Better Generalization in IC3

Zyad Hassan    Aaron R. Bradley    Fabio Somenzi

Department of Electrical, Computer, and Energy Engineering
University of Colorado at Boulder

Oct 23, 2013
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem</td>
<td>Solution</td>
<td>Results</td>
<td>Analysis</td>
<td>Conclusions</td>
</tr>
<tr>
<td>1</td>
<td>Problem</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Solution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Results</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Conclusions</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

1. Problem
2. Solution
3. Results
4. Analysis
5. Conclusions
IC3 [Bradley 2010, 2011]

- Model checking algorithm for invariance properties
- Attempts to construct an inductive strengthening of the property
- Construction is incremental: derives many simple lemmas
- Lemmas generation either:
  - Results in an inductive strengthening
  - Guides the search to a counterexample trace
- SAT-based: performs many relatively easy SAT queries
Generalization

- Key component of IC3
- Lifts IC3 from explicit to symbolic
- More successful generalization ⇔ Fewer individual states examined

What does IC3 generalize?
Generalization

- Key component of IC3
- Lifts IC3 from explicit to symbolic
- More successful generalization $\iff$ Fewer individual states examined

What does IC3 generalize?
Overview of IC3

- Prove the property by induction:
  - All initial states satisfy the property
  - All successors of good states are good
Overview of IC3

- Prove the property by induction:
  - All initial states satisfy the property
  - All successors of good states are good
Counterexamples to Induction (CTIs): The Troublemakers
Counterexamples to Induction (CTIs): The Troublemakers
Problem

Solution

Results

Analysis

Conclusions

Better Generalization in IC3

8/31
<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
<th>Results</th>
<th>Analysis</th>
<th>Conclusions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Better Generalization in IC3</td>
<td>8/31</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What does IC3 generalize?

A state is unreachable within $k$ steps to
A set of states is unreachable within $k$ steps
How does generalization work?

For each state-bit:

- Drop bit
- Find the smallest superset of states that have no predecessors outside of it (if exists)
Successful Generalization

```
100 110 011
101
111
CTI
```
Successful Generalization

Hassan, Bradley, Somenzi

Better Generalization in IC3 11/31
Successful Generalization
Successful Generalization
Successful Generalization
Failed Generalization

Hassan, Bradley, Somenzi

Better Generalization in IC3
Failed Generalization

CTI

Hassan, Bradley, Somenzi

Better Generalization in IC3
Failed Generalization

CTI

10—
Failed Generalization
Failed Generalization
Ineffective Generalization
Outline

1. Problem
2. Solution
3. Results
4. Analysis
5. Conclusions
Counterexamples to Generalization (CTGs)
Counterexamples to Generalization (CTGs)
Counterexamples to Generalization (CTGs)
Counterexamples to Generalization (CTG)

- State preventing some generalization (dropping a specific state-bit)
- Unlike CTIs, not necessarily backward reachable
- Blocking CTGs:
  - Backward reachable: if deep, saves IC3 explicit traversal
  - Neither forward nor backward: never addressed by IC3 but could continue to obstruct generalization
Instead of joining CTG with cube, turn attention to CTG
Like CTIs, prove unreachable within $k$ steps
If successful: generalize CTG, re-attempt CTI generalization
If failed: join
ctgDown

- Instead of joining CTG with cube, turn attention to CTG if limit is not exceeded
- Like CTIs, prove unreachable within $k$ steps
- If successful: generalize CTG, re-attempt CTI generalization
- If failed: or exceeded maxCTGs limit, join, reset maxCTGs limit
Resetting Limit After Joins
Resetting Limit After Joins
Resetting Limit After Joins
Resetting Limit After Joins
Resetting Limit After Joins
Resetting Limit After Joins
Resetting Limit After Joins
Resetting Limit After Joins
Outline

1. Problem
2. Solution
3. Results
4. Analysis
5. Conclusions
Experimental Setup

- HWMCC’10+11+12 (beemb substituted by beemf)
- 900s timeout
- Ilmc and ABC
- Light-weight preprocessing
- 5 random seeds
<table>
<thead>
<tr>
<th>Family</th>
<th>Size</th>
<th>Standard</th>
<th></th>
<th>With ctgDown</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Solved</td>
<td>Time (s)</td>
<td>Solved</td>
</tr>
<tr>
<td>139</td>
<td>99</td>
<td>99</td>
<td>2524</td>
<td>99</td>
</tr>
<tr>
<td>6s</td>
<td>120</td>
<td>19</td>
<td>93466</td>
<td>21</td>
</tr>
<tr>
<td>beem</td>
<td>86</td>
<td>48</td>
<td>38149</td>
<td>50</td>
</tr>
<tr>
<td>bob</td>
<td>149</td>
<td>122</td>
<td>25804</td>
<td>120</td>
</tr>
<tr>
<td>intel</td>
<td>60</td>
<td>23</td>
<td>35004</td>
<td>30</td>
</tr>
<tr>
<td>pdt</td>
<td>350</td>
<td>331</td>
<td>19291</td>
<td>336</td>
</tr>
<tr>
<td>other</td>
<td>280</td>
<td>271</td>
<td>11947</td>
<td>274</td>
</tr>
<tr>
<td>Total</td>
<td>1144</td>
<td>913</td>
<td>226790</td>
<td>930</td>
</tr>
<tr>
<td>Family</td>
<td>Size</td>
<td>Solved</td>
<td>Time (s)</td>
<td>Solved</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>--------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>139</td>
<td>99</td>
<td>99</td>
<td>701</td>
<td>99</td>
</tr>
<tr>
<td>6s</td>
<td>120</td>
<td>23</td>
<td>88401</td>
<td>30</td>
</tr>
<tr>
<td>beem</td>
<td>86</td>
<td>51</td>
<td>34098</td>
<td>56</td>
</tr>
<tr>
<td>bob</td>
<td>149</td>
<td>123</td>
<td>24292</td>
<td>124</td>
</tr>
<tr>
<td>intel</td>
<td>60</td>
<td>23</td>
<td>35665</td>
<td>26</td>
</tr>
<tr>
<td>pdt</td>
<td>350</td>
<td>329</td>
<td>22162</td>
<td>333</td>
</tr>
<tr>
<td>other</td>
<td>280</td>
<td>270</td>
<td>12591</td>
<td>274</td>
</tr>
<tr>
<td>Total</td>
<td>1144</td>
<td>916</td>
<td>218906</td>
<td>943</td>
</tr>
</tbody>
</table>
Outline

1. Problem
2. Solution
3. Results
4. Analysis
5. Conclusions
<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
<th>Results</th>
<th>Analysis</th>
<th>Conclusions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm reduction in length of explicit backward search</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understand effect on various IC3 metrics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Depth of CTGs vs. CTIs

![Graph showing the relationship between Average CTI Depth and Average CTG Depth]

- **Worse Performance** marked with +
- **Better Performance** marked with ×

Hassan, Bradley, Somenzi

Better Generalization in IC3
Effect on Maximum Depth of Priority Queue

IImc with ctgDown

Worse Performance
Better Performance

Hassan, Bradley, Somenzi

Better Generalization in IC3
Effect on Average Clause Size

Hassan, Bradley, Somenzi
Better Generalization in IC3
Outline

1. Problem
2. Solution
3. Results
4. Analysis
5. Conclusions
Conclusions

- Useful to divert IC3’s attention to address reason for failure of generalization
- Not too aggressive handling of CTGs so as not to lose property focus
- Decreases depth of explicit search
The End

Thank you.