CPA-SymExec:
Efficient Symbolic Execution in CPAchecker

Dirk Beyer and Thomas Lemberger

LMU Munich, Germany
Why you should listen

Interested in symbolic execution? (formal verification, test-case generation, program repair, equivalence checking, ...)
Analyze C programs, implemented in CPAchecker.
But...

Symbolic Execution can provide:

- Exhaustive (formal) verification
- Automatic test-case generation
- Symbolic and concrete program traces

But: It is not efficient (path explosion, SMT solving)

Existing tools:

- Crest/ConTest (dynamic)
- Klee (heuristics)
- Symbiotic (static slicing)
- ...

Our new tool:

- CPA-SymExec: Abstraction with CEGAR (+ heuristics)
Optimizations in CPA-SymExec

CPA-SymExec uses common optimizations:

- SMT result caching (+ subset caching)
- Model re-use
- Minimal SAT checks

And includes new optimizations:

- Computation of definite assignments
- Simplification of symbolic identifiers
- Counterexample-guided Abstraction Refinement (CEGAR)
Symbolic Execution

- Replace concrete test-values with generic symbolic values
- **Symbolic Memory**: Stores (symbolic) value assignments.
- **Path Constraints**: Constrain symbolic values.
Symbolic Execution

- Replace concrete test-values with generic symbolic values
- **Symbolic Memory**: Stores (symbolic) value assignments.
- **Path Constraints**: Constrain symbolic values.

```
1 unsigned char a = ?;
2 unsigned char b = ?;
3 unsigned char c = b + 1;
4 while (a < 100)
5     a++;
6 if (c == b)
7     error ();
```
Symbolic Execution

- Replace concrete test-values with generic symbolic values
- **Symbolic Memory**: Stores (symbolic) value assignments.
- **Path Constraints**: Constrain symbolic values.

```c
1 unsigned char a = ?;
2 unsigned char b = ?;
3 unsigned char c = b + 1;
4 while (a < 100)
5    a++;  
6 if (c == b)
7     error();
```

- Necessary to track (thanks CEGAR!):
  - Symbolic memory of `b` and `c`.
  - Constraint `c ≠ b`
Symbolic Execution

- Replace concrete test-values with generic symbolic values
- **Symbolic Memory**: Stores (symbolic) value assignments.
- **Path Constraints**: Constrain symbolic values.

```c
1 unsigned char a = ?;
2 unsigned char b = ?;
3 unsigned char c = b + 1;
4 while (a < 100)
  5   a++;
6 if (c == b)
  7   error();
```

- Necessary to track (thanks CEGAR!):
  - Symbolic memory of `b` and `c`.
  - Constraint `c ≠ b`
Counterexample-guided Abstraction Refinement

- **Program unsafe**
  - Check counterexample
    - counterexample feasible
    - counterexample spurious
  - Refine abstraction
  - restart

- **Program safe**
  - no counterexample found
  - Verify program
    - initial abstraction

- **Counterexample**
  - found

Thomas Lemberger
LMU Munich, Germany
Counterexample-guided Abstraction Refinement

- **Program unsafe**: counterexample feasible
  - counterexample found → Check counterexample
  - counterexample spurious → Refine abstraction

- **Initial abstraction** → Verify program
  - no counterexample found → Program safe
  - counterexample found → Refine abstraction

**Abstraction → memory and path constraints**
**Check → SymEx with all information**
**Refinement → Craig interpolation**
CPA-SymExec can create:

- Concrete and symbolic target paths
- Executable test cases (condition coverage)
- Interactive, visual reports

```c
1 unsigned char a = ?;
2 unsigned char b = ?;
3 unsigned char c = b + 1;
4 while (a < 100)
5   a++; 
6 if (c <= b)
7   error();
```

Symbolic target path:
- \( a = s_1 \wedge b = s_2 \wedge c = s_2 + 1 \)
- \( s_1 \geq 100 \) \(
  \begin{align*}
    &s_1 \geq 100 \\
    &s_2 + 1 \leq s_2 \\
  \end{align*}
\)
- \( \text{error()} \)

Concrete target path:
- \( a = 100 \wedge b = 255 \wedge c = 0 \)
- \( \text{error()} \)
Experimental Comparison

Find target path to function call/show none exists

5590 sequential programs from SV-COMP

<table>
<thead>
<tr>
<th>Tool</th>
<th>correct TRUE</th>
<th>correct FALSE</th>
<th>incorrect TRUE</th>
<th>incorrect FALSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPA-SymExec</td>
<td>2137</td>
<td>545</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>KLEE</td>
<td>446</td>
<td>899</td>
<td>6</td>
<td>33</td>
</tr>
<tr>
<td>SYMBIOTIC</td>
<td>1 201</td>
<td>848</td>
<td>3</td>
<td>9</td>
</tr>
</tbody>
</table>
Demo
More Information

- Video: https://youtu.be/qoBHTvPKtnw
- Artifact (tool and data): https://zenodo.org/record/1321181
- Technical Session: Tomorrow, Joffre CD, 13:30–15:00
Traditional Symbolic Execution Example

1. `unsigned char a = ?;`
2. `unsigned char b = ?;`
3. `unsigned char c = b + 1;`
4. `while (a < 100)`
   5. `a++;`
6. `if (c == b)`
   7. `error ();`