithms and neural networks*. Springer, 2019, pp. 43–55.
[2] Terence Parr, “antlr4,” https://github.com/antlr/antlr4, 2021, online; accessed Oct 2021.
[3] D. Ničković and T. Yamaguchi, “RTAMT: Online robustness monitors from STL,” in *International Symposium on Automated Technology for Verification and Analysis*, 2020, pp. 564–571.
[4] LG Silicon Valley Lab, “Maps content,” https://content.lgsvlsimulator.com/maps/, 2019, online; accessed Nov 2021.
[5] ——, “Vehicles content,” https://content.lgsvlsimulator.com/vehicles/, 2019, online; accessed Nov 2021.