Automatic Abstraction in Symbolic Trajectory Evaluation

Sara Adams

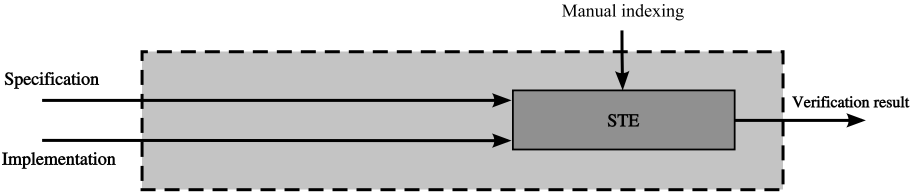
in collaboration with
Magnus Björk, Tom Melham, and Carl-Johan Seger

University of Oxford

13 November 2007

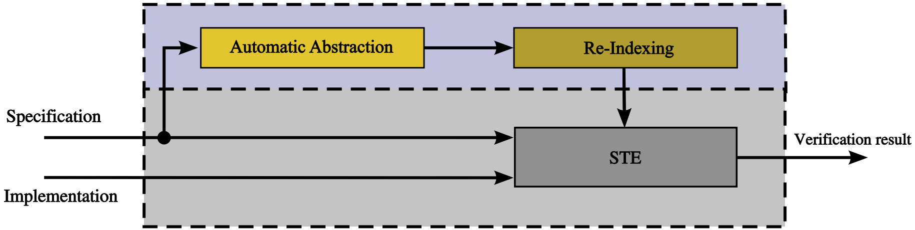
Our Contribution

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

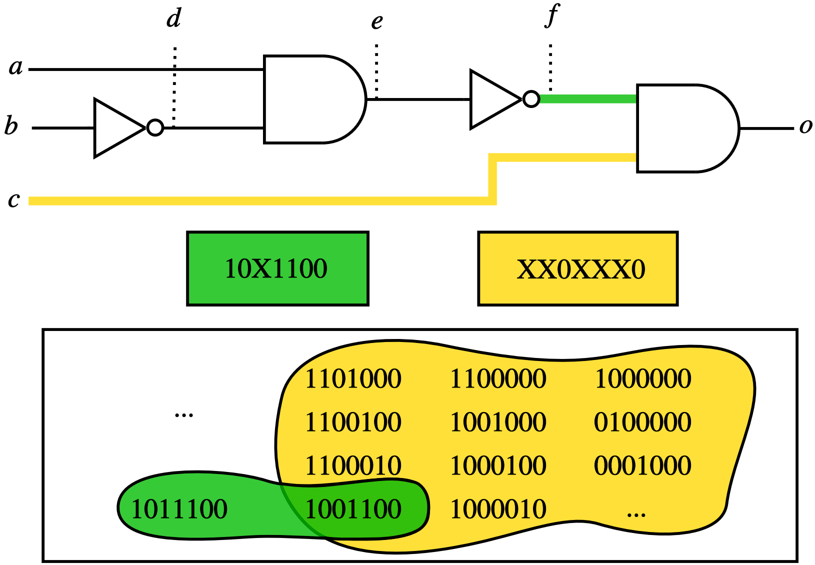


Our Contribution

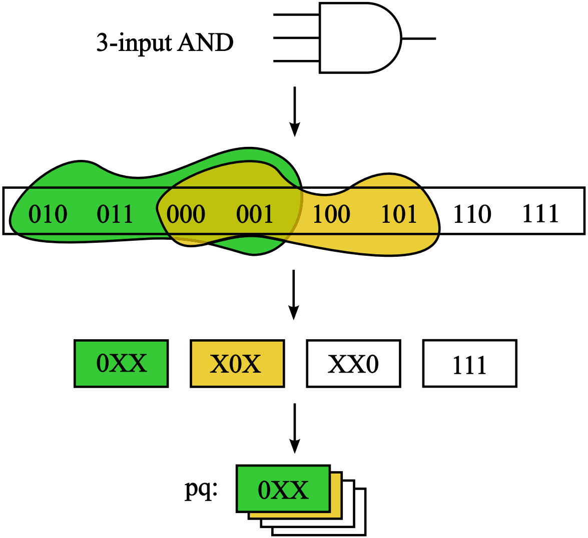
Automatic Discovery of highly non-trivial abstractions
that make verification of circuits possible
that could not be tackled with STE before



Abstraction in STE



Symbolic Indexing



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

Automatic Re-Indexing

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

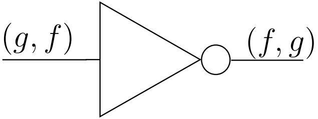
The algorithm

- 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

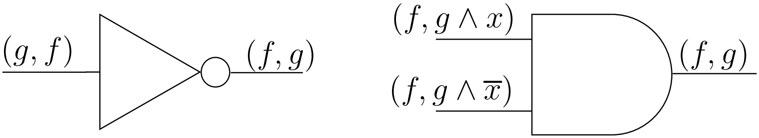


Basic idea

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

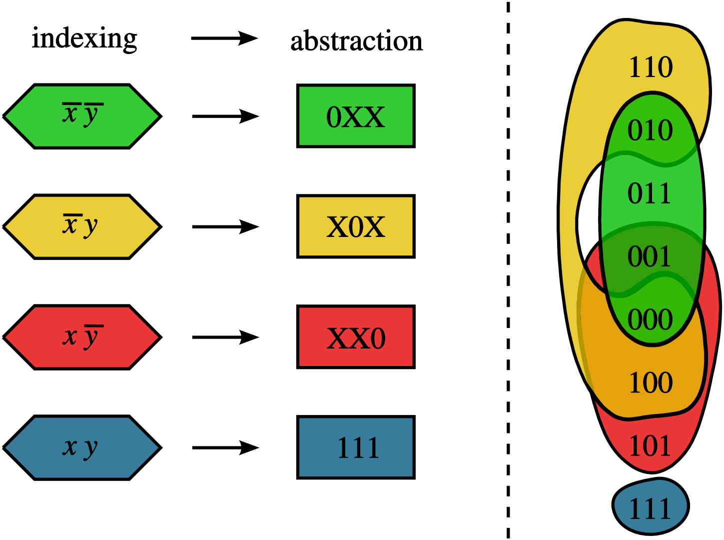


Basic idea

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



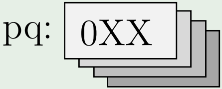
Making it work: Encoding

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



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

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

Content Accessible Memory and Memory

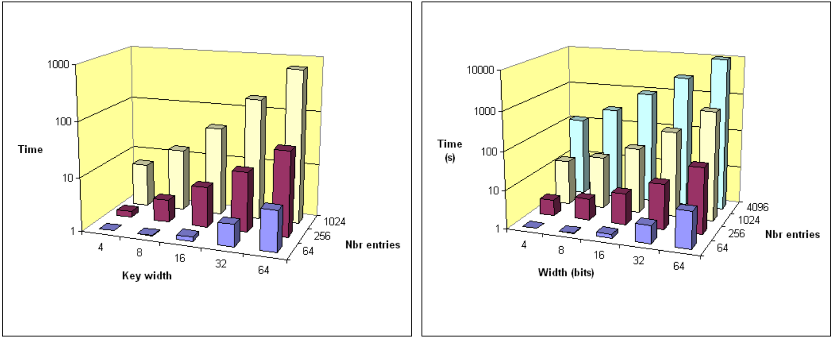


Figure: CAM (left) and Memory (right) verification

- Included: Automatic Abstraction, Re-Indexing, STE run
- Verification not feasible without symbolic indexing

Scheduler

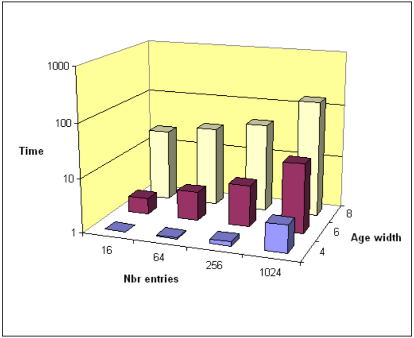
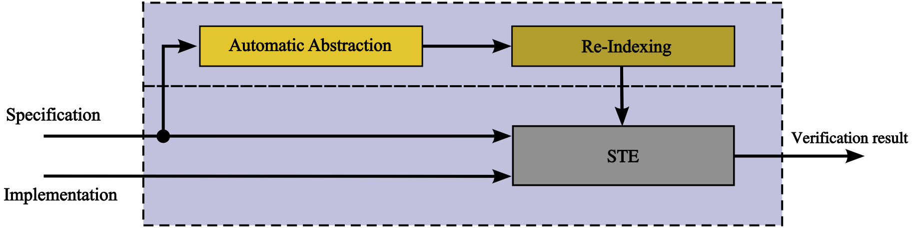


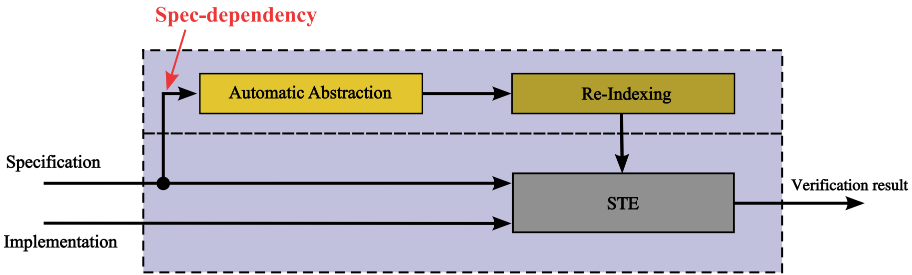
Figure: Scheduler verification

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

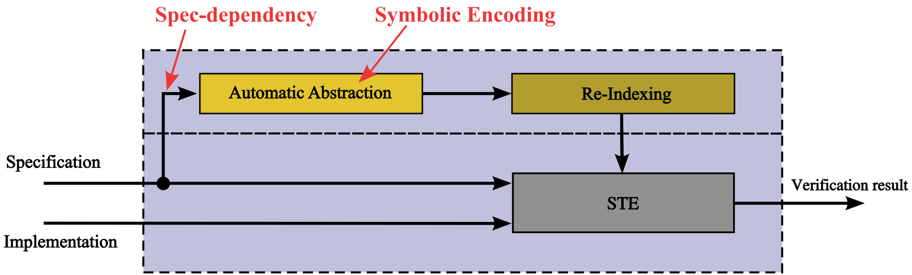
Future Work


Automatic Abstraction
Re-Indexing
Specification
STE
Verification result
Implementation

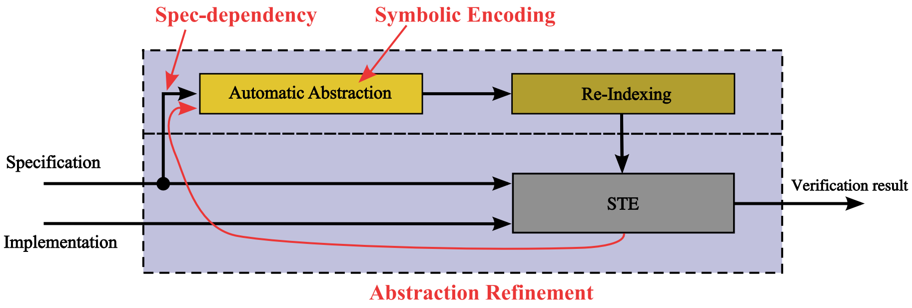
Future Work


Spec-dependency
Automatic Abstraction
Re-Indexing
Specification
STE
Verification result
Implementation

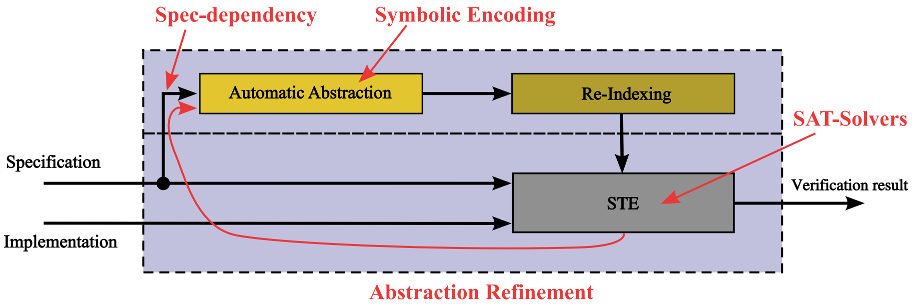
Future Work


Spec-dependency
Symbolic Encoding
Automatic Abstraction
Re-Indexing
Specification
STE
Verification result
Implementation

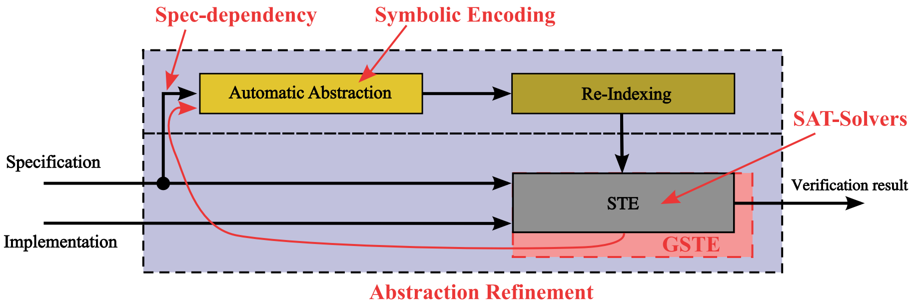
Future Work


Spec-dependency
Symbolic Encoding
Automatic Abstraction
Re-Indexing
Specification
STE
Verification result
Implementation
Abstraction Refinement

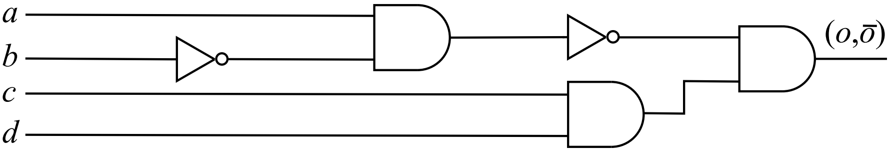
Future Work


Spec-dependency
Symbolic Encoding
Automatic Abstraction
Re-Indexing
SAT-Solvers
Specification
STE
Verification result
Implementation
Abstraction Refinement

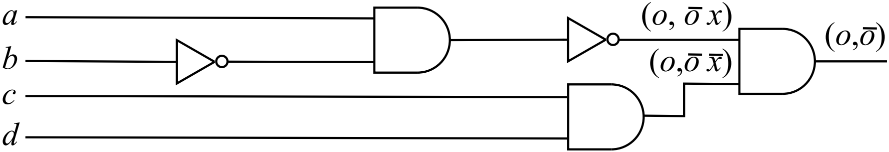
Future Work


Spec-dependency
Symbolic Encoding
Automatic Abstraction
Re-Indexing
SAT-Solvers
Specification
Verification result
STE
Implementation
GSTE
Abstraction Refinement

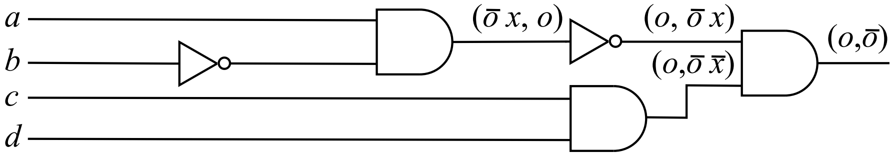
Backwards Propagation in Action



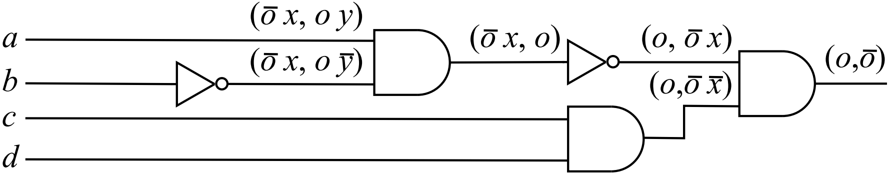
Backwards Propagation in Action



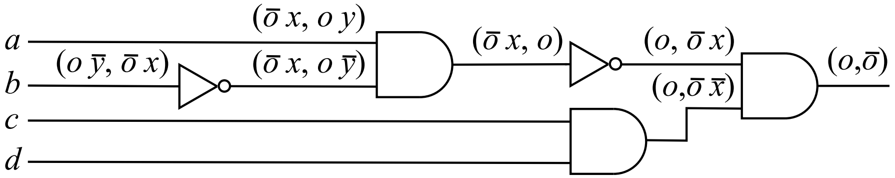
Backwards Propagation in Action



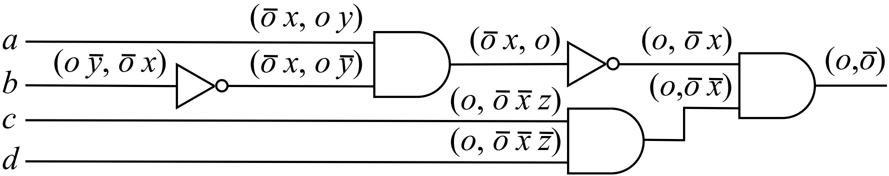
Backwards Propagation in Action



Backwards Propagation in Action



Backwards Propagation in Action



Backwards Propagation in Action