Detecting Semantic Merge Conflicts With Variability-Aware Execution

Hung Viet Nguyen
Iowa State University
My Huu Nguyen
Ho Chi Minh City University of Science, Vietnam
Son Cuu Dang
University of Technology Sydney, Australia
Christian Kästner
Carnegie Mellon University
Tien N. Nguyen
Iowa State University
Collaborative Software Development
Types of Merge Conflicts

- Textual merge conflicts
- Semantic merge conflicts
  - No textual conflicts
  - Resulting in build/test failures
  - 33% of 399 merges contained semantic conflicts (Brun et al., ESEC/FSE’11)
Example of a Semantic Conflict

$\text{calc}(1, 3) = 2; \text{[PASS]}$

$\text{calc}(1, 3) = 2; \text{[PASS]}$

$\text{calc}(1, 3) = 4; \text{[FAIL]}$
Problem Statement

Given a set of parallel changes, identify early the actual (minimum) set of changes that directly cause a semantic merge conflict.
• **Pairwise testing**
  - Quadratic complexity
  - Might execute redundant test cases
  - Still cannot explore higher-interaction errors

• **Brute-force approach**
  - Exponential number of combinations
Semex Approach

1. Encoding Parallel Changes

Variability-aware merged files

2. Variability-Aware Execution

Semantically conflicting files
Semex Approach

1. Encoding Parallel Changes

Variability-aware merged files

2. Variability-Aware Execution

Semantically conflicting files
Encoding Parallel Changes

\[
\text{function}\;\text{calc}(x, \; y)\{\text{
$a = x;$
$b = y;$
$c = 2 \times x;$
return $c;$
}\}
\]

\[
\text{function}\;\text{calc}(x, \; y)\{\text{
$a = x + y;$
$b = x - y;$
$c = a + b;$
return $c;$
}\}
\]

\[
\text{function}\;\text{calc}(x, \; y)\{\text{
if ($P1$) {
a = x;
b = y;
}\} \text{ else } {
\text{
$a = x + y;$
b = x - y;$
$c = a + b;$
return $c;$
}\}
\}
\]

\[
\text{function}\;\text{calc}(x, \; y)\{\text{
if ($P2$) {
$c = a + b;$
}\} \text{ else } {
\text{
$c = 2 \times x;$
return $c;$
}\}
\}
\]
Semex Approach

1. Encoding Parallel Changes
   - Variability-aware merged files

2. Variability-Aware Execution
   - Semantically conflicting files

Files in branch 1
Files in branch 2
Files in branch n
Brute-Force Testing

Program with configuration options

configure program

Programs with concrete selections of options

brute-force execution

Result for all configurations

aggregate results

Results for individual configurations
Is There a Shortcut?

Program + Options
Program with configuration options

Program + Options
Result for all configurations

Program + Options
Program with concrete selections of options

configure program

aggregate results

brute-force execution

Results for individual configurations
Variability-Aware Execution

Program + Options

Program with configuration options

Variability-aware execution

Program + Options

Result for all configurations

• Single run to explore the *entire configuration space*

• Yields equivalent results with brute-force execution on all configs
Taking advantage of sharing

```
function calc($x, $y) {
    if ($P1) {
        $a = $x;
        $b = $y;
    } else {
        $a = $x + $y;
        $b = $x - $y;
    }
    if ($P2) {
        $c = $a + $b;
    } else {
        $c = 2 * $x;
    }
    return $c;
}
```

```
<table>
<thead>
<tr>
<th>P1=1 P2=1</th>
<th>P1=1 P2=0</th>
<th>P1=0 P2=1</th>
<th>P1=0 P2=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
</tbody>
</table>
```
Key Idea of Variability-Aware Execution

Executing common code once
Multi-Value Data Representation

\[ \text{CHOICE}(\text{Cond}, \text{value1}, \text{value2}) \]

- Condition
  - Value when Cond=True
  - Value when Cond=False
function calc($x, $y) {
    if ($P1) {
        $a = $x;
        $b = $y;
    }
    else {
        $a = $x + $y;
        $b = $x - $y;
    }
    if ($P2) {
        $c = $a + $b;
    }
    else {
        $c = 2 * $x;
    }
    return $c;
}

assertEquals(calc(1, 3), 2);

FAIL when P1 & P2 are enabled

SEMANTIC CONFLICT DETECTED!
Variability-Aware Execution Preliminary Results (Nguyen et al., ICSE 2014)

• Scaled to $2^{50}$ configurations within 7 minutes

• Observed interaction of up to 16 plugins

• Pinpointed exact configurations in which plugin conflicts occurred
## Preliminary Experiment

<table>
<thead>
<tr>
<th>Example</th>
<th>Test cases</th>
<th>Branches</th>
<th>Semantic Conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>MathHelper</td>
<td>7</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Queue1</td>
<td>8</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Queue2</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Product</td>
<td>7</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Dog1</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Dog2</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Chess</td>
<td>12</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>50</strong></td>
<td></td>
<td><strong>19</strong></td>
</tr>
</tbody>
</table>
Semex: Detecting Semantic Merge Conflicts with Variability-Aware Execution

Problem Statement

Given a set of parallel changes, identify early the actual (minimum) set of changes that directly cause a semantic merge conflict.

Is There a Shortcut?

Program + Options
Program with configuration options

variability-aware execution

Program + Options
Result for all configurations

aggregate results

Programs with concrete selections of options

brute-force execution

Results for individual configurations

Semex Approach

1. Encoding Parallel Changes
2. Variability-Aware Execution
Semantically conflicting files

Files in branch 1
Files in branch 2
Files in branch n

Variability-aware merged files

P1
P2
P3
P4
P5
P6

P1
P2
P3
P4
P5
P6

Semex Approach

Preliminary Experiment

<table>
<thead>
<tr>
<th>Example</th>
<th>Test cases</th>
<th>Branches</th>
<th>Semantic Conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>MathHelper</td>
<td>7</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Queue1</td>
<td>8</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Queue2</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Product</td>
<td>7</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Dog1</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Dog2</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Chess</td>
<td>12</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Total</td>
<td>50</td>
<td></td>
<td>19</td>
</tr>
</tbody>
</table>