Integration of Formal Specification and Traffic Simulation for Scenario-Based Validation

Abstract—**Scenario-based testing has been used for validation and verification of autonomous vehicles in various settings. In these efforts, simulation platforms are extensively utilized to run and analyze test scenarios, which is crucial for the validation of perception, decision-making and action in different traffic situations. Providing formal specification for scenarios used in these simulations is also critical for standardization, repeatability and verification. In this paper, we propose an approach which integrates an open-source network-based traffic simulator with an open-source formal scenario description language designed for formal scenario specification of autonomous vehicle validation scenarios. This approach provides the capability to describe, generate, and share abstract traffic scenarios for validation. It also allows integration with higher fidelity or physical testing methods as it utilizes a widely used scenario specification methodology. We provide an integration of formal specification and traffic simulation and also show the potential of such an integration through simulation examples.**

Index Terms—**Autonomous Vehicles, Scenario-Based Testing, Validation, Formal Scenario Specification, Traffic Simulation**

I. Introduction

The road network is the domain where vehicles may freely and reliably travel from point A to point B so long as they conform to the physical properties of the road, and local regulations enforced by fines and punishment. This universal adherence to properties of a road network allows for modelability of transportation engineering problems, including traffic modeling. As autonomous vehicles operate on the same road network, and must also conform to all road network properties, it is apparent that traffic modeling techniques may compliment autonomous vehicle (AV) validation. Furthermore, this would be an optimal combination for decision making validation in particular when combined with a robust traffic network.

Other motivations towards the integration of formal specification and traffic simulation are:

- The neccesary to have a formal specification of scenarios for decision making validation of AV scenarios.
- It is advantageous or even necessary to have a scenario description method open-source so that not only the scenarios but also the method to create new scenarios is repeatable, shareable, and accessible.

Scenic [1] is an open–source tool for formal scenario specification (FSS), and SUMO [2] is a well-known, open–source, and widely used simulator for generating network-based traffic scenarios. These software are ideal to use to introduce a methodology of integrating FSS and network-based traffic simulation for AV decision validation through simulation.

The main contributions of this work can be listed as follows:

- To the best of our knowledge, we present the first integration of the open-source formal scenario description language Scenic and widely-used traffic scenario generation tool SUMO for scenario-based AV testing.
- We present an open-source scenario generation method for the validation of AVs.
- We generate example scenarios with different traffic network contents using our integrated methodology.
- We provide the open-source code repository of the SUMO–Scenic integration and all scene examples in this paper as a fork of Scenic V2.0.0.

The methodology provides the capability to generate formal models of validation scenarios and formal descriptions of safety metrics in simulation. While this methodology is designed to generate abstract simulation models for validation of decision making in AVs, formal specifications also enables the potential integration with higher fidelity simulation or physical testing.

The rest of the paper is organized as follows. Related Work is discussed in Section II. Afterwards, the integration methodology is presented in Section III. Next, example AV traffic scenarios showcase the integration in Section IV. Finally, the paper concludes in section V.

II. Related Work

Formal scenario definition of AV validation scenarios is neccesary in order to share scenarios and allow for reproducibility of scenarios. ASAM OpenSCENARIO v1.2 [3] is a standard for describing concrete and logical AV scenarios for AV simulators. ASAM OpenSCENARIO v2.0 [4] supports concrete and logical, as well as abstract scenarios. Abstract scenarios are an abstraction level above logical abstraction level featuring high level timing and execution constraints or vague location requirements etc. [5]. While similar in name, v1.2 and v2.0 are both prominent and independently developed standards, but are currently incompatible, however ASAM expects the two versions to converge in the upcoming years. The Measurable Scenario Description Language (M-SDL) by Foretellix [6] is the basis of ASAM OpenSCENARIO v2.0. An open-source release of M-SDL is available. M-SDL is used in the AV validation software Foretify by Foretellix [7], which is proprietary software.

Scenic [1] is another notable scenario description language which provides formal scenario description. Scenic is an open-source language with more popularity than M-SDL in related literature, utilized in AV validation research for AV scenarios. In addition to the examples in [1] with Grand Theft Auto 5 scenarios, Scenic is used with Carla to generate scenarios featuring road debris and oncoming vehicles avoidance, and traffic light and speed limit recognition [8]. It is also used with the VerifAI toolkit [9] to design and validate TaxiNet, a machine learning approach for an intelligent aircraft taxiing system [10]. Another recent example implements Scenic for LGSVL simulations [11].

Planning and decision making has become it's own layer of abstraction in AV validation, seperate from motion control and perception [12]. This distinction is becoming more prevalent in AV research [13]. As such network-based traffic simulators have become a ideal option for AV validation testing since a traffic network operates on definable rules including physical properties of vehicles and traffic rules of road edges, etc.

SUMO is well-known and widely used to simulate traffic scenarios for AV validation, recently Jing et al. [14] use SUMO validate a cooperative control algorithm for AVs on highway on-ramps, and Chamideh et al. [15] evaluates real-world autonomous intersection management methods using SUMO to manage large traffic networks. This research makes good use of SUMO's microscopic network management allowing the authors to analyses decision level problems while leaving the task of vehicle simulation to SUMO.

Simulation-based AV validation testing is an integral component in targeted scenario selection. Often, the intent of complex AV Validation scenarios is to find failure scenarios [16] or to score scenarios based on performance. Mullins et al. [17] classifies performance regions for black-box AV navigation scenarios in two and three dimensions (3D) to identify objects, boundaries, and regions of interest. More recently Sun et al. [18] locates safe or dangerous scenarios using concrete scenario configurations of a three parameter car following scenario, correlating the three parameters as a 3D configuration space with distinct safe/dangerous scores for any given configuration. The versatility of grouping scenarios based on performance is useful for failure scenarios due to safety violations of traffic [19] or recognizing guaranteed failure conditions [20].

III. Methodology

The main goal of this research is the integration of an FSS language with a network-based traffic simulator. To the best of our knowledge, our approach is the first open-source framework for this integration. The main components of the framework for this integration is presented in Figure 1.

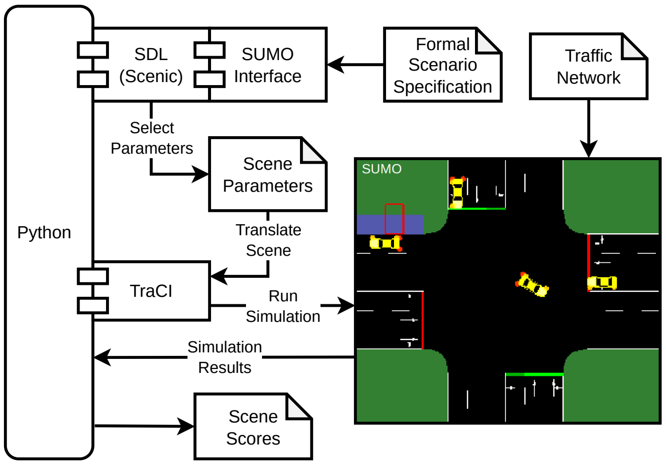

Fig. 1: The framework of the integration of a formal scenario specification language with a network-based traffic simulator for AV validation.

A. Integration Approach Requirements and Flow

We define the requirements of the integration framework as follows:

- An FSS of an AV traffic scenario, which defines the scenario in a way that is repeatable, reproducible, and shareable.
- A traffic network definition which describes the road layout, road rules, traffic infrastructure, traffic flows, and other physical and temporal information used to describe a road network.
- A scenario description language (SDL) which uses concrete parameter values from every parameter range of an FSS.
- A network-based traffic simulator to construct a scene given concrete parameters and to perform simulation tests.
- Quantitative scoring metrics which evaluate or describe the results of a simulation test.

Scenic SDL and SUMO Traffic Simulator are the ideal choices for formal scenario specification and simulation components respectively as they are both open source software tools, both are extendable and share a common scripting language, i.e. Python. AS shown in Fig.1, we also utilized the **Tra**ffic **C**ontrol **I**nterface (TraCI) [21] for microscopic control of SUMO simulations.

It's also important to understand the flow within the framework. In our flow, an FSS describes an AV validation scenario which occurs on a traffic network. Then the SDL (Scenic) selects scene parameters. The scene parameters are the configuration to a concrete scenario. Finally, the scenario is simulated and the results are evaluated.

Four of the framework components are displayed as documents in Figure 1. This is to highlight the formal components of scenario, network, and concrete scenarios:

- Formal Scenario Specification
- Traffic Network
- Scene Parameters
- Scene Scores

A scene is recorded as a parameter configuration→scene score pair, which is the representation of performance mode sampling for blackbox AV scenario testing. The correlation of parameter configuration→scene score is possible for formal, concrete scenarios as they are reproducible and should result in the same score when tested in the same environment. Mullins et al. [17] used a similar approach, however their main focus was navigation in two and three-dimensional environments.

B. Formal Object Specification

The objects in an FSS define position, behaviour, and other characteristics of the objects of a scene. The formal object specification for the objects within a network-based traffic simulator must be created in a way that formally specifies:

- The position in traffic-network terms, e.g. junction (intersection), edge (road), lane, etc.
- The mission of network-traversing objects, e.g. the route that the object will take from point A to point B.
- The objects adherence to network-rules, e.g. obeying speed limits, intersection priority rules, lane change restrictions, traffic lights, etc.
- Configuration and program logic of network-infrastructure, e.g. traffic light configuration and timing.

Next we will discuss how formal object specification in application using three well-known examples of cars, traffic lights, and pedestrians in SUMO. It is important to note that these are chosen to explain and demonstrate the concepts that may be applied to any object. However, concrete examples of objects commonly used in traffic scenarios are chosen to support the explanation of the methodology. In the next section, these examples will be shown within experiment scenarios. The nomenclature of the Car, Traffic Light, and Pedestrian object example descriptions are influenced by traffic-network terms.

The first example object is the Car. Cars and their variations, e.g. trucks, motorcycles, vans, etc., traverse a traffic network to get from point A to point B. After all, traffic is associated with many vehicles moving along the road. In traffic simulations, vehicle flows and trips of realistic or predicted traffic patterns may be included in the network configuration. In addition to the vehicles which are already a part of the network configuration, it is necessary to add individual vehicles to a scene such as adding an autonomous vehicle.

The properties of an example Car object are described in Table I. At a minimum, each Car object requires a unique name and a route. A route consists of explicitly defined and connecting edge identifiers (IDs) in the network to follow or two edge IDs. For the latter case, the edges of the route will be determined by a route-finding strategy in the network configuration.

TABLE I: Car Object Properties

Syntax	**Description**
name *string*	Vehicle ID.
route [*string*, ...]	Edge IDs of vehicle route.
distance *float*	Initial position along first edge in route.
lane *int*	Initial lane index of first edge in route.
xPos *float*	Initial X position.
yPos *float*	Initial Y position.
angle *float*	Initial angle (in degrees).
vehPlacement *int*	Vehicle placement options. Used when (x,y) position is explicitly set.
speed *float*	Vehicle speed (in mps).
speedMode *int*	Selects which network speed rules the vehicle follows.
track *boolean*	Follow vehicle in GUI.
color [*int,int,int,int*]	Vehicle color as a 32-bit RGBA.
changeSpeed [*float,float,float*]	Adjust speed of vehicle. [Speed (in mps), duration, At time...]
tau *float*	Desired minimum time headway.
carParam *boolean*	Enables collisions with cars in a junction.
parkPos [*string,float*]	Add a parking spot to vehicle route. [Parking Spot ID, Duration]
laneMode *int*	Lane change behavior.
laneChanges [*int,float*]	Add a lane change event. [Lane Index, At time...]

Angle, (X,Y) position, and lane properties are relevant for initial placement and non-moving actors only, as the vehicle will adhere to the network rules and traffic demands as it progresses along the route (e.g. a vehicle will stay in the rightmost lane unless it is faster to travel in another lane due to traffic or closed lanes).

The rules which a vehicle will follow and/or ignore must also be configurable. Fot instance, two important examples can be listed as follows:

- Changing the speedMode to ignore a red light or ignore the speed limit.

- Changing the laneMode to stay in the initial lane or merge without verifying there is adequate space for the vehicle.

Some properties such as speed or tau or planned events such as changeSpeed or laneChanges are properties included for Car as they add constraints or events to the mission which may effect the ability to complete a route or respond to a traffic event properly.

The second example object is the Traffic Light. Traffic lights are active components of a traffic network. Vehicles on the network obey the traffic light phases just as they do for yield-to-incoming-edge or all-way-stop junctions. The properties of an example Traffic Light object are described in Table II.

TABLE II: Traffic Light Object Properties

Syntax	**Description**
name *string*	Traffic Light ID.
state [*string*, ...]	Connection state specification for all phases.
duration [*float*, ...]	Phase duration.

A traffic network may have one or more traffic-light controlled junctions specified in the network configuration with a predefined program which controls the light phases of every lamp in the junction. A formal object specification in an FSS allows for an existing traffic light program to be overwritten. Each phase of a traffic light program consists of:

- The state of each connection (green, yellow, red).
- The duration of the phase before the next phase begins.

The third example object is the Pedestrian. Traffic networks may include sidewalks and vehicle-prohibited pathways for pedestrians to safely navigate from point a to point b, especially in urban networks. Just like car traffic, pedestrian traffic may be included in the network configuration as scheduled flows or patterns of pedestrians. The properties of an example Pedestrian object are described in Table III.

TABLE III: Pedestrian Object Properties

Syntax	**Description**
name *string*	Pedestrian ID.
route [*string*, ...]	Edge IDs of pedestrian route.
departTime *float*	Simulation time of trip departure.
distance *float*	Initial position along first edge in route.
arrivalPos *float*	Arrival position along last edge in route.
color [*int,int,int,int*]	Pedestrian color as a 32-bit RGBA.
egoWaitAtXing *bool*	Records timestamp when ego is waiting for this pedestrian to cross any pedestrian crossing.

Each Pedestrian requires a name and a route. The route must consist of connected edges which allow pedestrians or two edge IDs. For the latter case, the edges of the route will be determined by a route-finding strategy in the network configuration. Point A and B in a pedestrian's route is not always at the start or end of a path segment. As such The departure position on the first edge and the arrival position on the last edge of the pedestrian's route should be configurable.

Finally, A custom property is assigned to the Pedestrian object for reporting purposes. This property functions as a flag and when it is true, the simulation time will be recorded when the pedestrian is within a pedestrian crossing while an ego vehicle is present and stopped before that crossing. This can be critical for the simulation results analysis.

IV. Scenario Examples

This section provides three example scenarios to showcase the integration of formal description of AV validation scenarios to a network-based traffic simulator. For this purpose a SUMO interface for Scenic is created. The implementation of the SUMO–Scenic interface and examples described in this paper are open-source and available for access at [22].

To show the extedibility of this integration and for good readability, the experiments section is partitioned into three subsections and each subsection consists of a scenario which is an extension of the previous. In the first subsection, an existing AV validation scenario from literature is reimagined and integrated with a configured traffic network. In the second subsection, the programming of a traffic light in the network is overwritten and scene configurations of ego decision failure simulation tests are located. In the third subsection, pedestrians and a pedestrian crossing is added to the scenario and simulation tests are performed to determine the impact of pedestrians.

A. Adding Cars Within a Traffic Network

An example scenario is selected from the AV scenario database of [23]. The scenario database is accessible at [24] which contains modular and measurable AV scenarios formally specified using M-SDL. Each of these scenarios includes an ego vehicle that is under test and named by using a simple scheme, which leverages the sequential combination of the following scenario features:

- The number of non-ego actors.
- A letter which describes the location of the scenario. S for any road segment, T, X, and M for a 3/4/5-way intersection respectively.
- An iterable number which is the scenario's unique ID for the actor/location category.

The scenario selected to be respecified in Scenic is scenario 2S13, a scenario with two non-ego actors which takes place on any road segment. The non-ego actors are referred to as NPCs. One NPC is in the lane to the side of the ego while the second NPC is behind and in the same lane as the ego. We model the Scenario 2S13 in Scenic as follows:

```
from scenic.simulators.sumo.model import *
from scenic.core.distributions import *
import random

model scenic.simulators.sumo.model

kph2mps = 1/3.6

the_route = ["492300616#0", "159453012"]
lanes = [0,1]
ego_lane = random.choice(lanes)
lanes.remove(ego_lane)
npc1_lane = random.choice(lanes)
ego_dist = 100

ego = Car at 0 @ 0,
    with lane ego_lane,
    with distance ego_dist,
    with track True,
    with route the_route,
    with name "ego",
    with color [255,0,0,255],
    with speed Range(1,100) * kph2mps

npc1 = Car at 0 @ 2,
    with name "npc1",
    with lane npc1_lane,
    with color [255,255,0,255],
    with distance Range(-100,100) + ego_dist,
    with route the_route,
    with speed Range(0,100) * kph2mps,
    with laneChange [ego_lane, Range(0,60)]

npc2 = Car at 0 @ 4,
    with name "npc2",
    with lane ego_lane,
    with color [255,255,0,255],
    with distance Range(-100,-10) + ego_dist,
    with route the_route,
    with speed Range(0, 100) * kph2mps
```

The parameters of the 2S13 scenario are the actor lanes, actor speed, and NPC positions relative to the ego position. any parameter which is not explicitly set in the Scenic specification is controlled by the default SUMO controller. All vehicle actors travel the same route from the bottom to the top of the CurvyRoad network (Figure 2), which contains four traffic lights across three junctions.

Several moments of scenes generated from the 2S13 Scenic program are displayed in Figure 3. Figure 3A and 3B are two separate moments captured at the start of a scene where `ego` begins 100 meters along the network edge `492300616#0`, `npc1` is ahead of `ego` and in a separate lane compared to `ego`, and `npc2` is behind `ego` and in the same lane.

1) Results: It is important to note that the scenario in this section is not merely a re-implementation of the 2S13 scenario from M-SDL to Scenic, rather a reimagining of AV scenario generation, and the integration into a traffic-network.

The intent of the scenarios dataset in [23] is to be able to combine concise atomic scenarios from the dataset in serial to generate complex AV scenarios. In this experiment, the 2S13 scenario is implemented into a fully-configured traffic-

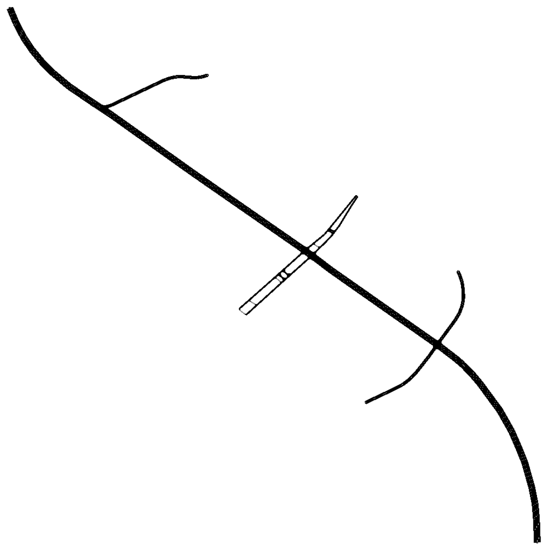

Fig. 2: The CurvyRoad network.

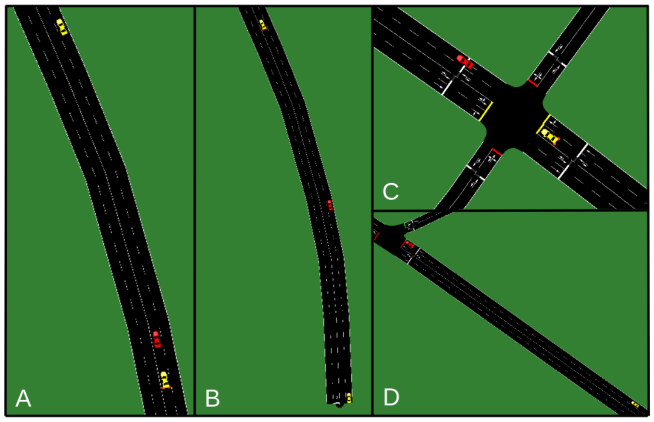

Fig. 3: Four scenes generated from the 2S13 Scenic program representing a traffic scenario.

network with its own speed limits and traffic lights which enables robust simulation testing of scene. Additionally, the mission of the cars is altered to be adapted to a traffic network.

All vehicles in the scenario are to travel along the 4-lane highway to the end and the scenario terminates when the `ego` completes it's mission instead of terminating when all vehicles have completed their atomic actions. The route-based missions allows for vehicle objects to be affected by the network. For example, in Figure 3C `ego` continues along the route while `npc1` stops at a yellow light, and in Figure 3D `npc1` approaches `ego` while `ego` obeys a red light.

To verify that unique scenes are being properly generated, some parameters are tracked in the formal scenario specification:

```
param ego_lane = ego.lane
param ego_speed = ego.speed
param npc1_lane = npc1.lane
param npc1_distance = npc1.distance
param npc1_speed = npc1.speed
param npc1_laneChange = npc1.laneChange[1]
param npc2_lane = npc2.lane
param npc2_distance = npc2.distance
param npc2_speed = npc2.speed
```

Then 1000 scenes are generated, simulated, and their parameter configurations recorded. Histograms for each of these parameters are created in order to visually verify that parameter ranges are sampled efficiently, which as Figure 4 depicts, shows the range sampling to be adequate for 1000 scenes. We check to see if the parameter ranges are sampled randomly but uniformly as expected given the law of large numbers. It is important to have fair sampling to maximize the generation of non-similar parameter configurations.

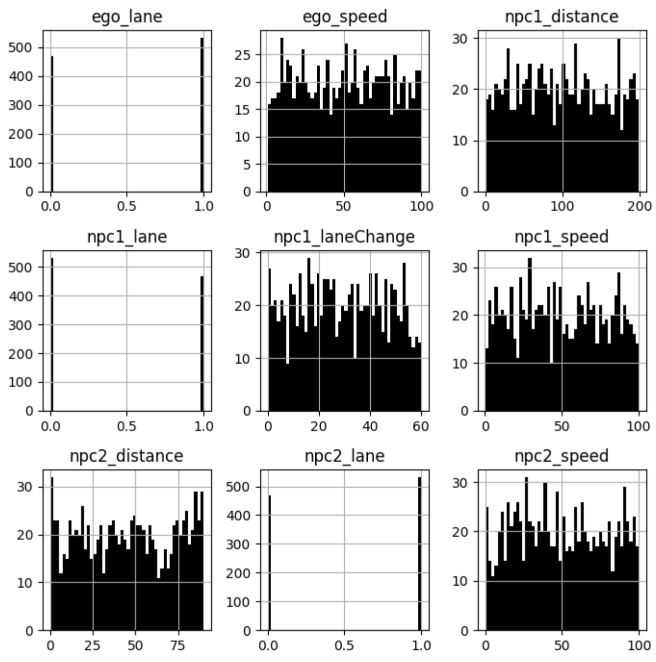

Fig. 4: Histograms of 1000 scene configurations.

B. *Traffic-Light Program Alteration*

It is important to be able to create scenarios with different components in validation. In this experiment, the 2S13 scenario is expanded with traffic light logic which has the potential to cause a recordable ego decision failure.

The logic of the first traffic light along the route is selected to be used. This traffic light is located at the junction shown in Figure 3C. Hence, a traffic light object is added to the scenario:

```
tl = TrafficLight at 0 @ 6,
    with name "1423875783",
    with state ["rrrrGGrrrrrrrr",
        "rrrryyrrrrrrrr","rrrrrrrrrrrrrr"],
    with duration [Range(1,30),
        Range(0,10),Range(1,30)]
```

The states of each traffic light phase are described in SUMO notation. Figure 5 displays a `rrrrGGrrrrrrrr` phase at the `1423875783` traffic lights. `rrrrGGrrrrrrrr` is the color of each traffic light unit, which is either red (`r`) or green (`G`). The connections at position 4 and 5 are given a green-yellow-red phase, while all other connections are always red. Three parameter ranges describe the duration of the green, yellow, and red phase durations.

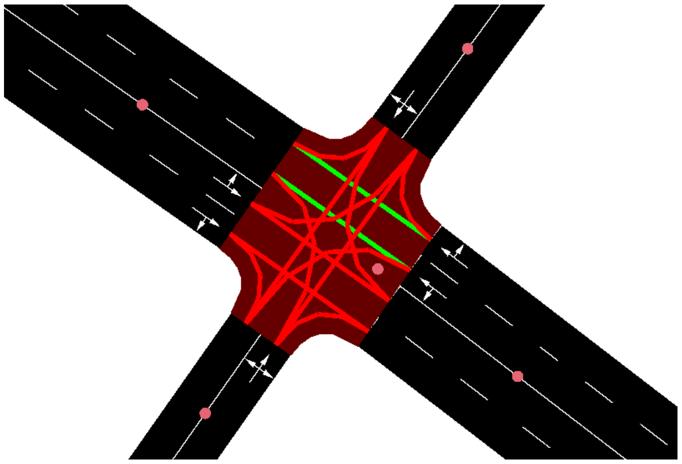

Fig. 5: A traffic light phase.

1) Results: Three additional Scenic parameters are added which track the green, yellow, and red traffic light phase durations:

```
param tl_g = tl.duration[0]
param tl_y = tl.duration[1]
param tl_r = tl.duration[2]
```

1000 scenes are generated and simulated. Histograms of the phase parameters are visualized in Figure 6 which verify that parameter ranges are sampled fairly during scene selection.

It is expected that some scene configurations in which during simulation tests, the ego fails to make a proper traffic light navigation decision to:

- Go through the light,
- or slow down in anticipation of a red light,
- or stop safely at a red light.

During these ego failure scenarios, the ego will emergency brake to obey a red light. Of the 1000 scenario configurations simulated, four of them are ego failure scenarios. These scenario numbers were 307, 574, 880, and 924. The scene

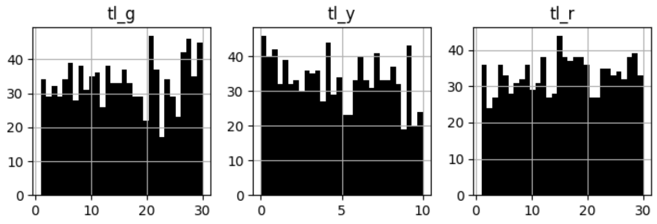

Fig. 6: Histograms of 1000 traffic light configurations.

configurations of the four ego failure scenarios are listed in Table IV.

TABLE IV: Configurations of ego failure scenarios.

Parameter	# 307	# 574	# 880	# 924
ego_lane	0.00	1.00	0.00	0.00
ego_speed (kph)	82.51	73.33	59.79	78.93
npc1_distance (m)	182.35	128.84	64.10	176.12
npc1_lane	1.00	0.00	1.00	1.00
npc1_laneChange (s)	27.73	10.29	3.10	41.33
npc1_speed (kph)	57.31	55.50	20.19	7.54
npc2_distance (m)	35.95	54.46	18.22	21.69
npc2_lane	0.00	1.00	0.00	0.00
npc2_speed (kph)	42.90	72.35	89.32	46.14
tl_g (s)	6.03	5.47	22.62	19.76
tl_r (s)	6.01	7.07	28.77	8.60
tl_y (s)	0.89	0.41	0.30	0.18

While four of 1000 data points is not sufficient for correlation analysis between the scene configurations and the ego failure outcome, we may extrapolate through operational design domain knowledge that sub 1-second yellow light phases is not a sufficient warning of an upcoming red light phase for fast moving vehicles using the default SUMO AI.

C. *Impact of Pedestrian-Friendly Additions*

A traffic network is constantly updating as the network ages or new requirements of a network model are added. One such example is adding pedestrians to a network in order to analyse impact of adding more pedestrian-friendly components such as a pedestrian crossing. Thus, the third iteration of the 2S13 scenario features pedestrians.

Some modifications to the CurvyRoad network in this example scenario are made to prepare the network for impactful and measurable pedestrian scenario simulation.

A pedestrian crossing is added to the network 114.62m after the controlled traffic light `tl`, which connects two sidewalks on opposite shoulders of a four lane highway. The crossing location is visualized and circled in Figure 7.

Three pedestrians also are added to the scenario before the pedestrian crossing and given a route such that they will travel through the crossing to the other side:

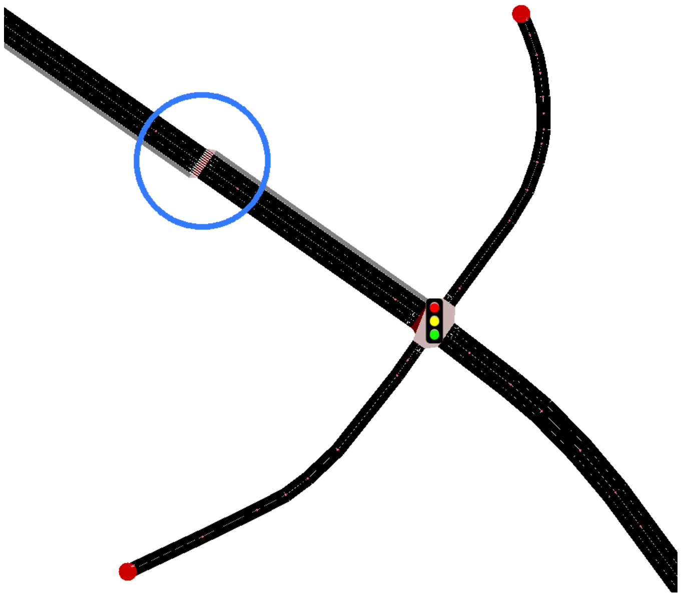

Fig. 7: A pedestrian crossing is added to the network 114.63m after the controlled traffic light.

```
ped_route = ["492300618#0", "-128973657#1"]
start_of_inter = 114.63

peds = []
ped_dist_from_inter = []
for i in range(3):
    dist_from_inter = Range(0,50)
    ped = Pedestrian at 0 @ (8+2*i),
        with name "ped_%d" % i,
        with route ped_route,
        with color [255,255,0,255],
        with departTime Range(0,40),
        with distance start_of_inter \
            - dist_from_inter,
        with egoWaitAtXing True
    peds.append(ped)
    ped_dist_from_inter.append(dist_from_inter)
```

Two parameter ranges are added to the scenario specification for each pedestrian added for a total of six new parameter ranges:

- The distance the pedestrian is placed from the crossing intersection in [0,50]m.
- The departure time of [0,40]s of simulation time. Note, the `ego` enters the simulation at time 0s.

The purpose of the pedestrian scenario is to locate scene configurations where the `ego` navigates the traffic network, then stops before the pedestrian crossing to yield to crossing pedestrians.

1) Results: The pedestrian scenario is represented as a formal scenario specification with eighteen (18) parameters in total, twelve (12) are inherited from the traffic-light scenario and six (6) are added to the pedestrian scenario.

The pedestrian departure times and distances are tracked as parameters:

```
param peds_departTime = \
    [ped.departTime for ped in peds]
param peds_distance = \
    [dist for dist in ped_dist_from_inter]
```

1000 scenes are generated and simulated. Histograms of the pedestrian parameters are visualized in Figure 8 which verify that the parameter ranges are sampled fairly during scene simulation.

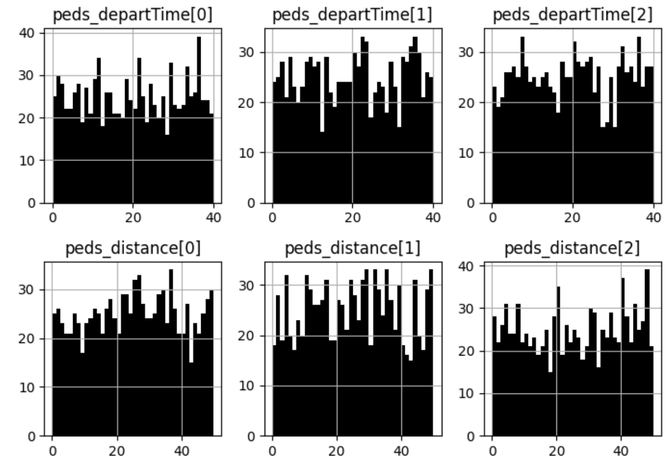

Fig. 8: Histograms of 1000 pedestrian configurations.

Some screenshots of simulated scenes are shown in Figure 9 of scenario outcomes. Each screenshot is taken at the crossing location, and all pedestrians in the image are circled. In Figure 9A, ego arrived at the crossing after all pedestrians have crossed already and did not have to yield. In Figure 9B, ego is arriving at the crossing while the second pedestrian recently finished crossing and the third pedestrian has not reached the crossing. ego did not yield in this simulation. In Figure 9C, ego is yielding to the last pedestrian who entered the crossing moments before the arrival of ego. In Figure 9D, ego is yielding while all three pedestrians are navigating the crossing.

Simulation statistics are collected from the scenario test reports. The first of these results is a summary of pedestrian–ego interaction at pedestrian crossings which is shown as two histograms in Figure 10. The number of pedestrians at the moment when ego is at the pedestrian crossing junction and the simulation time (in seconds) of the pedestrian–ego interaction is recorded. Out of 1000 simulation, 113 simulation observed 1 pedestrian at the crossing, 52 simulations with 2 pedestrians, and 11 simulations with 3 pedestrians.

Pedestrian–ego interactions at the crossing occur within [29,91]s of simulation time, which is the time it takes ego to traverse traffic (npc1 and npc2), and the controlled traffic light tl to reach the pedestrian crossing junction.

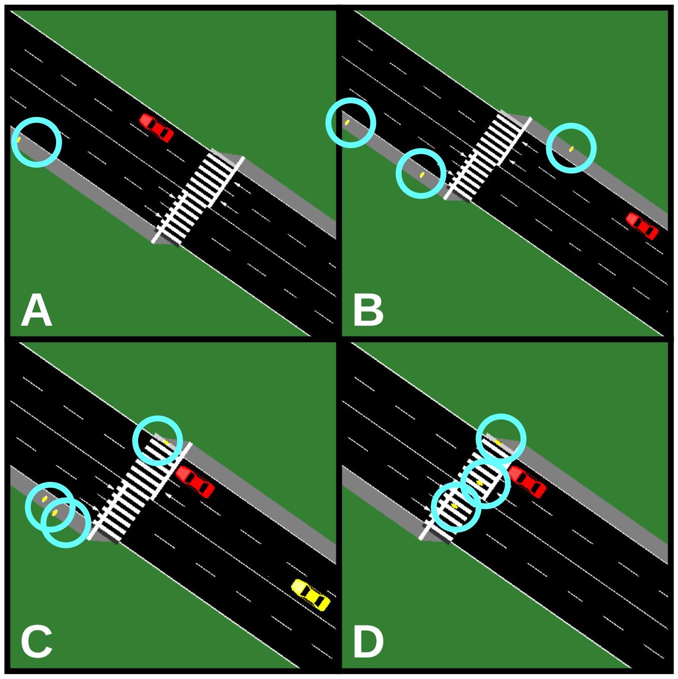

Fig. 9: Screenshots of simulation scenes when the ego vehicle is at the pedestrian crossing.

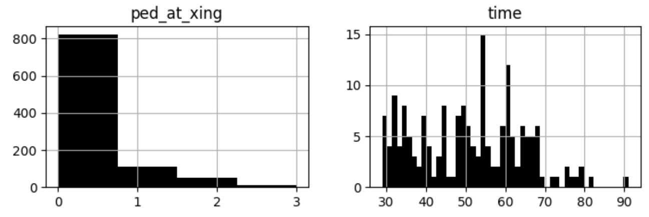

Fig. 10: Summary of pedestrian-ego interaction at pedestrian crossings data.

The concrete values of the 11 scene configurations which produced simulations which observe a pedestrian–ego event with three pedestrians present in the crossing are presented in Table V.

V. Conclusion

In conclusion, we introduced a methodology of integration of formal scenario specification and network-based traffic simulation following a framework which integrates the open-source software Scenic scenario description language and the well-known open-source SUMO traffic simulator for scenario-based AV validation testing. Then we followed up with three example implementation scenarios of increasing

TABLE V: Configurations of crosswalk interaction scenarios with three pedestrians crossing.

Parameter	# 2	# 85	# 111	# 199	# 375	# 422	# 619	# 681	# 741	# 923	# 930
ego_lane	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00	0.00	1.00
ego_speed (kph)	59.45	48.83	23.08	47.44	72.39	91.02	44.11	34.53	42.01	59.97	64.51
npc1_distance (m)	160.66	141.55	159.36	68.71	143.80	110.08	161.09	80.67	190.03	75.52	140.33
npc1_lane	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	1.00	0.00
npc1_laneChange (s)	54.49	30.37	8.47	0.40	28.09	5.15	57.46	6.04	59.13	22.33	6.44
npc1_speed (kph)	61.69	44.79	11.68	3.84	25.25	83.98	64.46	49.63	48.24	17.62	36.72
npc2_distance (m)	38.15	9.03	81.53	20.36	6.10	48.63	22.08	35.64	61.57	71.80	30.64
npc2_lane	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00	0.00	1.00
npc2_speed (kph)	46.81	80.05	77.38	52.49	59.99	28.49	43.68	14.61	27.68	36.45	46.45
tl_g (s)	9.29	16.96	14.66	10.88	3.10	7.55	2.29	26.61	1.66	19.97	18.11
tl_r (s)	2.49	15.41	26.85	21.62	6.36	21.58	13.83	2.01	10.59	19.20	9.44
tl_y (s)	7.48	4.18	4.21	8.87	5.25	9.26	6.32	8.79	3.42	9.72	4.17
peds_departTime[0] (s)	7.34	33.97	24.69	25.20	25.32	9.33	36.58	38.16	21.23	26.14	4.33
peds_departTime[1] (s)	14.08	7.34	38.23	21.75	8.94	12.06	26.34	34.61	1.27	25.10	25.09
peds_departTime[2] (s)	31.62	33.03	30.51	32.69	11.58	27.94	38.60	1.12	4.87	28.22	9.51
peds_distance[0] (m)	19.48	9.72	39.93	23.36	14.16	39.16	9.40	7.66	26.55	21.75	42.21
peds_distance[1] (m)	14.50	32.92	35.48	34.10	30.64	34.55	28.72	21.64	44.70	33.37	14.77
peds_distance[2] (m)	0.53	3.85	40.66	13.44	33.42	31.16	8.74	44.85	36.21	34.20	31.53

complexity as common traffic-network components added to the FSS. The results of the scenario examples included locating scene configurations of ego failure scenarios and analyzing the impact of adding pedestrians to a scenario. For future work, we want to optimize the scene configuration selection component to efficiently generate concrete scenarios of importance.

Finally, to reiterate on the advantages of integrating these tools, Scenic is open-source so we can share the scenario and anyone can focus on whatever part they want to focus on: whether it is scenario description, traffic modeling, or decision making. Additionally, SUMO is also open-source, and the traffic network is a modular component of the integration. This means that traffic network component of a scenario may be edited separately: whether it be adding traffic flows, pedestrian accessibility, or altering the road network by adding, removing, or closing roads.

REFERENCES

[1] D. J. Fremont, E. Kim, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: a language for scenario specification and data generation," *Mach. Learn.*, pp. 1–45, Feb. 2022.

[2] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, pp. 128–138, December 2012.

[3] "ASAM Open Scenario V 1.2.0," may 2022. ASAM Standard.

[4] "ASAM Open Scenario V 2.0.0," July 2022. ASAM Standard.

[5] "How do you get more out of your logical scenarios? And how do abstract scenarios take productivity and safety to a new level? - Foretellix," dec 2022. [Online; accessed 8. Dec. 2022].

[6] Foretellix Inc., "Measurable Scenario Description Language (M-SDL)," May 2021. M-SDL 21.05.0.1 Release.

[7] Foretellix, "Technology - Foretellix." https://www.foretellix.com/technology/, 2020.

[8] T. Shah, J. R. Lepird, A. T. Hartnett, and J. M. Dolan, "A Simulation-Based Benchmark for Behavioral Anomaly Detection in Autonomous Vehicles," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 2074–2081, IEEE, Sept. 2021.

[9] 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*, pp. 432–442, Springer, 2019.

[10] D. J. Fremont, J. Chiu, D. D. Margineantu, D. Osipychev, and S. A. Seshia, "Formal Analysis and Redesign of a Neural Network-Based Aircraft Taxiing System with VerifAI," in *Computer Aided Verification*, pp. 122–134, Cham, Switzerland: Springer, July 2020.

[11] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta, E. Agafonov, T. H. Kim, E. Sterner, K. Ushiroda, M. Reyes, D. Zelenkovsky, and S. Kim, "LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving," *arXiv*, May 2020.

[12] R. Razdan, W. Mahoney, E. Haritan, A. Kalia, J. Smith, T. Zarola, M. İ. Akbaş, J. Taiber, E. Straub, and R. Sell., "Unsettled Topics Concerning Automated Driving Systems and the Development Ecosystem," *Society of Automotive Engineers (SAE) EDGE^{TM} Research Report EPR2020004*, March 2020.

[13] A. Reyes-Muñoz and J. Guerrero-Ibáñez, "Vulnerable Road Users and Connected Autonomous Vehicles Interaction: A Survey," *Sensors*, vol. 22, p. 4614, June 2022.

[14] X. Jing, X. Pei, S. Yan, C. Han, Selpi, E. Andreotti, and J. Ding, "Safety benefit of cooperative control for heterogeneous traffic on-ramp merging," *transp saf environ*, vol. 4, p. tdac031, Dec. 2022.

[15] S. Chamideh, W. Tärneberg, and M. Kihl, "A Safe and Robust Autonomous Intersection Management System Using a Hierarchical Control Strategy and V2I Communication," *IEEE Syst. J.*, pp. 1–12, Nov. 2022.

[16] A. Corso, R. J. Moss, M. Koren, R. Lee, and M. J. Kochenderfer, "A survey of algorithms for black-box safety validation," *arXiv preprint arXiv:2005.02979*, 2020.

[17] G. E. Mullins, P. G. Stankiewicz, R. C. Hawthorne, and S. K. Gupta, "Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles," *Journal of Systems and Software*, vol. 137, pp. 197–215, Mar 2018.

[18] J. Sun, H. Zhou, H. Xi, H. Zhang, and Y. Tian, "Adaptive design of experiments for safety evaluation of automated vehicles," *IEEE*

Transactions on Intelligent Transportation Systems, vol. 23, no. 9, pp. 14497–14508, 2022.

[19] H. Tian, Y. Jiang, G. Wu, J. Yan, J. Wei, W. Chen, S. Li, and D. Ye, "MOSAT: finding safety violations of autonomous driving systems using multi-objective genetic algorithm," in *ESEC/FSE 2022: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 94–106, New York, NY, USA: Association for Computing Machinery, Nov. 2022.

[20] G. Chance, A. Ghobrial, S. Lemaignan, T. Pipe, and K. Eder, "An Agency-Directed Approach to Test Generation for Simulation-based Autonomous Vehicle Verification," *arXiv*, Dec. 2019.

[21] A. Wegener, H. Hellbruck, C. Wewetzer, and A. Lubke, "VANET Simulation Environment with Feedback Loop and its Application to Traffic Light Assistance," in *2008 IEEE Globecom Workshops*, pp. 1–7, IEEE, Nov 2008.

[22] Q. Goss and M. İ. Akbaş, "Scenic–Sumo." https://github.com/AkbasLab/scenic-sumo-most, Dec 2022. Github Repository, commit: 5f5356842211180372111111b5b35deeeb25ccdfe.

[23] Q. Goss, Y. AlRashidi, and M. İ. Akbaş, "Generation of modular and measurable validation scenarios for autonomous vehicles using accident data," in *IEEE Intelligent Vehicles Symposium (IV)*, July 2021.

[24] Q. Goss, Y. Alrashidi, and M. İ. Akbaş, "Autonomous Vehicle Validation and Verification." https://github.com/AkbasLab/AV-Atomic-Blocks-IV-2021, May 2021. Github Repository, commit: 727f3af20670577fe27e194200beff412841f687.