Using Execution Paths to Evolve Software Patches

ThanhVu Nguyen*, Westley Weimer**, Claires Le Gouges**, Stephanie Forrest*

* University of New Mexico
** University of Virginia

Tuesday, March 31, 2009
Introduction

• Software is **THE** problem

• Software Repair using Genetic Programming (SRGP)
  – Start with the original (buggy) program
  – Focus on execution path through AST
  – Restrict mutation and crossover to execution path
  – Don’t invent any new code

  – **Results**: successfully repair 13 *real* programs in over **140,000** lines of code in **feasible** time
SRGP Algorithm Outline

Preprocessing

Repeat

Fitness Evaluation

   If found an individual \( C \) with accepted fitness, \textbf{Return} \( C \)

Exploitation

   Select mating pool \( M \) consisting of high fit individuals

Exploration

   Perform recombination operator on \( M \) to get a new generation \( N \)
   Apply mutation operator on \( N \)

Until termination criteria are met
Example: Zunebug

- Millions of Microsoft Zune media players mysteriously froze up on December 31\textsuperscript{st}, 2008

- The bug: a date related function in Zune enters an *infinite loop* when the input is the last day of a leap year
1. `void` zunebug(`int` days) {
2.     `int` year = 1980;
3.     `while` (days > 365) {
4.         `if` (isLeapYear(year)) {
5.             `if` (days > 366) {
6.                 days -= 366;
7.                 year += 1;
8.             }
9.         `else` {
10.             }
11.     }
12.     `else` {
13.         days -= 365;
14.         year += 1;
15.     }
16. }
17. `printf("year is %d\n", year);
18. `}
void zunebug(int days) {
    int year = 1980;
    while (days > 365) {
        if (isLeapYear(year)) {
            if (days > 366) {
                days -= 366;
                year += 1;
            } else {
            }
        } else {
        }
    } else {
        days -= 365;
        year += 1;
    }
    printf("year is %d\n", year);
}
1. **void** zunebug**(int** days) {
2.       **int** year = 1980;
3.       **while** (days > 365) {
4.           **if** (isLeapYear(year)) {
5.               **if** (days > 366) {
6.                 days -= 366;
7.                 year += 1;
8.               }
9.           }
10.       }
11.   **else** {
12.       }
13.   }
14. }
15. else {
16.     days -= 365;
17.     year += 1;
18. }
19. }
20. printf(“year is %d
”, year);
21. }
Weighted Execution Path

Neg Exec Path
Stmt weight = 1.0
Weighted Execution Path

Pos Exec Path

j m t y d a e z v o

f w z b a c i k u r p t v
Weighted Execution Path

Stmts in both Neg Exec Path and Pos Exec Path
Weighted Execution Path

Weight = \{1.0, 0.1\}
Fitness Evaluation

• Take in a program source $P$ to be evaluated

• Compile $P$ to an executable program $P'$
  – If cannot compile, assign fitness $0$.

• Fitness score of $P'$: weighted sum of test cases that the $P'$ passes
  \[
  \text{Fitness}(P') = \# \text{ pos pass} \times W_{\text{pos}} + \# \text{ neg pass} \times W_{\text{neg}}
  \]
  – 5 positive test cases (weight = 1), 1 or 2 negative test cases (weight = 10)
  – If $P'$ passes all test cases, then $P$ is a solution candidate
  – Note: the original (buggy) program passes all positive test cases
Genetic Operators

- **Recombination** *(crossback)*
  - Cross the input individuals back with the *original* program (instead of crossing over each other)

- **Mutation**
  - \( delete(s) \). \( s = \{\} \);
  - \( insert(s, y) \). \( s = \{s; y;\} \);
  - \( swap(s, y) \). \( t = s; s = \{y\}; y = \{t\} \);
Original Program

```c
void zunebug(int days) {
    int year = 1980;
    while (days > 365) {
        if (isLeapYear(year)) {
            if (days > 366) {
                days -= 366;
                year += 1;
            }
        } else {
        }
    } else {
        days -= 365;
        year += 1;
    }
    printf("current year is %d\n", year);
}
```

Final Repair

```c
void zunebug_repair(int days) {
    int year = 1980;
    while (days > 365) {
        if (isLeapYear(year)) {
            if (days > 366) {
                days -= 366; // repair deletes
                year += 1;
            }
        } else {
        }
    } else {
        days -= 365; // repair inserts
        year += 1;
    }
    printf("current year is %d\n", year);
}
```
Evolution of Zunebug

![Graph showing the evolution of fitness over generations with two lines representing the average and the best individual. The graph indicates an increase in fitness over generations.]
## Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Version</th>
<th>LoC</th>
<th>Stmts</th>
<th>Path Len</th>
<th>Program Description</th>
<th>Fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcd</td>
<td>_</td>
<td>22</td>
<td>10</td>
<td>1.3</td>
<td>Handcrafted example</td>
<td>Infinite loop</td>
</tr>
<tr>
<td>look-s</td>
<td>svr 4.0 1.1</td>
<td>1363</td>
<td>100</td>
<td>32.4</td>
<td>Dictionary lookup</td>
<td>Infinite loop</td>
</tr>
<tr>
<td>atris</td>
<td>1.0.6</td>
<td>21553</td>
<td>6470</td>
<td>34.0</td>
<td>Graphical Tetris game</td>
<td>Local stack buffer exploit</td>
</tr>
<tr>
<td>uniq</td>
<td>ultrix 4.3</td>
<td>1146</td>
<td>81</td>
<td>81.5</td>
<td>Duplicate text processing</td>
<td>Segfault</td>
</tr>
<tr>
<td>look-u</td>
<td>ultrix 4.3</td>
<td>1169</td>
<td>90</td>
<td>213.0</td>
<td>Dictionary lookup</td>
<td>Segfault</td>
</tr>
<tr>
<td>deroff</td>
<td>ultrix 4.3</td>
<td>2236</td>
<td>1604</td>
<td>251.4</td>
<td>Document processing</td>
<td>Segfault</td>
</tr>
<tr>
<td>nullhttpd</td>
<td>0.5.0</td>
<td>5575</td>
<td>1040</td>
<td>768.5</td>
<td>Web server</td>
<td>Remote heap buffer exploits</td>
</tr>
<tr>
<td>indent</td>
<td>1.9.1</td>
<td>9906</td>
<td>2022</td>
<td>1435.9</td>
<td>Source code processing</td>
<td>Infinite loop</td>
</tr>
<tr>
<td>units</td>
<td>svr4.0 1.1</td>
<td>1504</td>
<td>240</td>
<td>2159.7</td>
<td>Metric conversion</td>
<td>Segfault</td>
</tr>
<tr>
<td>flex</td>
<td>2.5.4a</td>
<td>18775</td>
<td>3635</td>
<td>3836.6</td>
<td>Lexical analyzer generator</td>
<td>Segfault</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC</th>
<th>Path Len</th>
<th>Repair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fitness</td>
</tr>
<tr>
<td>gcd</td>
<td>22</td>
<td>1.3</td>
<td>41.0</td>
</tr>
<tr>
<td>look-s</td>
<td>1363</td>
<td>32.4</td>
<td>8.5</td>
</tr>
<tr>
<td>atris</td>
<td>21553</td>
<td>34.0</td>
<td>13.2</td>
</tr>
<tr>
<td>uniq</td>
<td>1146</td>
<td>81.5</td>
<td>9.5</td>
</tr>
<tr>
<td>look-u</td>
<td>1169</td>
<td>213.0</td>
<td>11.1</td>
</tr>
<tr>
<td>deroff</td>
<td>2236</td>
<td>251.4</td>
<td>21.6</td>
</tr>
<tr>
<td>nullhttpd</td>
<td>5575</td>
<td>768.5</td>
<td>79.1</td>
</tr>
<tr>
<td>indent</td>
<td>9906</td>
<td>1435.9</td>
<td>95.6</td>
</tr>
<tr>
<td>units</td>
<td>1504</td>
<td>2159.7</td>
<td>55.7</td>
</tr>
<tr>
<td>flex</td>
<td>18775</td>
<td>3836.6</td>
<td>33.4</td>
</tr>
</tbody>
</table>
### Most Recent Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Version</th>
<th>LoC</th>
<th>Stmts</th>
<th>Path Len</th>
<th>Program Description</th>
<th>Fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenLDAP</td>
<td>2.3.41</td>
<td>6519</td>
<td>25</td>
<td>1.3</td>
<td>Directory Protocol</td>
<td>Non-overflow denial of service</td>
</tr>
<tr>
<td></td>
<td>io.c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Php</td>
<td>5.2.1</td>
<td>26044</td>
<td>52</td>
<td>34.0</td>
<td>Scripting Language</td>
<td>Integer overflow</td>
</tr>
<tr>
<td>string.c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lighttpd</td>
<td>1.4.17</td>
<td>13984</td>
<td>136</td>
<td>32.4</td>
<td>Web server</td>
<td>Remote heap buffer overflow</td>
</tr>
<tr>
<td>fastcgi.c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wu-ftp</td>
<td>2.6.0</td>
<td>35109</td>
<td>149</td>
<td>81.5</td>
<td>Ftp server</td>
<td>Format string</td>
</tr>
<tr>
<td></td>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Traditional 1-point **crossover**
  - Works better than **crossback** in some programs and worse in others
Scalability

The graph shows the relationship between the log of the average number of fitness evaluations and the log of the weighted path length for various programs. The data points are labeled with names such as 'indent', 'units', 'nullhttpd', 'atris', 'uniq', 'deroff', 'look', 'look svr4', 'lighttpd', and 'openldap'. The graph exhibits a positive correlation, indicating that as the weighted path length increases, so does the average number of fitness evaluations required.
Conclusion

• **SRGP**
  – Focus on execution path to reduce search space complexity
  – Use GP to evolve code
  – Combine Positive and Negative test cases for fitness evaluation
    • Positive test cases: preserve the core functionality of the program
    • Negative test cases: identify the bug
  – Work on real world applications and different types of bugs

• **Future Work**
  – Explore different GP techniques
  – Integrate anomaly detection methods