Generating Range Fixes for Software Configuration

Yingfei Xiong*  Arnaud Hubaux  Steven She
Peking University  University of Namur  Krzysztof Czarnecki
China  Belgium  Canada

* The work was done when Yingfei was at University of Waterloo
Variability Models & Configurators

Variability Models

Configuration

Linux Kconfig, eCos CDL, pure::variants, ...

Variability Models
eCos Configurator - Errors
Error resolution and option activation both need to resolve violation of constraint.
Survey

• 97 Linux users and 9 eCos users

• Resolving a violation is hard
  – 20% Linux users need "a few dozen minutes" to activate an option in average
  – 56% eCos users consider activation to be a problem
eCos Configurator

Essentially, fixes work for both resolving errors and activating options
Fix Incompleteness

78% eCos users have encountered situations where the proposed fix is not useful.
How to complete fixes

PreloadSize = 8
PreloadSize = 7
PreloadSize = 6
PreloadSize = 5
...
PoolSize = 10
PoolSize = 11
PoolSize = 12
PoolSize = 13
...
Preload = false
Our Solution – Range Fixes

[PreloadSize <= 8]
[PoolSize >= 10]
[Preload = false]
Our Contributions

• Defining the range fix generation problem
  – Three desirable properties of range fixes
• Proposing a range fix generation algorithm
• Exploring the constraint interaction problem
  – Summarizing and adapting three strategies used in existing work
  – Comparing the strategies empirically
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

A complete set of desirable fixes:
- [PreloadSize <= 8]
- [PoolSize >= 10]
- [Preload = false]
### 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**

| \([\text{PreloadSize} \leq 9]\) | \([\text{PreloadSize} \leq 8, \text{Preload} = \text{false}]\) | \([\text{PreloadSize} \leq 7]\) |
Algorithm

• Based on Reiter’s theory of diagnosis
• Please check the paper for the details
Constraint Interaction

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

Increase PoolSize

Interacting constraint

Causing another error
Ignorance

Ignore the interaction
Elimination

Eliminate all changes that will violate other constraints
Propagation

Propagate the change along other constraints

\[
\begin{align*}
\text{PreloadSize} & \leq 8 \\
\text{PoolSize} & \geq 10 \quad & \text{BufferSize} & = \frac{\text{PoolSize}}{2} \\
\text{PoolSize} & \geq 10 \quad & \text{ObjectSize} & = \frac{4096}{\text{PoolSize}} \\
\text{Preload} & = \text{false}
\end{align*}
\]
Comparison of Strategies

<table>
<thead>
<tr>
<th></th>
<th>Ignorance</th>
<th>Elimination</th>
<th>Propagtion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution time</td>
<td>Shortest</td>
<td>Short</td>
<td>Possibly long</td>
</tr>
<tr>
<td>Complexity of fix lists</td>
<td>Simple</td>
<td>Simplest</td>
<td>Possibly complex</td>
</tr>
<tr>
<td>Introduction of new errors</td>
<td>Possible</td>
<td>Never</td>
<td>Never</td>
</tr>
<tr>
<td>Fix completeness</td>
<td>Complete (for one constraint)</td>
<td>Incomplete</td>
<td>Complete (for all constraints)</td>
</tr>
</tbody>
</table>
Experiments

• 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
## Execution Time

<table>
<thead>
<tr>
<th></th>
<th>Ignorance</th>
<th>Elimination</th>
<th>Propagation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution time</td>
<td>Average: 17ms</td>
<td>Average: 20ms</td>
<td>Average: 50ms</td>
</tr>
<tr>
<td></td>
<td>Maximum: 20ms</td>
<td>Maximum: 30ms</td>
<td>Maximum: 250ms</td>
</tr>
<tr>
<td>Complexity of fix lists</td>
<td>Simple</td>
<td>Simplest</td>
<td>Possibly complex</td>
</tr>
<tr>
<td>Introduction of new errors</td>
<td>Possible</td>
<td>Never</td>
<td>Never</td>
</tr>
<tr>
<td>Fix completeness</td>
<td>Complete (for one constraint)</td>
<td>Incomplete</td>
<td>Complete (for all constraints)</td>
</tr>
</tbody>
</table>

Our algorithm is sufficiently fast for each strategy.
# Complexity of fix lists

<table>
<thead>
<tr>
<th></th>
<th>Ignorance</th>
<th>Elimination</th>
<th>Propagation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Execution time</strong></td>
<td>Average: 17ms Maximum: 20ms</td>
<td>Average: 20ms Maximum: 30ms</td>
<td>Average: 50ms Maximum: 250ms</td>
</tr>
<tr>
<td><strong>Complexity of fix lists (Number of variables in a list)</strong></td>
<td>Max: 4 Median: 2 Average: 2.2</td>
<td>Max: 4 Median: 2 Average: 1.64</td>
<td>Max: 58 Median: 2 Average: 8.0</td>
</tr>
<tr>
<td><strong>Introduction of new errors</strong></td>
<td>Possible</td>
<td>Never</td>
<td>Never</td>
</tr>
<tr>
<td><strong>Fix completeness</strong></td>
<td>Complete (for one constraint)</td>
<td>Incomplete</td>
<td>Complete (for all constraints)</td>
</tr>
</tbody>
</table>

In propagation, 83% of the fix lists contain less than 10 variables
## Introduction of new errors

<table>
<thead>
<tr>
<th></th>
<th>Ignorance</th>
<th>Elimination</th>
<th>Propagation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Execution time</strong></td>
<td>Average: 17ms</td>
<td>Average: 20ms</td>
<td>Average: 50ms</td>
</tr>
<tr>
<td></td>
<td>Maximum: 20ms</td>
<td>Maximum: 30ms</td>
<td>Maximum: 250ms</td>
</tr>
<tr>
<td><strong>Complexity of fix lists</strong></td>
<td>Max: 4</td>
<td>Max: 4</td>
<td>Max: 58</td>
</tr>
<tr>
<td>(Number of variables in a list)</td>
<td>Median: 2</td>
<td>Median: 2</td>
<td>Median: 2</td>
</tr>
<tr>
<td></td>
<td>Average: 2.2</td>
<td>Average: 1.64</td>
<td>Average: 8.0</td>
</tr>
<tr>
<td><strong>Introduction of new errors</strong></td>
<td>44% of all violations</td>
<td>Never</td>
<td>Never</td>
</tr>
<tr>
<td><strong>Fix completeness</strong></td>
<td>Complete (for one constraint)</td>
<td>Incomplete</td>
<td>Complete (for all constraints)</td>
</tr>
</tbody>
</table>
Fix completeness

<table>
<thead>
<tr>
<th>Execution time</th>
<th>Ignorance</th>
<th>Elimination</th>
<th>Propagation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average: 17ms Maximum: 20ms</td>
<td>Average: 20ms Maximum: 30ms</td>
<td>Average: 50ms Maximum: 250ms</td>
</tr>
<tr>
<td>Complexity of fix lists (Number of variables in a list)</td>
<td>Max: 4 Median: 2 Average: 2.2</td>
<td>Max: 4 Median: 2 Average: 1.64</td>
<td>Max: 58 Median: 2 Average: 8.0</td>
</tr>
<tr>
<td>Introduction of new errors</td>
<td>44% of all violations</td>
<td>Never</td>
<td>Never</td>
</tr>
<tr>
<td>Fix completeness (coverage of user changes)</td>
<td>100%</td>
<td>57%</td>
<td>100%</td>
</tr>
</tbody>
</table>

eCos configurator: 73%
Conclusion

• Fix completeness can be achieved by organizing them into range fixes
• Range fixes can be generated automatically and efficiently
• Three strategies for constraint interaction
  – No absolutely best solution
  – Propagation strategy gives relatively better results than the other two
Thank you for your attention!

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