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
Our Contribution

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

Abstraction in STE

Adams, Björk, Melham, Seger (Oxford)

Automatic Abstraction in STE

13 November 2007 3 / 16
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
Example relation for a 3-input AND-gate

indexing → abstraction

\( \overline{x} \overline{y} \) → 0XX

\( \overline{x} y \) → X0X

\( x \overline{y} \) → XX0

\( xy \) → 111
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
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
Melham-Jones

- General relations
  - expensive quantifications
- Proof of coverage requirement

Modified version

- Specific structure on relations assumed
  - quantifications eliminated
  - proof in the paper
- Coverage by construction
  - proof in the paper
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

![Diagram showing the process of automatic abstraction in STE]

- **Specification** → **Automatic Abstraction** → **Re-Indexing** → **STE** → **Verification result**

**Future Work**

Adams, Björk, Melham, Seger (Oxford)
Future Work

Spec-dependency

Automatic Abstraction → Re-Indexing

STE

Verification result

Specification

Implementation
Future Work

![Diagram showing the process of Automatic Abstraction in STE]

1. Specification
2. Implementation
3. Automatic Abstraction
4. Re-Indexing
5. STE
6. Verification result

Key steps:
- Spec-dependency
- Symbolic Encoding
Future Work

Abstraction Refinement

Spec-dependency
Symbolic Encoding

Automatic Abstraction → Re-Indexing → STE → Verification result

Specification → Implementation
Future Work

Future Work

Adams, Björk, Melham, Seger (Oxford)
Future Work

Specification

Implementation

Abstraction Refinement

Automatic Abstraction

Re-Indexing

STE

GSTE

Spec-dependency

Symbolic Encoding

SAT-Solvers

Verification result