Milu: A Higher Order Mutation Testing Tool

Yue Jia
University College London

Joint work with Mark Harman and William Langdon
Agenda

Why Higher Order Mutation Testing?
Search for interesting HOMs
Milu mutation testing tool
Scalability and Extendability
Performance Study
Mutation Testing

First Order Mutants: A single change

- Simple faults / FOMs

Higher Order Mutant: Multiple changes

- Multiple faults / HOMs
Mutation Testing

Subset of First Order
Mutants are used

No Higher Order
Mutants at all!
Mutation Testing

Subset of First Order Mutants are used

No Higher Order Mutants at all!
Higher Order Mutation Testing

The space of all mutants (first and higher order) is a search space,

We should apply search based optimisation techniques to find mutants that are fit for purpose.
Higher Order Mutation Testing

Search for a small set of highly fit mutants within an enormous space, rather than to enumerate a complete set.

- Tabu Search
- Genetic Programming
- Particle Swarm Optimization
- Ant Colonies
- Genetic Algorithms
- Hill Climbing
- Genetic Programming
- Simulated Annealing
- Greedy
- Random
- Estimation of Distribution Algorithms
- LP
Higher Order Mutation Testing

Search for a small set of highly fit mutants within an enormous space, rather than to enumerate a complete set.

- Tabu Search
- Ant Colonies
- Hill Climbing
- Simulated Annealing
- Particle Swarm Optimization
- Genetic Algorithms
- Estimation of Distribution Algorithms
- Greedy
- LP
- Random
Higher Order Mutation Testing

Search for a small set of highly fit mutants within an enormous space, rather than to enumerate a complete set.

- Tabu Search
- Ant Colonies
- Hill Climbing
- Simulated Annealing
- Particle Swarm Optimization
- Genetic Algorithms
- Genetic Programming
- Greedy
- LP
- Random
- Estimation of Distribution Algorithms
Interesting HOMs

Most common case

FOM a is killed by \{ 1, 2, 3, 4 \}

Test set T
Interesting HOMs

Most common case

FOM a is killed by \{ 1, 2, 3, 4 \}

FOM b is killed by \{ 3, 4, 5, 6 \}
Interesting HOMs

Most common case

FOM a is killed by \{1, 2, 3, 4\}
FOM b is killed by \{3, 4, 5, 6\}
HOM ab is killed by \{1, 2, 3, 4, 5, 6\}
Interesting HOMs

Subsuming HOM

FOM a is killed by \{1, 2, 3, 4\}
FOM b is killed by \{3, 4, 5, 6\}
HOM ab is killed by \{2, 3, 5\}
Interesting HOMs

Strongly Subsuming HOM

- FOM a is killed by \{ 1, 2, 3, 4 \}
- FOM b is killed by \{ 3, 4, 5, 6 \}
- HOM ab is killed by \{ 3 \}
Interesting HOMs

Anti Coupling Effect HOM

FOM a is killed by \{ 1, 2, 3, 4 \}
FOM b is killed by \{ 3, 4, 5, 6 \}
HOM ab is killed by \{ 7 \}
Interesting HOMs

Equivalent HOM

- FOM a is killed by \{ 1, 2, 3, 4 \}
- FOM b is killed by \{ 3, 4, 5, 6 \}
- HOM ab is killed by \{ \}

Test set T

- Ta: 1, 2, 3, 4
- Tb: 3, 4, 5, 6
Milu

Strong mutation

First and Higher Order Mutants

For C program

Test harness
Data Representation

Index

Position

0 0 0 0 0
Milu

Data Representation

Index
Position

1 0 0 0 0
Milu

Data Representation

Index

Position

1 0 0 0
0 0 2 0
Milu

Data Representation

Index
Position

1 0 0 0
0 0 2 0
1 0 2 0
Data Representation

Index
Position

Mutation Id
Milu

Data Representation

Index

<table>
<thead>
<tr>
<th>1 0 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 2 0</td>
</tr>
<tr>
<td>1 0 2 0</td>
</tr>
</tbody>
</table>

Position

Mutation Id

/ / -
Milu

Limitations

Cannot scale up

Hard to extend
Solutions

Implement the mutation component as a pass into GCC
GCC Internal

Front End | Middle End | Back End

C
C++
Java
Fortran

GENERIC | GIMPLE | RTL | Assembly
GCC Internal

Source

Front End | Middle End | Back End

C

C++

Java

Fortran

GENERIC | GIMPLE | RTL | Assembly

Milu pass
Gimple

SIMPLE IR of McCat compiler

3 address representation

Control flow lowering

Cleanups and simplification
Gimple

If (foo (a + b, c))
  c = b++ /a ;
return c

t1 = a + b;
t2 = foo (t1, c)
if (t2 != 0)
{
  t3 = b
  b = b+ 1
  c = t3 / a
}
return c
Implementation

typedef void (*plugin_callback_func)(void *gcc_data, void *user_data);

struct register_pass_info
{
    struct opt_pass *pass;
    const char *reference_pass_name;
    int ref_pass_instance_number;
    enum pass_positioning_ops pos_op;
};
Advantages

Supports all major languages: C, C++, Java, Fortran 95, Ada, Objective-C, Objective-C++, Go, etc

Large number of platforms
Demo
Step 4: Check mutants

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>File</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>mut_0</td>
<td>STRP</td>
<td>tcas.c</td>
<td>ALIM</td>
</tr>
<tr>
<td>mut_1</td>
<td>OAAN</td>
<td>tcas.c</td>
<td>Inhibit_Bias</td>
</tr>
<tr>
<td>mut_2</td>
<td>OAAN</td>
<td>tcas.c</td>
<td>Inhibit_Bias</td>
</tr>
<tr>
<td>mut_3</td>
<td>OAAN</td>
<td>tcas.c</td>
<td>Inhibit_Bias</td>
</tr>
<tr>
<td>mut_4</td>
<td>OAAN</td>
<td>tcas.c</td>
<td>Inhibit_Bias</td>
</tr>
<tr>
<td>mut_5</td>
<td>OALN</td>
<td>tcas.c</td>
<td>Inhibit_Bias</td>
</tr>
<tr>
<td>mut_6</td>
<td>OALN</td>
<td>tcas.c</td>
<td>Inhibit_Bias</td>
</tr>
<tr>
<td>mut_7</td>
<td>OARN</td>
<td>tcas.c</td>
<td>Inhibit_Bias</td>
</tr>
<tr>
<td>mut_8</td>
<td>OARN</td>
<td>tcas.c</td>
<td>Inhibit_Bias</td>
</tr>
<tr>
<td>mut_9</td>
<td>OARN</td>
<td>tcas.c</td>
<td>Inhibit_Bias</td>
</tr>
<tr>
<td>mut_10</td>
<td>OARN</td>
<td>tcas.c</td>
<td>Inhibit_Bias</td>
</tr>
<tr>
<td>mut_11</td>
<td>OARN</td>
<td>tcas.c</td>
<td>Inhibit_Bias</td>
</tr>
<tr>
<td>mut_12</td>
<td>OARN</td>
<td>tcas.c</td>
<td>Inhibit_Bias</td>
</tr>
<tr>
<td>mut_13</td>
<td>STRP</td>
<td>tcas.c</td>
<td>Inhibit_Bias</td>
</tr>
<tr>
<td>mut_14</td>
<td>OEBN</td>
<td>tcas.c</td>
<td>Non_Cross</td>
</tr>
<tr>
<td>mut_15</td>
<td>OEBN</td>
<td>tcas.c</td>
<td>Non_Cross</td>
</tr>
<tr>
<td>mut_16</td>
<td>OEBN</td>
<td>tcas.c</td>
<td>Non_Cross</td>
</tr>
<tr>
<td>mut_17</td>
<td>ORRN</td>
<td>tcas.c</td>
<td>Non_Cross</td>
</tr>
</tbody>
</table>

Mutant

```c
int Inhibit_Biased_Climb ()
{
    return (Climb_Inhibit ? Up_Separation - 100 : Up_Separation);
}

bool Non_Crossing_Biased_Climb ()
{
    int upward_preferred;
    int upward_crossing_situation;
}
```

Original

```c
int Inhibit_Biased_Climb ()
{
    return (Climb_Inhibit ? Up_Separation + 100 : Up_Separation);
}

bool Non_Crossing_Biased_Climb ()
{
    int upward_preferred;
    int upward_crossing_situation;
}
```
### Mutation Score

Mutation Score: 0.303867 (11 test cases, 181 mutants)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Killing TCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>mut_142</td>
<td>Oeba</td>
<td>8</td>
</tr>
<tr>
<td>mut_143</td>
<td>Oeba</td>
<td>8</td>
</tr>
<tr>
<td>mut_149</td>
<td>Oeba</td>
<td>3</td>
</tr>
<tr>
<td>mut_150</td>
<td>Oeba</td>
<td>3</td>
</tr>
<tr>
<td>mut_151</td>
<td>OLLN</td>
<td>3</td>
</tr>
<tr>
<td>mut_100</td>
<td>ORRN</td>
<td>2</td>
</tr>
<tr>
<td>mut_102</td>
<td>ORRN</td>
<td>2</td>
</tr>
<tr>
<td>mut_116</td>
<td>ORRN</td>
<td>2</td>
</tr>
<tr>
<td>mut_117</td>
<td>ORRN</td>
<td>2</td>
</tr>
<tr>
<td>mut_154</td>
<td>Oeba</td>
<td>2</td>
</tr>
<tr>
<td>mut_155</td>
<td>Oeba</td>
<td>2</td>
</tr>
<tr>
<td>mut_156</td>
<td>OLLN</td>
<td>2</td>
</tr>
<tr>
<td>mut_24</td>
<td>Oeba</td>
<td>1</td>
</tr>
<tr>
<td>mut_25</td>
<td>Oeba</td>
<td>1</td>
</tr>
<tr>
<td>mut_27</td>
<td>OLLN</td>
<td>1</td>
</tr>
<tr>
<td>mut_28</td>
<td>ORRN</td>
<td>1</td>
</tr>
<tr>
<td>mut_30</td>
<td>ORRN</td>
<td>1</td>
</tr>
<tr>
<td>mut_31</td>
<td>ORRN</td>
<td>1</td>
</tr>
<tr>
<td>mut_36</td>
<td>Oeba</td>
<td>1</td>
</tr>
</tbody>
</table>

### Test Suite

```java
//---test1
int
test1()
{
    Cur_Vertical_Sep = 958;
    High_Confidence = 1;
    Two_of_Three_Reports_Valid = 1;
    Own_Tracked_Alt = 2597;
    Own_Tracked_Alt_Rate = 574;
    Other_Tracked_Alt = 4253;
    Alt_Layer_Value = 0;
    Up_Separation = 399;
}
```

### Mutant Original

```java
bool
Own_Above_Threat()
{
    return (Other_Tracked_Alt >= Own_Tracked_Alt);
}
```

```java
int
alt_sep_test()
{
    bool enabled, tcas_equipped, intent_not_known;
```
Performance

Source → Mutant Source → Mutant Binary → Test set
Source  Mutant Source  Mutant Binary  Test set

100-300 Loc, 1000 mutants, 100 test

7 secs  1 min  5-10 mins

Binary Injection  Gimple  MSG  Test harness
Conclusion

GCC Pass / Plugin

Mutating real world program

Multiple language mutation

Multiple platform

http://gcc.gnu.org/onlinedocs/gccint/Plugins.html

http://www.inf.kcl.ac.uk/pg/jiayue/milu/