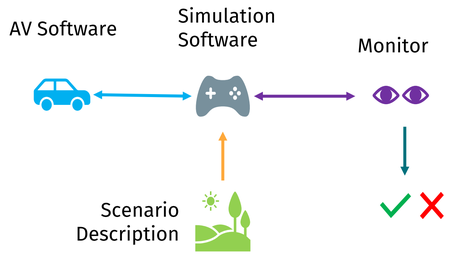
A Language for Autonomous Vehicles Testing Oracles

Ana Nora Evans, Mary Lou Soffa, Sebastian Elbaum
University of Virginia

End-to-end AV Testing



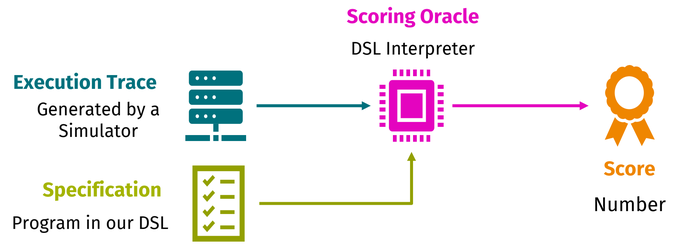
AV Specifications/Oracles

Sources:
1. Traffic rules and regulations
2. Local driving customs
3. Human driver behavior
4. Safe/defensive driving rules
5. Formal safety models

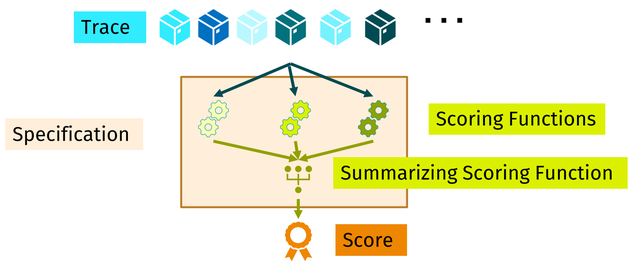
Conflicting Goals: Safety and Efficiency

Challenges:
1. Large number of smaller specifications
2. Either imprecise or too formal for most AV developers
3. Mostly pass/fail, can not be used in search and optimizations
4. A numerical score assigned to a an execution is useful for raking AV solutions.

Goal: A New Framework



DSL for AV Specifications



A **specification** is a set of scoring functions together with a summarizing scoring function.

A **trace** is a sequence of world snapshots, we call **trace elements**. Each trace element contains: a timestamp, traffic signs and traffic light information, state of the AV and the surrounding traffic.

A **scoring function** calculates a numerical score for a trace.

A **summarizing scoring function** calculates the score of a trace from the individual scores assigned by each scoring function to the trace.

Scoring Functions

Event: predicate on trace element.

Action: score update.

Notification: to another scoring function.

Sequence: consecutive trace elements for which the event is true.

Condition: predicate on a sequence.

Red/Green boxes are trace elements for which the event is false/true.
Red/Green rectangles are sequences with false/true condition.

Safety Specification

Speeding: Deduct one every time the speed limit is exceeded.

Event: test if the AV speed of the current trace element exceeds the speed limit.

Action and frequency: The score is updated every time the function triggers.

```
speeding = scoring_function(
       event = speed > MAX_SPEED,
       action = -1,
       frequency = action_sum)
```

Condition: not set. The function triggers every time the event is true.

Only the AV's speed is relevant for the `speeding` scoring function.

Speed	21	21.5	22	22.5	22.5	22	22.5	22	21
Score	0	0	0	-1	-2	-2	-3	-3	-3

Liveness Specification

Arrival Check: If the AV reaches a fixed point on the map, the score is one, otherwise it is zero.

```
arrival_test = scoring_function(
       event = (x-x_dest)² + (y-y_dest)² < 1,
       action = 1.0,
       frequency = first)
```

Frequency: If the event is true for a trace element, then the score is set to one and subsequent trace elements are not checked anymore.

Timeliness Specification

Lane Keeping: Every time the AV drives on the line for more than three seconds subtract one.

```
lane_keep = scoring_function(
    event = (road_normal > LW-TH and
       road_normal < LW+TH)
       or (road_normal > 2*LW-TH
       and road_normal < 2*LW+TH),
    condition=seq_time > 3,
    action=-1, frequency=action_sum)
```

Event: checks if the AV is on the line.
Condition: checks if the time the AV is on the line is longer than three seconds.
Action: For every sequence of driving on the line for more than three seconds, deduct one.

Temporal Specification

Deceleration before collision: If the AV decelerates within half of second for at least two seconds before a collision, then the score is one otherwise is zero.

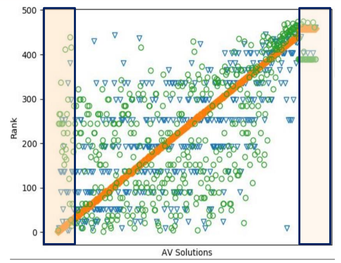
```
collisions = scoring_function(
       event = collision and expiration > 0,
       action = 1.0, frequency = all_sum)
deceleration = scoring_function(
       event = acceleration < 0 and not collision,
       condition = seq_time > 2, frequency = first,
       notifications =
           [(collisions, [(expiration, 0.5)])])
```

The deceleration scoring function checks if the AV decelerates for at least two seconds. When the deceleration triggers, the expiration variable of the collisions scoring functions is set to half of second.
Every time a trace element is processed, the expiration is decreased. If a collision is detected before the expiration becomes negative then the collisions function triggers.

Results and Conclusions

We encoded three open source specifications in the DSL. We used the simulator and 474 student solutions to the Path Planning Project of Udacity's Self Driving Car program.

The rankings are correlated.
The specifications agree on the worst, but not the best.



Trustworthy AVs Require Testing the Specification!

We propose a language for specifications and an oracle independent of the AV frameworks and simulators.

Future work:
1. Define test coverage criteria for specifications.
2. Develop testing techniques for specifications.
3. Static analyses to find similarities between specifications.