Multicore Protocols Challenge Problem

Milo M. K. Martin
University of Pennsylvania
Project Organization

- Design Methodology
- Computational Engines
- Challenge Problems
  - Apps for Mobile Platforms
  - Multicore Protocols
  - Networked Systems
  - Robotic Systems

Tools and Evaluation

Education and Knowledge Transfer
Opportunity: Specifying Multicore Protocols

• Started as a collaboration between comp. architecture (Martin) & formal methods (Alur)
  • Now broader (Bodik, Solar-Lezama, Tripakis, Seshia)

• Multicore protocols:
  • Distributed, multiple communicating agents
  • Asynchronous and unordered communication
  • Example: cache coherence protocols

• Challenging to get right...
  • Early success story for verification

• Still must be specified (in some modeling language)

Can we reduce this entry barrier using a new specification methodology?
Connections Across Themes

• Interactive Synthesis

• Multi-modal specifications inspired by *Sketch* (Bodik, Solar-Lezama)
Intel’s Current Validation Setup

Based on a slide by: Murali Talupur, Intel, ExCAPE PI Meeting, Jun 10 2013

- Validation can begin in earnest...
  - Only when both RTL and Testing Model are ready
- Differing interpretations possible at testing and design
Create executable, high level model which is formally analyzable
- Validation can begin early
- Leverage to create testing model
- Designers benefit from having unambiguous reference model

**But no viable way to create high level formal models**
A Traditional Specification Methodology
Traditional Specifications

Transitions from I to S.

(1) GetS
Req I→S
Dir I→S
S→S

(2) Data

(1) GetM
Req I→M
Dir I→M

(2) Data[ack=0]

(1) GetS
(2) Fwd-GetS

(3) Data

(3) Data[ack=0]
Traditional Specifications

Rule
State = D_BUSY & InMsg.MType = UNBLOCK_S =>
Begin
  State = D_M;
  sharers = SetUnion(Sharers, SetOf(InMsg.Sender));
  SendMsg({Type = ACK, Acks = 1, Dst = InMsg.Sender});
EndRule;
Traditional Specifications

Model Checker

Verified Protocol

Counter-example
I know what to do in this particular scenario...

But...

Still need to specify it
Can we change the methodology to make the process of specifying distributed protocols easier?

TRANSIT: Specifying Protocols with Concolic Snippets.
Udupa, Raghavan, Deshmukh, Mador-Haim, Martin, and Alur
PLDI 2013
Scenarios:

- Execution traces
- Translated from informal specs
- **Multimodal**, may be:
  - Concrete, or
  - Symbolic constraints, or
  - Both (concolic)

Inspired by *Sketch* (Bodik, Solar-Lezama)
Multi-modal Specification

Protocol Skeleton:
- Messages, processes, state variables

Scenarios
I know exactly what to do in this particular scenario!

and...

That’s what I’m going to specify with additional scenarios.
Goal

Protocol Skeleton
  +
  Scenarios
  +
  Invariants

Completed Protocol Specification
Tools and Evaluation

Design Methodology

Computational Engines

Challenge Problems

Apps for Mobile Platforms
Multicore Protocols
Networked Systems
Robotic Systems

Education and Knowledge Transfer
Scenarios to Protocol

Generate a completed protocol from scenario “snippets”

Find a minimal expression consistent with given scenarios

---

**Rule**

```plaintext
=>
Begin
    State = D_M;
    sharers = SendMsg({
        Type =
        Acks =
        Dst =});
EndRule;
```
Multi-modal Snippets: Example

• Consider the function $\text{max}(a, b)$
  • Informally: if $(a>b)$ then $a$ else $b$

• Expressed as:
  1. Concrete (examples):
     a = 5 & b = 10 $\Rightarrow$ $x = 10$
     a = 8 & b = 5 $\Rightarrow$ $x = 8$
     a = 0 & b = 2 $\Rightarrow$ $x = 2$
  2. Symbolic (logical constraints):
     $x \geq a \quad x \geq b$  $(x == a \mid x == b)$
  3. The desired code itself:
     $x := a > b ? a : b$

more choices, more flexible than just code
Expression Inference

• Consider only concrete snippets for the moment

• Enumerated in increasing order of size
  • Finds minimal expression

• To check consistency
  • Evaluate expression for each scenario

• Prune “indistinguishable” sub-expressions
  • Reduces search space, but uses more memory (dynamic programming)
Expression Inference

- Now consider symbolic snippets
- Satisfying assignment to symbolic snippet is a concrete snippet
- Counter-Example Guided Inductive Synthesis (Solar-Lezama, Seshia)
  - But with expression enumeration in inner loop
Expression Inference

• Now consider symbolic snippets
• Satisfying assignment to symbolic snippet is a concrete snippet
• Counter-Example Guided Inductive Synthesis (Solar-Lezama, Seshia)
  • But with expression enumeration in inner loop

![Diagram](image_url)

- SolveConcrete
- Concrete Snippets
- Consistent with all Symbolic snippets?
  - Yes: Output e
  - No: add concrete snippet corresponding to the witness for inconsistency of e
Cache Coherence Protocols

**Errata**

**AI39.** Cache Data Access Request from One Core Hitting a Modified Line in the L1 Data Cache of the Other Core May Cause Unpredictable System Behavior

**Problem:** When request for data from Core 1 results in a L1 cache miss, the request is sent to the L2 cache. If this request hits a modified line in the L1 data cache of Core 2, certain internal conditions may cause incorrect data to be returned to the Core 1.

**Implication:** This erratum may cause unpredictable system behavior.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.

- **TRANSIT:** a tool for specifying coherence protocols
  - Challenging to get right, bugs in existing protocols
  - Model checking commonly used for these protocols

- **Evaluation**
  - Scalability of inference algorithm
  - Usability via several simple case studies
Found the protocols used small expressions (< size 15)
Pruning enables sufficient scaling (for these protocols)
**Future direction:**
  - Improved scaling through *Syntax-Guided Synthesis*
A Usability Case Study

• Textbook protocol
  • Initial symbolic scenarios from informal desc.
  • Additional concrete scenarios added to fix errors

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># of initial snippets</td>
<td>96</td>
</tr>
<tr>
<td># of snippets in final version</td>
<td>108</td>
</tr>
<tr>
<td>Total Manual Effort</td>
<td>13 hours</td>
</tr>
<tr>
<td># of debug iterations</td>
<td>8</td>
</tr>
<tr>
<td># of counter-examples inspected</td>
<td>6</td>
</tr>
<tr>
<td># of updates/guards inferred</td>
<td>175/80</td>
</tr>
<tr>
<td># of states in verified protocol</td>
<td>1.5 M</td>
</tr>
</tbody>
</table>
Beyond Cache Coherence

• Using TRANSIT...
  • To synthesize the “Alternating-bit” transfer protocol
  • New collaboration (Tripakis)

• Initial findings:
  • Used methodology and synthesis successfully
  • Explored counterexample highlighting
  • Identified liveness checking as future direction
Education, Knowledge Transfer, Industrial Interaction, and Status

• Knowledge transfer & industrial interaction
  • Methodology & initial results published at PLDI
  • Students from project on industrial internships
    • Intel & MSR
  • Pursing further collaborations with Intel

• Directions, plans, and expanding collaborations:
  • Expression inference (via synthesis competition)
  • Inferring deeper protocol properties from scenarios
  • More in-depth case studies & classroom usage
Multicore Protocol Challenge Problem: Recap

• TRANSIT: a design methodology for...
  • Specifying protocols via: **scenarios & interactive synthesis**
  • With **multi-modal** intermixing of concrete and symbolic

• Findings:
  • Inference scalable enough to be useful
  • Used to successfully specify protocols

• Initial results published & industrial engagement ongoing

• Collaborating on synthlib & synthesis competition
  • Provides one of three baseline solvers
  • Also a source of benchmarks
Thank You

Questions?
Concrete Snippets?

- Specified two textbook protocols using only concrete snippets
- On average < 2 snippets required per transition
- Converged on a correct protocol in a few iterations
- Synthesis time: < 1 second
Scenarios and Snippets

- Scenario:
  - Sequence of message exchanges/transitions
  - Transcribed from informal specs
  - A collection of *snippets*

- Snippets:
  - Describe actions on a *single* transition
  - Relate current values of variables to updated values
  - Boolean-valued formulas
Usability: SGI Origin

• Specified the protocol used in SGI Origin
  • Complex, industrial protocol
• Intermixed concrete and symbolic scenarios
• Successfully converged to a correct protocol
• Computational effort:
  • Final synthesis took 30 minutes of CPU time
  • A small number of large expressions dominated
What we measured

• Expression Inference:
  • Does pruning based on concrete snippets help?

• Usability of the specification approach:
  • Tradeoff in using purely concrete vs. symbolic scenarios
  • Human effort involved
Expression Inference: Pruning

• Problem: Many redundant expressions explored
• Solution: Dynamic programming with pruning
• Recursively construct larger expressions
  • Using set of smaller indistinguishable expressions

\[
\begin{align*}
\{ a : 3, b : 4 \} & \equiv \text{indist} \\{ a : 5, b : 3 \}
\end{align*}
\]

\[
\begin{array}{c}
e_1(3, 4) = 7 \\
\text{indist} \\
e_1(5, 3) = 8
\end{array}
\]

\[
\begin{array}{c}
e_2(3, 4) = 7 \\
\text{indist} \\
e_2(5, 3) = 8
\end{array}
\]
Expression Inference: Pruning

- We’re still assuming all concrete snippets
  \{ a : 3, b : 4 \}
  \{ a : 5, b : 3 \}

- Pruning an expression of size $k$:
  - Reduces # expressions of size $> k$ that are considered
Concrete Snippets

Cache 1

State = S

State = S_M

Pend = 1;

Cache 2

Directory

WR_REQ

RSP { DATA = 0, ACKS = 1 }

INV { REQ = 1 }

ACK { CNT = 1 }

Pend = 0;
State = S_M;
Symbolic Snippets

State = S
State = S_M

Pend = Msg.Acks;

Pend -= Msg.Cnt;
Pend = 0 ➔
   State = M;
Pend != 0 ➔
   State = S_M;

Cache 1

Directory

Cache 2

WR_REQ

RSP { DATA = data, ACKS = |Sharers| }

INV { REQ = 1 }

ACK { CNT = 1 }