Checking Safety by Inductive Generalization of Counterexamples to Induction

Aaron R. Bradley and Zohar Manna
Stanford University

(Aaron is visiting EPFL and will be at CU Boulder)
Benchmark: intell_005
Solved: vis-grab (12 minutes, 178MB)

Our time: **11 seconds** (1 process)
Our memory: **13MB**

---

Proved

Time: **11 (1)**

VmPeak: 12820 kB

(Source: HWMCC'07)
Proved
Time: 285 (4)
VmPeak: 91748 kB
ID: 979581
#latch vars: 350
#coi vars: 182
[1 1 0 0 0% 0% 0% 0% 0% 0]
(1692 | !11354 | !11388)
[2 1 0 1 14% 22% 66% 17% 34]
(1706 | !1702 | !11388)
[3 1 0 5 22% 29% 64% 14% 68]
(1810 | 1874 | !1882 | !11388)
[4 1 0 18 33% 40% 62% 11% 136]
(1780 | 11102 | 11166 | !1772 | !11066 | !11150 | !11388)
[5 1 0 32 43% 45% 58% 8% 233]
...

[175 2 93 1167 66% 49% 50% 2% 12166]
(1800 | 1806 | !11056 | !11388)
[176 1 94 1176 66% 49% 51% 2% 12249]
(11086 | 11090 | !11388)
[177 1 97 1177 66% 49% 51% 2% 12315]
[178 2 98 1178 66% 49% 50% 2% 12358]
Proved
Time: 49 (2)
VmPeak: 29204 kB
Parallel Scaling
ID: 962250
#latch vars: 1307
#coi vars: 608
[1 1 0 0 0% 0% 0% 0% 0]
(12606 | !15154 | !15216)
[2 1 0 3 24% 21% 69% 11% 34]
(!12616 | !12612 | !15216)
[3 1 0 5 31% 27% 61% 10% 57]
(14430 | !12616 | !15216)
[4 1 0 14 42% 33% 55% 8% 100]
(12616 | !12634 | !15216)
[5 1 0 18 45% 35% 54% 7% 122]
...
[238 1 0 1813 82% 47% 52% 0% 14661]
(14426 | 14806 | !13680 | !15216)
[239 1 0 1821 82% 47% 52% 0% 14732]
(13554 | 15018 | 15046 | !15216)
[240 1 0 1828 82% 47% 52% 0% 14800]
(!15114 | !15110 | !15216)
[241 1 0 1834 82% 47% 52% 0% 14856]
Proved
Time: 439 (4)
VmPeak: 37752 kB
Other hard instances from HWMCC'07

spec10-and-env (AMBA)
  8 processes: 1.5 hours, 900MB/process

nusmv.reactor^2.C (TIP)
  1 process: 26 minutes, 22MB
  8 processes: 4 minutes, 19MB/process

nusmv.reactor^6.C (TIP)
  1 process: 43 minutes, 30MB
  8 processes: 5 minutes, 19MB/process

Not a “magic bullet”: utterly fails on
cmu.dme[1/2].Beijk.bs*,...
But perhaps a promising approach?

Different set of benchmarks in paper (PicoJava II).
# The Verification Team Analogy

**Goal**: Inductive strengthening of property

<table>
<thead>
<tr>
<th>Verification Team</th>
<th>Inductive Generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Individuals</td>
<td>1. Processes</td>
</tr>
<tr>
<td>2. Lemmas</td>
<td>2. Inductive Clauses</td>
</tr>
<tr>
<td>3. Property</td>
<td>3. Property</td>
</tr>
</tbody>
</table>

**Lemma**: Summary of *observation* and *proof*
Lemma: Inductive Clause

1. Counterexample to induction:

State $s$: \[ \neg l_{2606} \land \ldots \land \neg l_{5154} \land \ldots \land \neg l_{5216} \]

Clause $\neg s$: \[ l_{2606} \lor \ldots \lor \neg l_{15154} \lor \ldots \lor \neg l_{15216} \]

No counterexample? Then property is valid.
Lemma: Inductive Clause

2. Minimal inductive subclause:

Original Clause $\sim$s:

\[
\begin{array}{ccc}
12606 & | & \ldots & | & !15154 & | & \ldots & | & !15216 \\
\end{array}
\]

608 literals. Inductive? Maybe, maybe not.

Minimal Inductive Subclause:

\[
\begin{array}{ccc}
12606 & | & !15154 & | & !15216 \\
\end{array}
\]

3 literals (informative!). Inductive relative to property and previous clauses.
Inductive Generalization

Clause \( \neg s \): \(12606 \mid \ldots \mid !l5154 \mid \ldots \mid !15216\)

**Maximal** inductive subclause:
- Unique.
- Best approximation of computing preimage to fixpoint.
- **Weak**: Excludes “only” states that can reach \(s\).

**Minimal** inductive subclause:
- Not unique.
- Minimal: Strict subclauses are not inductive.
- **Strong**: Also excludes many states that cannot reach \(s\).

Inductive explanation of why \(s\) and similar states are unreachable.
Discovery of MI Subclause

[1 1 0 0 0% 0% 0% 0% 0% 0]
(12606 | !15154 | !15216)

[2 1 0 3 24% 21% 69% 11% 34]
(!12616 | !12612 | !15216)

[3 1 0 5 31% 27% 61% 10% 57]
(14430 | !12616 | !15216)

[4 1 0 14 42% 33% 55% 8% 100]
(12616 | !12634 | !15216)

[5 1 0 18 45% 35% 54% 7% 122]

608 literals.
But <100 SAT problems/iteration.
Discovery of MI Subclause

Many “easy” SAT queries.

1. $O(n)$ SAT queries to find maximal IS $c_1$. 
   
   In practice: many fewer than $n$

2. $O(m \lg n)$ SAT queries to find “small” $m$-literal inductive subclause $c_2$ of $c_1$. 
   
   In practice: $m$ is very small

3. Brute force to guarantee minimality. 
   
   In practice: Algorithm 2 minimizes effects
Related Work

- Interpolation-based model checking [McMillan]
- CEGAR (Jain et al., Clarke et al., ...)
  
  Abstract transition relation

- BMC, k-induction [Biere et al., Sheeran et al., ...]
  Reduce to large SAT/QBF queries.

- Strengthening in k-induction
  [de Moura et al., Vimjam et al., Awedh et al., ...]
  Based on preimage of counterexample.
  Weak, so k-induction is main principle.
Ongoing & Future Work


2. Combine with BMC for better debugging. Add clauses to BMC SAT query \textit{online}.

3. Other types of lemmas?

Conclusions

- **Principle**: Iterative discovery of lemmas.  
  Control resource usage.  
  Run in parallel.

- **Principle**: Use induction to generalize.

- **Mechanism**:  
  Fast discovery of minimal inductive subclauses.

Questions? Comments?