Role of Formal Verification in Certifying Autonomous Vehicles

Parasara Sridhar Duggirala
CSE@UCONN
psd@uconn.edu
Some Recent Developments

Mountain View, CA—Since 2009

Google self-driving car so-far had 13 accidents (none fatal).
Some Recent Developments

Google self driving car so-far had 13 accidents (none fatal).

Self-Driving Tesla Was Involved in Fatal Crash, U.S. Says

By BILL VLASIC and NEAL E. BOUDETTE  JUNE 30, 2016

CALIFORNIA’S FINALLY READY FOR TRULY DRIVERLESS CARS

JACK STEWART  TRANSPORTATION  03.11.17  7:00 AM
Some Recent Developments

Google self-driving car so-far had 13 accidents (none fatal).

Self-Driving Tesla Was Involved in Fatal Crash, U.S. Says

By BILL VLASIC and NEAL E. BOUDETTE  JUNE 30, 2016

CALIFORNIA’S FINALLY READY FOR TRULY DRIVERLESS CARS

Uber has grounded its self-driving cars in Arizona after one was involved in an accident

Johana Bhuiyan
Saturday, 25 Mar 2017 | 3:29 PM ET

Uber resumes self-driving program three days after Arizona crash
Doomsday in 10 Years!

Tuesday, April 1, 2025

Google Car Claims 100 Lives

In a shocking incident in Menlo Park, CA, a Google car claimed the lives of 100 individuals. Sources from Google seem to suggest that the cause for this accident is a software bug because of some un-reviewed code written by a teenage intern. Tech companies like Microsoft, Facebook, and Amazon issued public statements condemning Google because 90 of the 100 people dead in the accident worked in one of these companies.
Doomsday in 10 Years!

Massive Amazon cloud service outage disrupts sites

Elizabeth Weise, USA TODAY  Published 1:51 p.m. ET Feb. 28, 2017 | Updated 6:56 a.m. ET March 1, 2017

Here’s Why Amazon’s Cloud Suffered a Meltdown This Week

Unfortunately, one of the inputs to the command was entered incorrectly and a larger set of servers was removed than intended. The servers that were inadvertently removed supported two other S3 subsystems.
With great software, comes great risks!
Avoiding The Doomsday

Certified Autonomous Vehicle
This Talk: How Formal Verification Can Help in Certifying Autonomous Vehicles.

Certified Autonomous Vehicle
Outline

✓ Avoiding Doomsday For Autonomous Vehicles
  • Formal Verification 101
  • Success Stories of Formal Verification
  • Roadmap for Certification of Autonomous Vehicles
  • Conclusions
Formal Verification 101

- Model Checking: Algorithmically verifying if your given model satisfies a given property.

- Used extensively in hardware, software, and protocol verification

- (There are no buffer overflow possibilities)
Model Checking $\neq$ Extensive Testing

- Model Checking considers all possible executions and will either return a proof or counterexample.

Model Checking Tool

- Model
- Software
- Property
  - (There are no buffer overflow possibilities)

Used extensively in hardware, software, and protocol verification.
Verification of Autonomous Vehicle

Model Of AV

Certification Software

Property
(Doesn’t “violate” traffic laws)

Certificate

Counterexample
Why Software For Certification?

Ex. Aerospace domain: Certification for avionic systems is done manually.
Why Software For Certification?

Ex. Aerospace domain: Certification for avionic systems is done manually.

Consequence: *Modifying a single line of code would require certification of the entire system.*
Why Software For Certification?

Ex. Aerospace domain: Certification for avionic systems is done manually.

Consequence: *Modifying a single line of code would require certification of the entire system.*

Avionics ≠ Automotive – widely different markets.
Why Software For Certification?

Ex. Aerospace domain: Certification for avionic systems is done manually.
Consequence: *Modifying a single line of code would require certification of the entire system.*

Avionics ≠ Automotive – widely different markets.

As the complexity of the system increases, manual certification cost increases exponentially. **Automating the certification process using Formal Verification is our only hope.**
Success Stories of Formal Verification
Disaster Scenarios

Bad software caused some serious damage!

Intel Pentium bug caused loss of reputation and money.

Ariane 5 crashed within a few minutes after launch.

Software race condition caused northeast blackout of 2003.

Software bug caused Toyota to recall 1.2M Prius cars.
Avoiding Doomsday

Intel now uses Synopsys/Cadence tools for formal verification.

Static analysis could catch the bug during the analysis.

NASA review of Toyota’s code mentions using Formal verification tools.
AbsInt: A Tool For Software Verification Of Flight Control Software.

The flight control software in some AIRBUS systems have been fully verified to not have any buffer overflow or division by zero errors.

For over a decade, Airbus France has been using our tools in the development of safety-critical avionics software for several airplane types, including the flight control software of the A380, the world's largest passenger aircraft.

Honda has been using our tools in developing the FADEC software of a turbofan engine.

The Technical University of Munich is using our tools in the development, testing and optimization of flight control and navigation algorithms.
SLAM: Static Verifier for Windows Drivers

Used for verifying device drivers for Windows.

"Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we're building tools that can do actual proof about the software and how it works in order to guarantee the reliability."

Bill Gates, April 18, 2002. Keynote address at WinHec 2002

Low rate of false positives – around 4%. 

IEEE Property Specification Language

• Combined industry + academia effort for standardizing the specification language for hardware.

• Part of Verilog, VHDL, SystemC, and SystemVerilog.

• Intel, IBM, Freescale, Synopsis, Cadence, etc.
**SeL4: A Formally Verified Microkernel**

Uses mechanical proof checker Isabelle/HOL

Properties of microkernel are encoded as theorems and proved in Isabelle/HOL

SeL4 has been deployed on Unmanned Little-Bird helicopter.

**RED Team** hackers couldn’t hack it even after 6 months while having access to source code!
How To Certify Autonomous Vehicles: A Formal Verification Approach And Research Directions
Levels of Abstraction

1. High level traffic rules
   “Always stop at a red light”

2. Motion primitives
   Controllers for turning car and avoiding collisions.

3. Real time correctness
   A command issued will run with a maximum latency of 20ms
High Level Traffic Rules

• For the humans, by the humans.

Official driver’s manual provided by State of Connecticut Department of Motor Vehicles.

60 page document describing the rules of the road.
High Level Traffic Rules

• For the humans, by the humans.

Official driver’s manual provided by State of Connecticut Department of Motor Vehicles.

60 page document describing the rules of the road.

How does a computer understand these rules?
Traffic Rules For A Computer?

- Computer can understand formulas and logic.
Traffic Rules For A Computer?

- Computer can understand formulas and logic.
- Solution: encode traffic rules as logic formulae.
- Is it even possible? Evidence suggests, yes.
Traffic Rules For A Computer?

• Computer can understand formulas and logic.
• Solution: encode traffic rules as logic formulae.
• Is it even possible? Evidence suggests, yes.

• Example PSL specification for hardware circuits:
  "an acknowledgement is issued within 4 cycles of receiving a request"
  
  \[
  \text{always(req -> \{[*4]; ack;\})}
  \]

• Example from Connecticut DMV manual:
  "merging with any traffic should take at most 2 seconds"
  
  \[
  \text{always(mergeBegin -> \{[*2]; mergeEnd;\})}
  \]
Linear Temporal Logic – A Logic For Describing Temporal Properties

• Linear Temporal Logic (LTL): A logical framework for expressing temporal specification of behaviors.
  \[
  \text{always}(\text{req} \Rightarrow \{[*4]; \text{ack};\})
  \]

• Variants of temporal logic (timed logics) for specifying real-time behaviors.

Research Direction
Motion Primitives
Motion Primitives

- Behavior of car can be modeled in a "bicycle model"
Motion Primitives

- Behavior of car can be modeled in a "bicycle model"

\[ \dot{x} = f(x, u) \]

- Given the model \( \dot{x} = f(x, u) \), are the controllers for turning, passing, and breaking, safe?
Verification of Motion Primitives.

• Hybrid Systems Verification (Computer Science + Control)

CSE5905 HW Prob.: Generate parameters for autonomous car [simpler model] Controller and verify using C2E2 (my tool) if the safety specifications are satisfied

Research Direction
Scalable Verification Tools For Handling Complex ODE Model
Real-Time Behavior

• In theory, the hardware just runs one process, in practice, the hardware juggles various processes.

• Example: the mars rover had 50 processes while landing!

• Each of these processes are handled by scheduler.
How To Guarantee Real-Time Behavior?

• Real-Time Systems in Computer Science.

• **Worst Case Execution Time (WCET) Problem:** Given a program $P$ and a hardware platform $H$ what is the worst case running time of $P$ on $H$?

• Very mature software tools for analyzing WCET.
Formal Verification For Certified Autonomous Vehicles

1. High level traffic rules
   "Always stop at a red light"
   **Use Temporal Logic**

2. Motion primitives
   Controllers for turning car and avoiding collisions.
   **Use Hybrid Systems Verification Tools**

3. Real time correctness
   A command issued will run with a maximum latency of 20ms
   **Use WCET Tools From Real-Time Systems**
Prototype – Lab Version

Miniature version of autonomous vehicle

Thank You. Questions?