Implementing Termination Analysis using Configurable Software Verification

Sebastian Ott
Termination

- No infinite execution
- Liveness property
- Important property of programs:
  - partial correctness $\land$ termination $\Rightarrow$ total correctness
- Undecidable in general
LassoRanker

- Java library from Ultimate Automizer
- Synthesis of
  - Termination arguments
  - Non-termination arguments
- Template based approach
- SMT solver as back-end
- Lasso as input
Lasso

• Simple loop program

\((x', x) \in \text{Loop} \iff A \begin{pmatrix} x' \\ x \end{pmatrix} + b \leq 0\)

• SMT formula in DNF
Composition of Termination Arguments

• *Loop* is well-founded if *Loop* \( \subseteq T \) and *T* is well-founded.
• Disjunctively well-founded relation *R* \( \subseteq T_1 \cup T_2 \ldots \)
• *R* is well-founded if its transitive hull is disjunctively well-founded.
Termination Algorithm

TerminationCPA + safety analysis → LassoBuilder

counterexample

ranking relation + invariants

LassoRanker

lassos
TerminationCPA

- Searches for potentially non-terminating lassos
- Separation of stem and loop
- Program instrumentation at Honda
  - Stem-loop-transition: \( x' = x; y' = y; \)
  - Loop head --[! ranking relation] \( \rightarrow \) error location
- WrapperCPA
- ARGCPA – TerminationCPA – CompositeCPA
Restrictions and Challenges

- No support for recursion
- Unbounded arrays
- Encoding of termination arguments
  - Linear combination of pointers
  - Array cells: $a'[i] > a[i] \land a'[i] > 0$
- Number of disjunctions in lasso formulas
  - Pointer
  - $a \neq b \rightarrow (a < b) \lor (a > b)$
Evaluation

• Termination Algorithm + Predicate Analysis
• Participants of SV-COMP 2016
  • AProVE
  • SeaHorn
  • Ultimate Automizer
• 733 loop programs
• Limitations
  • 2 CPU cores
  • 900 s CPU time
  • 15 GB memory
## Evaluation

<table>
<thead>
<tr>
<th></th>
<th>AProVE</th>
<th>CPAchecker</th>
<th>SeaHorn</th>
<th>Ultimate Automizer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRUE (569)</strong></td>
<td>278</td>
<td>272</td>
<td>259</td>
<td>430</td>
</tr>
<tr>
<td><strong>FALSE (136)</strong></td>
<td>71</td>
<td>60</td>
<td>82</td>
<td>111</td>
</tr>
<tr>
<td><strong>incorrect results</strong></td>
<td>3</td>
<td>1</td>
<td>46</td>
<td>0</td>
</tr>
<tr>
<td><strong>∅ CPU time</strong></td>
<td>409 s</td>
<td>339 s</td>
<td>170 s</td>
<td>134 s</td>
</tr>
<tr>
<td><strong>∅ memory</strong></td>
<td>2870 MB</td>
<td>1600 MB</td>
<td>64,8 MB</td>
<td>1150 MB</td>
</tr>
<tr>
<td><strong>∅ CPU time (correct results)</strong></td>
<td>45,8 s</td>
<td>45,6 s</td>
<td>12,7 s</td>
<td>33,1 s</td>
</tr>
<tr>
<td><strong>∅ memory (correct results)</strong></td>
<td>1300 MB</td>
<td>596 MB</td>
<td>40,0 MB</td>
<td>528 MB</td>
</tr>
</tbody>
</table>
Evaluation

![Graph showing CPU time (s) vs. n-th fastest result for different tools: AProVE, CPAchecker, SeaHorn, and Ultimate Automizer.]
Evaluation (without pointers)
Future Work

• More types of termination arguments
• Other tool for construction of (non-)termination arguments
• Better support of arrays
• Counterexample check
• Validation of witnesses
Conclusion

• Termination analysis in CPAchecker
• Based on the CPA concept
• Good result on programs without pointers
• Construction of lassos is inefficient for pointers
Questions?