Component-based CEGAR
(under review)
Dirk Beyer, Jan Haltermann,
Thomas Lemberger, Heike Wehrheim
01.10.2021
Motivation: Classic CEGAR

Abstract-Model Exploration

Task: $P, \varphi$

Precision Refinement

Feasibility Check

Feasibility Check

Precision Refinement

Program correct

Program incorrect

Infeasible counterexample

Potential counterexample

Precision increment

Component-based CEGAR — (under review)
Dirk Beyer, Jan Haltermann, Thomas Lemberger, Heike Wehrheim, —
Motivation: Classic CEGAR - Problem

Problem:
- Many tools employ CEGAR (statefull)
- Common underlying schema
- New idea ⇒ New Implementation
Component-based CEGAR (C-CEAR)

Defined components and interfaces:
- Components (stateless):
  - Abstract Model Explorer
  - Feasibility Checker
  - Precision Refiner
- Interfaces (existing formats):
  - Violation Witness
  - Path Witness
  - Invariant Witness
- Construction for off-the-shelf components

Defined components and interfaces:
- Components (stateless):
  - Abstract Model Explorer
  - Feasibility Checker
  - Precision Refiner
- Interfaces (existing formats):
  - Violation Witness
  - Path Witness
  - Invariant Witness
- Construction for off-the-shelf components
Usage of Off-the-Shelf Components - Model Explorer

Component-based CEGAR — (under review)
Dirk Beyer, Jan Haltermann, Thomas Lemberger, Heike Wehrheim, —
Usage of Off-the-Shelf Components - Feasibility Checker

Component-based CEGAR — (under review)
Dirk Beyer, Jan Haltermann, Thomas Lemberger, Heike Wehrheim, —
Usage of Off-the-Shelf Components - Precision Refiner
Implementation

CPAchecker
(predicateAnalysis w/o. Interpol)

Abstract-Model
Explorer

Precision
Refiner

Feasibility
Checker

Path
Witness

Invariant
Witness

Violation
Witness

CPAchecker
(predicateAnalysis + Interpol),
UAUTOMIZER

program correct

program incorrect

Component-based CEGAR — (under review)
Dirk Beyer, Jan Haltermann, Thomas Lemberger, Heike Wehrheim, —
Evaluation

Research Questions:
- RQ1: Overhead of a component-based approach (with predmap)
- RQ2: Cost using existing formats
- RQ3: Use of off-the-shelf components

Dataset: SV-BENCHMARKS (4510 tasks), SV-COMP setting
RQ1: Overhead of component based design

<table>
<thead>
<tr>
<th></th>
<th>correct</th>
<th>overall</th>
<th>incorrect</th>
<th>alarm</th>
<th>proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pred</td>
<td>2 183</td>
<td>1 343</td>
<td>840</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>C-Pred</td>
<td>2 105</td>
<td>1 297</td>
<td>808</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

(Using predmap as exchange format)
RQ1: Overhead of component based design

Comparison of CPU time for Pred and C-Pred

Median factor of run-time increase by C-Pred compared to Pred.
Overall median increase is 3.2
RQ2: Cost of Standardized Formats

<table>
<thead>
<tr>
<th></th>
<th>correct overall</th>
<th>correct proof</th>
<th>correct alarm</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-Pred</td>
<td>2 105</td>
<td>1 297</td>
<td>808</td>
</tr>
<tr>
<td>C-PredWit</td>
<td>1 573</td>
<td>978</td>
<td>595</td>
</tr>
</tbody>
</table>

- Effectiveness reduces by 25%
- Reasons:
  - Not all predicates discovered are exported
  - No loop unrollings in witness

Comparison of CPU time for C-Pred and C-PredWit
RQ3: C-Cegar using different components

RQ 3.1: C-PredWit + different feasibility checker (with precision refiner CPACHECKER)

<table>
<thead>
<tr>
<th></th>
<th>overall</th>
<th>proof</th>
<th>unique</th>
<th>alarm</th>
<th>unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPACHECKER</td>
<td>1573</td>
<td>978</td>
<td>61</td>
<td>595</td>
<td>317</td>
</tr>
<tr>
<td>FSHELL-WITNESS2TEST</td>
<td>612</td>
<td>515</td>
<td>0</td>
<td>97</td>
<td>56</td>
</tr>
<tr>
<td>UAUTOMIZER</td>
<td>1225</td>
<td>918</td>
<td>1</td>
<td>307</td>
<td>20</td>
</tr>
</tbody>
</table>
RQ3: C-Cegar using different components

RQ 3.2: C-PredWit + different precision refiner (with feasibility checker CPACHECKER)

<table>
<thead>
<tr>
<th></th>
<th>overall</th>
<th>proof</th>
<th>unique</th>
<th>correct</th>
<th>alarm</th>
<th>unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPACHECKER</td>
<td>1573</td>
<td>978</td>
<td>301</td>
<td>595</td>
<td></td>
<td>303</td>
</tr>
<tr>
<td>UAUTOMIZER</td>
<td>1016</td>
<td>716</td>
<td>41</td>
<td>300</td>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>
Summary - C-CEGAR

- Clear defined components and interfaces
- Implementation in COVERITeam
- Evaluation show advantages:
  - Same expressiveness (with lower efficiency (3.2))
  - Existing formats can be used
  - Usage of existing off-the-shelf components
- Starting point for further development