AUGMENTING PREDICATE ANALYSIS
WITH AUXILIARY INVARIANTS

Thomas Stiegmaier
September 23, 2016
University of Passau
Motivation

Predicate Analysis

- SMT-based
- Abstraction of program, computed from a set of predicates $\pi$
- CEGAR for refining $\pi$
- Craig interpolation for discovering precision increments
Predicate Analysis

- SMT-based
- Abstraction of program, computed from a set of predicates $\pi$
- CEGAR for refining $\pi$
- Craig interpolation for discovering precision increments

precision $\pi$

path formula $\phi$

abstraction formula $\psi$

Abstraction computation

$\psi' = (\phi \land \psi)^\pi$

$\phi = \text{TRUE}$
Generating Invariants

- Several tools available: INVGEN, DAIKON
- Often not SMT-based
Generating Invariants

- Several tools available: INVGEN, DAIKON
- Often not SMT-based

Use invariants in other analyses

- Add new (helpful) information to a predicate analysis
- Speed up the analysis
  - less refinements
  - less dependent on interpolants
\[ \psi' = (\phi \land \psi \land \text{INV})^{\pi} \]
\[ \psi' = (\phi \land \psi)^\pi \cup \{\text{INV}\} \]
\[ \psi' = (\phi \land \psi)^\pi \land \text{INV} \]
\[ \psi' = (\phi \land \psi \land \text{INV})^{\pi \cup \{\text{INV}\}} \land \text{INV} \]
Location ②

- Abstraction location
- $\pi = \{ i < 10 \}$
- invariant $i = 2$
### Predicate Analysis — Example

<table>
<thead>
<tr>
<th>Strategy</th>
<th>New Abstract State</th>
<th>Possible Transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Inv</td>
<td>(i &lt; 10, \text{ TRUE})</td>
<td>2 → 3, 2 → 4</td>
</tr>
<tr>
<td>Prec</td>
<td>(i = 2 \land i &lt; 10, \text{ TRUE})</td>
<td>2 → 3</td>
</tr>
<tr>
<td>PF</td>
<td>(i &lt; 10, \ i = 2)</td>
<td>2 → 3</td>
</tr>
<tr>
<td>AF</td>
<td>(i = 2 \land i &lt; 10, \text{ TRUE})</td>
<td>2 → 3</td>
</tr>
<tr>
<td>Prec + PF</td>
<td>(i = 2 \land i &lt; 10, \ i = 2)</td>
<td>2 → 3</td>
</tr>
<tr>
<td>Prec + AF</td>
<td>(i = 2 \land i = 2 \land i &lt; 10, \text{ TRUE})</td>
<td>2 → 3</td>
</tr>
<tr>
<td>PF + AF</td>
<td>(i = 2 \land i &lt; 10, \ i = 2)</td>
<td>2 → 3</td>
</tr>
<tr>
<td>Prec + PF + AF</td>
<td>(i = 2 \land i = 2 \land i &lt; 10, \ i = 2)</td>
<td>2 → 3</td>
</tr>
</tbody>
</table>
Auxiliary Invariants

- fast computation
- high success rate
- useful invariants

→ no negative impact on the main analysis
Auxiliary Invariants

- fast computation
- high success rate
- useful invariants
  → no negative impact on the main analysis
Auxiliary Invariants — Lightweight Heuristics

PredicateCPA specific

- Inductive weakening of path formulas
- Checking conjuncts of path formulas on invariance
- Checking interpolants on invariance
Auxiliary Invariants — Lightweight Heuristics

PredicateCPA specific

- Inductive weakening of path formulas
- Checking conjuncts of path formulas on invariance
- Checking interpolants on invariance

Applicable to other analyses

- Path invariants
Auxiliary Invariants — Sequential Analyses

Compute invariants from reached sets of earlier analyses

- Bounded Predicate Analysis
- Unbounded Predicate Analysis
- Predicate Analysis with Invariants

Precision

Reached Set
Auxiliary Invariants — Parallel Analyses

- $k$-induction uses concurrently running invariant generation
  - not usable for other concurrent analyses
  → new CPACHECKER feature

- Algorithm for executing several analyses in parallel
- Communication between analyses via reached sets
One manager class
  - Exposes general methods for retrieving and generating invariants
  - Hides exact configuration
  - Lazy computation of invariants during refinement
  - Mixing generation and usage strategies possible
One manager class
- Exposes general methods for retrieving and generating invariants
- Hides exact configuration
- Lazy computation of invariants during refinement
- Mixing generation and usage strategies possible

Two users
- Refinement (precision increment)
- PrecisionAdjustment (path -and abstraction formula)
2.6 GHz Octa Core CPUs (Intel E5-2650 v2)
8 GB memory
300 s or 600 s CPU time
trunk r23084
Measured with BENCHEXEC

3488 verification tasks taken from SV-COMP’16
• Inductive weakening and checking conjuncts of path formulas failed
• Checking interpolants on invariance is very slow due to prefix generation
• Path invariants are too slow overall, but good on tasks in the loops category
Evaluation — Path Invariants

- Two configurations:
  - Predicate Analysis + Path Invariants with InvariantsCPA
  - Predicate Analysis + Path Invariants with PolicyCPA
```c
int main() {
    int i;
    for (i = 0; i < 1000000; i++) ;
    assert (i == 1000000);
    return 0;
}
```

- Interpolation unrolls the loop
- ✓ found invariant: $i = 1000000$ for location of `assert` call
**Evaluation — Parallel Analyses**

- Combination of:
  - An analysis with the PredicateCPA, and
  - An analysis with the InvariantsCPA (continuously-refined)

- 600 s CPU time (300 s per analysis)

- 7 configurations: abs, prec, path, abs-path, ...

- 3 baselines
  - 300 s and 600 s predicate analyses
    base300, base600
  - 600 s parallel analysis without invariant generation
    basePar
EVALUATION — PARALLEL ANALYSES

![Graph showing CPU time (s) vs. n-th fastest correct result]

- base600
- base300
- basePar
- async-abs

CPU time (s)

n-th fastest correct result
all baselines are strictly worse than configurations with invariants

- async-abs is the best configuration
  - 4 % better than base600
  - 8 % better than base300
  - 3 % better than basePar
  → wall time is comparable to base300

- async-prec is slow

- async-prec-path almost as good as async-abs
Combination of:
- bounded predicate analysis (100 s)
- unbounded predicate analysis without refinement (100 s)
- predicate analysis using invariants (300 s)

7 configurations (invariants): abs, prec, path, abs-path, ...

1 configuration (only precision): restart2

2 baselines, 300 s and 600 s predicate analyses base300, base600
EVALUATION — SEQUENTIAL ANALYSES
CONCLUSION & OUTLOOK

- Heuristics for invariant generation need more time than expected
- More intelligent heuristics needed:
  - When should invariants be generated
  - Filtering of found invariants

Combination of analyses increases performance

Performance is even better if the analyses communicate → Aim: Make communication easier usable
CONCLUSION & OUTLOOK

- Heuristics for invariant generation need more time than expected
- More intelligent heuristics needed:
  - When should invariants be generated
  - Filtering of found invariants

✓ Combination of analyses increases performance
✓ Performance is even better if the analyses communicate
→ Aim: Make communication easier usable
Table 1: Details on analyses using path invariants for generating auxiliary invariants and their baseline

<table>
<thead>
<tr>
<th></th>
<th>correct</th>
<th>wrong</th>
<th>Invariants (equal)</th>
<th>CPU time (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>safe</td>
<td>unsafe</td>
<td>time (h)</td>
<td>tries</td>
</tr>
<tr>
<td>base300</td>
<td>1 391</td>
<td>553</td>
<td>27</td>
<td>149</td>
</tr>
<tr>
<td>path-inv</td>
<td>1 327</td>
<td>519</td>
<td>27</td>
<td>2.36</td>
</tr>
<tr>
<td>path-policy</td>
<td>1 337</td>
<td>529</td>
<td>27</td>
<td>3.84</td>
</tr>
<tr>
<td>400s-inv</td>
<td>1 364</td>
<td>575</td>
<td>27</td>
<td>196</td>
</tr>
<tr>
<td>400s-policy</td>
<td>1 371</td>
<td>576</td>
<td>27</td>
<td>196</td>
</tr>
</tbody>
</table>
Table 2: A selection of tasks and their results with path invariants

<table>
<thead>
<tr>
<th>file name</th>
<th>path-inv</th>
<th>path-policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>loop-acceleration/array_true-unreach-call3.i</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>loop-acceleration/functions_true-unreach-call1.i</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>loop-acceleration/nested_true-unreach-call1.i</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>loop-acceleration/simple_true-unreach-call1.i</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>loop-new/count_by_1_true-unreach-call.i</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>loop-new/count_by_1_variant_true-unreach-call.i</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>loop-new/count_by_nondet_true-unreach-call.i</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Table 3: Details on all parallel analyses using invariants and their baselines

<table>
<thead>
<tr>
<th></th>
<th>correct</th>
<th></th>
<th>wrong</th>
<th></th>
<th>Main Succ</th>
<th>Wall time (h)</th>
<th>CPU time (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>correct</td>
<td>all</td>
<td>equal</td>
</tr>
<tr>
<td>base300</td>
<td>1391</td>
<td>553</td>
<td>0</td>
<td>27</td>
<td>1944</td>
<td>128</td>
<td>13.8</td>
</tr>
<tr>
<td>base600</td>
<td>1434</td>
<td>588</td>
<td>0</td>
<td>27</td>
<td>2022</td>
<td>240</td>
<td>13.9</td>
</tr>
<tr>
<td>basePar</td>
<td>1509</td>
<td>541</td>
<td>0</td>
<td>18</td>
<td>1109</td>
<td>152</td>
<td>15.6</td>
</tr>
<tr>
<td>abs</td>
<td>1532</td>
<td>572</td>
<td>0</td>
<td>18</td>
<td>1154</td>
<td>147</td>
<td>14.4</td>
</tr>
<tr>
<td>path</td>
<td>1536</td>
<td>561</td>
<td>1</td>
<td>17</td>
<td>1148</td>
<td>146</td>
<td>14.2</td>
</tr>
<tr>
<td>prec</td>
<td>1526</td>
<td>549</td>
<td>0</td>
<td>18</td>
<td>1108</td>
<td>149</td>
<td>15.3</td>
</tr>
<tr>
<td>prec-path</td>
<td>1525</td>
<td>561</td>
<td>1</td>
<td>17</td>
<td>1111</td>
<td>148</td>
<td>15.1</td>
</tr>
<tr>
<td>abs-path</td>
<td>1528</td>
<td>568</td>
<td>1</td>
<td>18</td>
<td>1148</td>
<td>146</td>
<td>14.4</td>
</tr>
<tr>
<td>prec-abs</td>
<td>1526</td>
<td>557</td>
<td>0</td>
<td>18</td>
<td>1110</td>
<td>149</td>
<td>15.2</td>
</tr>
<tr>
<td>prec-abs-path</td>
<td>1531</td>
<td>551</td>
<td>1</td>
<td>18</td>
<td>1106</td>
<td>148</td>
<td>15.0</td>
</tr>
</tbody>
</table>
### Table 4: Details on all sequential combinations of analyses using invariants and their baselines

<table>
<thead>
<tr>
<th></th>
<th>correct</th>
<th>wrong</th>
<th>Analyses</th>
<th>Wall time (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>true</td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>base300</td>
<td>1391</td>
<td>553</td>
<td>27</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128</td>
</tr>
<tr>
<td>base600</td>
<td>1434</td>
<td>588</td>
<td>27</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>240</td>
</tr>
<tr>
<td>restart2</td>
<td>1420</td>
<td>612</td>
<td>27</td>
<td>1.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>182</td>
</tr>
<tr>
<td>abs</td>
<td>1415</td>
<td>557</td>
<td>27</td>
<td>2.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>201</td>
</tr>
<tr>
<td>path</td>
<td>1416</td>
<td>547</td>
<td>28</td>
<td>2.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>200</td>
</tr>
<tr>
<td>prec</td>
<td>1409</td>
<td>550</td>
<td>27</td>
<td>2.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>202</td>
</tr>
<tr>
<td>prec-path</td>
<td>1409</td>
<td>557</td>
<td>28</td>
<td>2.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>201</td>
</tr>
<tr>
<td>abs-path</td>
<td>1414</td>
<td>555</td>
<td>28</td>
<td>2.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>200</td>
</tr>
<tr>
<td>prec-abs</td>
<td>1407</td>
<td>555</td>
<td>27</td>
<td>2.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>202</td>
</tr>
<tr>
<td>prec-abs-path</td>
<td>1414</td>
<td>552</td>
<td>26</td>
<td>2.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>201</td>
</tr>
</tbody>
</table>