Range Fixes and Their Application on Software Configuration

Yingfei Xiong, Peking University, 2012

Based on an ICSE’12 paper with Arnaud Hubaux (U. Namur), Steven She and Krzysztof Czarnecki (U. Waterloo)
We produce errors everyday

```java
public static void main(String[] args) {
    output("Hello, world");
}
```

The method `output(String)` is undefined for the type `Main`

1 quick fix available:

- Create method 'output(String)'

Press 'F2' for focus
We use fixes everyday

```java
public static void main(String[] args) {
    output("Hello, world");
}

private static void output(String msg) {
    // TODO Auto-generated method stub
}
```
How much do we know about fixes?

• How much are fixes needed?
• What fixes are desirable?
• Can we generate fixes automatically?
Study domain: operating system configuration

Variability Models

Linux Kconfig, eCos CDL, ...

Variability Models
eCos Configurator - Errors

- Preload Size: 10
- Requires PreloadSize <= PoolSize
- Property Value:
  - Value: 10
  - Default: 10
  - Flavor: data
  - Requires: PreloadSize <= PoolSize
  - DefaultValue: 10
Contributions

• How much are fixes needed?
  – A survey revealing manual fixes take minutes

• What fixes are desirable?
  – A new type of fix, range fix, and evaluated desirable properties of fixes

• Can we generate fixes automatically?
  – An algorithm generating range fixes in tens of milliseconds
How much are fixes needed?

A survey showing manual fixes take minutes
Survey

- 97 Linux users and 9 eCos users
- Resolving a violation is hard
  - 20% Linux users need "a few dozen minutes" to resolve a violation on average
  - 56% eCos users consider violation resolution to be a problem
What fixes are desirable?

A new type of fixes, range fixes, and evaluated desirable properties
eCos configurator has built-in fixes
78% eCos users have encountered situations where the proposed fix is not useful.
How to complete fixes

<table>
<thead>
<tr>
<th>PreloadSize</th>
<th>PoolSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Preload = false
Our Proposal – Range Fixes

[PreloadSize <= 8]
[PoolSize >= 10]
[Preload = false]
Fix Generation Problem – a General Definition

Typed Variables
- Preload: Bool
- PreloadSize: Int
- PoolSize: Int

Assigned Values
- Preload = true
- PreloadSize = 10
- PoolSize = 8

A logic constraint
- Preload → PreloadSize <= PoolSize

Fix Generator

Desirable fixes
- PreloadSize <= 8
- PoolSize >= 10
- Preload = false

A complete set of desirable fixes
## Desired Properties of Fixes

<table>
<thead>
<tr>
<th>Correctness</th>
<th>Minimality of variables</th>
<th>Maximality of ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any change represented by a range fix will satisfy the constraint</td>
<td>There is no way to change a subset of variables to satisfy the constraint</td>
<td>A range fix represents the maximal ranges over the variables</td>
</tr>
</tbody>
</table>

**A desirable one:** \([\text{PreloadSize} \leq 8]\)

**Undesirable ones**

- \([\text{PreloadSize} \leq 9]\)
- \([\text{PreloadSize} \leq 8, \text{Preload} = \text{false}]\)
- \([\text{PreloadSize} \leq 7]\)
Constraint Interaction

[PreloadSize <= 8]
[PoolSize >= 10]
[Preload = false]
Constraint Interaction

Increase PoolSize

Causing another error

Interacting constraint
Three Strategies

• Ignorance
• Elimination
• Propagation

• Summarized from existing approaches
Propagation Strategy

Make a conjunction of all satisfied constraints plus the violated one

Preload → PreloadSize ≤ PoolSize / \[PoolSize == BufferSize * 1024 / ObjectSize\]

\[PreloadSize ≤ 8\]
\[PoolSize ≥ 10 \& BufferSize = PoolSize / 2\]
\[PoolSize ≥ 10 \& ObjectSize = 4096 / PoolSize\]
\[Preload = false\]
Evaluation

• Source
  – Version histories from 5 open source projects

• Steps
  – Compare each pair of consecutive versions
  – Replay the user changes in different orders
  – Generate fixes for the violations and compare with user changes
Results

• Coverage of User changes: 100%

• Complexity of fix lists
  – measured by adding up the number of variables in each fix
  – Median: 2
  – Maximum: 58
  – 83% of the fix lists contain less than 10 variables
Can we generate fixes automatically?

An algorithm generating range fixes in tens of milliseconds
Interface of our algorithm

- **Typed Variables**:
  - Preload: Bool
  - PreloadSize: Int
  - PoolSize: Int

- **Assigned Values**:
  - Preload = true
  - PreloadSize = 10
  - PoolSize = 8

- **A logic constraint**:
  - Preload → PreloadSize <= PoolSize

- **Fix Generation Algorithm**
  - [PreloadSize <= 8]
  - [PoolSize >= 10]
  - [Preload = false]

- **A complete set of desirable fixes**
Algorithm Outline

• Step 1: find the variables to change
  – Basic idea: translating to an SMT problem
    1. treat configurations also as constraints
    2. ask an SMT solver for unsatisfiable cores
    3. combine the unsatisfiable cores

• Step 2: find the range of the variables
  – Basic idea: simplify the constraint
    1. replace unchangeable variables with their current values
    2. simplify the constraint and convert to CNF
Performance of the algorithm

- Published results
  - Average: 50ms
  - Maximum: 250ms

- We have recently improved the performance
Thank you for your attention!

EccFixer: http://gsd.uwaterloo.ca/eccfixer