# Formal Verification of Autonomous Lane-Changing

Alyssa Byrnes

## I. Objective

The use of formal methods has been used to produce provably correct software for many systems [1]. The correctness of a system is determined by formalizing expected behavior of the system.

Formal methods are an efficient way to minimize testing needed. For example, if we sample 10 different assignments for every state variable of a 7-dimensional car, that would mean running $10^7$ simulations. If we want to test an interaction between two cars, that would be $10^{14}$ simulations.

Using model checking, we can express each state variable as a *range* of values, and say something about the safety of the vehicle given this range of inputs.

An example question we could answer is, "Given a starting velocity between $10.8-11.1$ mph, an x position between $0-.1$ and a y position between $0-.1$, will a car be able to safely pass another car of a different velocity and position range?"

There is a tool that exists that can represent such a scenario and test it for safety, called Apex [2]. However, in the paper the authors only test it on a two-step process given a manual input. I would like to test it on an input of an actual lane-changing algorithm and see if it's actually practical to use.

## II. Related Work

The tool I plan to use is called APEX [2]. It uses a 7D bicycle model [3]. It is written in Python and C++, and verifies safety properties of autonomous vehicles. Mainly, it focuses on path trajectories and collision avoidance. The code is available to the public on Github.

APEX has many parts, but the main flow is that it takes a range of state variable inputs, basic environmental data such as lane width, and a goal state. It uses a trajectory planner to predict the trajectory associated going from the current state to the goal state. It writes the state variable inputs, environmental data, $\delta$, and trajectory to a ".drh" file along with a "safety metric" expressed in Metric Interval Temporal Logic to determine if there will be a collision. This file is input into dReach [4], and it returns whether or not there exist any scenarios where the car will get within $\delta$ of a collision.

The lane changing data I intend to test is from a paper by Dong et al. [5].

## III. Problem Definition

I wanted to know if I could input a specific starting state, road state, car information, and goal state into APEX to test existing work. The goal state would be of the form $(s_x, s_y, \theta, v, \kappa)$, where $(s_x, s_y)$ is the goal position of the vehicle, $\theta$ is heading angle, $v$ is the velocity of the vehicle, and $\kappa$ is the curvature. This would be input into APEX, along with a distance $\delta$, and APEX will output whether or not there is a scenario where the car comes within $\delta$ of an unsafe scenario.

## IV. Methods

A lot of my time was spent fully understanding how APEX works and to get it up and running. This has been a challenge because this work has not been publicly maintained and isn't designed to be user-friendly. However, I communicated with the author about issues.

My biggest challenge was with the SMT-solving tool they use to test their model, called dReach [4]. It is constantly updated, and the version used in APEX isn't supported anymore. This involved making some changes to input syntax.

To make it so I can input specific variables, I had to make changes directly to their ".drh" basefile being input into dReach.

I wrote a python script that prompts the user for the goal state information, which is input into APEX's trajectory planner. Then, once the trajectory is generated, the script prompts the user for the following information:

- Starting x position
- Starting y position
- Starting velocity
- Road Width
- Car Weight

### A. *dReach*

dReach is a tool that is a smaller part of the larger tool, dReal. dReach works as an SMT solver given differential inputs of a physical scenario.

There are 6 main parts to a ".drh" file:

1) Constants: here are the definitions of the constants in the scenario, such as the car weight and lane width.
2) Basic Functions: here are definitions of functions that describe basic physical concepts, such as acceleration.
3) Non-constant variables: Next, the variables that exist within a *range* of values, such as starting position, are specified.
4) Scenario Description: In this section, the physics of a scenario are described, broken into events.
5) Initialization: These are the initial values for the variables described in number 3.
6) Goal State: This is the logical description of the goal state. For our purposes, the goal state is $((s_y < w) \wedge (s_x - \epsilon > s_{xenv}))$ [2]. The first part of the conjunction states that the car doesn't go off road, and the second part states that the car doesn't collide with the car in front of it.

The two main sections of the ".drh" file that I edited are 1 and 5.

## V. Results

I tested 20 different scenarios from the data provided by Dong et al. [5] with $\delta = .1$, and APEX returned "UNSAT" for all of them, which means that an unsafe state was not reached.

From this we can conclude that, if APEX presents an accurate model, then Dong et al.'s findings were correct.

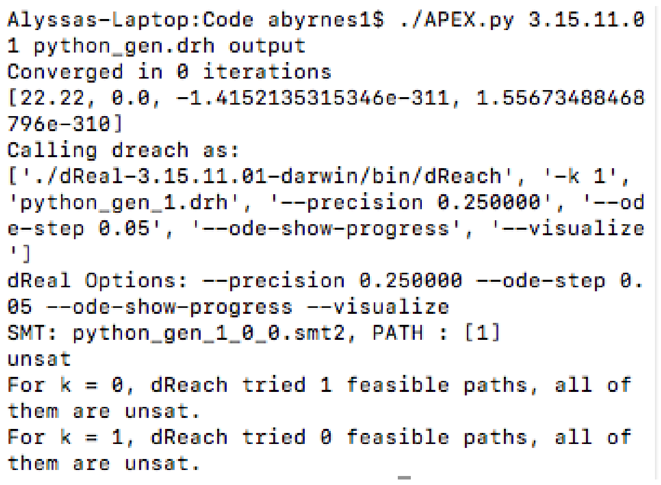
Figure 1 demonstrates an output of APEX before I modified it. This is what it would output when it didn't find any incidents of an unsafe outcome.

Figure 2 demonstrates what the new interface and output for APEX looks like. The data input into APEX is one of the scenarios provided by Dong et al. [5]. Figure 3 shows a pictorial representation of that scenario.

## VI. Conclusion and Future Work

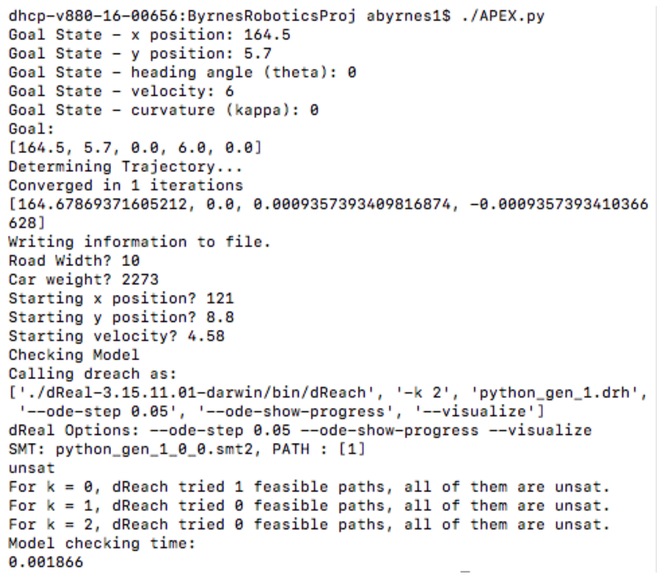
In conclusion, APEX has shown value in verifying lane-changing scenarios, and it's a shame that it doesn't seem to have been used or modified for 2 years.

An immediate issue that should be fixed is that the user can't easily modify the ranges of the starting



```
Alyssas-Laptop:Code abyrnes1$ ./APEX.py 3.15.11.0
1 python_gen.drh output
Converged in 0 iterations
[22.22, 0.0, -1.4152135315346e-311, 1.55673488468
796e-310]
Calling dreach as:
['./dReal-3.15.11.01-darwin/bin/dReach', '-k 1',
'python_gen_1.drh', '--precision 0.250000', '--od
e-step 0.05', '--ode-show-progress', '--visualize
']
dReal Options: --precision 0.250000 --ode-step 0.
05 --ode-show-progress --visualize
SMT: python_gen_1_0_0.smt2, PATH : [1]
unsat
For k = 0, dReach tried 1 feasible paths, all of
them are unsat.
For k = 1, dReach tried 0 feasible paths, all of
them are unsat.
```

Fig. 1. Example of old output from APEX



```
dhcp-v880-16-00656:ByrnesRoboticsProj abyrnes1$ ./APEX.py
Goal State - x position: 164.5
Goal State - y position: 5.7
Goal State - heading angle (theta): 0
Goal State - velocity: 6
Goal State - curvature (kappa): 0
Goal:
[164.5, 5.7, 0.0, 6.0, 0.0]
Determining Trajectory...
Converged in 1 iterations
[164.67869371605212, 0.0, 0.0009357393409816874, -0.0009357393410366
628]
Writing information to file.
Road Width? 10
Car weight? 2273
Starting x position? 121
Starting y position? 8.8
Starting velocity? 4.58
Checking Model
Calling dreach as:
['./dReal-3.15.11.01-darwin/bin/dReach', '-k 2', 'python_gen_1.drh',
 '--ode-step 0.05', '--ode-show-progress', '--visualize']
dReal Options: --ode-step 0.05 --ode-show-progress --visualize
SMT: python_gen_1_0_0.smt2, PATH : [1]
unsat
For k = 0, dReach tried 1 feasible paths, all of them are unsat.
For k = 1, dReach tried 0 feasible paths, all of them are unsat.
For k = 2, dReach tried 0 feasible paths, all of them are unsat.
Model checking time:
0.001866
```

Fig. 2. Example of new output from APEX

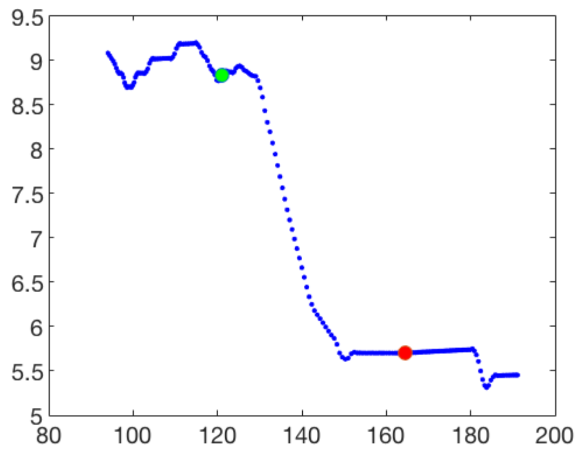


Fig. 3. Visual representation of scenario

variables, so the variables are only being tested in the ranges set in the .drh file.

One future goal is to add the option for a user to test the scenario for a car on the road with a specified starting velocity and position. Then, I would like to extend that to $n$ other vehicles on the road.

Another future goal would be to use APEX to help train a planner offline. Unfortunately, APEX isn't fast enough to be used online for planning.

## VII. Acknowledgements

I would like to thank Houssam Abbas, one of the creators of APEX, for talking to me about his work and answering any questions I had.

## References

[1] C. Sturton, "Designing a voting machine for testing and verification," Master's thesis, EECS Department, University of California, Berkeley, Dec 2012. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-253.html

[2] M. O'Kelly, H. Abbas, S. Gao, S. Shiraishi, S. Kato, and R. Mangharam, "Apex: Autonomous vehicle plan verification and execution," 2016.

[3] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[4] S. Gao, S. Kong, and E. M. Clarke, "dreal: An smt solver for nonlinear theories over the reals," in *International Conference on Automated Deduction*. Springer, 2013, pp. 208–214.

[5] C. Dong, Y. Zhang, and J. M. Dolan, "Lane-change social behavior generator for autonomous driving car by non-parametric regression in reproducing kernel hilbert space," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 4489–4494.

## VIII. Code

The code is located at `https://github.com/AlyssaByrnes/ByrnesRoboticsProj`.