Milu: A Higher Order Mutation Testing Tool

Yue Jia
University College London

Joint work with Mark Harman and William Langdon

CREST

Agenda

Why Higher Order Mutation Testing?

Search for interesting HOMs

Milu mutation testing tool

Scalability and Extendability

Performance Study

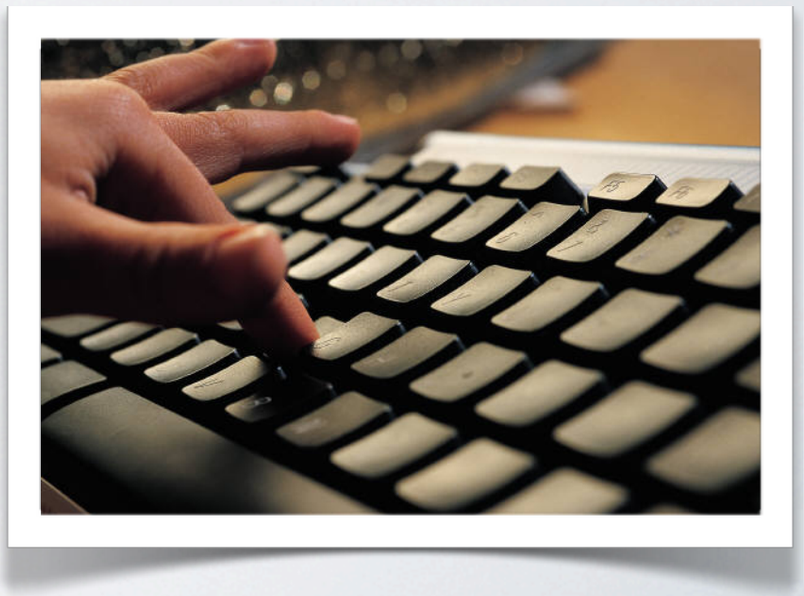
Mutation Testing

First Order Mutants : A single change

Simple faults / FOMs

Higher Order Mutant : Multiple changes

Multiple faults / HOMs



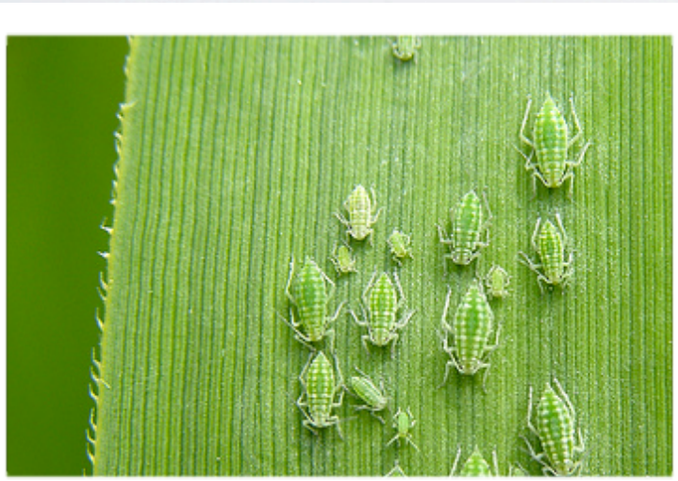
Mutation Testing

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

Ant Colonies

Particle Swarm Optimization

Genetic Algorithms

Hill Climbing

Genetic Programming

Simulated Annealing

Random

Greedy

LP

Estimation of Distribution Algorithms

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

Particle Swarm Optimization

Genetic Algorithms

Hill Climbing

Genetic Programming

Simulated Annealing

Random

Greedy

LP

Estimation of Distribution Algorithms

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

Particle Swarm Optimization

Genetic Algorithms

Hill Climbing

Genetic Programming

Simulated Annealing

Random

Greedy

LP

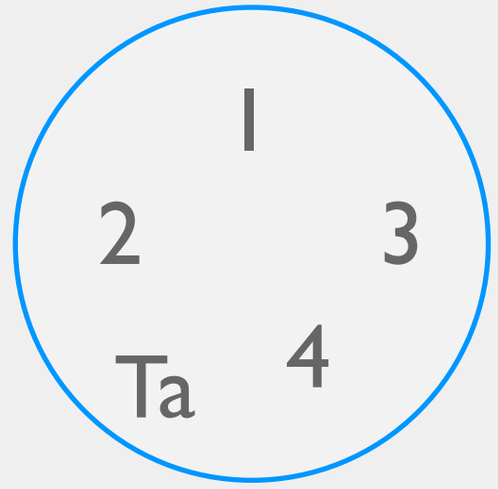
Estimation of Distribution Algorithms

Interesting HOMs

Most common case

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

Test set T



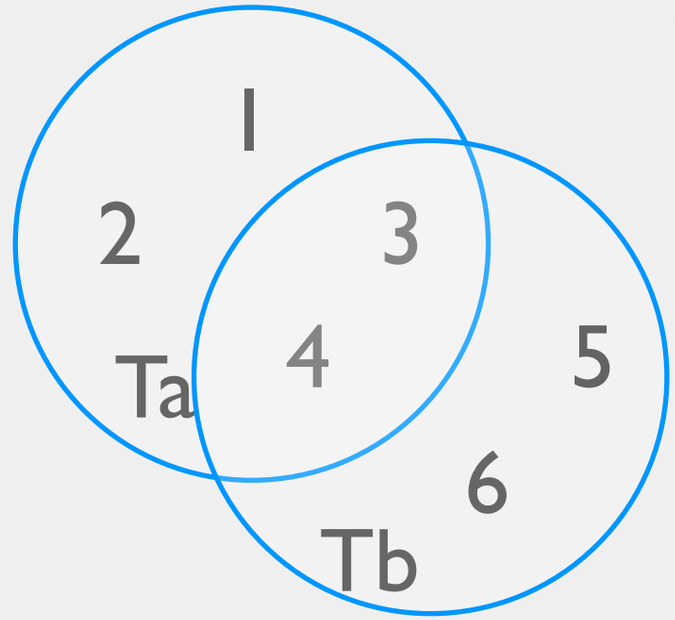
Interesting HOMs

Most common case

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

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

Test set T



Interesting HOMs

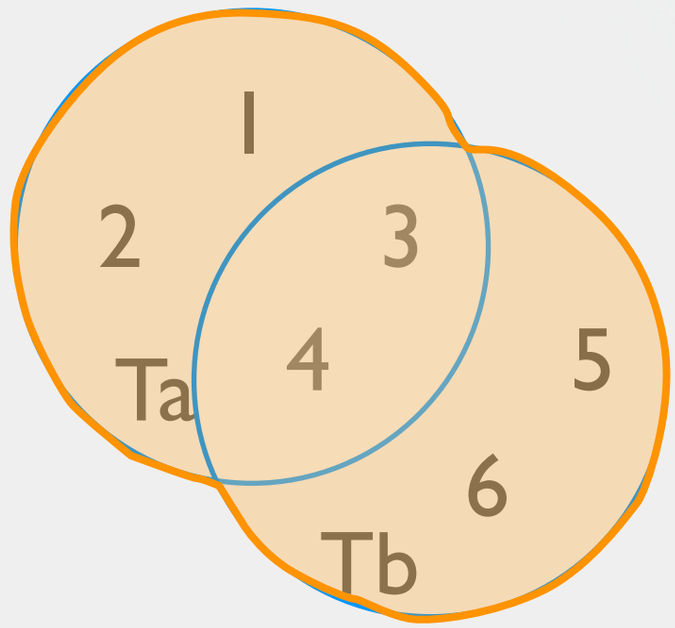
Most common case

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

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

HOM ab is killed by
{1, 2, 3, 4, 5, 6}

Test set T



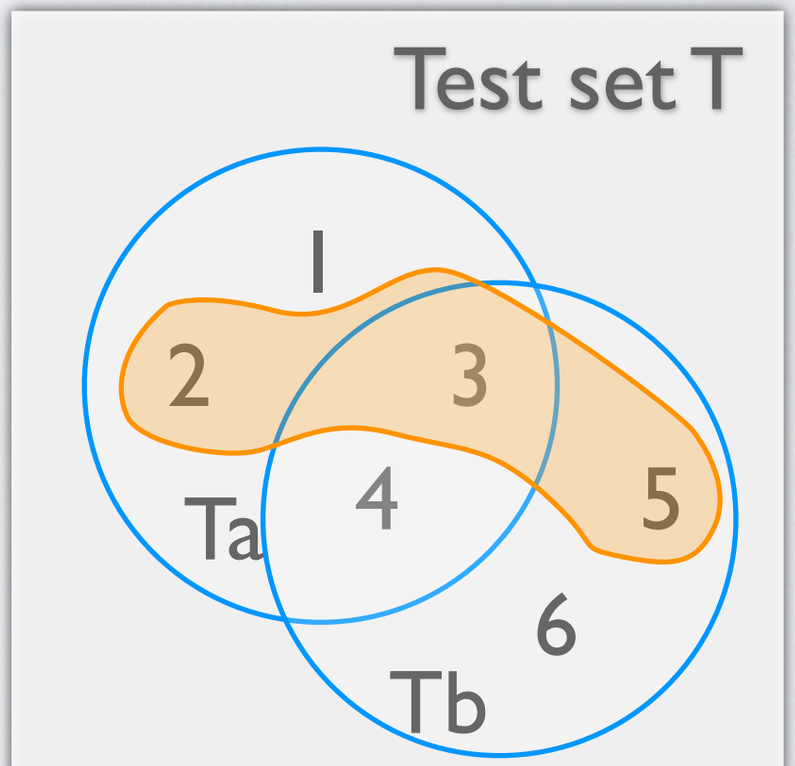
Interesting HOMs

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}



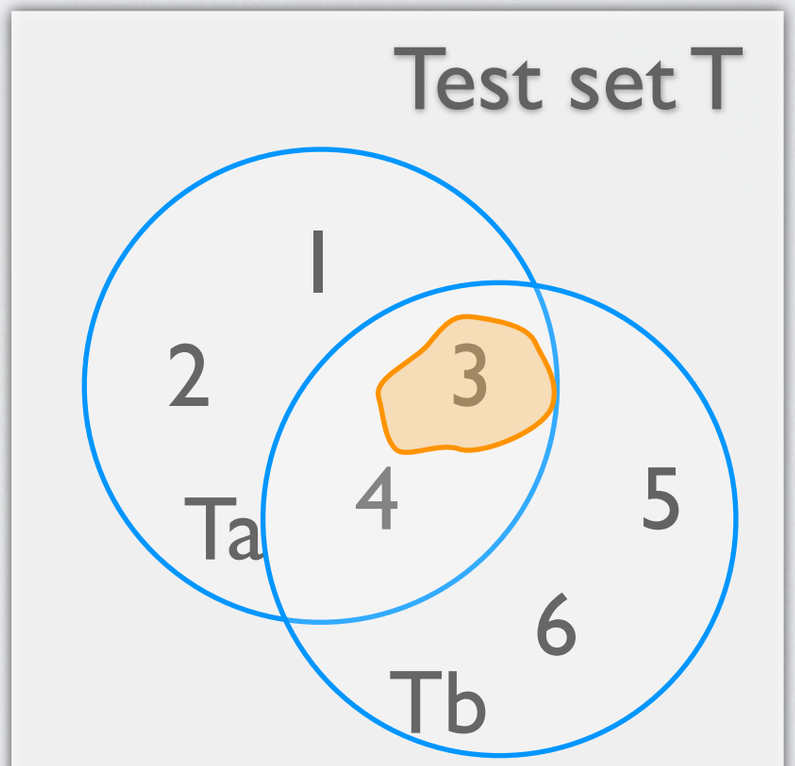
Interesting HOMs

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 }



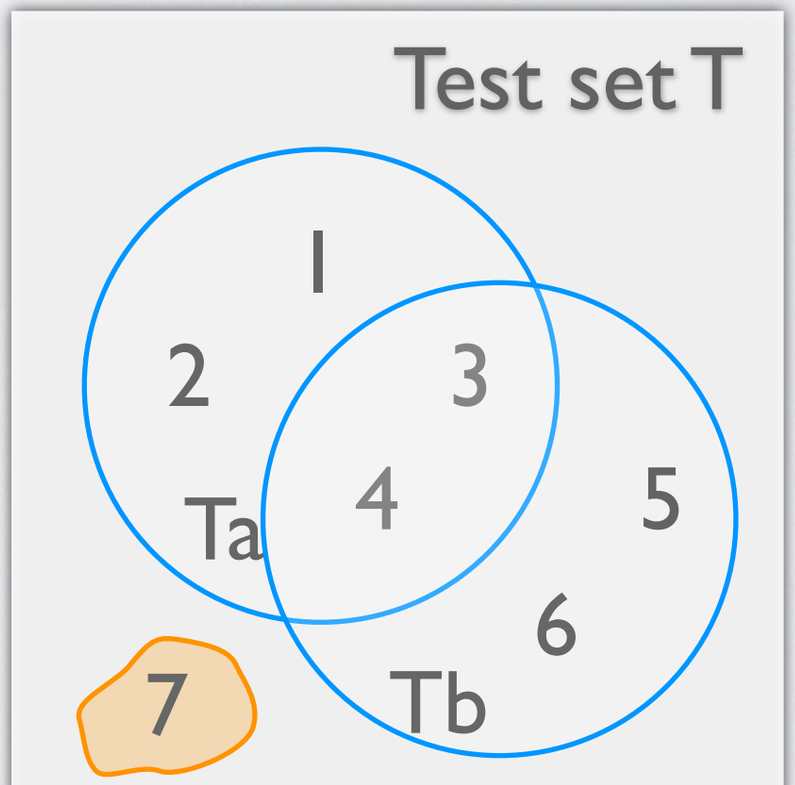
Interesting HOMs

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 }



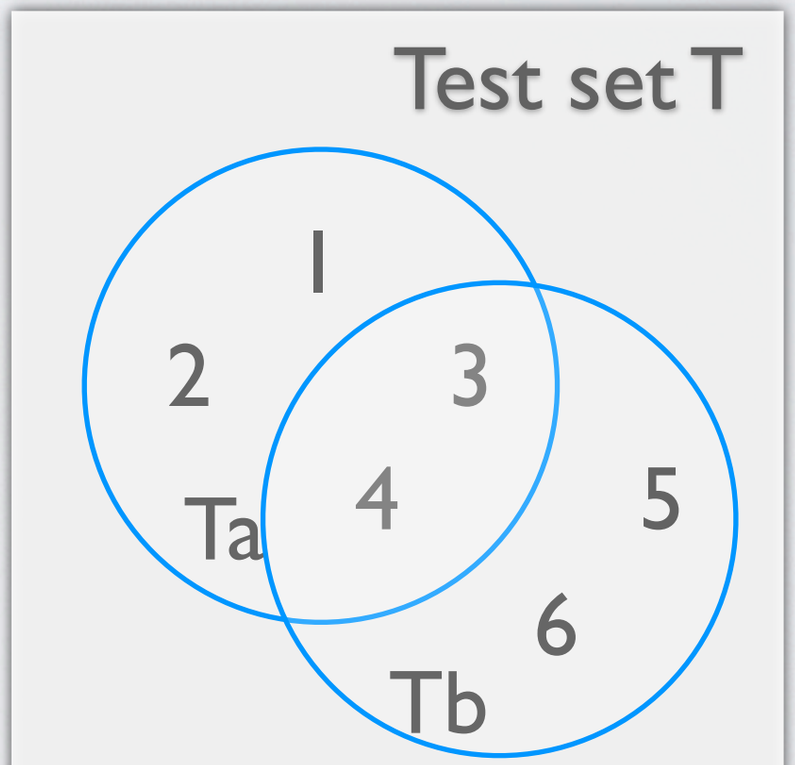
Interesting HOMs

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
{ }



Milu

Strong mutation

First and Higher Order Mutants

For C program

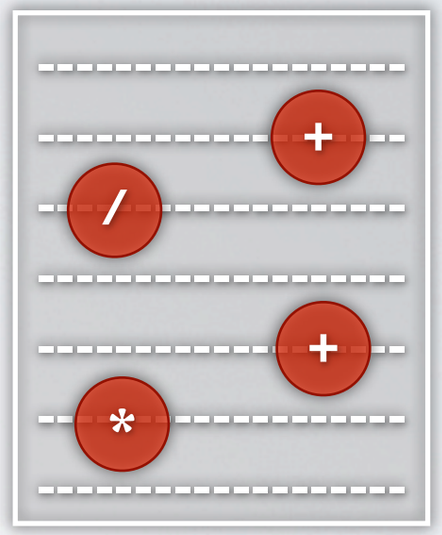
Test harness

Milu

Data Representation

Index

Position

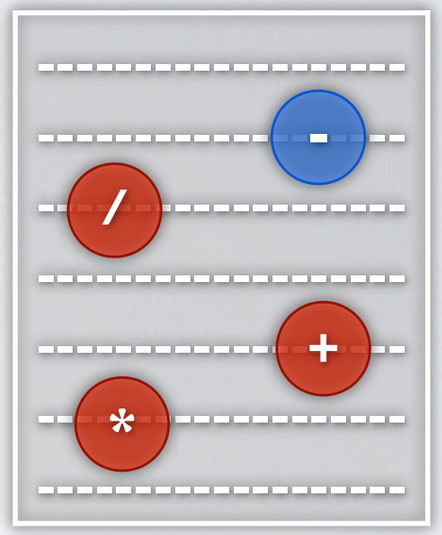


Milu

Data Representation

Index

Position

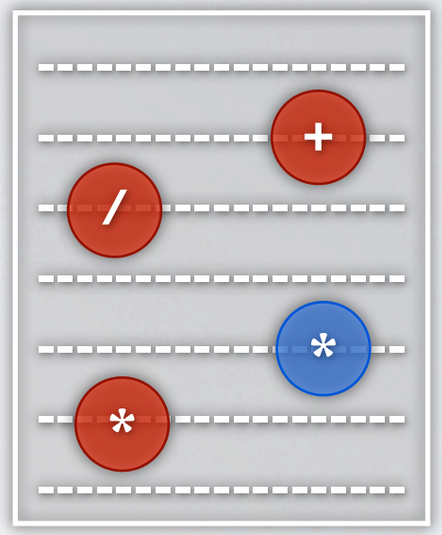


Milu

Data Representation

Index

Position

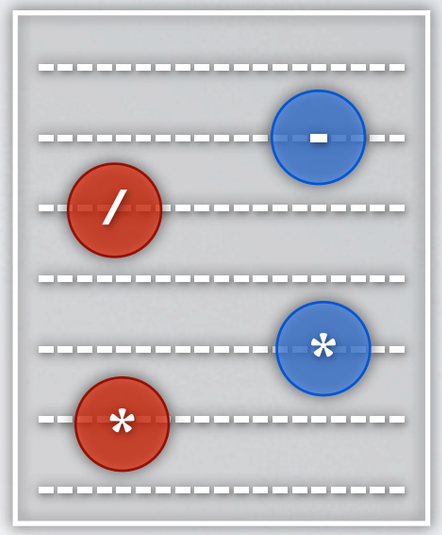


Milu

Data Representation

Index

Position



1	0	0	0

0	0	2	0

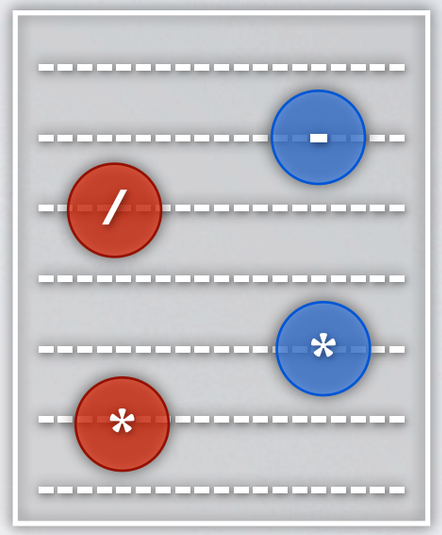
1	0	2	0

Milu

Data Representation

Index

Position



1 0 0 0

0 0 2 0

1 0 2 0

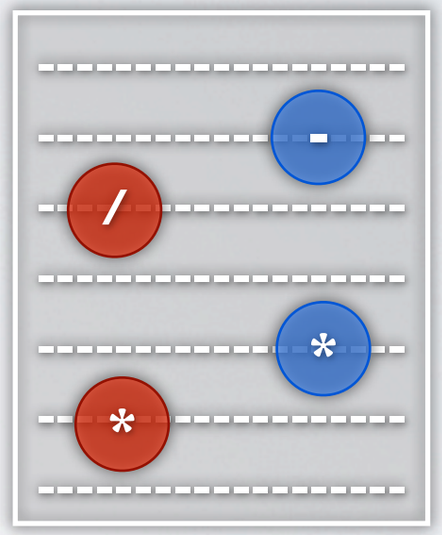
Mutation Id

Milu

Data Representation

Index

Position



1 0 0 0

0 0 2 0

1 0 2 0

Mutation Id

Milu

Limitations

Cannot scale up

Hard to extend

Solutions

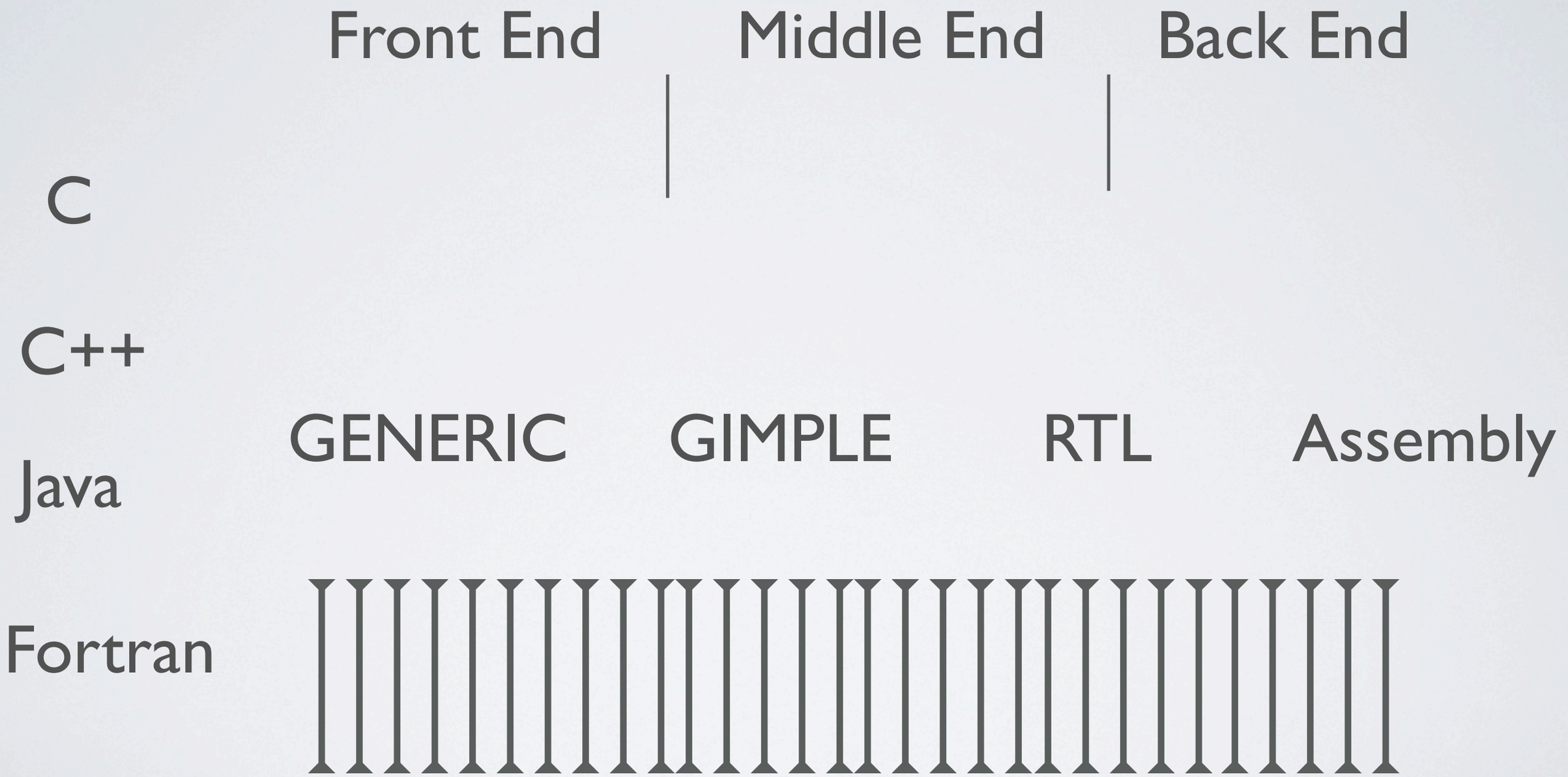
Implement the mutation component as a pass into GCC

GCC Internal

	Front End	Middle End	Back End
C			
C++			
Java	GENERIC	GIMPLE	RTL
Fortran			

Assembly

GCC Internal



GCC Internal

Source | Front End | Middle End | Back End

C

C++

Java

Fortran

GENERIC GIMPLE RTL Assembly

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

Milu

Step 4: Check mutants

Name	Type	File	Function
mut_0	STRP	tcas.c	ALIM
mut_1	OAAN	tcas.c	Inhibit_Bia
mut_2	OAAN	tcas.c	Inhibit_Bia
mut_3	OAAN	tcas.c	Inhibit_Bia
mut_4	OAAN	tcas.c	Inhibit_Bia
mut_5	OALN	tcas.c	Inhibit_Bia
mut_6	OALN	tcas.c	Inhibit_Bia
mut_7	OARN	tcas.c	Inhibit_Bia
mut_8	OARN	tcas.c	Inhibit_Bia
mut_9	OARN	tcas.c	Inhibit_Bia
mut_10	OARN	tcas.c	Inhibit_Bia
mut_11	OARN	tcas.c	Inhibit_Bia
mut_12	OARN	tcas.c	Inhibit_Bia
mut_13	STRP	tcas.c	Inhibit_Bia
mut_14	OEBA	tcas.c	Non_Cross
mut_15	OEBA	tcas.c	Non_Cross
mut_16	OEBA	tcas.c	Non_Cross
mut_17	ORRN	tcas.c	Non_Cross

Mutant

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

bool
Non_Crossing_Biased_Climb ()
{
  int upward_preferred;
```

Original

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

bool
Non_Crossing_Biased_Climb ()
{
  int upward_preferred;
```

Generated 181 mutants

Save & Exit

Run Test

Milu

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

Name	Type	KillingTCs
mut_142	OEBA	8
mut_143	OEBA	8
mut_149	OEBA	3
mut_150	OEBA	3
mut_151	OLLN	3
mut_100	ORRN	2
mut_102	ORRN	2
mut_116	ORRN	2
mut_117	ORRN	2
mut_154	OEBA	2
mut_155	OEBA	2
mut_156	OLLN	2
mut_24	OEBA	1
mut_25	OEBA	1
mut_27	OLLN	1
mut_28	ORRN	1
mut_30	ORRN	1
mut_31	ORRN	1
mut_36	OEBA	1

Test suite

```
//----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**

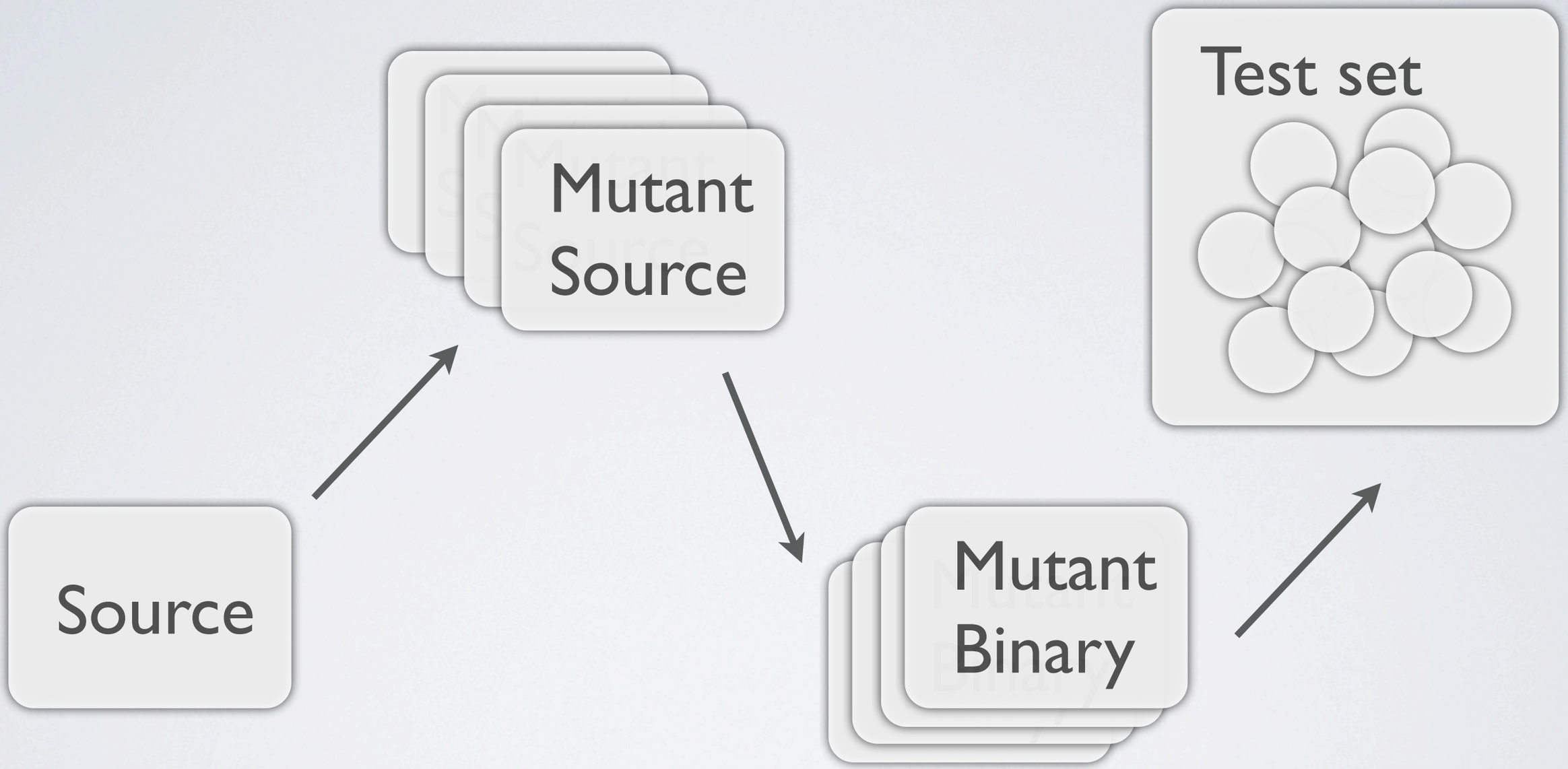
```
bool
Own_Above_Threat ()
{
  return (Other_Tracked_Alt >= Own_Tracked_Alt);
}

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

55 mutants are killed and 126 mutants are surviving

Quit Save

Performance



Source

Mutant Source

Mutant Binary

Test set

100-300 Loc , 1000 mutants, 100 test

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/