Steering Symbolic Execution to Less Traveled Paths

You Li

Joint with Zhendong Su, Linzhang Wang, Xuandong Li
Background and Motivations

- Testing is important, but can be ineffective
  - Software is complex with large or infinite state space
  - Manual testing is tedious and ad hoc
  - Random testing is not systematic

- Symbolic execution is promising
  - Systematically explores a program
  - Generates test cases with high coverage
Symbolic Execution

- Uses symbolic values for inputs to explore a program
- Forks at branch conditions
- Follows both directions by updating path constraints
- Solves path constraints to generate test cases
Main Challenges

- Complex constraints
- Path explosion

Goal: Guide Symbolic Execution to Profitable Paths
Key issue: How to guide toward profitable paths?
Less Traveled Paths

- **Benefits**
  - Cover the program better
  - Locate more bugs

- **Difficulties**
  - Define "footprints"
  - Use "footprints" to guide path exploration
Subpath-Guided Path Exploration

- How to define “footprints”?
  - Length-$n$ Subpath Program Spectra

- How to use "footprints" to guide path exploration?
  - Subpath-Guided Search (SGS)
Program Spectra

- Program profiling
  - Counting different program execution events

Profiling of different events provides various program spectra
  - Branch Hit Spectra
  - Branch Count Spectra
  - Complete Path Spectra
  - Path Spectra
  - Path Count Spectra
Length-$n$ Subpath Program Spectra

- Each subpath has $n$ branches
- Contiguous sub-sequences of execution paths
- Varying $n$ leads to a spectrum of modeling precision
- Fills the gap between branch coverage & complete path coverage
Subpath Guided Search (SGS)

- Maintain a structure $\langle \pi_n, f \rangle$
  - $\pi_n$ is a length $n$ subpath
  - $f$ is the frequency of $\pi_n$

- For each execution, track the most recent length-$n$ path segment

- Pick a pending execution with the lowest $f$ to explore next
  - Break ties randomly
main (x, y) {
  if (x > y)
    x = f(x);
  else
    ;
  g (x, y);
  return;
}

int f (a) {
  if (a > 0)
    ABORT;
  else
    return -a;
}

g (a, b) {
  if (a == 0)
    if (b == 0)
      ABORT;
    else
      ;
    else
      print a/b;
  return;
}
Total Number of Paths: 7
Sub-path Frequency

- $S_{0t}, 0 \ast$
- $S_{0f}, 1$
- $S_{0fS_{8t}}, 0 \ast$
- $S_{0fS_{8f}}, 1$

Diagram:

- Entry
- $S_{0t}, 0$
- $S_{0}$
- $S_{5}$
- $S_{6}$
- $S_{0fS_{8t}}, 0$
- $S_{8}$
- $S_{0fS_{8fS_{4}}}$
- $S_{4}$

2013/10/31
Sub-path Frequency

- $S_{0t}, 0$
- $S_{0f}, 1$
- $S_{0f}S_{8t}, 1$
- $S_{0f}S_{8f}, 1$
- $S_{8t}S_{9t}, 0$
- $S_{8t}S_{9f}, 0$
- $S_{8t}S_{9f}, 0$

Flowchart:
- Entry
  - $S_{0t}, 0$ to $S_{0}$
    - $S_{5}$
      - $S_{6}$
    - $S_{8}$
      - $S_{9}$
        - $S_{10}$
      - $S_{8t}S_{9t}, 0$
    - $S_{8t}S_{9f}, 0$
  - $S_{8t}S_{9f}, 0$ to $S_{4}$

Date: 2013/10/31
Sub-path Frequency

- $S_{0t}, 1$
- $S_{0f}, 1$
- $S_{0f}S_{8t}, 1$
- $S_{0f}S_{8f}, 1$
- $S_{8t}S_{9t}, 1$
- $S_{8t}S_{9f}, 1$
- $S_{0t}S_{5t}, 0^*$
- $S_{0t}S_{5f}, 0^*$

2013/10/31
Sub-path Frequency

- $S_{0t}, 1$
- $S_{0f}, 1$
- $S_{0f}S_{8t}, 1$
- $S_{0f}S_{8f}, 1$
- $S_{8t}S_{9t}, 1$
- $S_{8t}S_{9f}, 1$
- $S_{0t}S_{5t}, 0^*$
- $S_{0t}S_{5f}, 1$
- $S_{5f}S_{8t}, 0^*$
- $S_{5f}S_{8f}, 0^*$

Diagram:

- Entry
- $S_0$
- $S_5$
- $S_6$
- $S_7$
- $S_9$
- $S_10$
- $S_4$
**Sub-path Frequency**

- $S_{0t}, 1$
- $S_{0f}, 1$
- $S_{0f}S_{8t}, 1$
- $S_{0f}S_{8f}, 1$
- $S_{8t}S_{9t}, 1^*$
- $S_{8t}S_{9f}, 1^*$
- $S_{0t}S_{5t}, 0^*$
- $S_{0t}S_{5f}, 1$
- $S_{5f}S_{8t}, 1$
- $S_{5f}S_{8f}, 1$

---

2013/10/31
Evaluation: Research Questions

- What impact do different choices of $n$ have?
- Can they be effectively combined?
- How does our strategy compare to existing strategies?
Evaluation Setup

- Implement SGS in KLEE
- Evaluation subjects: GNU core utilities
- Evaluated search strategies
  - Length-n SGS with varying n (n = 1, 2, 4, 8)
  - Existing strategies implemented in KLEE
- Evaluation metrics
  - How well a program is covered?
  - How effective in locating bugs?
KLEE Strategies

- DFS
- Random State
- Random Path
- Non-Uniform Random Selection
  - covnew
  - depth
  - icnt
  - md2u
Program Coverage

- 75 programs (2K - 10K LOC in size)
- Run each strategy for 1 hour
- Output test cases exploring new statements or triggering errors
- Re-execute test cases to measure statement coverage
Coverage Distribution

- 90%-100%
- 80%-90%
- 70%-80%
- 60%-70%
- 60%-

- RSS
- DFS
- RPS
- NURS covernew
- NURS depth
- NURS icnt
- NURS md2u

2013/10/31
Average Coverage (%)
"Best" Counts

<table>
<thead>
<tr>
<th></th>
<th>SGS-1</th>
<th>SGS-2</th>
<th>SGS-4</th>
<th>SGS-8</th>
<th>RSS</th>
<th>DFS</th>
<th>RPS</th>
<th>NURS cn</th>
<th>NURS depth</th>
<th>NURS icnt</th>
<th>NURS md2u</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>25</td>
<td>38</td>
<td>29</td>
<td>24</td>
<td>20</td>
<td>16</td>
<td>23</td>
<td>14</td>
<td>21</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

2013/10/31
Results Recap

- Result 1: SGS yields higher coverage
- Result 2: No uniform best $n$ for SGS
Combined SGS

- Run SGS with length 1, 2, 4, 8 for 15 minutes each
- Combine all the generated test cases
Average Coverage (%)
Coverage Distribution

![Coverage Distribution Chart](chart)

- SGS-1
- SGS-2
- SGS-4
- SGS-8
- Com

- RSS
- DFS
- RPS
- NURS covernew
- NURS depth
- NURS icnt
- NURS md2u

Legend:
- 90%-100%
- 80%-90%
- 70%-80%
- 60%-70%
- 60%-

Date: 2013/10/31
### Average Coverage (%)

<table>
<thead>
<tr>
<th></th>
<th>Avg.</th>
<th>SGS-1</th>
<th>SGS-2</th>
<th>SGS-4</th>
<th>SGS-8</th>
<th>Com</th>
<th>RSS</th>
<th>DFS</th>
<th>RPS</th>
<th>NURS cn</th>
<th>NURS depth</th>
<th>NURS icnt</th>
<th>NURS md2u</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>69.67</td>
<td>72.87</td>
<td>72.35</td>
<td>66.95</td>
<td>80.67</td>
<td>60.56</td>
<td>56.12</td>
<td>68.38</td>
<td>46.87</td>
<td>64.35</td>
<td>51.72</td>
<td>50.5</td>
</tr>
</tbody>
</table>

2013/10/31
## "Best" Counts

<table>
<thead>
<tr>
<th></th>
<th>SGS-1</th>
<th>SGS-2</th>
<th>SGS-4</th>
<th>SGS-8</th>
<th>Com</th>
<th>RSS</th>
<th>DFS</th>
<th>RPS</th>
<th>NURS cn</th>
<th>NURS depth</th>
<th>NURS icnt</th>
<th>NURS md2u</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>25</td>
<td>38</td>
<td>29</td>
<td>24</td>
<td>57</td>
<td>20</td>
<td>16</td>
<td>23</td>
<td>14</td>
<td>21</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

2013/10/31
Result 3: Combined SGS performs uniformly the best
Result 4: SGS yields more bug reports

Result 5: SGS has acceptable overhead
Bug Detection: Killing Mutants

- 40 programs (which produce deterministic output)
- Run each different strategy for 1 hour
- Output all terminated test cases
- Generate mutants of the 40 programs
- Re-execute test cases on both original program and mutants
- Compare their outputs to see if mutants were killed
Average Kill Rate(%)
Total Kill Number

<table>
<thead>
<tr>
<th></th>
<th>SGS-1</th>
<th>SGS-2</th>
<th>SGS-4</th>
<th>SGS-8</th>
<th>Combined</th>
<th>RSS</th>
<th>DFS</th>
<th>RPS</th>
<th>NURS.cn</th>
<th>NURS.depth</th>
<th>NURS.icnt</th>
<th>NURS.md2u</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td>14529</td>
<td>16072</td>
<td>14930</td>
<td>13951</td>
<td>19719</td>
<td>12267</td>
<td>9632</td>
<td>13349</td>
<td>10211</td>
<td>11105</td>
<td>9987</td>
<td>9208</td>
</tr>
</tbody>
</table>

2013/10/31
"Best" Counts

Result 6: SGS kills more mutants
Impact of Different Length \( n \)

- Shorter length => less contextual information
- Longer length => more contextual information
- Combined SGS strikes a good balance
  - Efficiency
  - Effectiveness
Summary

- Introduced length-n path spectra to guide path exploration
  - Uniform, parameterized technique
  - Steering toward less traveled paths

- Implemented in KLEE and extensively evaluated
  - SGS outperforms existing search strategies
  - SGS exhibits different behavior with varying length $n$
  - Combined SGS performs the best

Thanks!