Towards Using Multiple Counterexamples for Abstraction Refinement
Ákos Hajdu, Zoltán Micskei

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group
http://inf.mit.bme.hu/en

Context

<table>
<thead>
<tr>
<th>Theta Framework</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Framework for abstraction refinement-based algorithms</td>
<td>Formalisms and language front-ends</td>
</tr>
<tr>
<td>• Generic, modular, configurable</td>
<td>Transition systems</td>
</tr>
<tr>
<td>• Easy development, evaluation and combination of algorithms</td>
<td>Control flow automata</td>
</tr>
<tr>
<td>• Support for various formalisms</td>
<td>Timed automata</td>
</tr>
<tr>
<td>• Applicable for systems with different aspects (e.g. CPS)</td>
<td>SMT solver interface</td>
</tr>
<tr>
<td>• Open source</td>
<td></td>
</tr>
<tr>
<td>github.com/FTSRG/theta</td>
<td></td>
</tr>
</tbody>
</table>

Research Question

Considering multiple counterexamples for abstraction refinement:
overhead ↔ better refinements (?)

Preliminary Results

Experiments on SV-COMP, HWMCC, PLC models identified two kinds of counterexample structures

<table>
<thead>
<tr>
<th>Multiple paths, single erroneous state</th>
<th>Multiple paths, multiple erroneous states</th>
</tr>
</thead>
</table>

Refinement ideas

• Prune path in prefix/suffix: no benefit
• Prune path in middle: eliminate all counterexamples in a single iteration
  • Fewer but larger iterations
  • Explore k counterexamples → configurable
• Prefer strategy that prunes closest to the initial state
• Calculate refinement for each path and determine (coarsest) common precision eliminating all counterexamples