Programmable Test Track for AVs

Aviral Shrivastava
Arizona State University

With inputs and support from:
Lina Karam, Georgios Fainekos, Heni Ben Amor, Rida Bazzi, Ram Pendyala, Sethuraman Panchanathan (faculty @ ASU)
Edward Andert, Mohammad Khayatin (my students @ ASU)
Marissa Walker, Kevin Biesty, Matt Clark (state of Arizona)
Greg Leeming (Intel), Jack Weast (Intel), Jeffrey Wishart (Exponent), Nils Hoffman (Local Motors)
About myself

- Computer Scientist by training
  - IIT Delhi, and UC Irvine
- Professor of CSE @ Arizona State University

- NSF, NIST, Industry projects on
  - Scaling real-time compute-power of processors
  - Tick-talk: Timing API for distributed CPS
  - Testing the timing of CPS

- AV-related research
  - Help build some AVs
  - Design of algorithms for traffic intersections of AVs [DAC 2017][RTSS 2018]
Software correctness is hard!!

```c
bool flag[2] = {false, false};
int turn;

flag[0] = true;
turn = 1;

while (flag[1] == true && turn == 1)
{
    // busy wait
}

// critical section
...
// end of critical section
flag[0] = false;

flag[1] = true;
turn = 0;

while (flag[0] == true && turn == 0)
{
    // busy wait
}

// critical section
...
// end of critical section
flag[1] = false;
```

Peterson’s algorithm for mutual exclusion of two threads

Web page: aviral.lab.asu.edu
Software correctness is hard!!

The Ariane 501 crash
- Start.
- 37 seconds of flight.
- KaBOOM!
- 10 years and 7 billion dollars are turning into dust.

Why visibility matters—the Ariane 5 crash
- Velocity was represented as a 64-bit float
- A conversion into a 16-bit signed integer caused an overflow
- The current velocity of Ariane 5 was too high to be represented as a 16-bit integer
- Error handling was suppressed for performance reasons

```
-- Vertical velocity bias as measured by sensor
L_M_BV_32 :=
    TBD.T_ENTIER_32S((1.0/C_M_LSB_BV) *
    G_M_INFO_DERIVE(T_ALG.E_BV));
-- Check, if measured vertical velocity bias can be
-- converted to a 16 bit int. If so, then convert
if L_M_BV_32 > 32767 then
    P_M_DERIVE(T_ALG.E_BV) := 16#7FFF#;
elsif L_M_BV_32 < -32768 then
    P_M_DERIVE(T_ALG.E_BV) := 16#8000#;
else
    P_M_DERIVE(T_ALG.E_BV) :=
        UC_16S_EN16NS(TDB.T_ENTIER_16S(L_M_BV_32));
end if;
-- Horizontal velocity bias as measured by sensor
-- is converted to a 16 bit int without checking
P_M_DERIVE(T_ALG.E_BH) :=
    UC_16S_EN16NS(TDB.T_ENTIER_16S((1.0/C_M_LSB_BH) *
    G_M_INFO_DERIVE(T_ALG.E_BH)));
```

*Source: http://moscova.inria.fr/~levy/talks/10enslongo/enstlongo.pdf
Consensus-driven testing of AVs

- No test can prove the safety of a CAV
- Confidence building measure
- Measurable target for the developers
- Clear definition of due diligence
- Confidence building
- AV developer/manufacturer independent/agnostic

Web page: aviral.lab.asu.edu
Programmable test track

- Environmental Tunnel
  - Wind, snow, rain

- Cityscape
  - Intersections, stop signs

- Roadway interactions
  - Bicyclists, other vehicles, deer on a highway

- Program the timing of the events and interactions
- Can be set using a script

Web page: aviral.lab.asu.edu
How to create the tests?

- Choose commonly occurring scenarios
- Choose among the known NHTSA crash scenarios
- Driving scenarios to test sensor and sensor-fusion weak points
  - Sudden start/stop of rain, light
- Driving scenarios to test ML vulnerabilities
  - Adversarial attacks
- Driving scenarios to test software vulnerabilities
  - Module interaction and reuse, exceptions handlers
- Driving scenarios to test TIM situations
  - How does the AV behave in an accident
- Driving scenarios to test AV-driver interface
  - Is there enough time for a driver to be alerted and that they can meaningfully intervene
- Driving scenarios to test basic security vulnerabilities
  - Jeep attack

Varying illumination and obstacle types
Runtime safety monitor

- Mutually agreed-to safe driving rules (e.g. RSS)
- Safety monitor that will test whether the vehicle is driving safely at all times
  - Needs only coarse-level information, like the speed of the vehicle, position, acceleration of the vehicle
- Useful for internal testing for a manufacturer/developer
- Conflict resolution
  - If non-tamperable (encrypted), the collected data will be compelling evidence to defend the actions of your vehicle
- Fundamental tension of using own sensors/using vehicle data
  - Use low-level information – will be useful to validate software’s decisions
Conclusion

- Software correctness is hard!!!

- Before deployment - Need Mutually agreed-to test for AVs
- After deployment - Need runtime safety monitors

- Exciting times are ahead