A DRIVER'S LICENSE TEST FOR DRIVERLESS VEHICLES

BY HOUSSAM ABBAS
MATTHEW E. O'KELLY
ALENA RODIONOVA
RAHUL MANGHARAM

DEPARTMENT OF
ELECTRICAL AND
SYSTEMS ENGINEERING
UNIVERSITY OF
PENNSYLVANIA

Autonomous vehicles (AVs) have already driven millions of miles on public roads, but even the simplest maneuvers such as a lane change or vehicle overtake have not been certified for safety. Current methodologies for testing of Advanced Driver Assistance Systems, such as Adaptive Cruise Control, cannot be directly applied to determine AV safety as the AV actively makes *decisions* using its perception, planning and control systems for both longitudinal and lateral motion. These systems increasingly use machine learning components whose safety is hard to guarantee across a range of driving scenarios and environmental conditions.

New approaches are needed to bound and minimize the risk of AVs to reassure the public, determine insurance pricing and ensure the long-term growth of the domain. So what type of evidence should we require before giving a driver's license to an autonomous vehicle? To answer this question, consider the major components which make up an AV. An AV is typically equipped with multiple sensors, such as a LIDAR (a laser range finder) and several cameras (**Figure 1**(1)). The readings of these sensors are processed by algorithms that extract a model of the current scene, like object detectors, in order to understand *who's doing what and where*. This information is then fused together to provide the AV with its state estimate, such as position and velocity, and that of the other agents in the scene. The AV must then decide where to go next (a discrete decision taken by the behavioral planner), what continuous trajectory to follow to get there (a computation performed by the trajectory planner) and how to actuate steering and acceleration to follow that trajectory (performed by the trajectory tracker). Add to this the interaction with other vehicles, changing weather conditions and the respect of traffic laws, and it is clear that verifying correctness of AV behavior is a gargantuan task.

WHOLE-AV TESTING

Such considerations have led AV researchers to *formal methods* to provide a high level of assurance. This term encompasses a wide field of theory, techniques and tools for answering the following question: Given a *mathematical model* of a System Under Test (SUT), and a *formal specification* of correct system behavior, does the SUT model satisfy the specification? A formal tool's answer is *complete and sound* [1]. If the SUT model is incorrect, the tool *will* find an example violation, also called a *counterexample*. And if the tool returns that "The model is correct", then the model is indeed correct and does not violate the specification. Unlike testing, there is no question of 'Could we have found a bug if we had tested more?'

Formal methods applied to the problem of AV verification

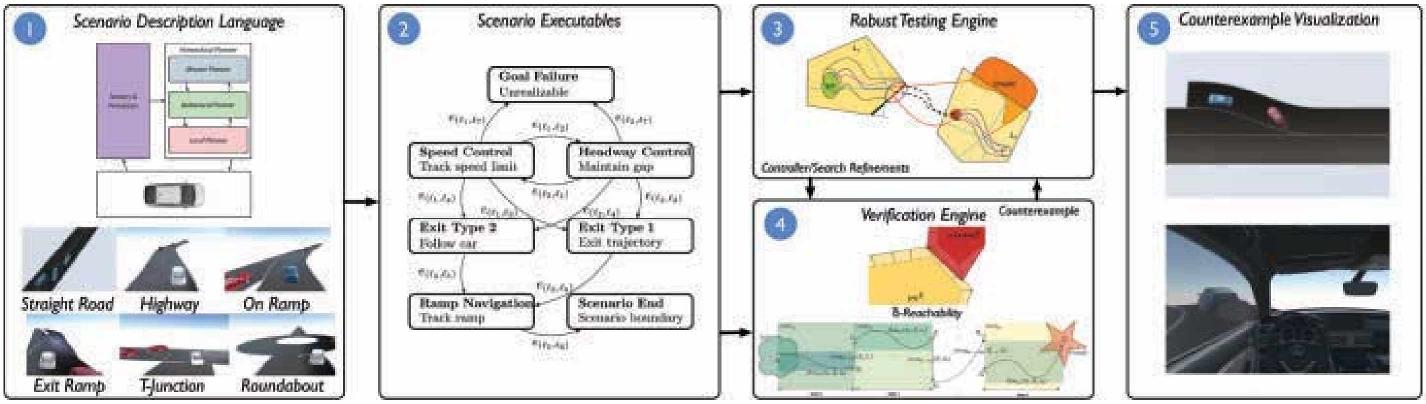


FIGURE 1 The AVCAD toolchain: (1) A Scenario Description Language allows quick creation of driving scenarios (2) The scenarios are translated into formats that can be processed by the testing and verification tools (3) Robust Testing [G. Fainekos] (4) Formal Verification Engine [S. Kong] (5) Requirement violations are visualized for an intuitive understanding of the violation.

[1] Though some provide approximate answers for more complicated models.

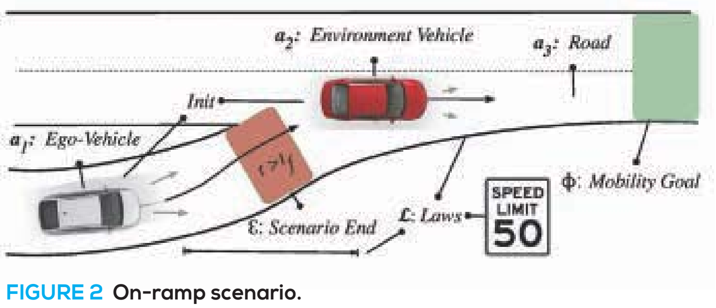


FIGURE 2 On-ramp scenario.

include theorem proving [1, 2], reachability analysis [3], synthesis [4, 5, 6], and maneuver design [7]. Theorem proving is an interactive technique in which the computer is largely responsible for demonstrating that the model satisfies the specification, with occasional help from the user. The latter provides lemmas and axioms that the tool leverages to advance the proof towards its conclusion. While this interactivity allows us to tackle more complex models, it also limits the scalability of the approach.Additionally, existing work such as [1] utilizes unrealistically conservative lemmas (such as, vehicle spacing of at least 291 feet) [8].

Reachability analysis is a popular formal technique where the reachable state space of the model is over-approximated and tested for intersection with a set of unsafe states. Reachability analysis is used in [9] to verify the operation of the AV *during navigation*. This provides an extension of on-board diagnostics to whole-AV operation, where the diagnosis does not concern one component's requirements, but the safety of the entire AV.

Another approach is to design *correct-by-construction* controllers from pre-verified maneuvers. The basic idea is that one builds a library of maneuvers, like Left-Turn and Right-Turn, and verifies (by Lyapunov analysis, say) that the car can perform these maneuvers from any initial state. Online, we restrict the AV's motion to be a composition of such maneuvers. This technique is closely related to motion planning algorithms and is limited to specific types of correctness criteria, like dynamical feasibility. Along these lines, a vigorous area of research concerns *controller synthesis from formal specifications* [4, 5], where formal verification techniques are adapted to *create* controllers that are correct relative to specifications in some temporal logic. Most of this work requires a discretization of the AV model's state-space and faces computational complexity barriers. Nonetheless, it forms the basis of a promising approach that divides the verification challenge into a design phase where correct-by-construction controllers are synthesized, and a post-design phase where these are used in a whole-AV verification effort.

COMBINING TESTING AND FORMAL VERIFICATION FOR WHOLE-CAR TESTING IN IDEALIZED ENVIRONMENTS

The guidance issued by NHTSA on the elements of a safety assurance case for AVs [10] is a starting point for standardizing the *type* of safety and correctness evidence needed for deployment of AVs. However, it does not prescribe *how* such evidence should be obtained. AV correctness, including safety, is a continuous spectrum: we should be able to rank vehicles by their relative safety, and compare one AV's performance across different scenarios. This is routinely done for human-driven cars, which receive safety ratings based, for example, on their crash performance and the technology they carry, like collision sensors. For AVs, such a continuous measure of correctness can and should be obtained *at design time, and measured throughout the design cycle*.

To illustrate the sorts of requirements that need to be formalized and measured, consider a Highway On-ramp scenario (**Figure 2**). The ego-vehicle, which is the AV under test, must merge while satisfying the following requirements:

1. At all times, stay at least 2 m from any fixed obstacle.

2. If the ego-vehicle is already in the merge point and an approaching vehicle on the highway is closer than 6 m, reach a speed of 45 mph within 6 sec.

3. Either reach the green rectangle within 20 secs or stay at the starting position until the road is clear.

These requirements increase in complexity: the first is a static no-collision requirement, the second adds a *reactive* element, and the third adds a pure *temporal* element. What are meaningful continuous measures of correctness for these requirements? For Req. 1, a meaningful measure ρ would be the minimum distance between the vehicle and any fixed obstacle over the course of the simulation. Req. 2 is more complicated since it involves different possibilities. It is reasonable to say that the correctness measure ρ in this case equals either the minimum distance between the two cars if it is above 7 m (so the minimum speed requirement is irrelevant), otherwise it equals the difference between the maximum car speed and 45 mph over the 6 sec window. What about the third requirement? Things are even more complicated because of the temporal 'until' component: should the correctness measure reward entering the intersection earlier? Should it differentiate between two different behaviors after the road clears? And what if *all* three requirements are part of the vehicle specification? How do we balance between all of them?

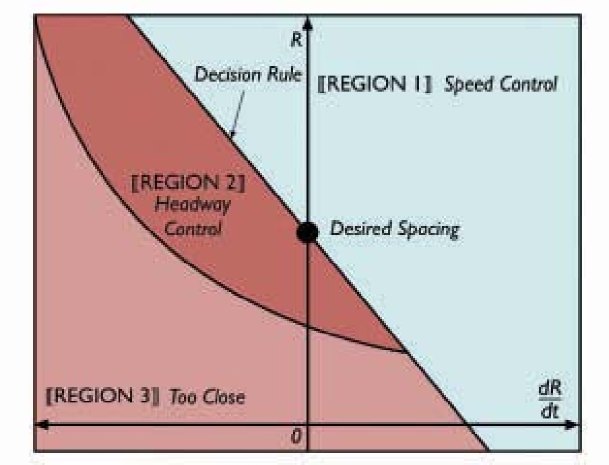


FIGURE 3 Hybrid Adaptive Cruise Controller. In Region 1 Speed Control, the AV tries to maintain a desired speed. It switches to Headway Control if a minimum time to collision constraint is violated. In Headway Control, the AV tries to maintain a given separation from the leading vehicle. R is the spacing to lead vehicle. In Region 3, the vehicle brakes to avoid collision.

It becomes clear that we need a *systematic way* of calculating this correctness measure for *arbitrary* specifications involving reactive, spatio-temporal requirements. Such a measure of correctness is provided by the *robustness function* of Metric Temporal Logic (MTL) requirements [11]. Specifically, it is possible to express the AV requirements as a *formula* φ in MTL, which is a formal mathematical language for writing temporal specifications. Using a formal logic, like MTL, removes ambiguity from the requirements, and enables the use of *automatic* correctness checking tools that go a long way toward flushing out difficult bugs that could not be found by manually-created test cases.

Given a (reachability) MTL formula φ, the highest level of assurance is provided by *reachability analysis*, described earlier. To run such a powerful tool requires the development of an appropriate mathematical model of the *whole* AV, which is very challenging. Moreover, reachability tools can have very long runtimes.

To counter the second issue, the *robustness* ρ_φ of φ can be leveraged [12]. The robustness $\rho_\varphi(x)$ of system execution x is a real number that measures two things about x: its sign tells whether x satisfies the spec ($\rho_\varphi(x) > 0$) or violates it ($\rho_\varphi(x) < 0$). Moreover, the trajectory x can be disturbed by an amount $|\rho_\varphi(x)|$ without changing its truth value (e.g., if it is correct, the disturbed trajectory is also correct). Thus, robustness is a continuous measure of correctness of the AV relative to the desired properties: if $\rho_\varphi(x_1) > \rho_\varphi(x_2) > 0$, this means x_1 is more robustly correct than x_2 since it can sustain a greater disturbance without violating the correctness specification.

The idea behind *robustness-guided verification* [13] is that we can first search the set of behaviors to find those executions with low robustness. Assuming continuity of behavior, low-robustness executions are surrounded by other low-robustness executions, and possibly by executions with negative robustness (**Figure 4**). The latter, then, are violations of φ. The reachability tool is run on a neighborhood of these low-robustness executions: rather than waste time on robustly-correct behavior, we focus on behavior that may reveal bugs. Formal verification and robustness, and the tools that implement them, are illustrated in the following example from the AV testing tool AVCAD [14].

Scenario 1 *(On-Ramp,* ***Figure 2****) There are two cars, the AV* a_1 *and an environment vehicle* a_2*. The AV is getting on the highway via an on-ramp, which is a cubic spline. The shape of the on-ramp matters because the tracking performance of the AV is altered by sharp curvatures. The AV uses a hybrid Adaptive Cruise Controller (ACC) shown in* ***Figure 3****. This ACC design has been utilized extensively on real vehicles, but is designed for operating conditions involving highway driving tasks with straight roads.*

AVCAD, ***Figure 1****, supports two tools:* S-TaLiRo *[12] and dReach [15].* S-TaLiRo *is a specification-guided automatic test generator for cyber-physical systems. By minimizing* ρ_φ *over the space of AV behaviors* x*,* S-TaLiRo *can find many different ways in which the AV violates the specification, thus promoting good coverage of the test space. dReach is a formal reachability tool that can exhaustively determine whether a dynamical system violates its specification.*

Robust testing in S-TaLiRo *was able to identify a design flaw within 8 seconds. In contrast, dReach also returned UNSAFE, but ran for 5+ hours. This raises the general point that when analyzing new controller designs, robust testing produces interpretable results more quickly than reachability. Once major design issues have been addressed in testing, then reachability can be used to certify the scenario as error-free, or find to corner case errors.*

Additionally, robust testing can quickly identify potential safe sub-regions. ***Figure 4*** *shows the robustness of system trajectories as a function of the initial velocity of the environment vehicle, its x-coordinate, and the goal region of the AV. Green points denote safe executions.* ***Figure 4*** *suggests that the system is robust on longer ramps (AV goal between 39 and 50 meters). dReach is able to prove that this region is safe in about 3 minutes, which should be contrasted with the 5+ hours it took to process the entire set of behaviors. This approach is useful because it can precisely answer regulatory questions such as: under what conditions is the system safe to operate?*

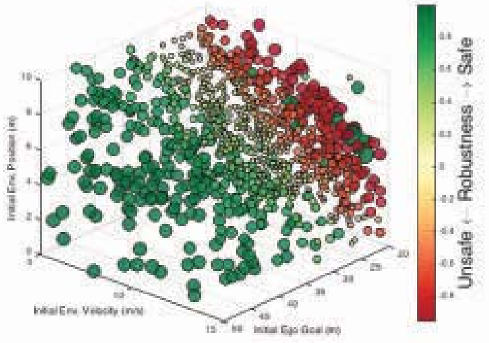


FIGURE 4
Robustness of On-Ramp scenario as a function of 3 initial state variables (1000 runs).

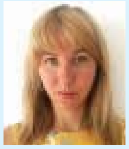
INCORPORATING A WORLD SIMULATOR INTO WHOLE-AV TESTING

An idealized mathematical model of the environment and other cars is not required for a testing tool like S-TaLiRo. The latter only requires the ability to execute the system under test (SUT). The SUT, in fact, could be the actual AV software that will execute on the physical hardware. Therefore, we can leverage advanced simulators that provide the AV perception pipeline with realistic input, such as video and depth data. The

ABOUT THE AUTHORS

Houssam Abbas is a Postdoctoral Fellow in the Department of Electrical and Systems Engineering at the University of Pennsylvania. His research interests include formal verification and testing of cyber-physical systems.

Matthew E. O'Kelly is a doctoral candidate in the Department of Electrical and Systems Engineering at the University of Pennsylvania, with research interests that include design of safe autonomous systems.



Alena Rodionova is a doctoral candidate in the Department of Electrical and Systems Engineering at the University of Pennsylvania with research interests in design of safe autonomous systems.

Rahul Mangharam is an Associate Professor in the Department of Electrical and Systems Engineering at the University of Pennsylvania whose interests are at the intersection of machine learning, control systems and formal methods for life-critical systems. He received the 2016 US Presidential Early Career Award for Scientists and Engineers (PECASE).

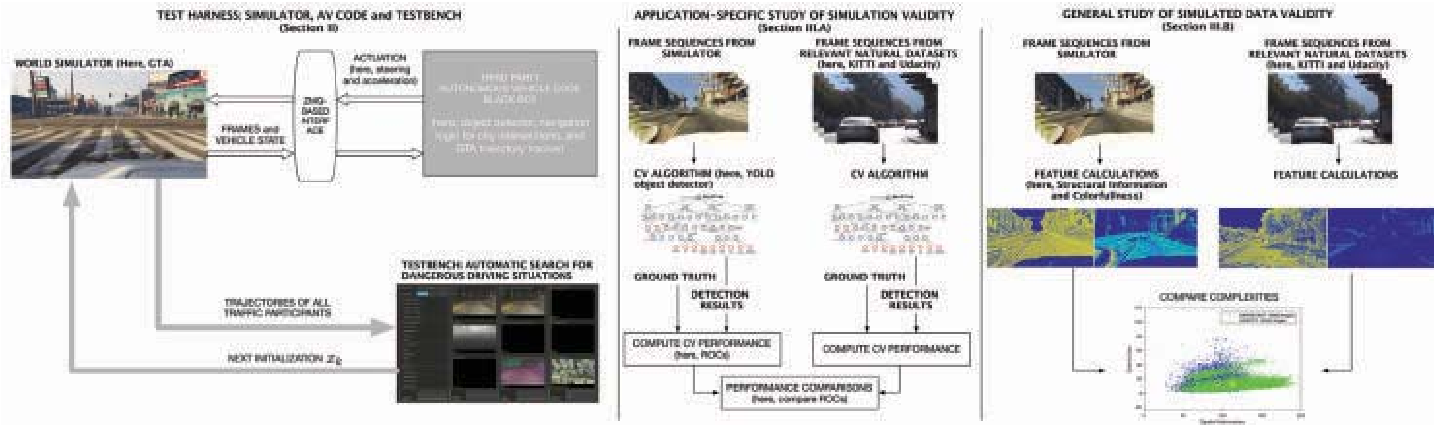

TEST HARNESS: SIMULATOR, AV CODE and TESTBENCH
(Section II)
WORLD SIMULATOR (Here, GTA)
ACTUATION
(here, steering and acceleration)
ZMQ-BASED INTERFACE
FRAMES and VEHICLE STATE
THIRD PARTY AUTONOMOUS VEHICLE CODE BLACK BOX
(here: object detector, navigation logic for city intersections, and GTA trajectory tracker)
TESTBENCH: AUTOMATIC SEARCH FOR DANGEROUS DRIVING SITUATIONS
TRAJECTORIES OF ALL TRAFFIC PARTICIPANTS
NEXT INITIALIZATION x_k
APPLICATION-SPECIFIC STUDY OF SIMULATION VALIDITY
(Section III.A)
FRAME SEQUENCES FROM SIMULATOR
FRAME SEQUENCES FROM RELEVANT NATURAL DATASETS
(here, KITTI and Udacity)
CV ALGORITHM (here, YOLO object detector)
CV ALGORITHM
GROUND TRUTH
DETECTION RESULTS
GROUND TRUTH
DETECTION RESULTS
COMPUTE CV PERFORMANCE
(here, ROCs)
COMPUTE CV PERFORMANCE
PERFORMANCE COMPARISONS
(here, compare ROCs)
GENERAL STUDY OF SIMULATED DATA VALIDITY
(Section III.B)
FRAME SEQUENCES FROM SIMULATOR
FRAME SEQUENCES FROM RELEVANT NATURAL DATASETS
(here, KITTI and Udacity)
FEATURE CALCULATIONS
(here, Structural Information and Colorfulness)
FEATURE CALCULATIONS
COMPARE COMPLEXITIES

FIGURE 5 The test harness. *(Left)* Robustness-guided search for unsafe behavior. The harness selects the initial position and velocity of the AV. It also selects initial environment conditions: positions and velocities of other cars, and time of day, which allows control of the illumination conditions. This initialization is sent to the world simulator (here, GTA), which simulates the scenario in lock-step with the AV code. Every frame produced by the game is sent to the AV to be processed by its perception pipeline. The AV controllers then compute the next actuation that is sent to the game to move the AV. *(Middle)* To validate simulation results, the perception code is run on simulated frames (from the world simulator) and on real datasets, and the performances are compared. *(Right)* The visual complexity of simulated and real datasets are compared to further assess whether simulated data can act as proxy for real data.

perception code then processes this input and extracts from it information for the AV's controllers, such as position and speed of obstacles in the environment.

In [16], a test harness is presented that allows an AV to drive in a simulated world in real-time, as illustrated in **Figure 5**. A notable aspect of this harness is that it allows weather conditions to vary, thus stressing the perception pipeline. This is very important: the 2016 fatal accident in Florida involving a Tesla Auto-Pilot was partially due to a failure of the car's visual sensors to detect the truck blocking the AV's path against the bright sky. Issues like validity of simulated data are also addressed in [16].

Scenario 2 *The game Grand Theft Auto V (GTA) is used as a world simulator. At a T-junction in the GTA city map, the objective of the AV is to make a safe right turn, and obey the Stop Sign. Robust testing automatically found a non-trivial accident between the AV and another car in under 100 simulations. This was due to the right combination of poor lighting (robust testing automatically chose twilight conditions) and similar speeds for the AV and another car.*

MOVING FORWARD: TOWARD RISK ANALYSIS FOR AUTONOMOUS VEHICLES

Ultimately, after all the testing and verification, non-technical issues like insurance and liability must be settled for autonomous vehicles to become a commercial reality. Insurance speaks the language of risk: what is the probability of a terrible accident in this city? How often is this car model involved in minor collisions? With autonomous vehicles, we have a chance to answer these questions *before* the AV hits the road: by a careful choice of simulations, and with large amounts of traffic data, we can build a *risk profile* of an AV to guide the insurance pricing. The above techniques, from formal verification to testing, further this goal by giving complementary ways of quantifying the likelihood of an accident and its severity. An autonomous vehicle thus brings together disparate fields of inquiry, and may well be the first autonomous robot that deals directly with social questions like "What level of risk are we prepared to explicitly accept, and for what benefit and to whom?" ■

REFERENCES

1 Sarah M. Loos, André Platzer, and Ligia Nistor. Adaptive cruise control: Hybrid, distributed, and now formally verified. In *International Symposium on Formal Methods*, Springer, 2011, pp. 42–56.

2 Albert Rizaldi and Matthias Althoff. Formalising traffic rules for accountability of autonomous vehicles. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems,* IEEE, 2015, pp. 1658–1665.

3 Matthias Althoff and John M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4), 2014, pp. 903–918.

4 Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M. Murray. Receding horizon control for temporal logic specifications. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control, ACM*, 2010, pp. 101–110.

5 Aakar Mehra, Wen-Loong Ma, Forrest Berg, Paulo Tabuada, Jessy W. Grizzle, and Aaron D. Ames. Adaptive cruise control: Experimental validation of advanced controllers on scale-model cars. In *2015 American Control Conference (ACC)*, IEEE, 2015, pp. 1411–1418.

6 Werner Damm, Hans-Jörg Peter, Jan Rakow, and Bernd Westphal. Can we build it: formal synthesis of control strategies for cooperative driver assistance systems. Mathematical Structures in Computer Science, 23(04), 2013, pp. 676–725.

7 Anirudha Majumdar, Mark Tobenkin, and Russ Tedrake. Algebraic verification for parameterized motion planning libraries. In *2012 American Control Conference (ACC)*, IEEE, 2012, pp. 250–257.

8 Theodore P. Pavlic, Paolo A.G. Sivilotti, Alan D. Weide, and Bruce W. Weide. Comments on adaptive cruise control: hybrid, distributed, and now formally verified. *OSU CSE Dept TR22*, 2011.

9 Matthias Althoff and John M. Dolan. Reachability computation of low-order models for the safety verification of high-order road vehicle models. In *American Control Conference (ACC)*, 2012, IEEE, 2012, pp. 3559–3566.

10 NHTSA. Federal automated vehicles policy, September 2016.

11 Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4), 1990, pp. 255–299.

12 Yashwanth S. R. Annapureddy and Georgios E. Fainekos. Ant colonies for temporal logic falsification of hybrid systems. In *Proc. of the 36th Annual Conference of IEEE Industrial Electronics*, 2010, pp. 91–96.

13 Houssam Abbas, Matthew O'Kelly, and Rahul Mangharam. Relaxed decidability and the robust semantics of metric temporal logic: Technical report. University of Pennsylvania Scholarly Commons, 2017.

14 Matthew O'Kelly, Houssam Abbas, and Rahul Mangharam. Computer-aided design for safe autonomous vehicles. In *Resilience Week*, 2017.

15 Soonho Kong, Sicun Gao, Wei Chen, and Edmund M. Clarke. dreach: Delta-reachability analysis for hybrid systems. In *Tools and Algorithms for the Construction and Analysis of Systems-21st International Conference*, TACAS 2015, 2015, pp. 200–205.

16 Houssam Abbas, Matthew O'Kelly, Alena Rodionova, and Rahul Mangharam. Safe at any speed: A simulation-based test harness for autonomous vehicles. 2018. Submitted.