Bridging the Gap between Research and Practice: Formal Methods for Powertrain Control Software

Jyotirmoy V. Deshmukh
What is a Powertrain?

- Main components in delivering power
- Automotive context: engine and transmission
- This talk: engine, fuel cell stack, engine + electric motor, etc.
What is Powertrain Control Software

- **Real-time Control Algorithms**
  - Examples: Air/Fuel Ratio Control, Idle Speed Control, Exhaust-gas recirculation, boost control, Electronic throttle control, battery management systems, etc.

- **On-board Diagnostic Algorithms**
  - Fuel and air metering, emissions controls, misfire indication, telematics, fleet tracking, etc.
What do we mean by formally verified?

Safety
Low exhaust gas emissions
Good Fuel Efficiency
Drivability
Comfort
Complexity of powertrain software is increasing

- 1977: first GM car with embedded software
- 1981: GM: 50K LOC for entire US fleet
- 70 to 100 ECUs in modern luxury cars, close to 100M LOC
- Engine control: 1.7M LOC
  - F-22 raptor: 1.7M, Boeing 787: 6.5M
- Frost & Sullivan: 200M to 300M LOC
- Electronics & Software: 35-40% of luxury car cost


UC Berkeley, Dream Seminar, Oct 15
Overview

- Model-based Development: Verification & Validation

- Promising Techniques
  - Some success stories

- Challenge Problems
Dev Target: Keep A/F ratio to 5% of 14.7

- Catalytic converters reduce HC, CO2, and NOx emissions
- Conversion efficiency optimal at stoichiometric value

MBD process

Development target

Prototype Design

System evaluation

Legacy Code

Prototype modeling and implementation

UC Berkeley, Dream Seminar, Oct 15
Controller Hybrid Automaton

- No Feedback Control
- Only feedforward estimator

- Feedback Control
  + Feedforward estimator

startup

\[ \lambda \downarrow \text{ref} = 14.7 \]

normal

\[ \lambda \downarrow \text{ref} = 14.7 \]

sensor failure

\[ \lambda \downarrow \text{ref} = 14.7 \]

power

\[ \lambda \downarrow \text{ref} \uparrow \text{power} = 12.5 \]

\( \theta \leq 50 \uparrow \theta \)

\( \theta \geq 70 \uparrow \theta \)
MBD process

Development target

Feasibility study / requirement analysis

Prototype Design

Prototype modeling and implementation

System evaluation

Legacy Code

New design

Requirement

Controller Specification Model

Testing

Auto-Code Generation

Lots and lots of testing, now by driving!

More testing

Integrated code

More Testing

UC Berkeley, Dream Seminar, Oct 15
# Challenge: Verify these Temporal Logic Requirements

\[ \mu = \lambda - 14.7 / 14.7 \]

Normalized A/F Ratio

**Existing formal verification techniques struggle**

## Normal Mode Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>□ ( \downarrow(\tau \downarrow s, T) ) (</td>
<td>\mu</td>
</tr>
<tr>
<td>□ ( \downarrow(\tau \downarrow s, T) ) (rise&amp;fall⇒□ ( \downarrow(\eta, \zeta / 2 ) ) (</td>
<td>\mu</td>
</tr>
<tr>
<td>◇ ( \downarrow(T,T) ) ( \sqrt{1/t - \tau \downarrow I} \int_0^t { (\lambda(t) - 14.7)^2 u(\tau - \tau \downarrow I) } d\tau ) ( &lt; 0.05 )</td>
<td>RMS error is less than 0.05</td>
</tr>
<tr>
<td>□ ( \downarrow(\tau \downarrow s, T) ) ( \mu &lt; 0.05 )</td>
<td>Maximum overshoot is 0.05</td>
</tr>
</tbody>
</table>

UC Berkeley, Dream Seminar, Oct 15
V&V in the industry: both tools and practice

- Mostly focused on “verifying” the controller
  - By that, I mean “verifying” the generated C code
  - By that, I mean testing the generated C code

- Code coverage metrics, test-case generation

- Some formal methods being marketed by tool vendors
  - Static analysis: Dead code detection, Divide by zero
  - Property Proving
  - Do not scale to large open-loop or simple closed-loop models

- Dearth of tools for closed-loop testing/verification?
Scaling and Industrializing MBD V&V: A case for simulation-guided formal analysis
Spectrum of Analysis Techniques

Can apply to real designs? (scalability)

Software Testing
Control Theory Techniques

- Simulation
- Linear Analysis (numerical)
  - Test Vector Generation for Model Coverage
- Linear Analysis (symbolic)

Program Analysis
Formal Verification

- The One Tool to Rule Them All
- Concolic Testing
  - (Bounded) Model Checking
  - Stability Proofs
  - Reachability Analysis
- Theorem Proving

How formal/exhaustive?
Why start from simulations?

- Visual feedback, bug-finding
- Can use existing design artifacts
- Does not require knowledge of:
  - Temporal Logic, SAT modulo theories, Bounded Model Checking, Undecidability, Hybrid Automata, …
- Rich heritage of dynamic analysis, runtime verification in formal methods/verification community
  (SciDuction, Seshia et al., Proofs from Tests, Gupta et al., Concolic Testing, Sen et al., Monitoring Temporal Logic (cf. Leucker et al.))

- Control Designers already use them a lot
- What we learned: If we want production engineers to pay attention, we have to speak to them in a language they understand!
A Few Promising Techniques

- Requirement Falsification
- Mining Temporal Requirements
- Conformance Testing

- Simulation-guided dynamical analysis
- Simulation-guided reachability analysis
Requirement Falsification

- Given model $M$; property $\varphi$
- Find input $u(t)$ and initial conditions $x\downarrow0$ s.t.
- $M(x\downarrow0, u)$ does not satisfy $\varphi$
- Not verification, but systematic bug-finding, aka “super testing”
- No guarantees of completeness (except asymptotic/probabilistic)
Signal/Metric Temporal Logic for Requirements

\[ \square_{[0,100]}(x > 1 \land x < 3) \]

Always between time 0 and 100
Eventually at some time $t$ between time 20 and 60

From that time $t$, always till the end of the signal trace

$\diamondsuit [20, 60] \square [0, \infty) \text{ } |x| < 0.1$
Quantitative Semantics for Real-time Temporal Logics

- Robust satisfaction$^{1,2}$ of temporal logic property $\varphi$ by given simulation trace $y(\cdot)$:
  - Function mapping $\varphi$ and $y$ to $\mathbb{R}$
  - Positive number = $y$ satisfies $\varphi$
  - Negative number = $y$ does not satisfy $\varphi$
  - Moving towards zero = moving towards violation

Quantitative Semantics for Real-time Temporal Logics

\[ [0,0.7] \Diamond [0,0.3] x \geq 1.5 \]

\[ \mu = x \mid 1.5 \]

\[ \mu \leftarrow 0 \quad 1 \quad 0.5 \quad -0.5 \quad 0.5 \quad -1 \quad 0 \quad 0.5 \]

\[ \Diamond [0,0.3] \mu \]

supremum over each interval
Quantitative Semantics for Real-time Temporal Logics

\[ \rho \left( \square[0,0.7] \Diamond[0,0.3] x \geq 1.5, x, 0 \right) = 0.5 \]

\[ \mu = x \mid 1.5 \]

\[ \Diamond[0,0.3] \mu \]

\[ \square[0,0.7] \Diamond[0,0.3] \mu \]

infimum over result from previous step
Falsification by optimization

u(t)

M(u(t), x<0)

Optimizer:
Minimize robust satisfaction value

- Use powerful optimization heuristics to get close to global optimum
Falsification with Parameterized Inputs

**S-TaLiRo** [Fainekos, Sankaranarayanan, et al., TACAS ’11, HSCC ’10, ACC ’12]
- Metric Temporal Logic based robustness computation
- Supports: simulated annealing, cross-entropy, ant-colony, genetic algorithms

**Breach** [Donzé et al., CAV ‘10, NSV ‘13]
- Signal Temporal Logic based robustness computation
- Supports Nelder-Mead optimizer
Falsification + State-space Coverage

**RRT-REX** [Dreossi, Donzé, Dang + Toyota MBD, NFM ‘15]

- RRT-based optimal search

Choosing Goal Point:
- Maximize Space-Coverage
- Pick in region of low robust satisfaction value

Pick Neighbor to grow from
- Pick neighbor with lowest robust satisfaction value

Local Choice for Input
- Decrease robust satisfaction value of partial trace

x↓2
BAD

Different choices for $u(\Delta t)$

Different choices for $u(2\Delta t)$

x↓1

UC Berkeley, Dream Seminar, Oct 15
Falsification + Input Space Refinement

**SITAR** [Toyota MBD + Oded Maler, ATVA ‘15]

- Discretize Input Signal Space
Falsification + Input Space Refinement

**SITAR** [Toyota MBD + Oded Maler, ATVA ‘15]

- Evaluate cost at neighbors
- Descend to neighbor with lowest cost
- Too many neighbors? Stochastically pick a subset
- Maintain Tabu list to avoid revisiting neighbors
- Havoc when local optimum or slow convergence
Falsification + Input Space Refinement

**SITAR** [Toyota MBD + Oded Maler, ATVA ‘15]

- Refine Input Signal Space from “promising inputs”
- Permit nonuniform gridding and refinement
In Practice?

- Found “hunting” behavior in an experimental Toyota airpath control model
  - Over 4000 Simulink blocks
  - Advanced Control Scheme

- Found undesirable overshoot in an experimental version of control software in a powertrain application
  - Large model
  - Takes 5 to 10 seconds to simulate one second of real-time
  - Hours of falsification required
Requirements in an Industrial Setting

- Formal methods/verification engineers love them

- Control designers write them in Word documents
  - (in Japanese and/or German)

- Control designers (and occasionally temporal logic pundits) have a hard time writing them
Mining Requirements

- How do control designers check correctness?

- What do you do when your chief engineer leaves?

- Formal techniques need machine-checkable requirements

- Key Idea: Identify Temporal Logic Patterns from Simulation Data!
Counterexample Guided Inductive Synthesis

Find “Tightest” Answers

Settling Time is ??
Overshoot is ??
Bounds on x are ??

Settling Time is 5 ms
Overshoot is 5 KPa
Upper Bound on x is 3.6

∃ trace ⌫ Property?
Counterexample Guided Inductive Synthesis

∃ trace \not\models \text{Property}?

Counterexamples

Find “Tightest” Answers

Settling Time is 5.8 ms
Overshoot is 5.3 KPa
Upper Bound on x is 4

Settling Time is ??
Overshoot is ??
Bounds on x are ??
Countercexample Guided Inductive Synthesis

Find “Tightest” Answers

Settling Time is ??
Overshoot is ??
Bounds on $x$ are ??

Settling Time is 6 ms
Overshoot is 5.5 KPa
Upper Bound on $x$ is 5

∃ trace \not\models Property?

NO
Conformance Testing

- Is behavior of models $M_1$ and $M_2$ “similar”?
- Multiple Distance metrics in theory
  - Skorokhod metric
  - Kossentini-Caspi metric
    - [C. Kossentini, P. Caspi, *Approximation, Sampling and Voting in Hybrid Computing Systems*]
  - Metric by R. Goebel, R. Sanfelice and A. Teel
  - $(T, J, (\tau, \varepsilon))$-closeness metric
  - $\delta$-approximate bisimulation relations
Key enablers

- Efficient algorithms to compute distance metrics
  - R. Majumdar, V. Prabhu, *Computing the Skorokhod Distance between Polygonal Traces*, HSCC 2015

- Simulation & optimization-guided conformance testing

UC Berkeley, Dream Seminar, Oct 15
Here are some grand challenges....
Grand Challenge I: Requirement Engineering

Key challenge for Toyota, Bosch, and others


During the position (cp) regulation after a step input on demand (dp), when the absolute value of the maximum torque limit (tl) decreases with a step (pre-condition), the absolute value of the actuator response in torque (ct) must be less than the torque limit plus 10% in less than 10 ms (postcondition).

\[
p_1 \cup_{(a_1,b_1)} (p_2 \cup_{(a_2,b_2)} (p_3 \cup_{(a_3,b_3)} (p_4 \cup_{(a_4,b_4)} G p_5)))
\]
Grand Challenge I: Requirement Engineering

- Key challenge for Toyota, Bosch, and others
  - How do you present requirements to control designers?
  - How do they convey their intention without using formalisms?
  - Is Temporal Logic the right requirement language?
Grand Challenge II: Impending invasion of concurrency

- Huge push for multi-core and many-core ECUs
- Immediate Challenges: Real-time +
  - Partitioning/Parallelizing
  - Load balancing
  - Memory mapping,
  - Cross-core reads, overhead?
  - Legacy software?
- Philosophical challenges
  - Concurrency is a known hard problem
  - Programming Languages
  - Domain-specific Determinism: semantic equivalence vs. predictable run-time
  - Control designers rarely know about memory models, parallelization, etc.
Credits

- Jim Kapinski, Xiaoqing Jin, Hisahiro “Isaac” Ito, Koichi Ueda, Ken Butts: Toyota
- Alexandre Donzé, Sanjit Seshia: UC Berkeley
- Oded Maler, Tommaso Dreossi, Thao Dang: Verimag
- Georgios Fainekos: Arizona State University
- Sriram Sankaranarayanan: Univ. of Colorado

- Thank You, Questions?