Automatic Abstraction in Symbolic Trajectory Evaluation

Sara Adams

in collaboration with
Magnus Björk, Tom Melham, and Carl-Johan Seger

University of Oxford

13 November 2007
Our Contribution

Automatic Discovery of highly non-trivial abstractions that make verification of circuits possible that could not be tackled with STE before
Automatic Discovery of highly non-trivial abstractions that make verification of circuits possible that could not be tackled with STE before.
Abstraction in STE

- Diagram of a digital circuit with inputs a, b, and c, and output o.
- Truth tables for inputs and outputs:
  - 10X1100
  - XX0XXX0

- Binary values displayed in the bottom right:
  - 1101000 1100000 1000000
  - 1100100 1001000 0100000
  - 1100010 1000100 0001000
  - 1011100 1001100 1000010
  - ...
Symbolic Indexing

3-input AND

010 011 000 001 100 101 110 111

0XX X0X XX0 111

pq: 0XX
Symbolic Indexing

What’s good about it?

Powerful abstraction mechanism
- can transform exponential verification to linear
- critical enabler for content accessible memory and memory verification

What’s the problem?

Manual derivation tedious
- discovery of good indexing schemes hard
- coverage requirement (else: false positives)
- composition non-trivial
Melham-Jones Algorithm

Input:
- verification task using abstraction scheme A
- relation between scheme A and scheme B

Output:
- verification task using abstraction scheme B

Special case

Start with no abstraction scheme

Coverage condition

Relation has to guarantee that scheme A and scheme B cover the same cases; usually: cover all possible cases
Automatic Re-Indexing

What’s good about it?
- Correctness of indexing scheme machine-checkable
- Compositionality and reasoning of verification

What’s the problem?
- Manual derivation of relation tedious
- Coverage check can be exponential
  - loss of re-indexing profits

Automatic Abstraction
- Generate relation automatically
- Coverage requirement satisfied by construction
Automatic Re-Indexing

What’s good about it?
- Correctness of indexing scheme machine-checkable
- Compositionality and reasoning of verification

What’s the problem?
- Manual derivation of relation tedious
- Coverage check can be exponential
  - loss of re-indexing profits

Automatic Abstraction
- Generate relation automatically
- Coverage requirement satisfied by construction
Backward Propagation

The algorithm

- Input: Specification (as a circuit)
- Output: Indexing Relation (needed for Melham-Jones)

Why use the specification?

- Expresses essential properties
- Uncluttered
Backward Propagation: Using the specification

Basic idea

On the specification determine:
which input combinations force the output to true and false respectively

- start from output
- determine which inputs force the output to be true or false resp.
- when given a choice, introduce an indexing variable
Backward Propagation: Using the specification

Basic idea

On the specification determine:
which input combinations force the output to true and false respectively
- start from output
- determine which inputs force the output to be true or false resp.
- when given a choice, introduce an indexing variable
Example relation for a 3-input AND-gate
Making it work: Encoding

**Basic algorithm**
- 2-input AND-gates
- Fresh indexing variables on every choice

**Efficient algorithm**
- n-input AND-gates, XNOR-gates
- Reuse indexing variables for better sharing

pq: 0XX
Automatic Abstraction

Making it work: Over-abstraction

Basic algorithm
Abstraction dependent on specification only

Efficient algorithm
- Allow declaration of symbolic constants
- specify which inputs not to abstract
Making it work: Automatic Re-Indexing

<table>
<thead>
<tr>
<th>Melham-Jones</th>
</tr>
</thead>
<tbody>
<tr>
<td>General relations</td>
</tr>
<tr>
<td>- expensive quantifications</td>
</tr>
<tr>
<td>Proof of coverage requirement</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modified version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specific structure on relations assumed</td>
</tr>
<tr>
<td>- quantifications eliminated</td>
</tr>
<tr>
<td>- proof in the paper</td>
</tr>
<tr>
<td>Coverage by construction</td>
</tr>
<tr>
<td>- proof in the paper</td>
</tr>
</tbody>
</table>
Results

Content Accessible Memory and Memory

Figure: CAM (left) and Memory (right) verification
- Included: Automatic Abstraction, Re-Indexing, STE run
- Verification not feasible without symbolic indexing
Scheduler

Figure: Scheduler verification

- Specification: *retrieve the oldest ready entry*
- Verification not feasible without symbolic indexing
- Indexing and abstraction highly non-obvious
Future Work

- Automatic Abstraction
- Re-Indexing
- STE
- Verification result

Specification

Implementation
Future Work

Adams, Björk, Melham, Seger (Oxford)

Automatic Abstraction in STE

13 November 2007 16 / 16
Future Work

Future Work

Adams, Björk, Melham, Seger (Oxford)

Automatic Abstraction in STE

13 November 2007
Future Work

Future Work

13 November 2007 16 / 16
Backup slides

Backwards Propagation in Action

\( (o, \bar{o}) \)
Backwards Propagation in Action

\[ (o, \bar{o} x) \]
\[ (o, \bar{o} \bar{x}) \]
\[ (o, \bar{o}) \]
Backwards Propagation in Action
Backwards Propagation in Action
Backwards Propagation in Action
Backwards Propagation in Action

Backup slides

\[ (\overline{o \ x}, o \ y) \]
\[ (\overline{o \ y}, \overline{o \ x}) \]
\[ (\overline{o \ x}, o) \]
\[ (o, \overline{o \ x}) \]
\[ (o, \overline{o \ x} \overline{z}) \]
\[ (o, \overline{o \ x} \overline{z}) \]
\[ (o, \overline{\overline{o}}) \]