Search Strategies in Dynamic Symbolic Execution

Hyunmin Seo
The Hong Kong University of Science and Technology
29 July 2014
Dynamic Symbolic Execution

- **Automatic test input** generation technique
  - No redundant input for the same path
  - High coverage
  - No False Positive
- Symbolic execution-based
- Exploits dynamic information
DSE Example

\(i_0: \ x = y = z = 0\)
\(i_1: \ x = y = 0, \ z = 1\)

```plaintext
tdef (x, y, z)
    x = sym_input();
y = sym_input();
z = sym_input();

    a = x + y;
    if (z > a)
        b = x - y;
    else
        b = 2 * y;
...
```

<table>
<thead>
<tr>
<th>Var</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>s</td>
</tr>
<tr>
<td>y</td>
<td>s</td>
</tr>
<tr>
<td>z</td>
<td>s</td>
</tr>
<tr>
<td>a</td>
<td>s</td>
</tr>
<tr>
<td>b</td>
<td>2*s</td>
</tr>
</tbody>
</table>

PC: \(s_2 \leq s_0 + s_1\)

PC': \(s_2 > s_0 + s_1\)
Overview of DSE

PC = pc_1 \land pc_2 \land pc_3 \ldots \land pc_n
Path Explosion Challenge

Execution Tree - A set of all possible execution paths
Path Explosion in DSE
Search Strategies

- Prioritise branches
- Explore **high priority** branches only

  - Run $P$ with a concrete input $I$
    - Execution Path $\pi = b_1b_2b_3...b_n$
    - Select a branch $b_i$
    - Generate $I'$ for $\pi' = b_1b_2b_3...b'_i$
      - By symbolic execution and constraint solving
CarFast

• How many branches can be covered by taking this branch?

• Calculate the number of branches transitively control dependent on each branch

* CarFast - [Park ’12]
Generational Search

- How many branches are covered by taking this branch?

* Generational Search - [Godefroid '08]
Context of Branch

• **Context of b**
  • State of P before reaching b
  • How the execution reached b
... if (member_level >= PRIME)
    shipping = FAST; // b1
else
    shipping = SLOW; // b2
if (total >= 100)
    discount = get_discount(total); // b3
else
    discount = 0; // b4
if (total >= 200)
    points = get_points(total - discount); // b5
else
    points = 0; // b6
...

\[ \text{total} < 100 \& \text{total} \geq 200 \]
\[ \text{UNSAT} \]
\[ \text{total} \geq 100 \& \text{total} \geq 200 \]
\[ \text{SAT} \]
Context-Guided Search (CGS)

$k$-Context

Preceding Branches

Dominators

Select

Skip

Context Cache
Deeply Nested CFG

3-Context of $b_{11}$ on $\pi_1$

$= (b_7, b_9, b_{11})$

3-Context of $b_{11}$ on $\pi_2$

$= (b_7, b_9, b_{11})$
Dominator

- $d \text{ dom } n$ iff every path from the entry node to $n$ must go through $d$

- If $p_1, p_2, \ldots, p_k$ are predecessors of $n$ and $d \neq n$, then $d \text{ dom } n$ iff $d \text{ dom } p_i$ for each $i$

\[
\begin{align*}
\text{Dom}(n_0) &= \{n_0\} \\
\text{Dom}(n) &= \left( \bigcap_{p \in \text{preds}(n)} \text{Dom}(p) \right) \cup \{n\}
\end{align*}
\]

k-Context - Refine

- **k-Context** - a sequence of non-dominating $k$-preceeding branches

$\text{dom}(b_{11}) = \{b_3, b_5, b_7, b_9, b_{11}\}$

2-Context of $b_{11}$ on $\pi_1$

$= (b_1, b_{11})$

2-Context of $b_{11}$ on $\pi_2$

$= (b_2, b_{11})$
Optimal \( k \)

- **1-Context** \( \rightarrow \) Similar to Greedy Search
  - Select \( b \) if \( b \) has not been selected

- **\( \infty \)-Context** \( \rightarrow \) BFS
  - Select every \( b \) in BFS order
Incremental Search

1-context

2-context

3-context

BFS traversal

Examine $b_i$
Evaluation - RQ

• RQ1
  • Given the same testing budget, how many branches can each strategy cover?

• RQ2
  • Given a target coverage goal, how many iterations does each strategy require to achieve the goal?
## Evaluation Subjects

<table>
<thead>
<tr>
<th>Subject</th>
<th>Testing Tool</th>
<th>Language</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>grep</td>
<td>CREST</td>
<td>C</td>
<td>19K</td>
</tr>
<tr>
<td>replace</td>
<td>CREST</td>
<td>C</td>
<td>0.5K</td>
</tr>
<tr>
<td>expat</td>
<td>CREST</td>
<td>C</td>
<td>18K</td>
</tr>
<tr>
<td>cdaudio</td>
<td>CREST</td>
<td>C</td>
<td>2k</td>
</tr>
<tr>
<td>floppy</td>
<td>CREST</td>
<td>C</td>
<td>1.5K</td>
</tr>
<tr>
<td>kbfiltr</td>
<td>CREST</td>
<td>C</td>
<td>1K</td>
</tr>
<tr>
<td>tp300</td>
<td>CarFastTool</td>
<td>Java</td>
<td>0.3K</td>
</tr>
<tr>
<td>tp600</td>
<td>CarFastTool</td>
<td>Java</td>
<td>0.6K</td>
</tr>
<tr>
<td>tp1k</td>
<td>CarFastTool</td>
<td>Java</td>
<td>1.5K</td>
</tr>
<tr>
<td>tp2k</td>
<td>CarFastTool</td>
<td>Java</td>
<td>2.4K</td>
</tr>
<tr>
<td>tp5k</td>
<td>CarFastTool</td>
<td>Java</td>
<td>5.8K</td>
</tr>
<tr>
<td>tp10k</td>
<td>CarFastTool</td>
<td>Java</td>
<td>28K</td>
</tr>
</tbody>
</table>
## Coverage in CREST

<table>
<thead>
<tr>
<th>Subject</th>
<th>Strategy</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
<th>4000</th>
</tr>
</thead>
<tbody>
<tr>
<td>grep</td>
<td>CGS</td>
<td>1523.0</td>
<td>1643.3</td>
<td>1690.4</td>
<td>1721.6 (+1.8%)</td>
</tr>
<tr>
<td></td>
<td>CFG</td>
<td>1494.5</td>
<td>1503.7</td>
<td>1475.8</td>
<td>1495.8 (+1.1%)</td>
</tr>
<tr>
<td></td>
<td>Random Branch</td>
<td>1317.0</td>
<td>1371.9</td>
<td>1397.3</td>
<td>1412.7 (+1.1%)</td>
</tr>
<tr>
<td></td>
<td>Generational</td>
<td>1032.5</td>
<td>1199.3</td>
<td>1224.6</td>
<td>1255.6 (+2.5%)</td>
</tr>
<tr>
<td></td>
<td>DFS</td>
<td>948.1</td>
<td>989.9</td>
<td>1087.2</td>
<td>1099.9 (+1.2%)</td>
</tr>
<tr>
<td></td>
<td>CarFast</td>
<td>1197.9</td>
<td>1223.1</td>
<td>1240.8</td>
<td>1253.9 (+1.1%)</td>
</tr>
<tr>
<td>replace</td>
<td>CGS</td>
<td>180.0</td>
<td>180.8</td>
<td>181.0</td>
<td>181.0 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>CFG</td>
<td>175.1</td>
<td>176.3</td>
<td>176.6</td>
<td>177.0 (+0.2%)</td>
</tr>
<tr>
<td></td>
<td>Random Branch</td>
<td>167.4</td>
<td>171.9</td>
<td>173.2</td>
<td>174.2 (+0.5%)</td>
</tr>
<tr>
<td></td>
<td>Generational</td>
<td>165.2</td>
<td>170.7</td>
<td>175.8</td>
<td>175.8 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>DFS</td>
<td>84.3</td>
<td>157.0</td>
<td>169.6</td>
<td>170.6 (+0.6%)</td>
</tr>
<tr>
<td></td>
<td>CarFast</td>
<td>151.0</td>
<td>152.9</td>
<td>154.0</td>
<td>155.8 (+1.2%)</td>
</tr>
<tr>
<td>expat</td>
<td>CGS</td>
<td>1040.3</td>
<td>1131.6</td>
<td>1201.2</td>
<td>1248.0 (+3.9%)</td>
</tr>
<tr>
<td></td>
<td>CFG</td>
<td>899.7</td>
<td>972.7</td>
<td>1036.8</td>
<td>1073.7 (+3.6%)</td>
</tr>
<tr>
<td></td>
<td>Random Branch</td>
<td>677.2</td>
<td>677.4</td>
<td>677.5</td>
<td>677.6 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>Generational</td>
<td>703.0</td>
<td>715.0</td>
<td>717.0</td>
<td>719.2 (+0.3%)</td>
</tr>
<tr>
<td></td>
<td>DFS</td>
<td>670.0</td>
<td>670.0</td>
<td>670.0</td>
<td>670.0 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>CarFast</td>
<td>739.8</td>
<td>764.9</td>
<td>789.7</td>
<td>819.0 (+3.7%)</td>
</tr>
<tr>
<td>cdaudio</td>
<td>CGS</td>
<td>250.0</td>
<td>250.0</td>
<td>250.0</td>
<td>250.0 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>CFG</td>
<td>246.0</td>
<td>249.0</td>
<td>249.6</td>
<td>249.7 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>Random Branch</td>
<td>220.8</td>
<td>233.8</td>
<td>239.3</td>
<td>241.9 (+1.1%)</td>
</tr>
<tr>
<td></td>
<td>Generational</td>
<td>250.0</td>
<td>250.0</td>
<td>250.0</td>
<td>250.0 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>DFS</td>
<td>242.0</td>
<td>242.0</td>
<td>242.0</td>
<td>242.0 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>CarFast</td>
<td>122.0</td>
<td>122.0</td>
<td>122.0</td>
<td>122.0 (+0.0%)</td>
</tr>
<tr>
<td>floppy</td>
<td>CGS</td>
<td>205.0</td>
<td>205.0</td>
<td>205.0</td>
<td>205.0 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>CFG</td>
<td>199.8</td>
<td>203.8</td>
<td>204.6</td>
<td>204.9 (+0.1%)</td>
</tr>
<tr>
<td></td>
<td>Random Branch</td>
<td>133.4</td>
<td>150.4</td>
<td>159.2</td>
<td>165.0 (+3.7%)</td>
</tr>
<tr>
<td></td>
<td>Generational</td>
<td>205.0</td>
<td>205.0</td>
<td>205.0</td>
<td>205.0 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>DFS</td>
<td>186.3</td>
<td>186.3</td>
<td>186.3</td>
<td>186.3 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>CarFast</td>
<td>49.0</td>
<td>49.0</td>
<td>49.0</td>
<td>49.0 (+0.0%)</td>
</tr>
<tr>
<td>kbfiltre</td>
<td>CGS</td>
<td>149.0</td>
<td>149.0</td>
<td>149.0</td>
<td>149.0 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>CFG</td>
<td>147.6</td>
<td>149.0</td>
<td>149.0</td>
<td>149.0 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>Random Branch</td>
<td>143.9</td>
<td>147.7</td>
<td>148.6</td>
<td>148.8 (+0.1%)</td>
</tr>
<tr>
<td></td>
<td>Generational</td>
<td>149.0</td>
<td>149.0</td>
<td>149.0</td>
<td>149.0 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>DFS</td>
<td>137.0</td>
<td>137.0</td>
<td>137.0</td>
<td>137.0 (+0.0%)</td>
</tr>
<tr>
<td></td>
<td>CarFast</td>
<td>109.0</td>
<td>109.0</td>
<td>109.0</td>
<td>109.0 (+0.0%)</td>
</tr>
</tbody>
</table>
Iterations in CREST

- grep (Target:1099)
- replace (Target:155)
- expat (Target:670)
- cdaudio (Target:241)
- floppy (Target:165)
- kbfilt (Target:137)
### CGS Summary

- **Context-Guided Search**

  - $k$-Context
  - Preceding Branches
  - Dominators
  - Context Cache
  - Select
  - Skip
  - BFS traversal
  - Examine $b_i$
  - $1$-context
  - $2$-context
  - $3$-context

- **Evaluation on 12 subjects**

  How We Get There: A Context-Guided Search Strategy in Concolic Testing, Hyunmin Seo and Sunghun Kim, FSE ‘14
Goal of Search

- Prioritisation criteria of search strategy varies by the goal
  - Coverage in general
  - Coverage in target area
Regression Testing

- Regression testing or patch testing
- Generate inputs for recently changed code
Challenges in Regression Testing

- To test changed code, we need to cover $b$ first
- However, SMT solver may return **UNSAT** results
UNSAT Problem

\[ pc_1 = x > 200 \]

\[ pc_2 = x > 0 \]

\[ PC' = pc_1 \land \neg pc_2 \]

\[ PC' = x > 200 \land x \leq 0 \]

UNSAT
Covering UNSAT Branch

- UNSAT ≠ Unreachable
- Trying the same branch from different paths may return SAT
Possible Approach

- Symbolic Execution along all paths
- Dijkstra’s weakest precondition
Heuristic-Based Approach

- Try conflict core first
- Try different def location

\[ pc_1 = x > 200 \]
\[ pc_2 = x > 0 \]
Guidance by Pre-Analysis

Abstract Value of $y$

$= [-\infty, 100]$ if $(y > 1)$
Summary

• Dynamic Symbolic Execution

• Path Explosion Challenge

• Search Strategies
  • Coverage in general - CGS
  • Coverage for target area
Future Work

• Search strategies for bug finding

• Guide execution toward \( b_1 \) with maximum possible value of \( a \) to make buffer \( z \) to overflow

```plaintext
if (x)
    y = z[a];  // b_1
...
```