Requirements Extraction from Models of Automotive Software

Rance Cleaveland

Department of Computer Science
University of Maryland

1 November 2009

Joint work with Sam Huang, Chris Ackermann (UMD); Arnab Ray (Fraunhofer CESE); Charles Shelton, Beth Latronico (Robert Bosch)

©2009 Fraunhofer USA Inc.
The Model Checking Problem

\[ M \models \varphi \]

system / model

property / requirement

satisfies / possesses
The Synthesis Problem

? ⊨ \varphi
The Requirements-Extraction Problem

\[ M \models ? \]
Motivation for Requirements Extraction

- System comprehension
- Specification reconstruction
  - Missing / incomplete / out-of-date documentation
  - “Implicit requirements” (introduced by developers)
Requirements Extraction for Automotive Software

• Joint project: UMD, Fraunhofer, Bosch

• Outline
  – Automotive software development
  – Reqts-extraction via machine learning
  – Pilot study
  – Conclusion
Automotive Software

• Driver of innovation

90% of new feature content based on sw [GM]
50M+ lines of code [GM]

• Rising cost

20% of 2006 vehicle cost due to software [Conti]

• Warranty, liability, quality

High-profile recalls in Germany, Japan, US
Automotive Software Development

• Ensure high quality of automotive software
  – ... while preserving time to market
  – … at reasonable cost
• How?
  – Model-based development (MBD)
    *Efficiencies in production*
  – Automated testing
    *Efficiencies in verification and validation (V&V)*
Models: Simulink®

- Block-diagram modeling language of The MathWorks, Inc.
- Hierarchical modeling
- Simulation
- Continuous, discrete semantics

©2009 Fraunhofer USA Inc.
Models: Stateflow®
Semantics

- Simulink has different “solvers” (= semantics)
  - Continuous: inputs / outputs are signals
  - Discrete: inputs / outputs are data values
- Analog modeling: continuous solvers
- Digital-controller modeling: discrete solvers
  - Synchronous
  - Run-to-completion
  - Time-driven

©2009 Fraunhofer USA Inc.
(Model-Based) Development

- Models formalize specifications, design
- Models support V&V, testing, code generation
- Models facilitate communication among teams

©2008 Fraunhofer USA Inc.
Requirements Extraction

• The extraction problem
  – Given: system (M)
  – Produce: requirements (φ)

• Approach
  – Generate test data satisfying coverage criteria
  – Use machine learning to propose invariants
  – Check invariants using instrumentation-based verification
Coverage Testing via Guided Simulation

- Test = simulation run = sequence of I/O vectors
- Goal: maximize model coverage
  e.g. branch, state, transition, MC/DC, etc.
- Method: guided simulation
  - Simulate model, BUT
  - Choose input data to guide simulation to uncovered parts
  - Turn simulation runs into test data
- Input selection by Monte Carlo, constraint solving
- Implemented in Reactis® model-based testing and verification environment

©2009 Fraunhofer USA Inc.
Machine Learning

- Tools for inferring relationships among variables based on time-series data
  - Input: table
  - Output: relationships ("association rules")
  
<table>
<thead>
<tr>
<th>Time</th>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- e.g. $0 \leq x \leq 3 \implies y \geq 0$
Instrumentation-Based Verification

- Formulate requirements as monitor models
  - Inputs: signals in model
  - Outputs: boolean flags
    - Flag = true: no violation so far
    - Flag = false: violation detected

- Instrument main model with monitors

- Test instrumented model to search for violations

“If speed is < 30, cruise control must remain inactive”
Machine Learning and Requirements Extraction

• General idea
  – Treat tests (I/O sequences) as experimental data
  – Use to infer relationships between inputs, outputs

• Our insight
  – Ensure test cases satisfy coverage criteria (e.g. branch coverage) to ensure “thoroughness”
  – Use IBV to double-check proposed relationships
Pilot Study: Production Body-Electronic Application

• Artifacts
  – Simulink model (ca. 75 blocks)
  – Requirements formulated as state machine
  – Requirements correspond to 42 invariants defining transition relation

• Goal: our approach, random testing [Raz]
  – Completeness (% of 42 detected?)
  – Accuracy (% false positives?)
Pilot Study: Tool Chain

• Automated test-generation tool: Reactis
• Machine-learning tool: Magnum Opus
• Additional tooling
  – Test-format conversions
  – Automated generation of monitor models, instrumentation
Experimental Design

• Repeat five times
  1. Generate coverage tests (Reactis)
  2. Create invariants (Magnum Opus)
  3. Use IBV to double-check invariants (Reactis)
  4. Combine original, IBV tests, rerun 2, 3

• Repeat five times
  1. Generate random tests (Reactis)
  2. Create invariants (Magnum Opus)
  3. Use IBV to double-check invariants (Reactis)
  4. Create second set of random tests, combine with first
  5. Repeat 3
Experimental Results

• Hypothesis: coverage-testing yields better invariants than random testing

• Coverage results:
  95% of inferred invariants true
  97% of requirements inferred
  Two missing requirements detected

• Random results:
  55% of inferred invariants true
  40% of requirements inferred

• Hypothesis confirmed
Conclusions and Directions for Future Research

- Coverage-testing yields better requirements
- IBV double-checks generated invariants effectively
- Future directions
  - Extraction of temporally complex requirements
  - Visualization of generated requirements
  - Requirements extraction as tool for model understanding, exploration, validation
Thank You!

Rance Cleaveland
University of Maryland

rance@cs.umd.edu

301-405-8572

www.cs.umd.edu/~rance