Search-Based Test Generation

Gordon Fraser, University of Sheffield

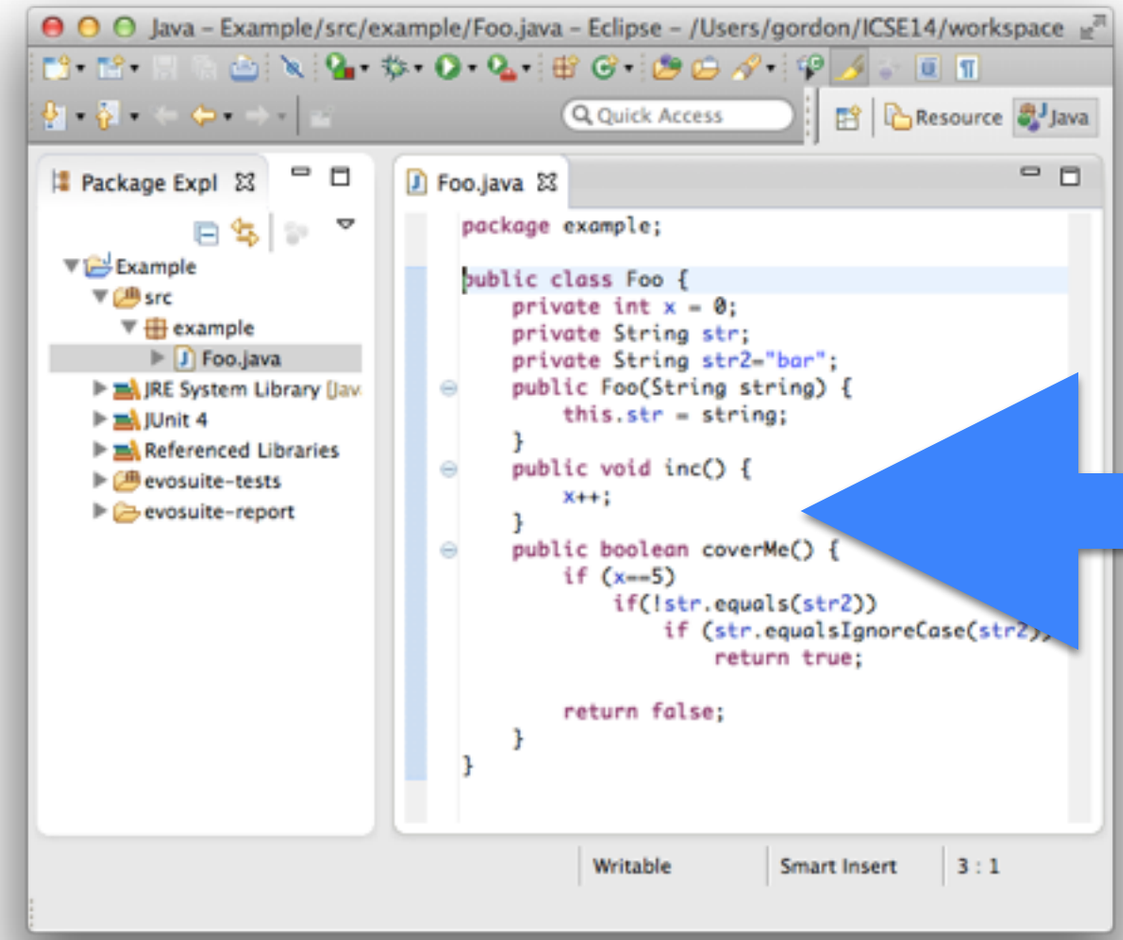
Acknowledgements

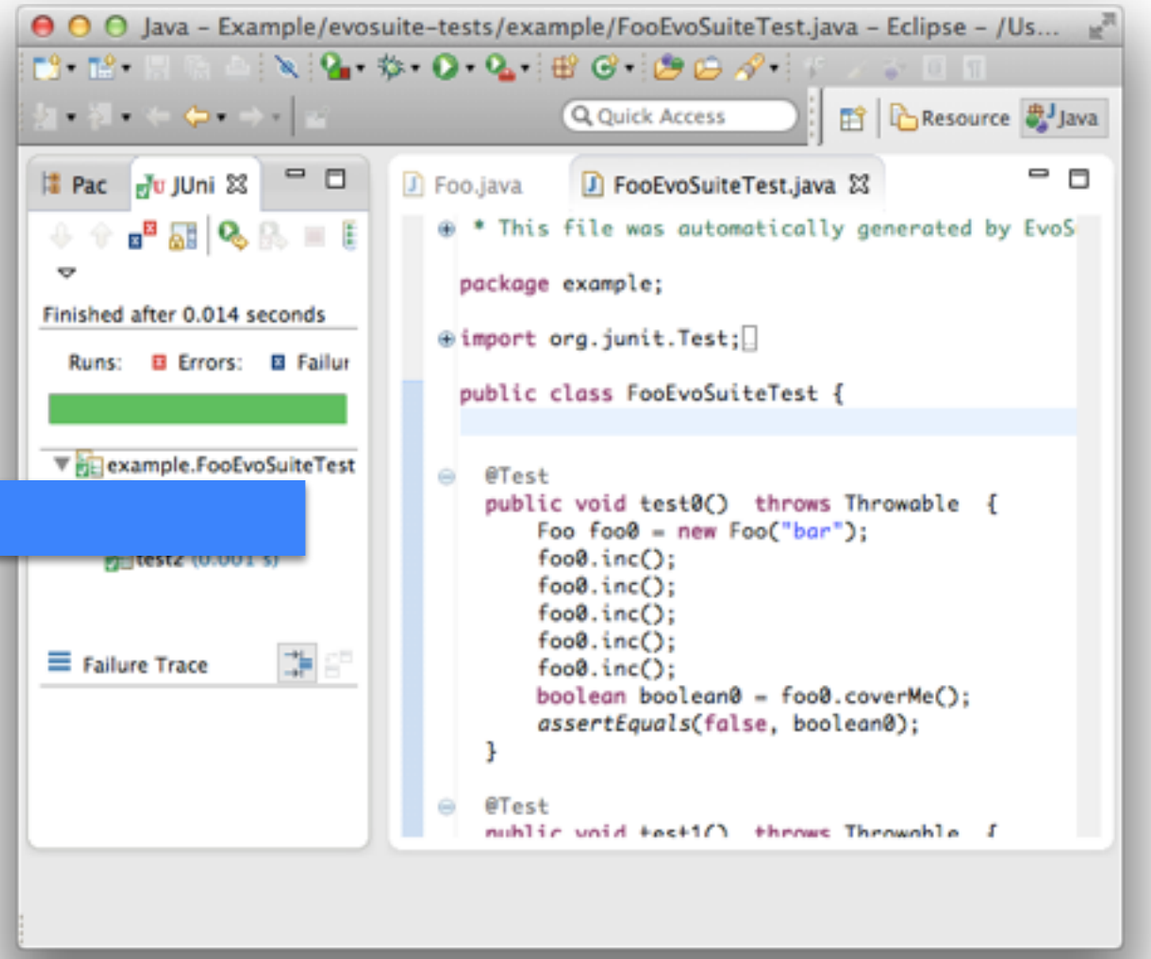
The material in some of these slides has kindly been provided by:

Phil McMinn (University of Sheffield)
Mark Harman (KCL/UCL)
Joachim Wegener (Berner & Matner)

Contents

Contents

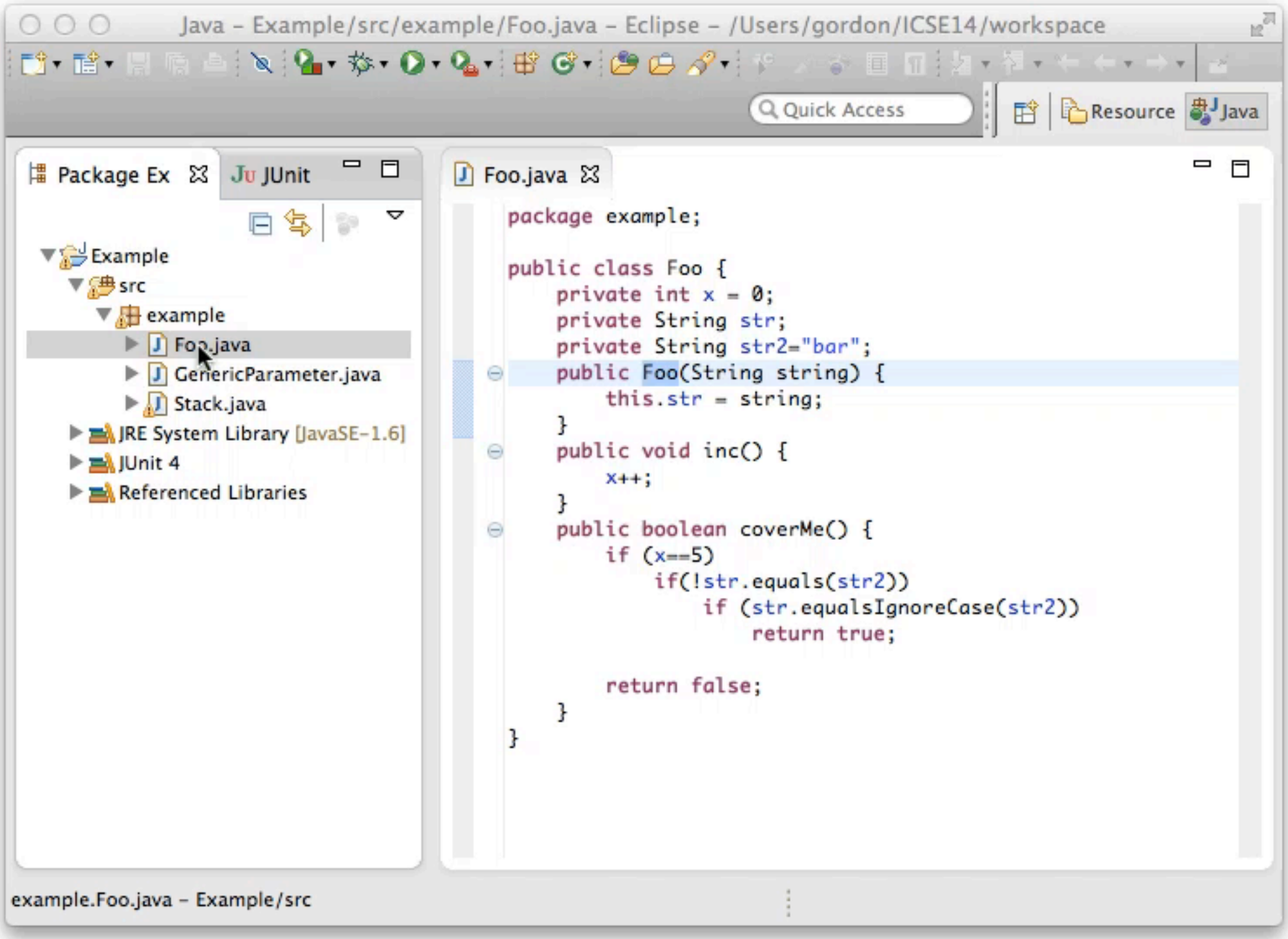


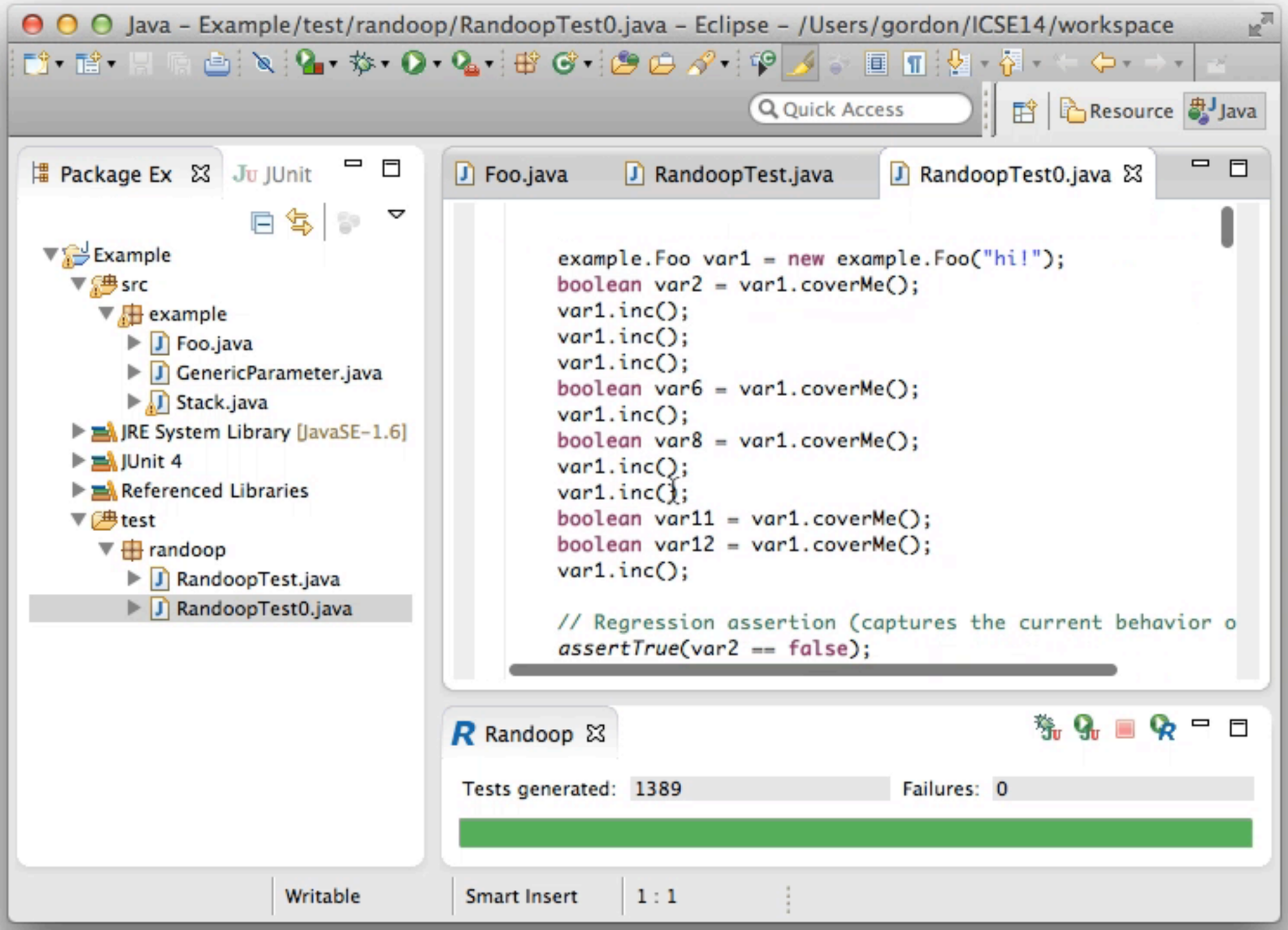


Source code

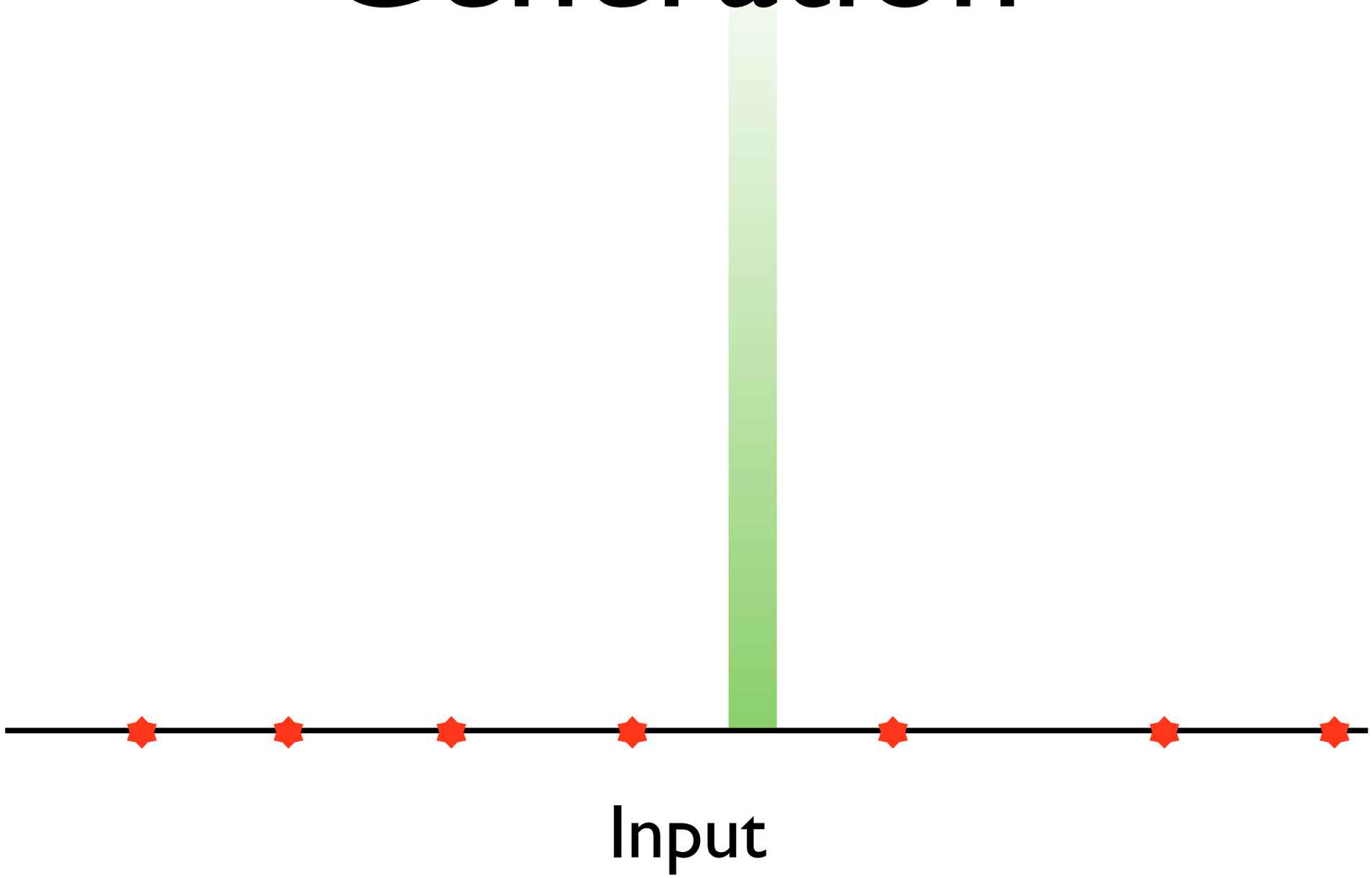
Tests

Automated test generation


Java – Example/src/example/Foo.java – Eclipse – /Users/gordon/ICSE14/workspace
Quick Access
Resource
Java
Package Ex
JUnit
Example
src
example
Foo.java
GenericParameter.java
Stack.java
JRE System Library [JavaSE-1.6]
JUnit 4
Referenced Libraries
Foo.java
package example;
public class Foo {
private int x = 0;
private String str;
private String str2="bar";
public Foo(String string) {
this.str = string;
}
public void inc() {
x++;
}
public boolean coverMe() {
if (x==5)
if(!str.equals(str2))
if (str.equalsIgnoreCase(str2))
return true;
return false;
}
}
example.Foo.java – Example/src


Java – Example/test/randoop/RandoopTest0.java – Eclipse – /Users/gordon/ICSE14/workspace
Quick Access
Resource
Java
Package Ex
JUnit
Example
src
example
Foo.java
GenericParameter.java
Stack.java
JRE System Library [JavaSE-1.6]
JUnit 4
Referenced Libraries
test
randoop
RandoopTest.java
RandoopTest0.java
Foo.java
RandoopTest.java
RandoopTest0.java
example.Foo var1 = new example.Foo("hi!");
boolean var2 = var1.coverMe();
var1.inc();
var1.inc();
var1.inc();
boolean var6 = var1.coverMe();
var1.inc();
boolean var8 = var1.coverMe();
var1.inc();
var1.inc();
boolean var11 = var1.coverMe();
boolean var12 = var1.coverMe();
var1.inc();
// Regression assertion (captures the current behavior o
assertTrue(var2 == false);
Randoop
Tests generated: 1389
Failures: 0
Writable
Smart Insert
1 : 1

Random Test Data Generation



Pex for fun – from Microsc ×

pexforfun.com

My Duels▼ | Settings▼ | Sign In

Curious? Learn More!

Pex for fun

Random Puzzle | Learn | APCS | New | 1,525,714 clicked 'Ask Pex!' | C# | Visual Basic | F#

The code is a puzzle. Do you understand what the code does? Click **Ask Pex!** to find out.

```
using System;

public class Foo {
        private int x = 0;
        private String str;
        private String str2="bar";
        public Foo(String str){
                this.str = str;
        }
        public void inc() {
                x++;
        }
        public bool coverMe() {
                if (x==5)
                        if(!str.Equals(str2))
                                if (str.Equals(str2
```

Ask Pex!

Permalink

Click Here!

Community Live Feed Publications About

Microsoft Research RiSE

for Windows Phone

Pex for fun – from Microso

pexforfun.com

Curious? Learn More!

Pex for fun

Random Puzzle | Learn | APCS | New | 1,525,714 clicked 'Ask Pex!' | C# | Visual Basic | F#

The code is a puzzle. Do you understand what the code does? Click **Ask Pex!** to find out.

```
                this.str = str;
        }
        public void inc() {
                x++;
        }
        public bool coverMe() {
                if (x==5)
                        if(!str.Equals(str2))
                                if (str.Equals(str2,
                                    StringComparison.OrdinalIgnoreCase))
                                        return true;

                return false;
        }
}
```

Ask Pex!

Click Here!

Permalink

Community | Live Feed | Publications | About

Microsoft Research

RiSE

for Windows Phone

Puzzle history (show Permalinks): initial;

Pex for fun – from Microso

pexforfun.com

Pex for fun

Random Puzzle | Learn | APCS | New | 1,525,714 clicked 'Ask Pex!' | C# | Visual Basic | F#

The code is a puzzle. Do you understand what the code does? Click **Ask Pex!** to find out.

```
        public bool coverMe() {
                if (x==5)
                        if(!str.Equals(str2))
                                if (str.Equals(str2,
                                    StringComparison.OrdinalIgnoreCase))
                                        return true;

                return false;
        }

        public static bool Puzzle(Foo foo) {
          return foo.coverMe();
        }

}
```

Ask Pex! *Done. 2 interesting inputs found.* How does Pex work? Permalink

	foo	result	Output/Exception	Error Message
✗	null		NullReferenceException	Object reference not set to an instance of an object.
✓	new Foo{}	false		

Coding Duel Name: Turn This Puzzle Into A Coding Duel Help

Pex and Moles

Generating vs Checking

Conventional Software Testing Research

Write a method to construct test cases

Search-Based Testing

Write a method
to determine how good a test case is

Generating vs Checking

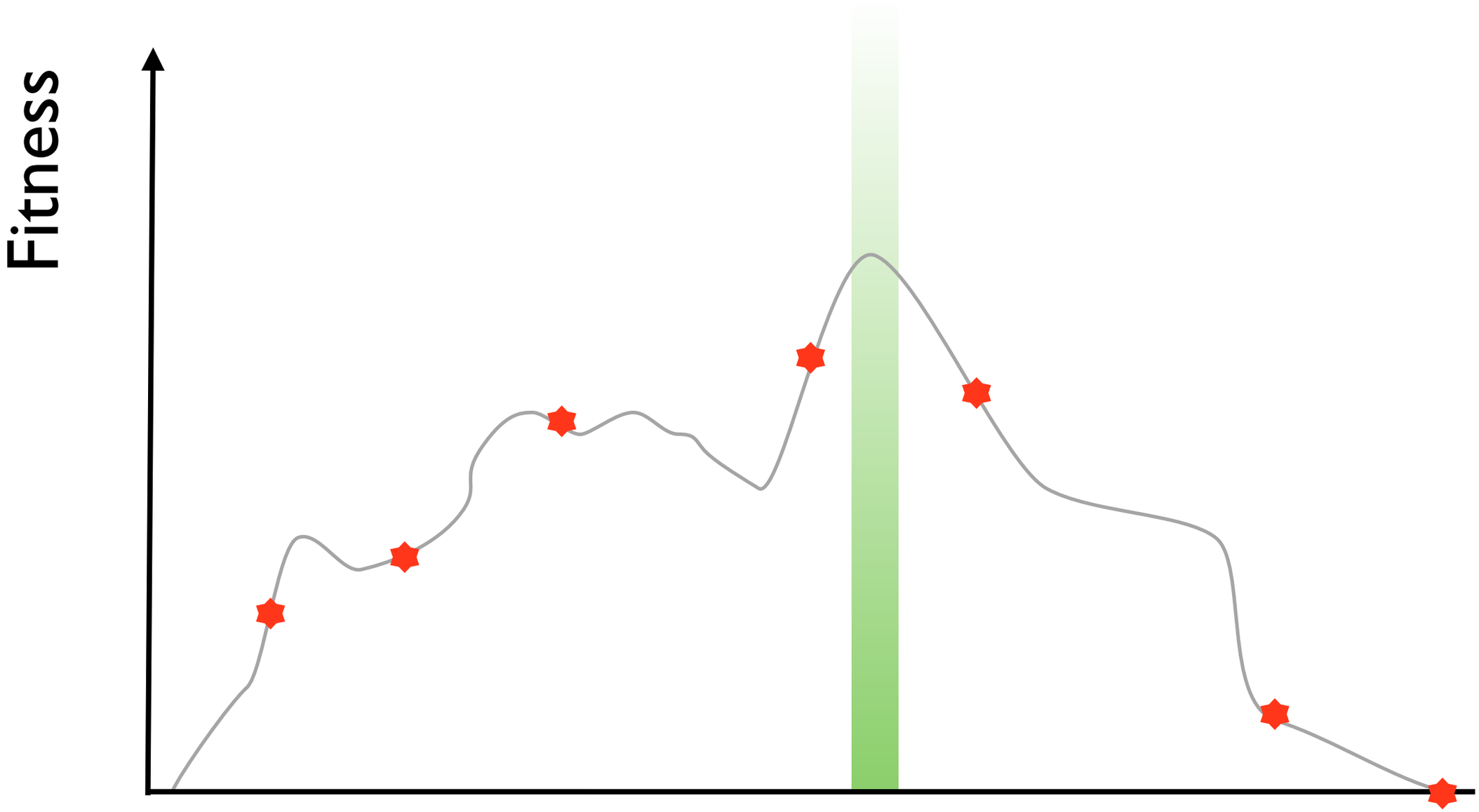
Conventional Software Testing Research

Write a method to construct test cases

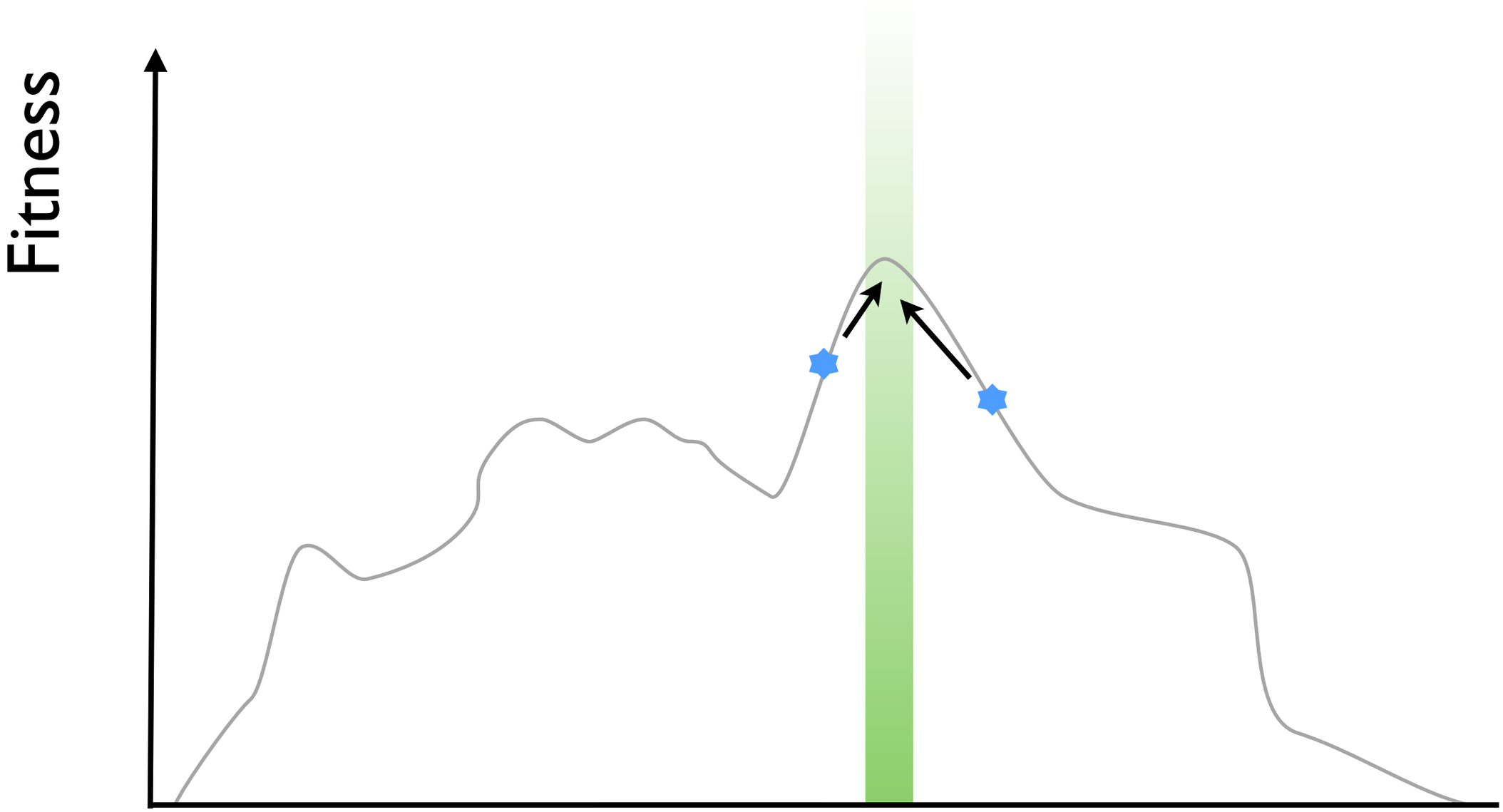
Search-Based Testing

Write a fitness function
to determine how good a test case is

Fitness-guided search

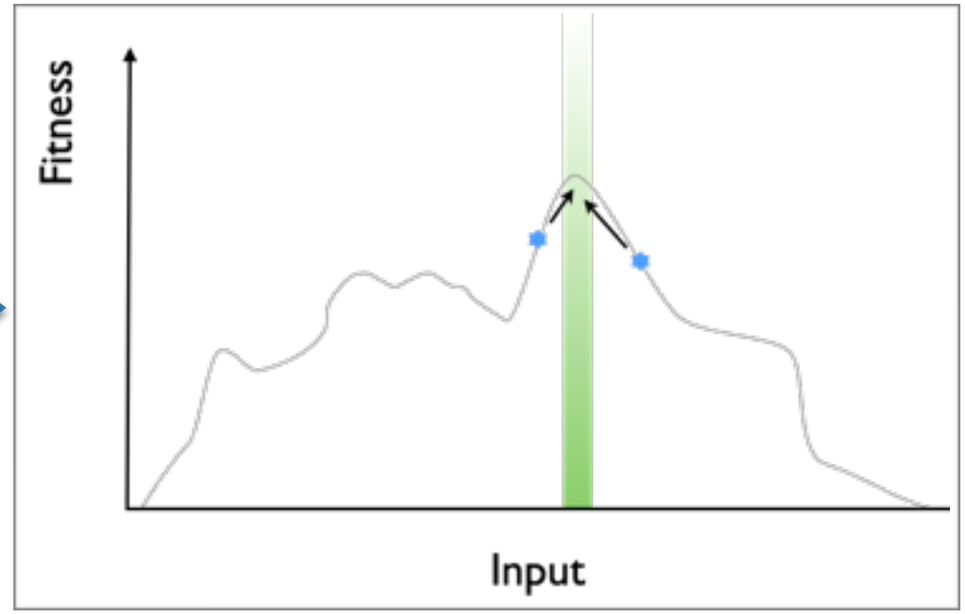


Fitness-guided search



Components of an SBST Tool

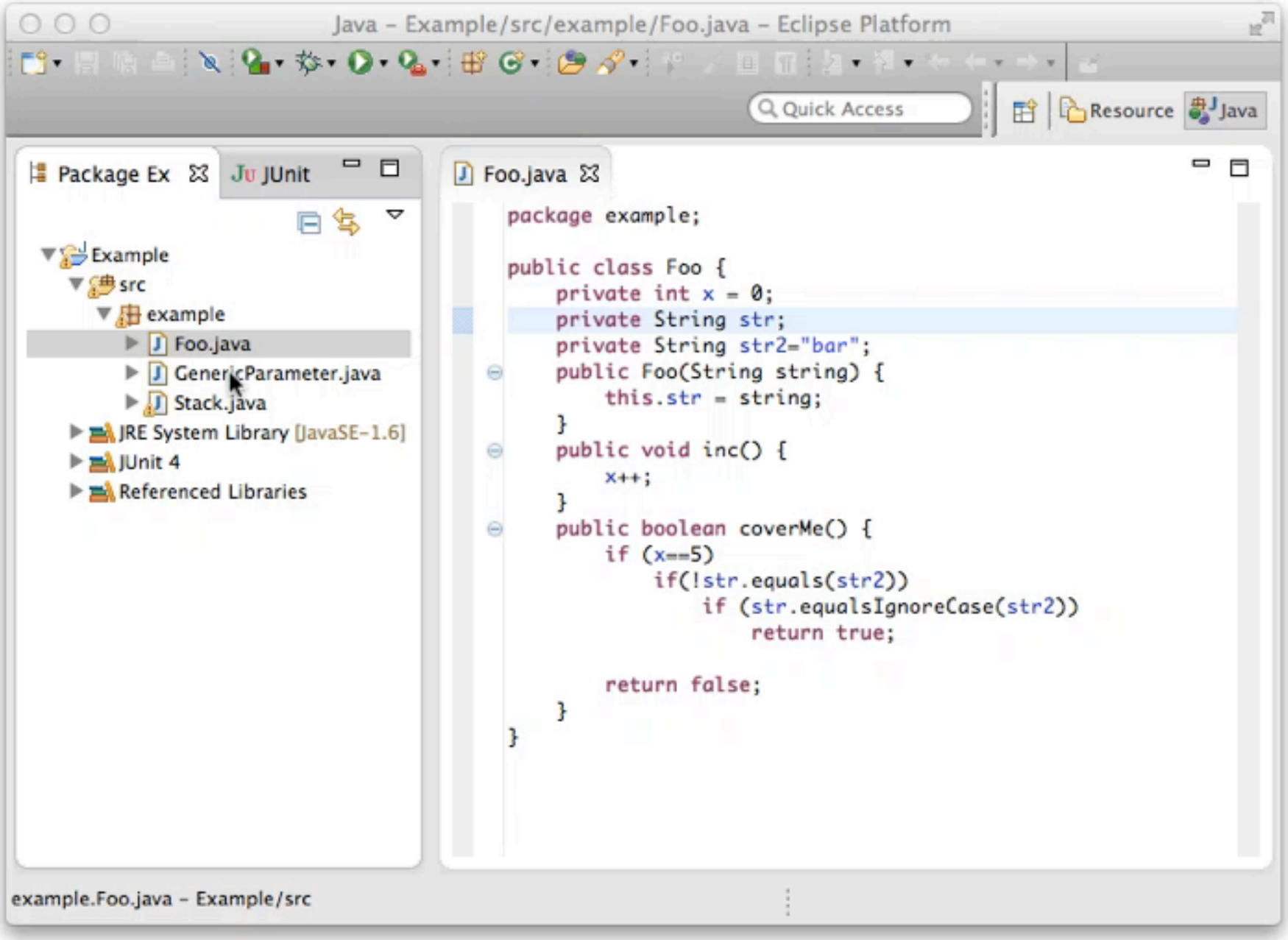
```
def testMe(x, y):
    if x == 2 * (y + 1):
        return True
    else:
        return False
```



- Search Algorithm
- Representation
- Fitness Function

Components of an SBST Tool

Search Algorithm	Meta-heuristic algorithm
Representation	Encoding of the problem solution
Search Operators	Modifications of encoded solutions
Fitness Function	Measure how good a candidate solution is
Test Execution	Execute tests
Instrumentation	Collect data/traces for fitness calculation during execution


Java – Example/src/example/Foo.java – Eclipse Platform
Quick Access
Resource
Java
Package Ex
JUnit
Example
src
example
Foo.java
GenericParameter.java
Stack.java
JRE System Library [JavaSE-1.6]
JUnit 4
Referenced Libraries
Foo.java
package example;
public class Foo {
private int x = 0;
private String str;
private String str2="bar";
public Foo(String string) {
this.str = string;
}
public void inc() {
x++;
}
public boolean coverMe() {
if (x==5)
if(!str.equals(str2))
if (str.equalsIgnoreCase(str2))
return true;
return false;
}
}
example.Foo.java - Example/src

Coverage Progress

83.64 %

Address Book

New contact

New category

First name | Last name | E-mail | Phone | Mobile

All

Create a category

Category name:

eO"l" already exists

Abbrechen | OK

First name

Last name

Phone

Mobile

Notes

Second e-mail

URL

Apply

Ke Mao, Mark Harman, Yue Jia. Sapienz: Multi-objective automated testing for Android applications, ISSTA 2016

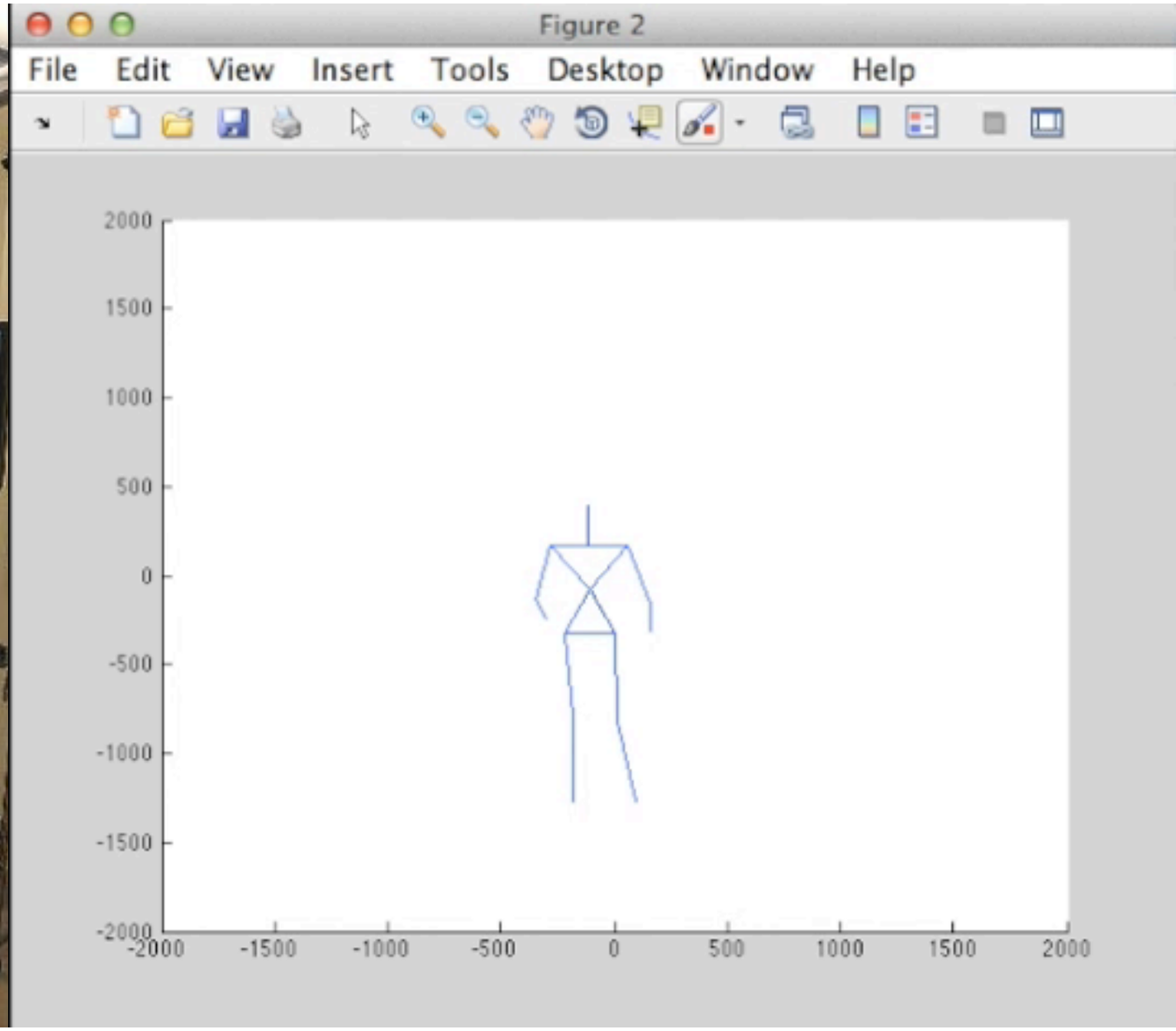


Figure 2
File Edit View Insert Tools Desktop Window Help
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2000 -1500 -1000 -500 0 500 1000 1500 2000

First publication on SBST

Webb Miller

David Spooner

Automatic Generation of Floating-Point Test Data
IEEE Transactions on Software Engineering, 1976

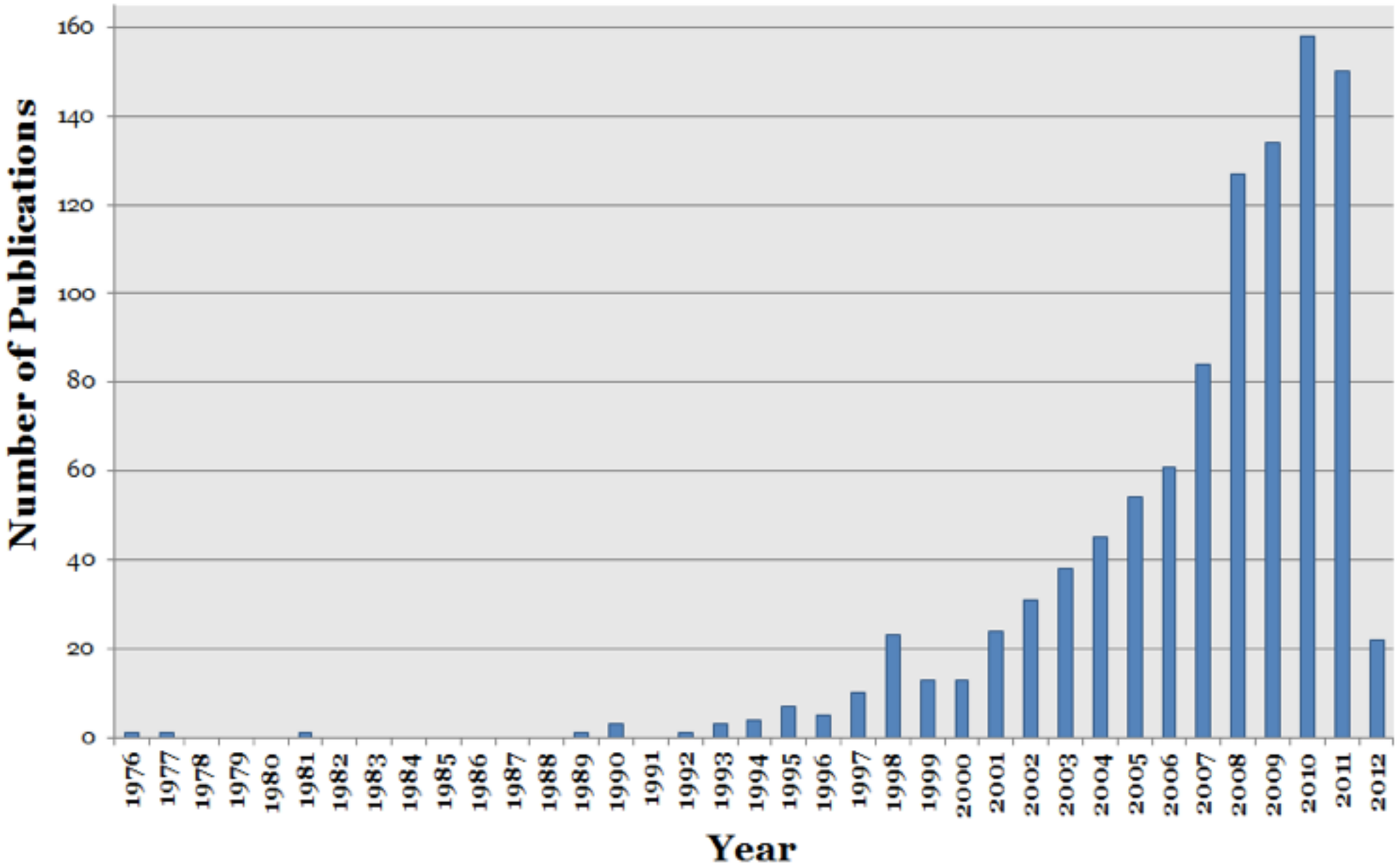
First publication on SBST

Webb Miller

David Spooner

Automatic Generation of Floating-Point Test Data
IEEE Transactions on Software Engineering, 1976

Publications since 1976



source: SEBASE publications repository

http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/

Getting started in SBSE

M. Harman and B. Jones: Search-based software engineering. Information and Software Technology, 43(14): 833–839, 2001.

D. Whitley: An overview of evolutionary algorithms: Practical issues and common pitfalls. Information and Software Technology, 43(14):817–831, 2001.

Phil McMinn: Search-based software test data generation: a survey. Software Testing, Verification and Reliability 14(2): 105-156, 2004

Wasif Afzal, Richard Torkar and Robert Feldt: A Systematic Review of Search-based Testing for Non-Functional System Properties. Information and Software Technology, 51(6):957-976, 2009

Shaukat Ali, Lionel Briand, Hadi Hemmati and Rajwinder Panesar-Walawege: A Systematic Review of the Application and Empirical Investigation of Search-Based Test-Case Generation. IEEE Transactions on Software Engineering, 36(6), pp.742-762, 2010

Mark Harman, Yue Jia and Yuanyuan Zhang: Achievements, open problems and challenges for search based software testing. ICST, 2015

International Search-Based Testing Workshop

2008 ICST, Lillehammer, Norway
2009 ICST, Colorado, Denver
2010 ICST, Paris, France
2011 ICST, Berlin, Germany
2012 ICST, Montreal, Canada
2013 ICST, Luxembourg
2014 ICSE, Hyderabad, India
2015 ICSE, Florence, Italy
2016 ICSE, Austin, USA
2017 ICSE, Buenos Aires, Argentina?

http://www.searchbasedsoftwaretesting.org/

Raleigh, North Carolina

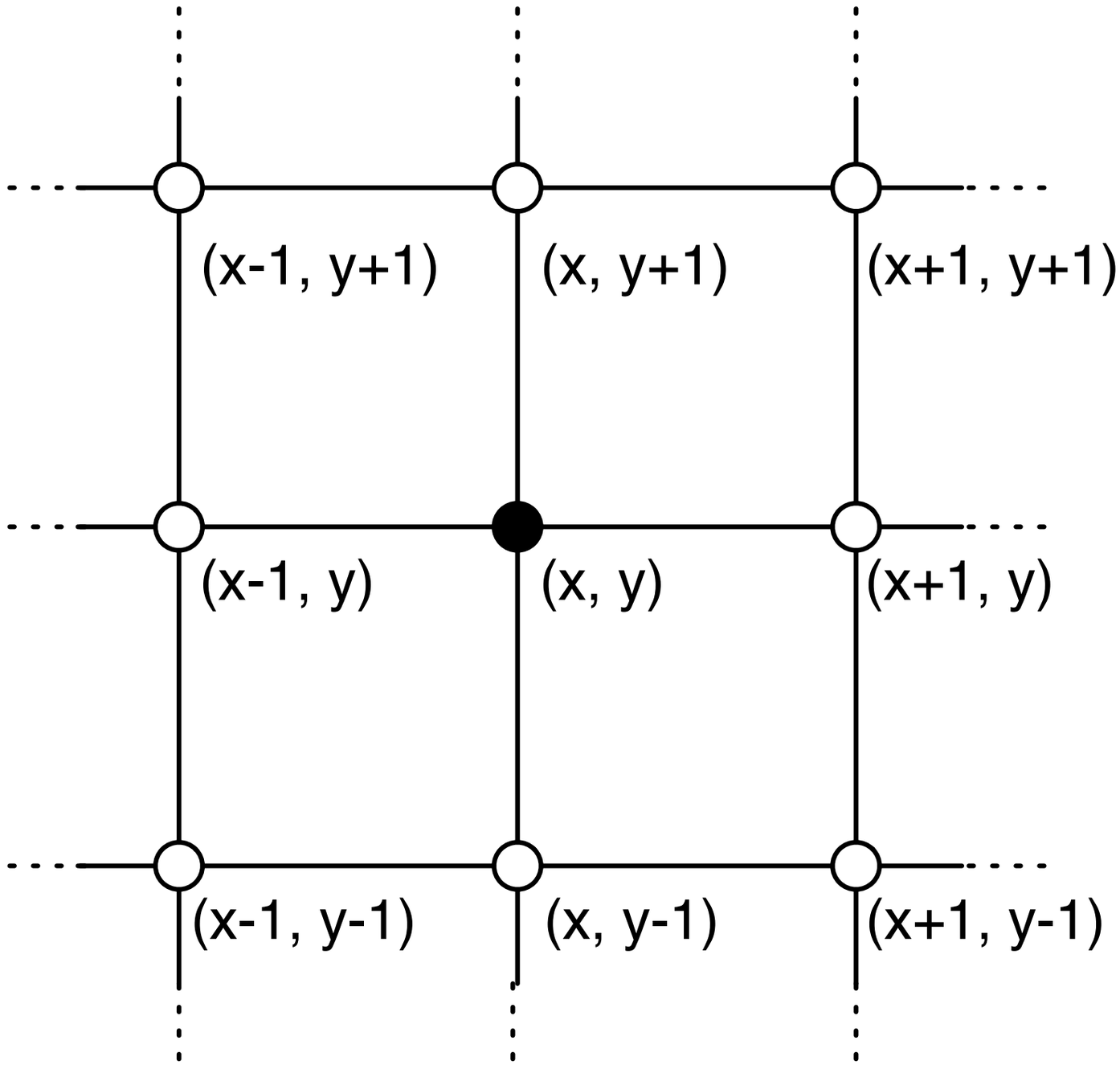
www.ssbse.org

Contents

Contents

```
def testMe(x, y):
    if x == 2 * (y + 1):
        return True
    else:
        return False
```

http://goo.gl/8mj8HC - 1.zip


(x-1, y+1)
(x, y+1)
(x+1, y+1)
(x-1, y)
(x, y)
(x+1, y)
(x-1, y-1)
(x, y-1)
(x+1, y-1)

Components of an SBST Tool

Search Algorithm — Hill-climbing

Representation — Tuple (x, y)

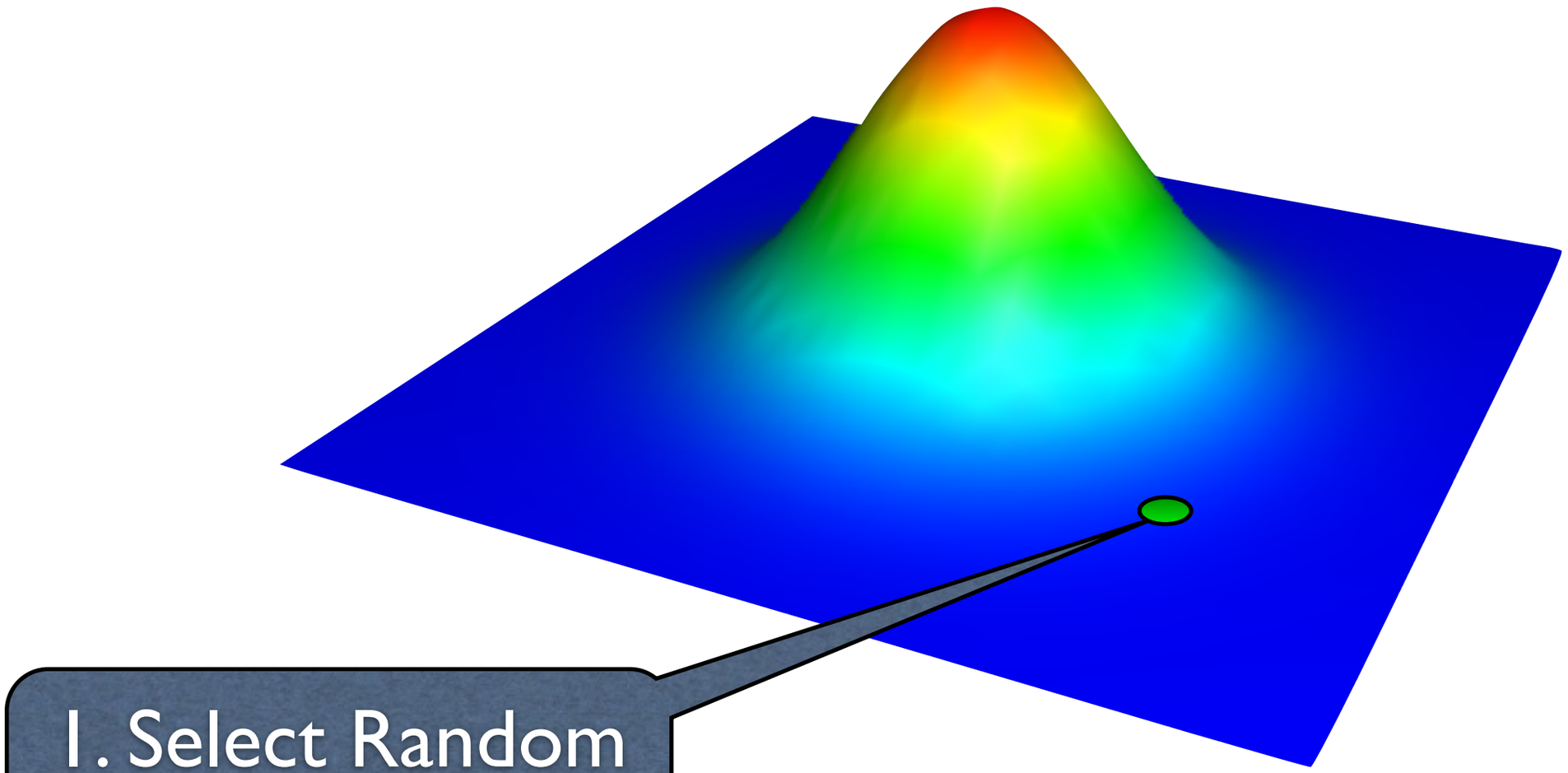
Search Operators — Neighbourhood of (x, y)

Fitness Function

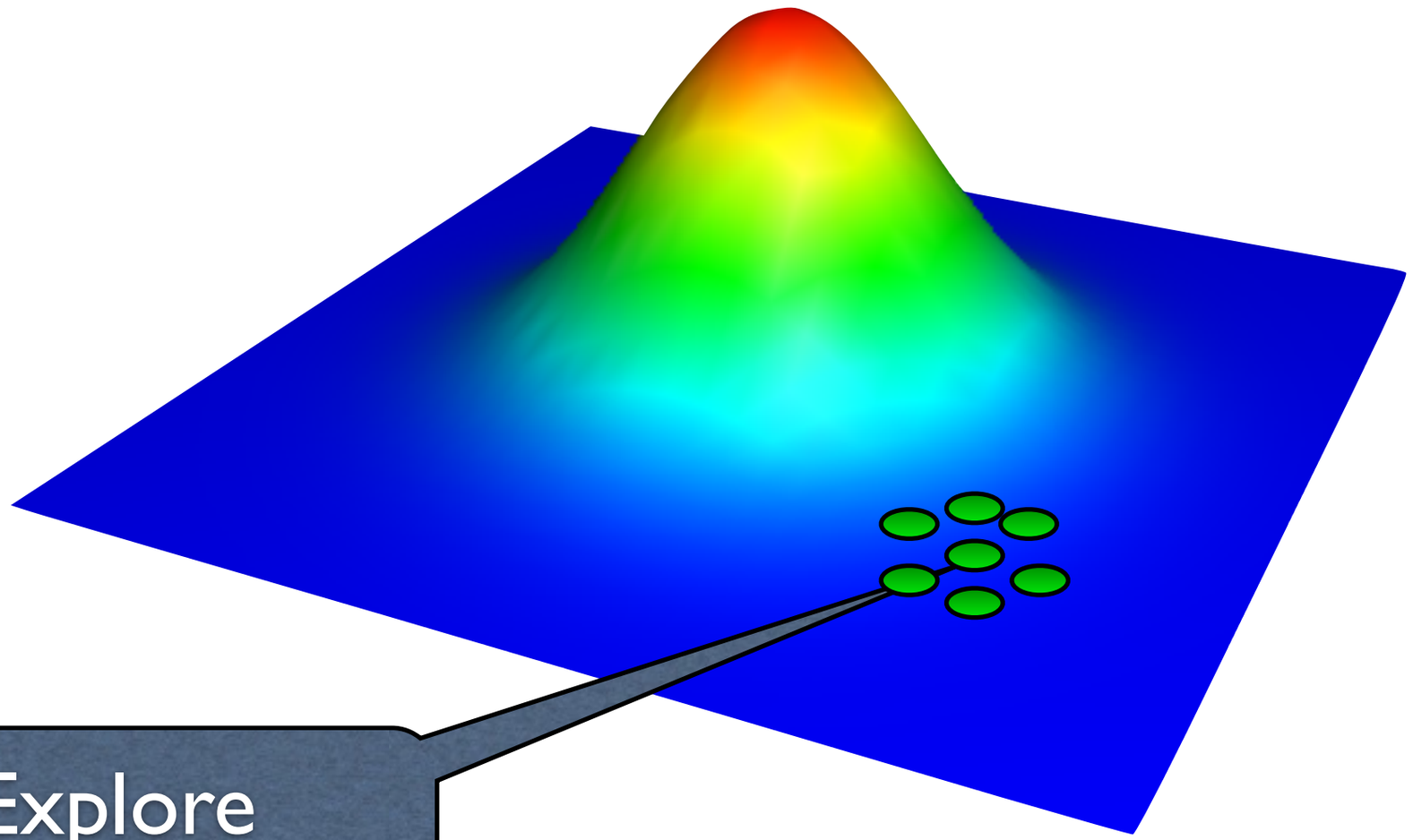
Test Execution

Instrumentation

Hill Climbing

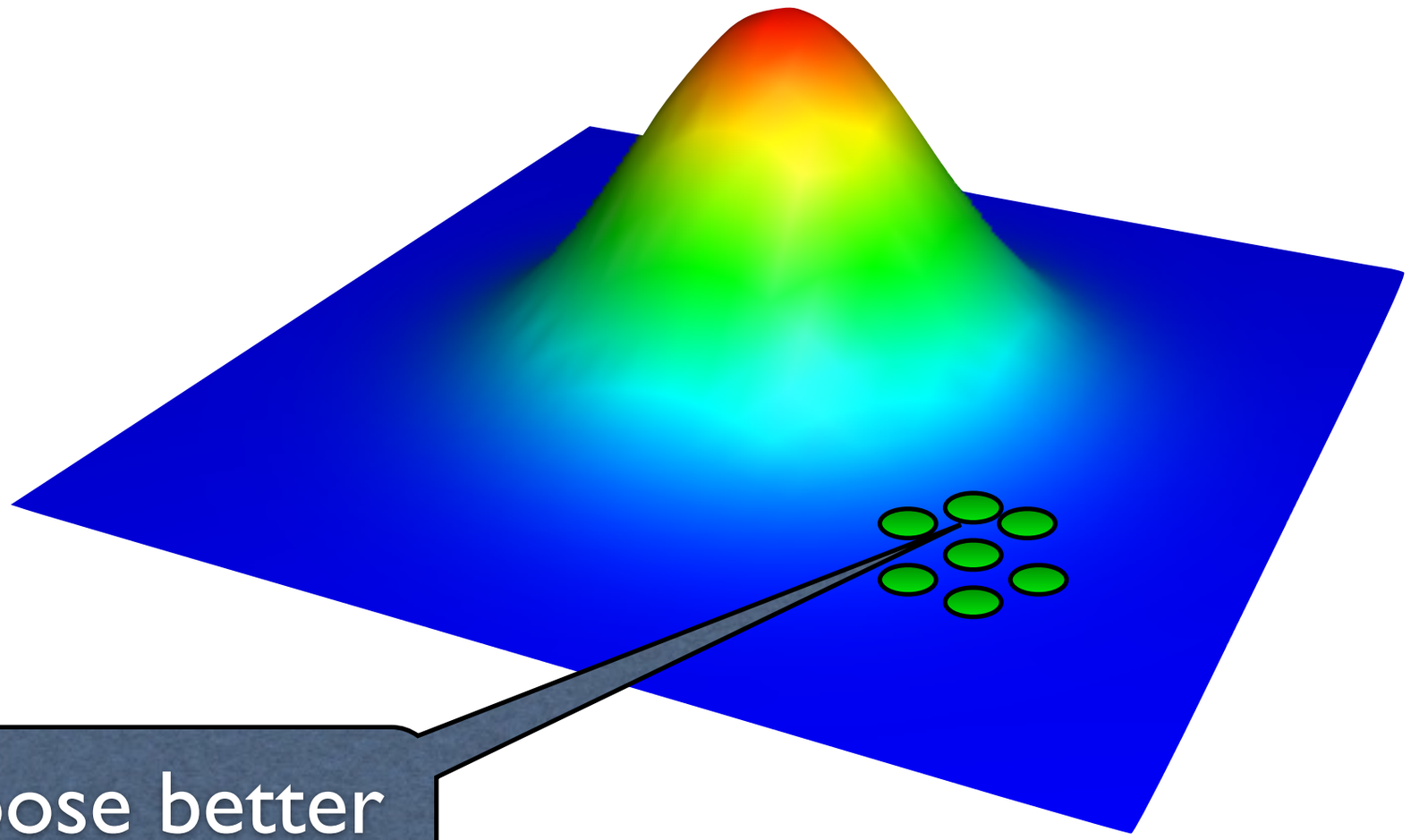


Hill Climbing



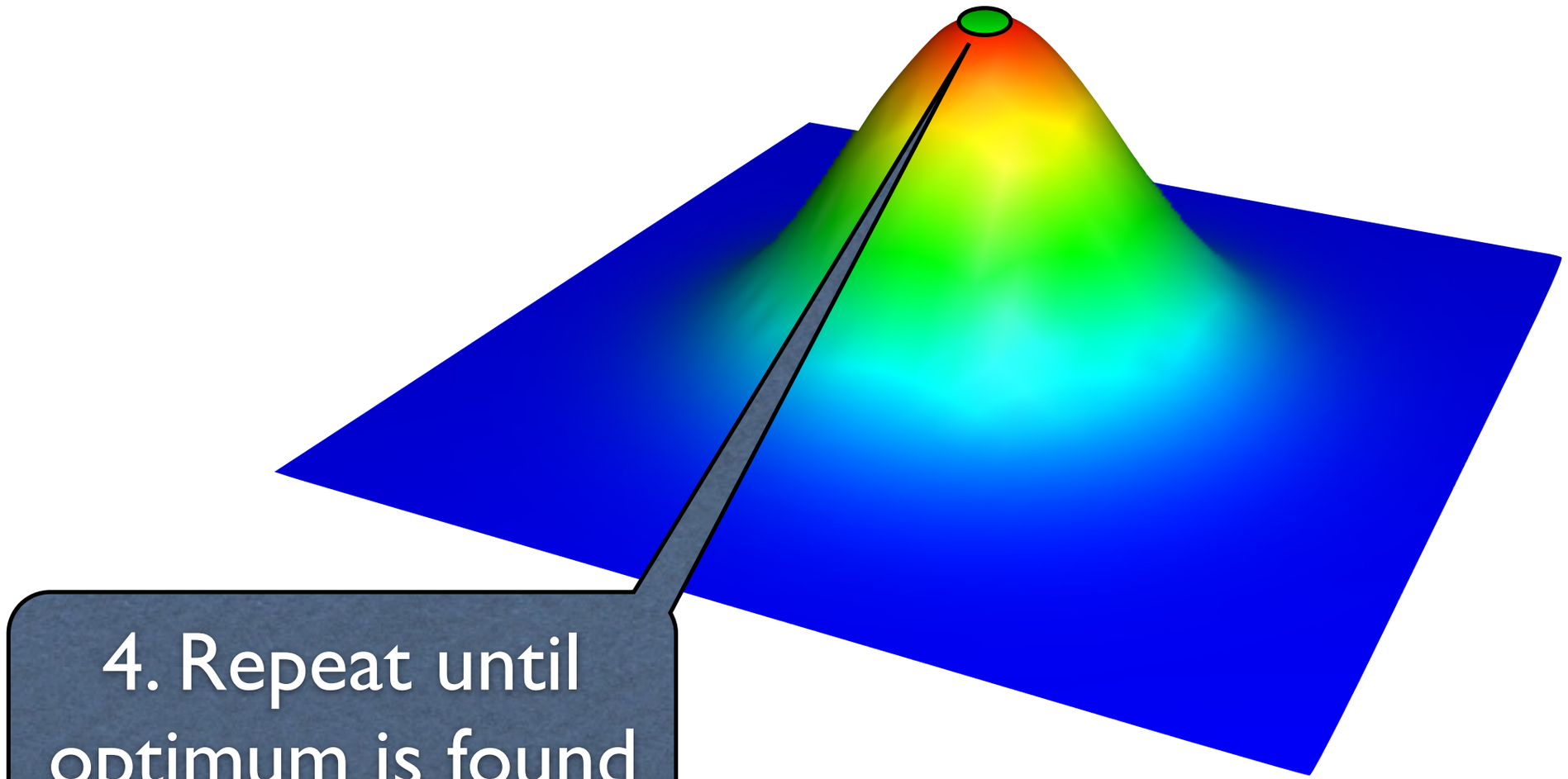
2. Explore Neighbourhood

Hill Climbing



3. Choose better neighbour

Hill Climbing



Components of an SBST Tool

Search Algorithm — Hill-climbing

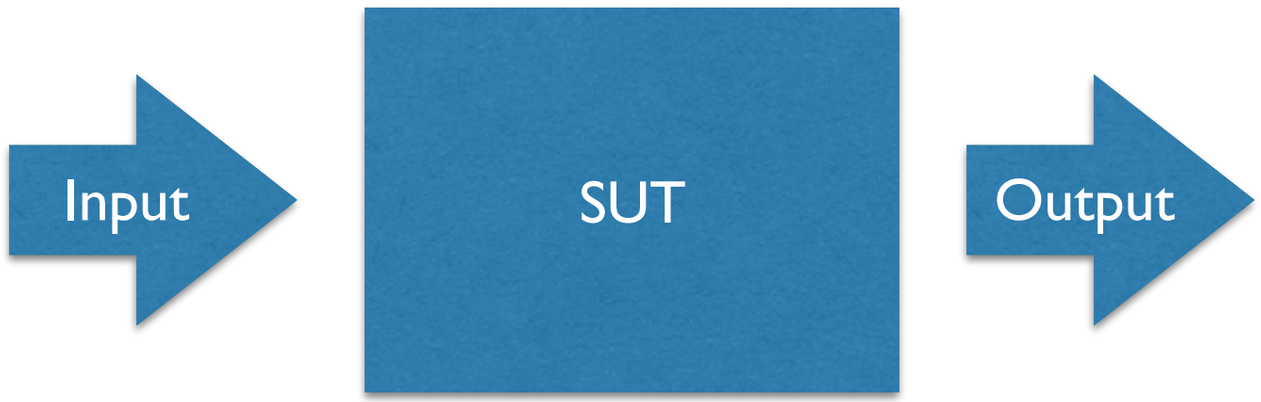
Representation — Tuple (x, y)

Search Operators — Neighbourhood of (x, y)

Fitness Function

Test Execution

Instrumentation


Input
SUT
Output

Test Data

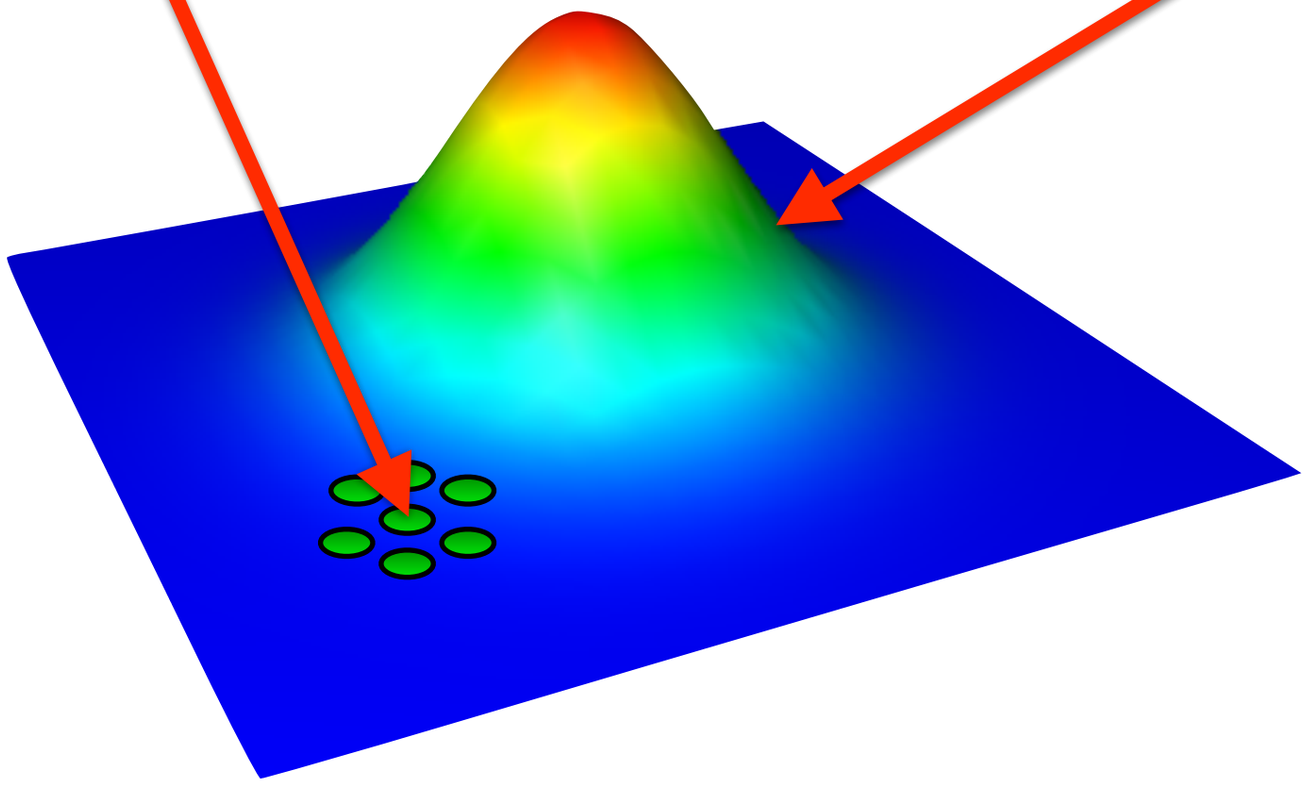
Input

Instrumented SUT

Output

Trace

Fitness

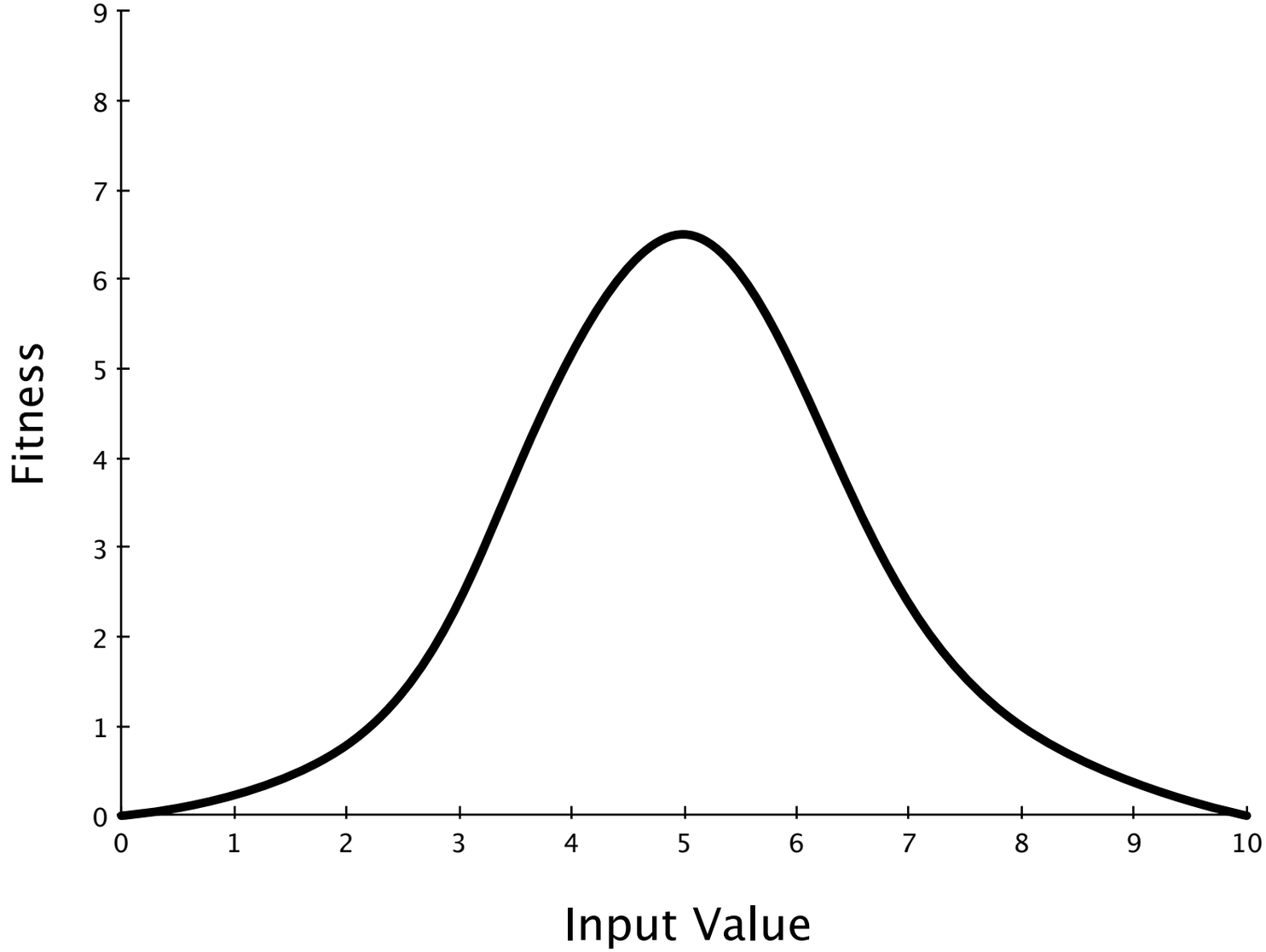


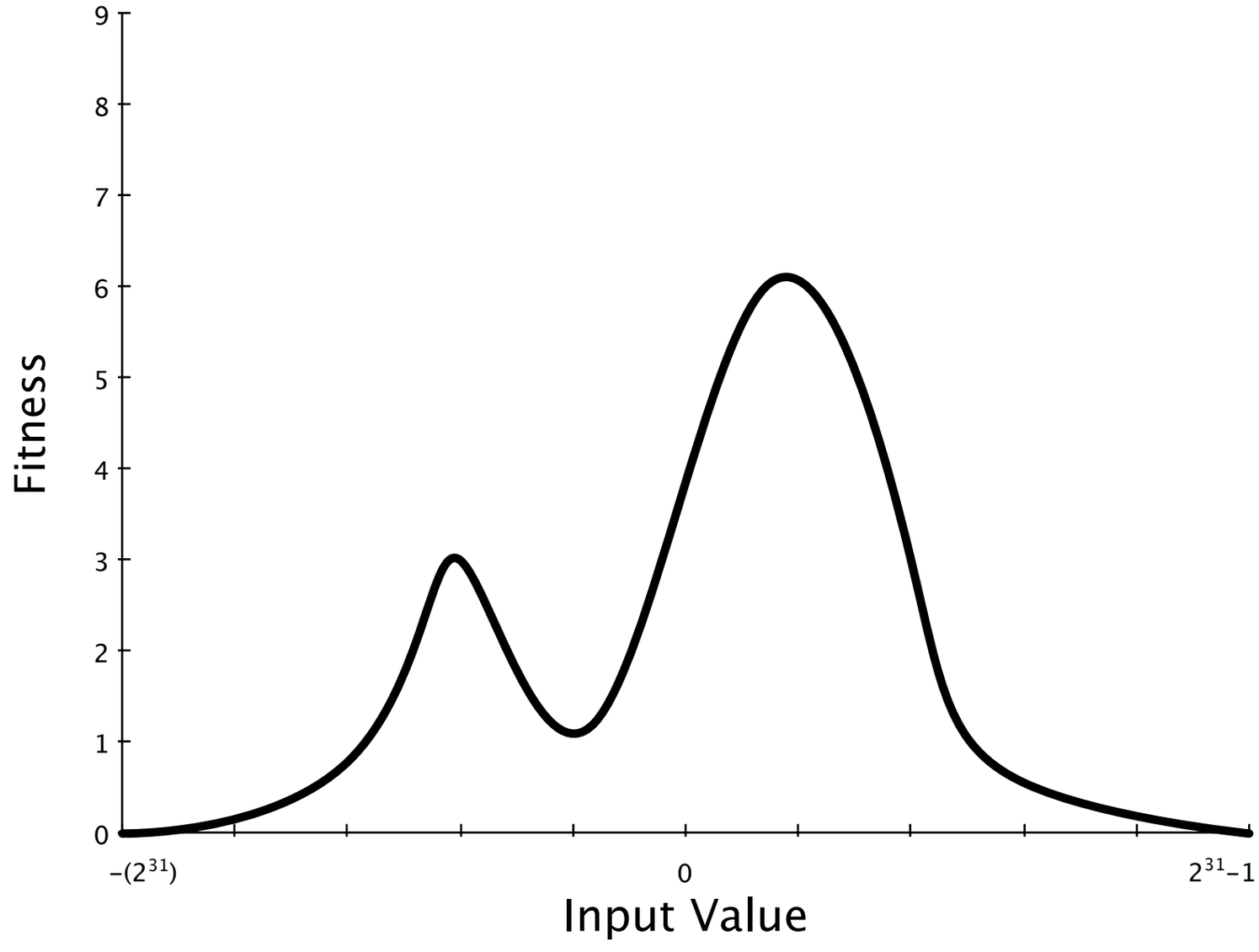
```
def testMe(x, y):
    if x == 2 * (y + 1):
        return True
    else:
        return False
```

http://goo.gl/8mj8HC - 2.zip

Components of an SBST Tool

Search Algorithm	Hill-climbing
Representation	Tuple (x, y)
Search Operators	Neighbourhood of (x, y)
Fitness Function	Branch distance
Test Execution	Call method
Instrumentation	Global variable


Fitness
Input Value
0
1
2
3
4
5
6
7
8
9
10


Fitness
9
8
7
6
5
4
3
2
1
0
-(2^31)
0
2^31-1
Input Value

```
def testMe(x, y):
    if x == 2 * y and y > 1:
        return True
    else:
        return False
```

http://goo.gl/8mj8HC - 3.zip

Branch Distance

Expression	Distance True	Distance False
x == y	|x - y|	1
x != y	1	|x - y|
x > y	y - x + 1	x - y
x >= y	y - x	x - y + 1
x < y	x - y+ 1	x - y
x <= y	x - y	x - y + 1

```
def testMe(x, y):
    if x == 2 * y and y > 1:
        return True
    else:
        return False
```

http://goo.gl/8mj8HC - 3.zip

```
from Instrumentation import BranchPredicate

def testMe(x, y):
    if (BranchPredicate(1, 'Eq', x, 2 * y) and
        BranchPredicate(2, 'Gt', y, 1)):
        return True
    else:
        return False
```

http://goo.gl/8mj8HC - 3.zip

Components of an SBST Tool

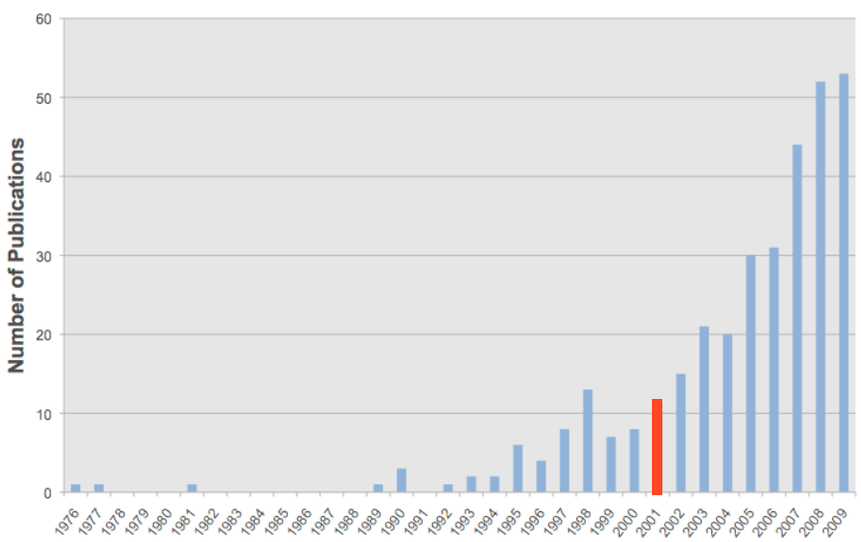
Search Algorithm	Hill-climbing
Representation	Tuple (x, y)
Search Operators	Neighbourhood of (x, y)
Fitness Function	Branch distance
Test Execution	Call method
Instrumentation	BranchDistance instrumentation

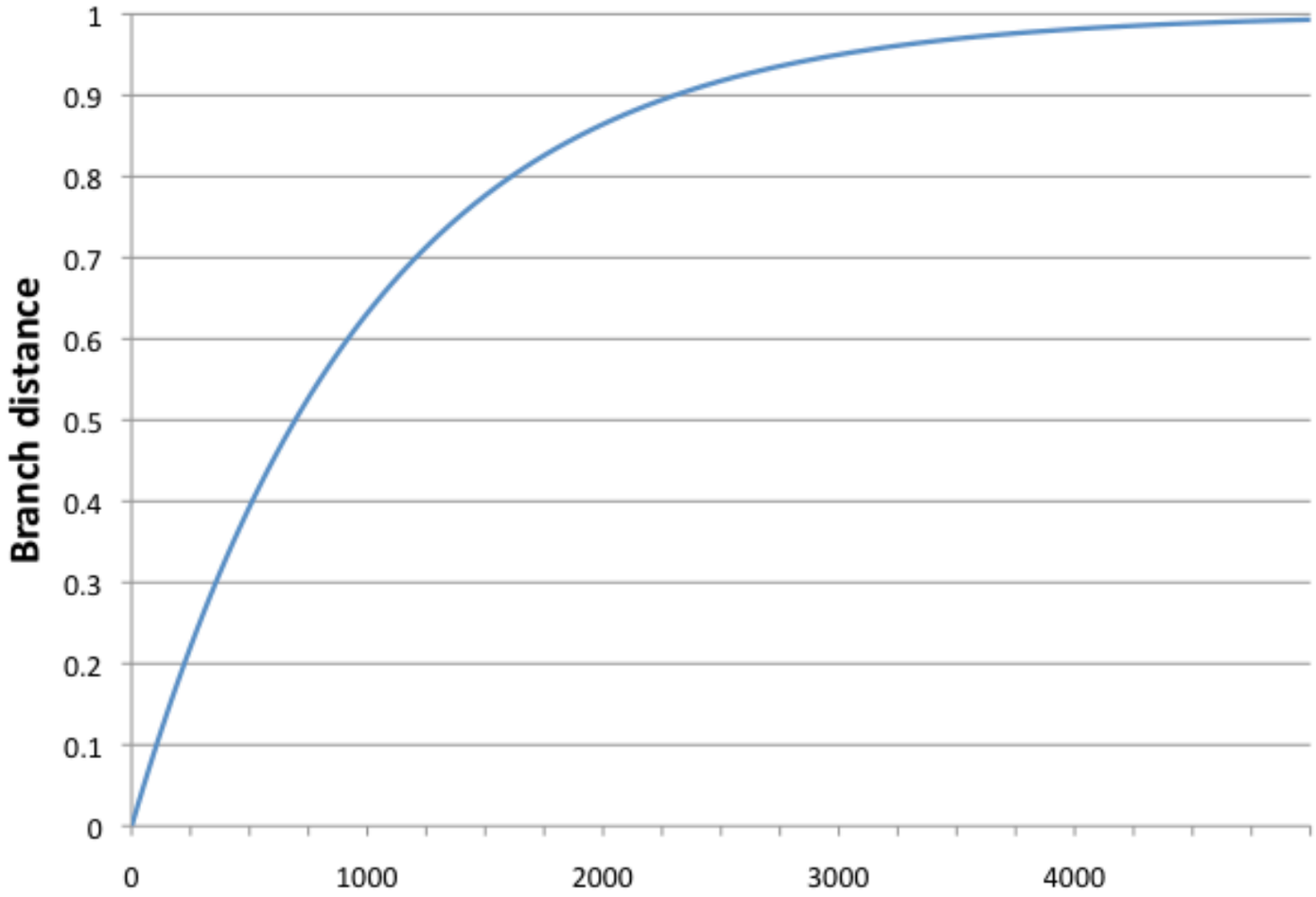
Normalisation Functions

Since the ‘maximum’ branch distance is generally unknown we need a non-standard normalisation function

$$1 - \alpha^{-x}$$

Baresel (2000), alpha = 1.001



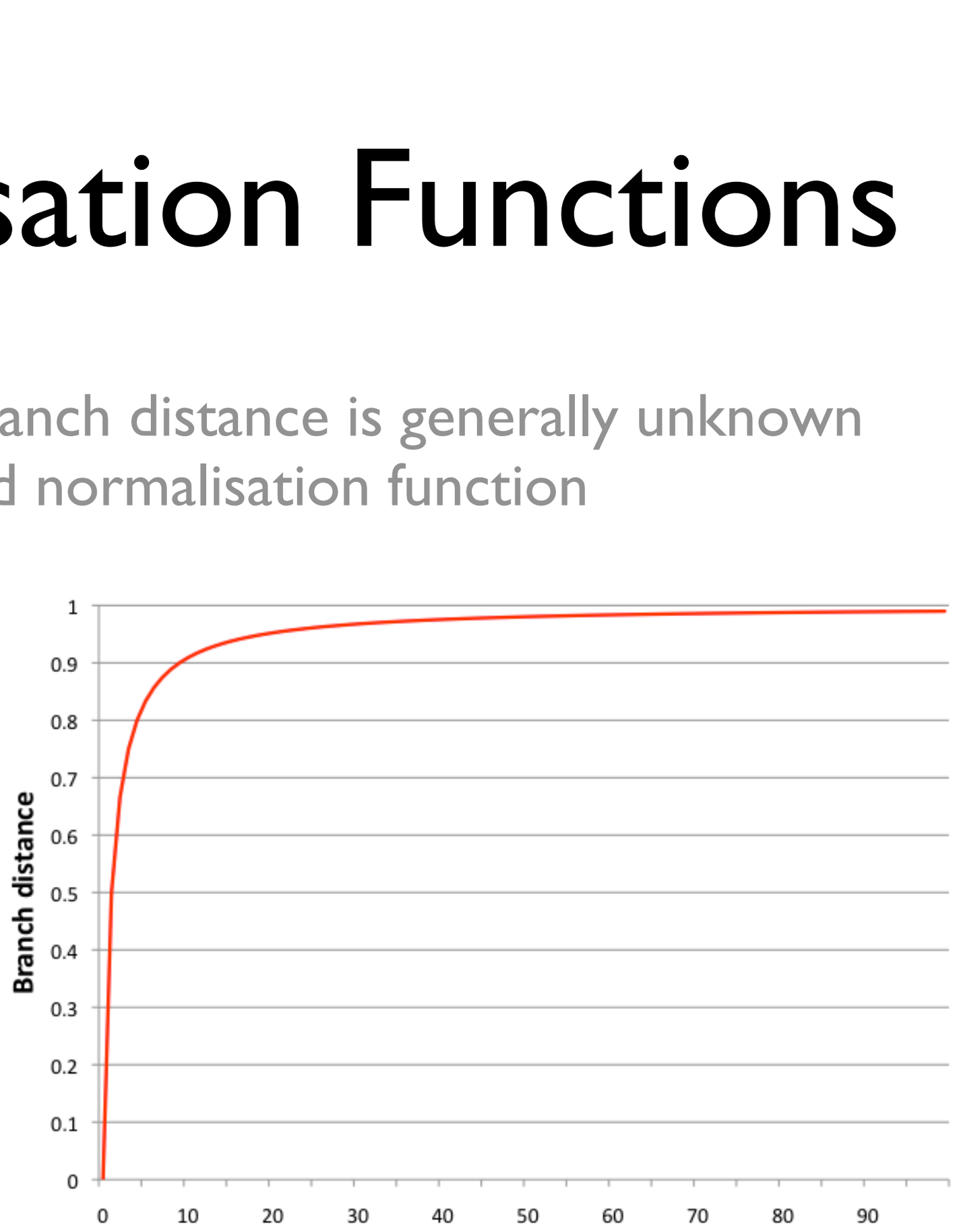


Normalisation Functions

Since the ‘maximum’ branch distance is generally unknown
we need a non-standard normalisation function

$$\frac{x}{x+\beta}$$

Arcuri (2010), beta = 1

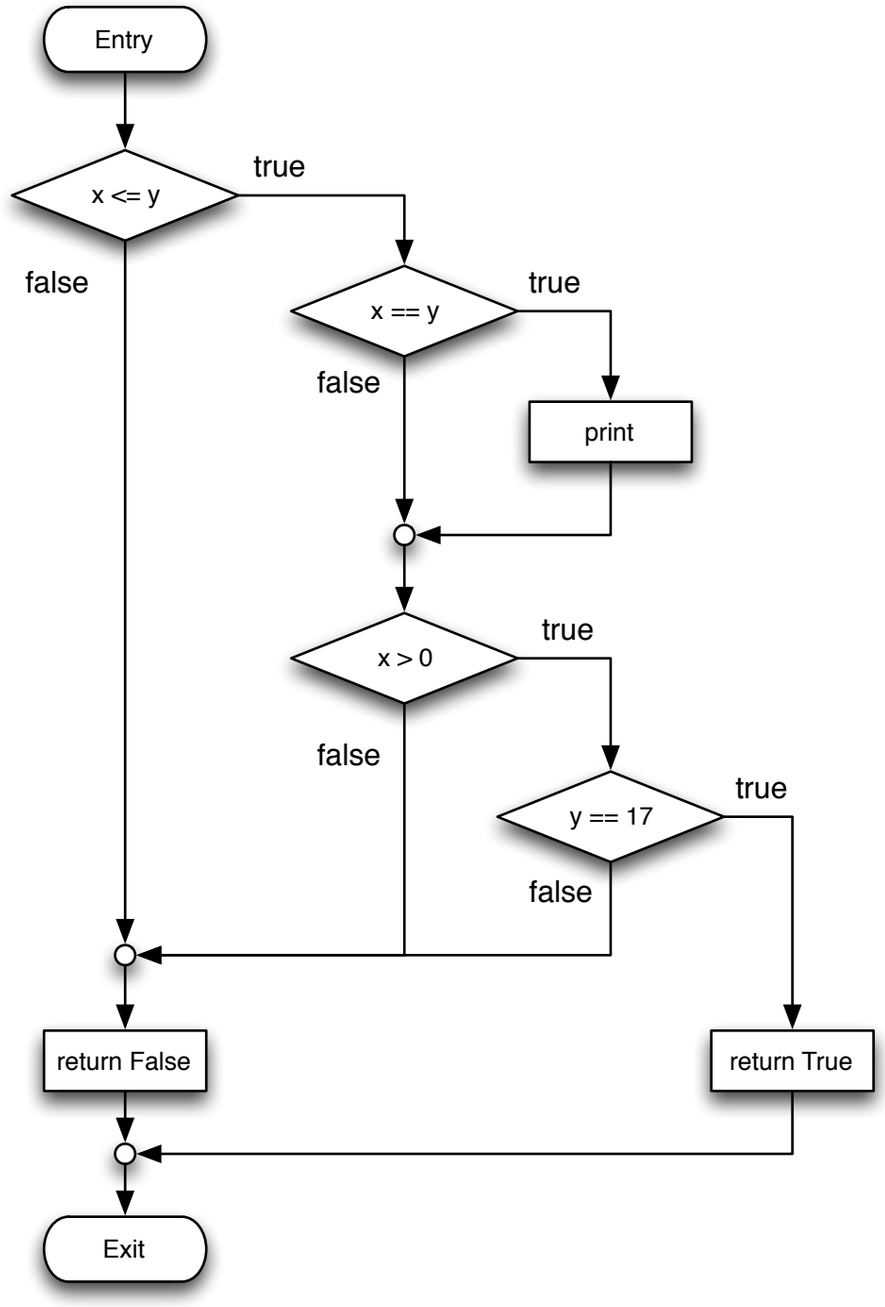


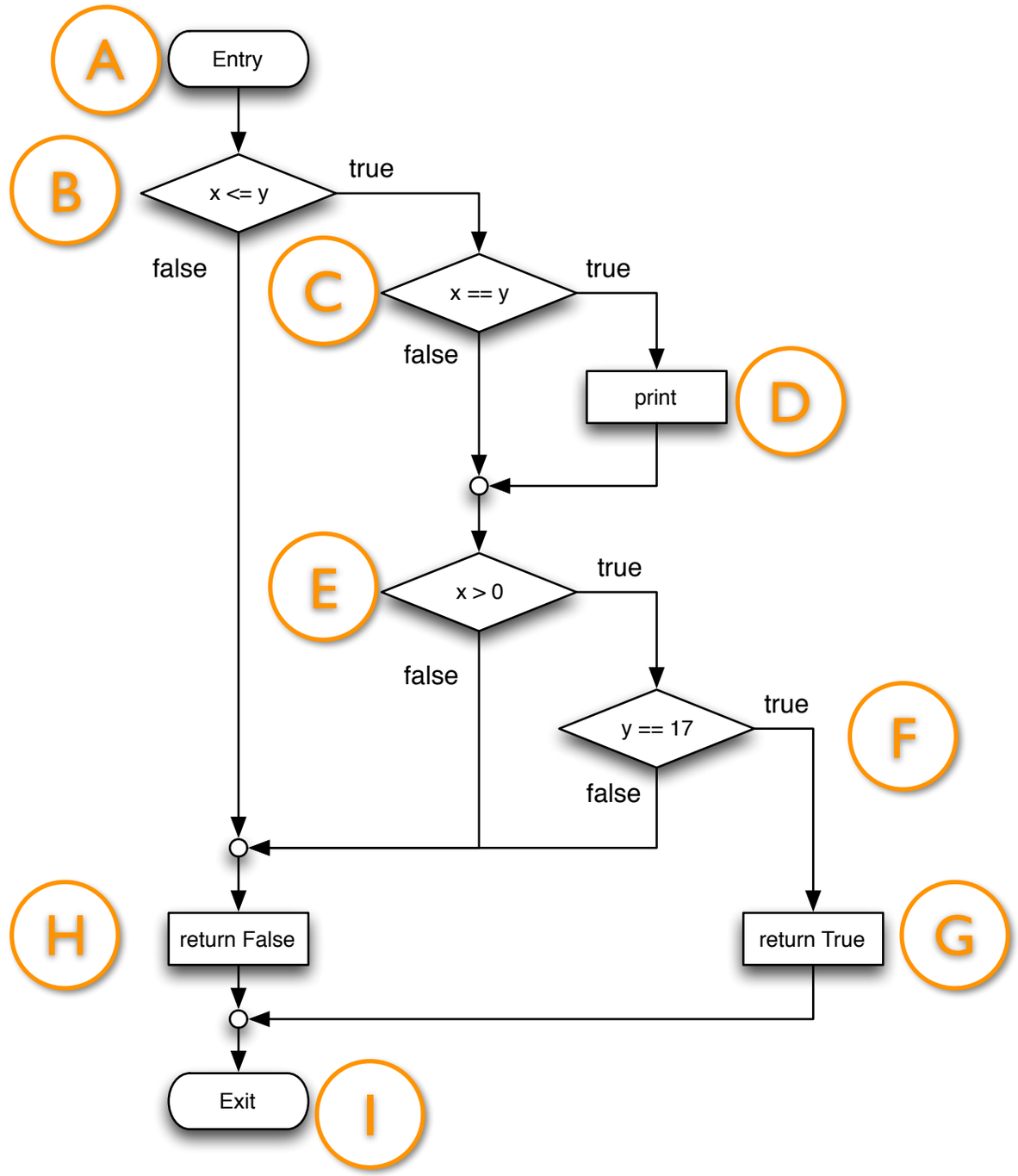
```
from Instrumentation import BranchPredicate

def testMe(x, y):
    if (BranchPredicate(1, 'Eq', x, 2 * y) and
        BranchPredicate(2, 'Gt', y, 1)):
        return True
    else:
        return False
```

http://goo.gl/8mj8HC - 3.zip

```
def testMe(x, y):
    if x <= y:
        if x == y:
            print("Some output")
        if x > 0:
            if y == 17:
                # Target Branch
                return True
return False
```

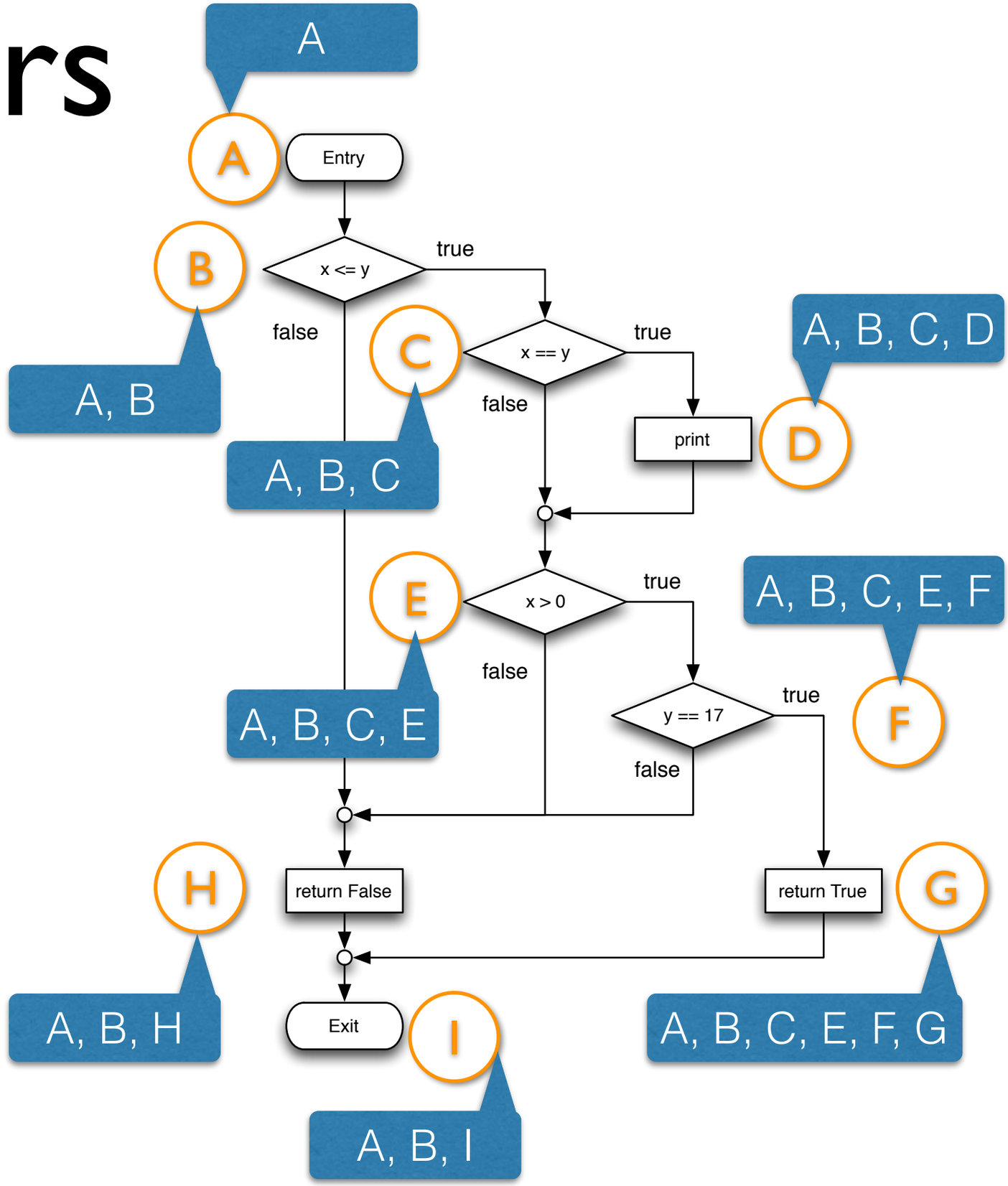
```
def testMe(x, y):
    if x <= y:
        if x == y:
            print("Some output")
    if x > 0:
        if y == 17:
            # Target Branch
            return True
return False
```




A
Entry
B
x <= y
true
false
C
x == y
true
false
print
D
E
x > 0
true
false
y == 17
true
false
F
H
return False
return True
G
Exit
I

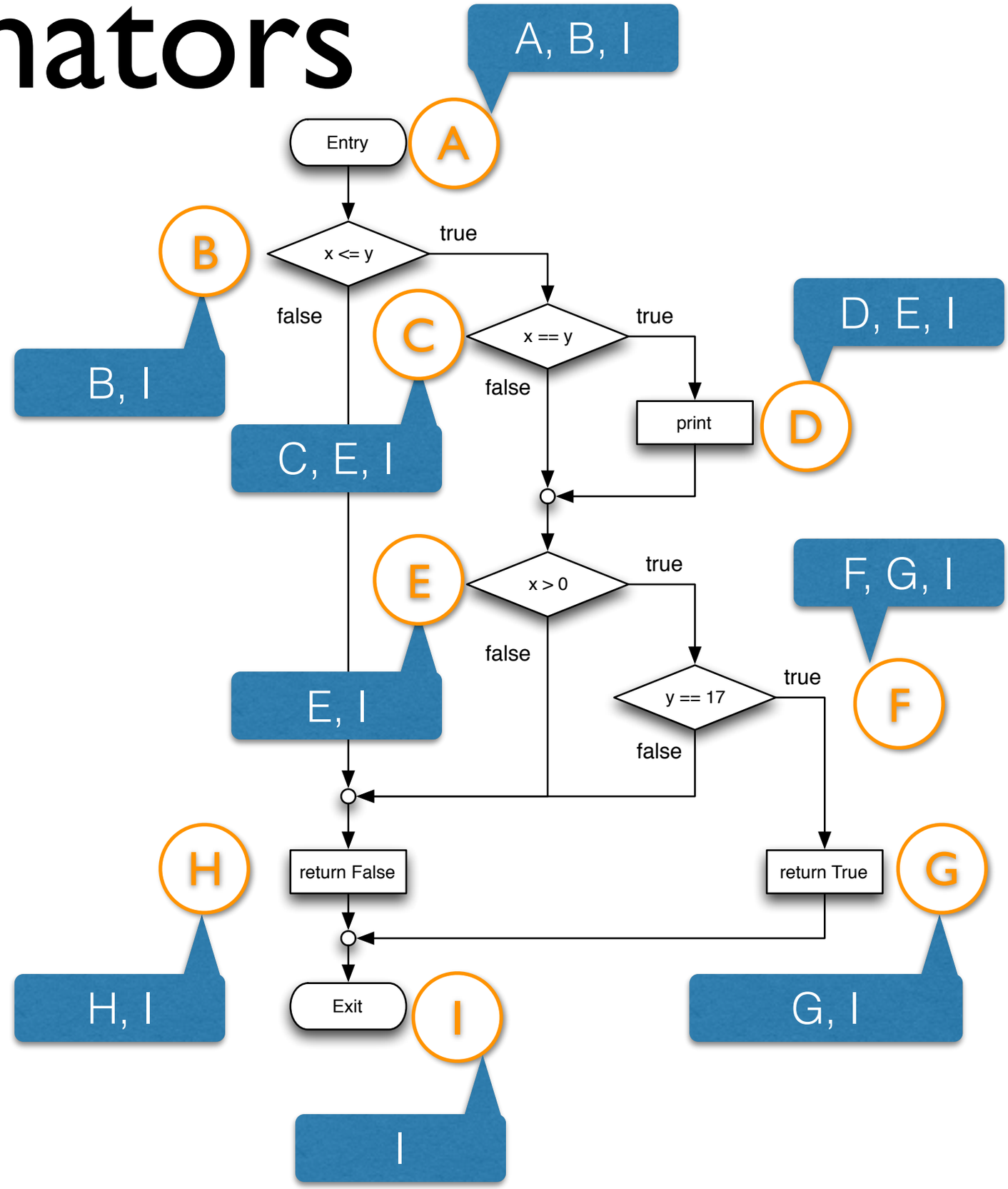
Dominators

- Node A dominates B if every path to B goes through A



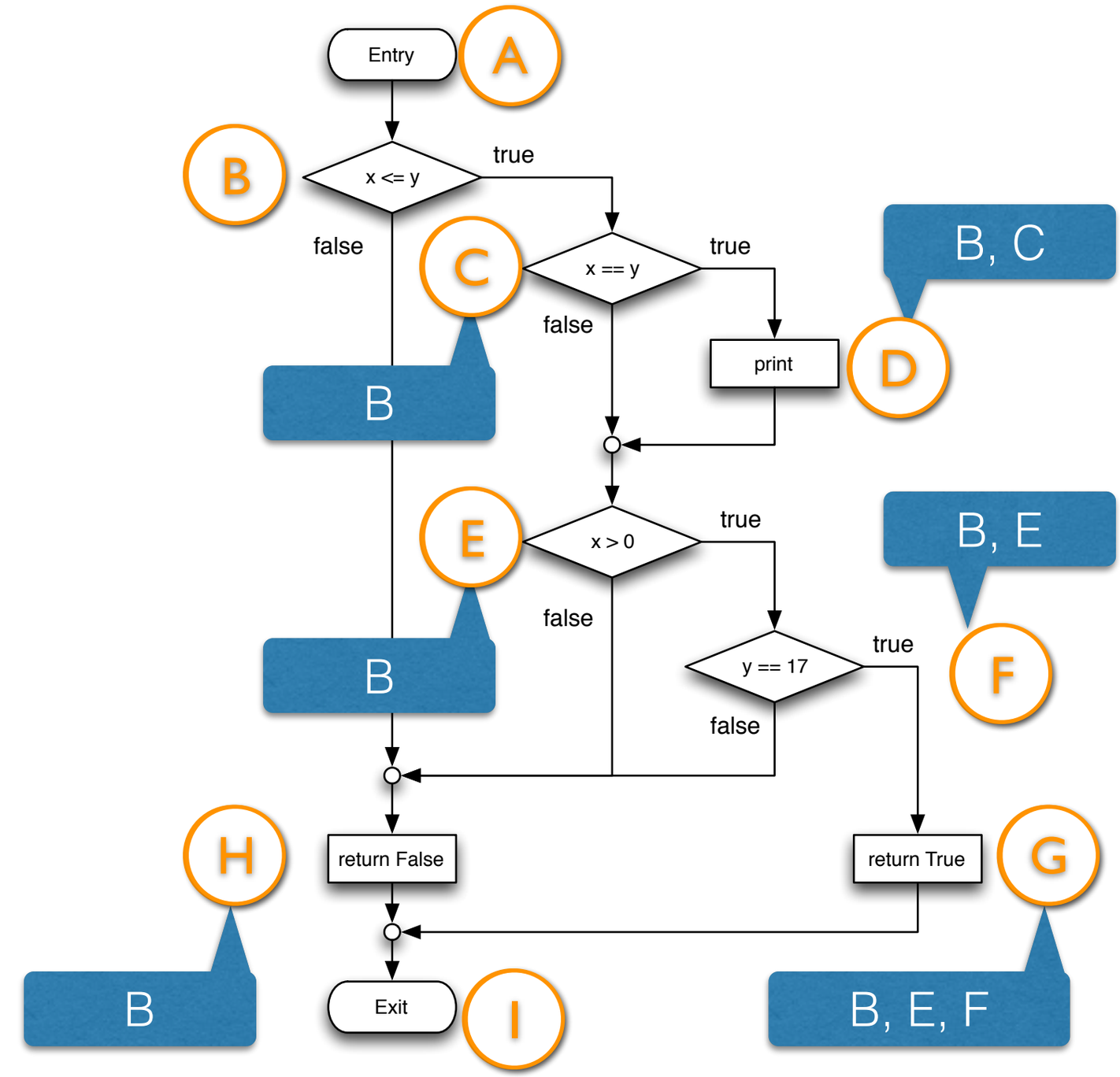
Post-Dominators

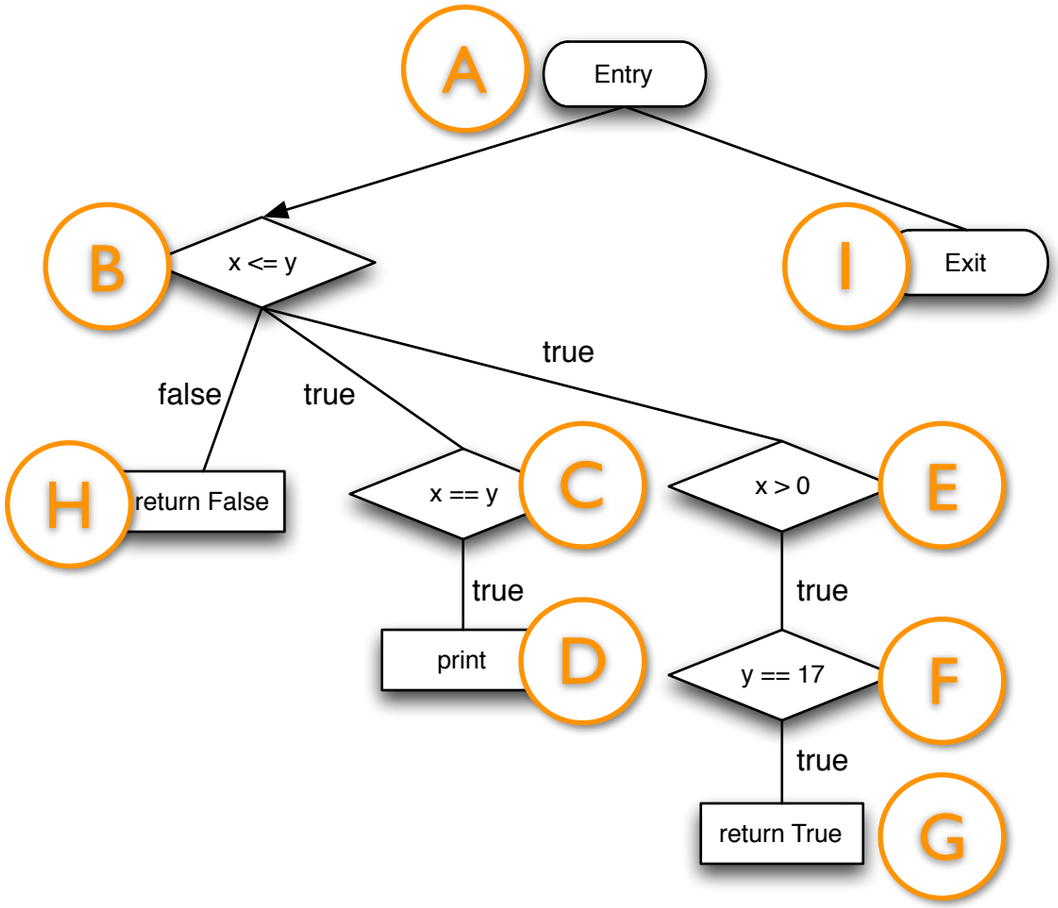
- Dominators viewed in reverse (paths from exit node)

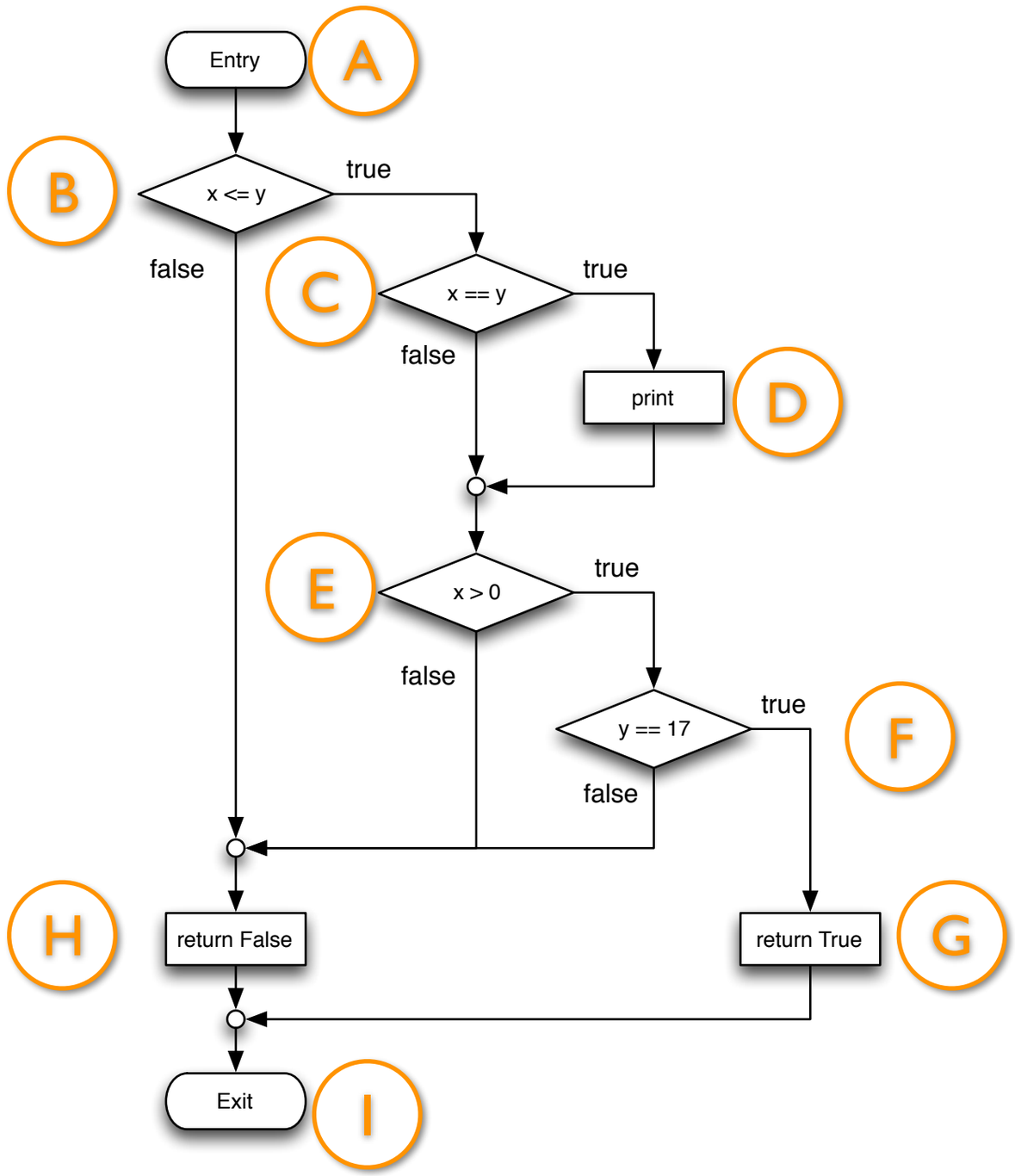


Control Dependence

- A is control dependent on B if:
- B has at least two successors in the CFG
- B dominates A
- B is not post-dominated by A
- There is a successor of B that is post-dominated by A

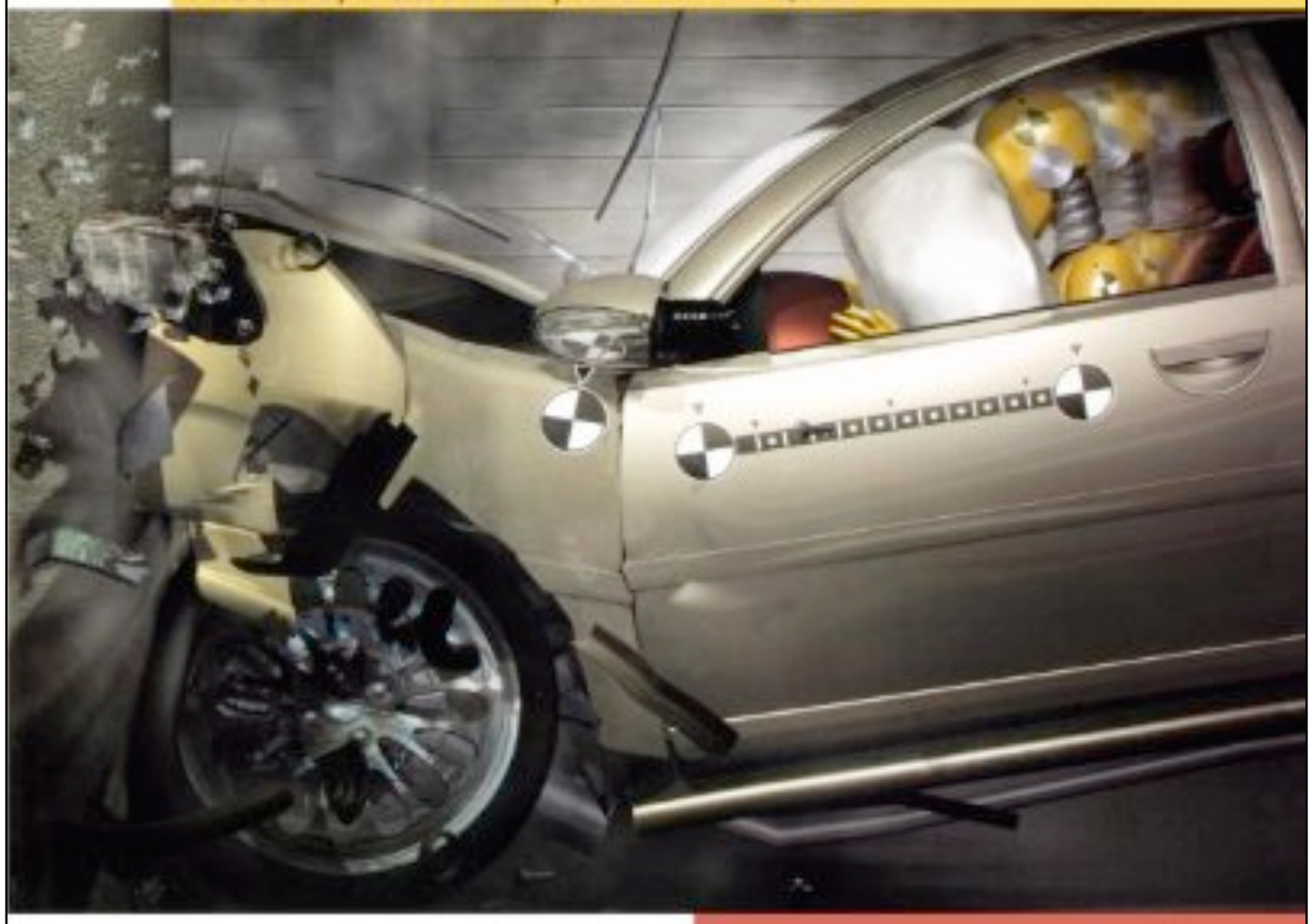



A
Entry
B
x <= y
I
Exit
true
false
true
H
return False
x == y
C
x > 0
E
true
true
print
D
y == 17
F
true
return True
G


Entry
A
B
x <= y
true
false
C
x == y
true
false
print
D
E
x > 0
true
false
y == 17
true
false
F
H
return False
return True
G
Exit
I

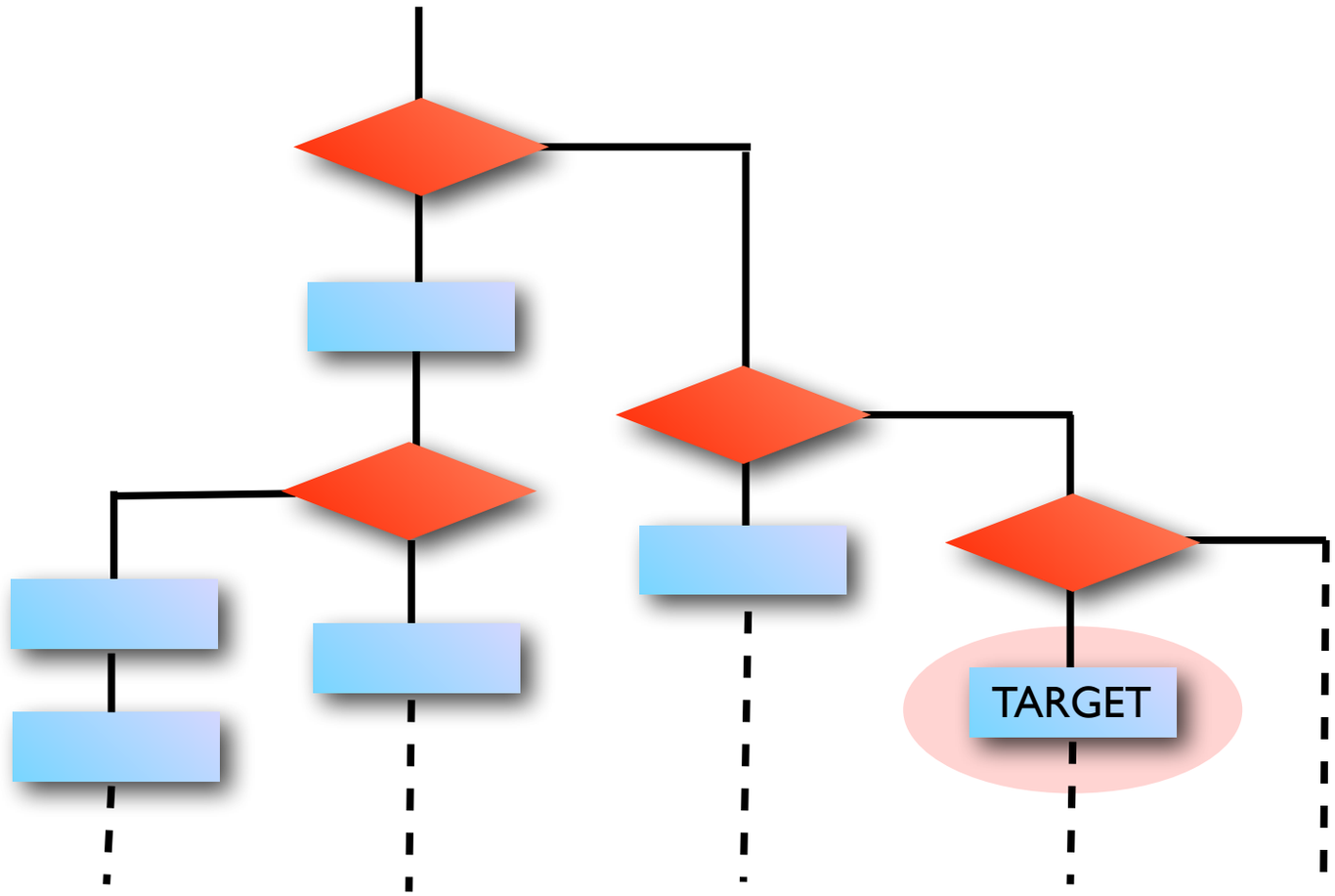
SOFTWARE TESTING AND ANALYSIS

PROCESS, PRINCIPLES, AND TECHNIQUES

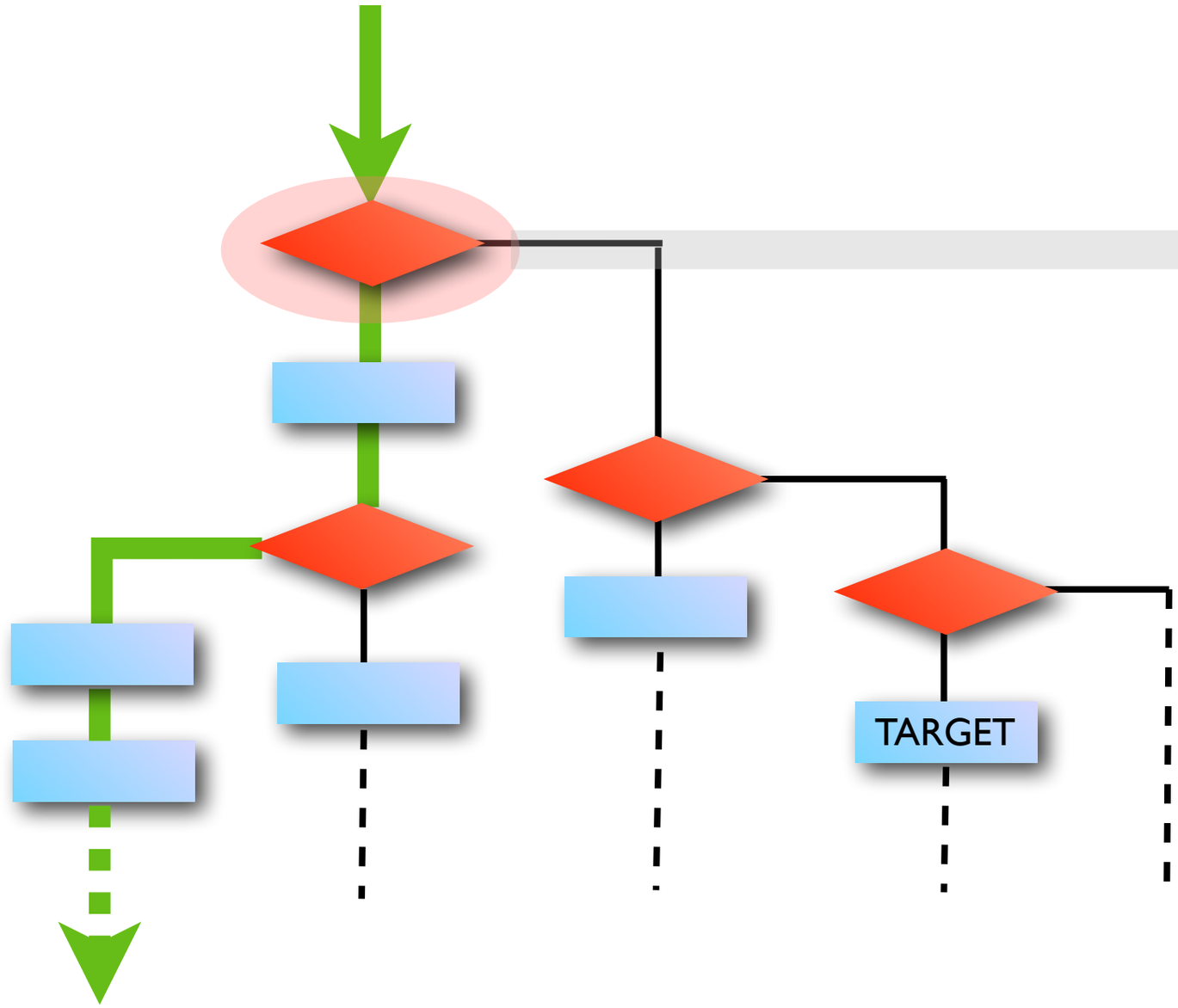


Mauro Pezzè
Michal Young

Covering a structure

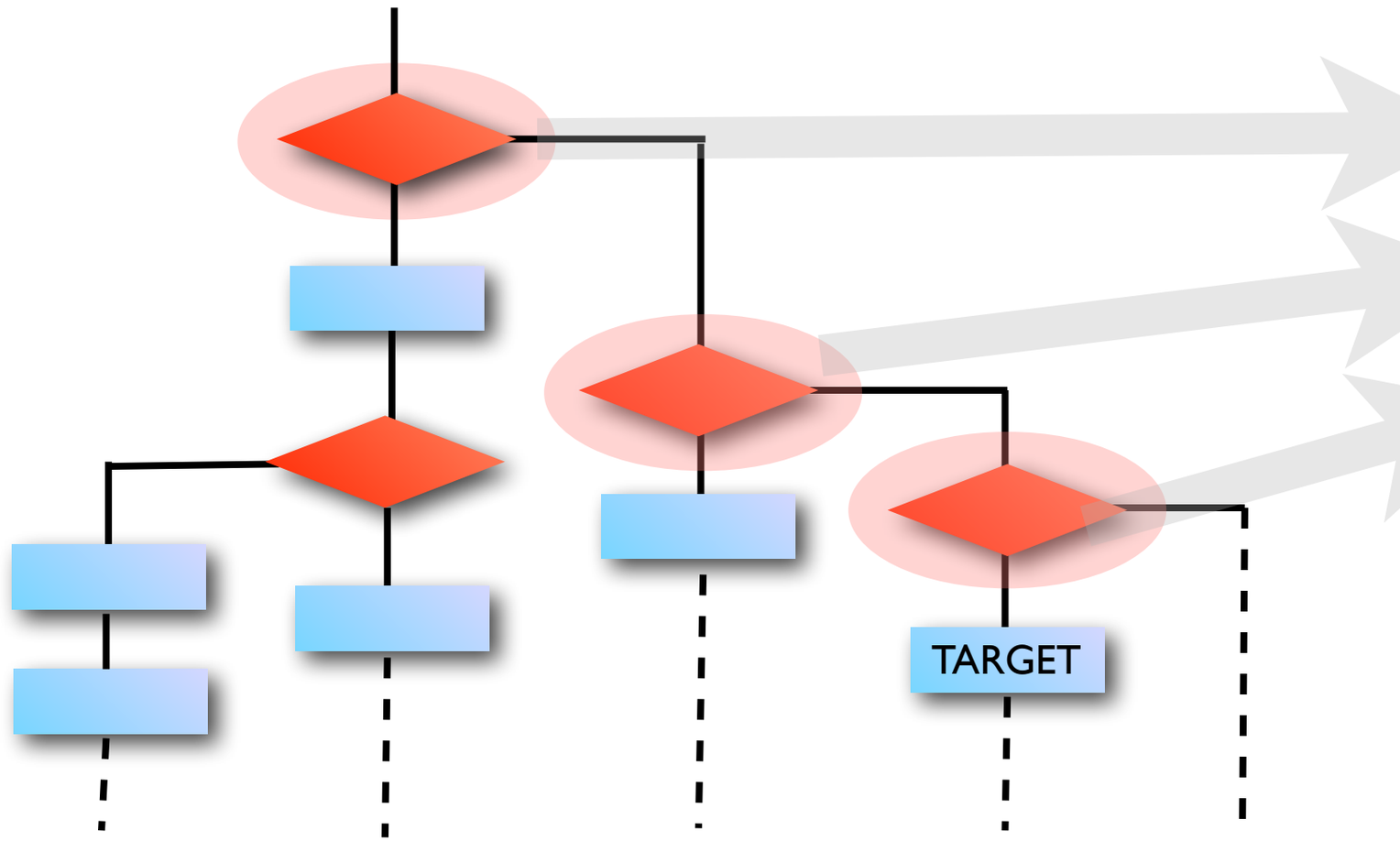


Fitness evaluation



The test data executes the ‘wrong’ path

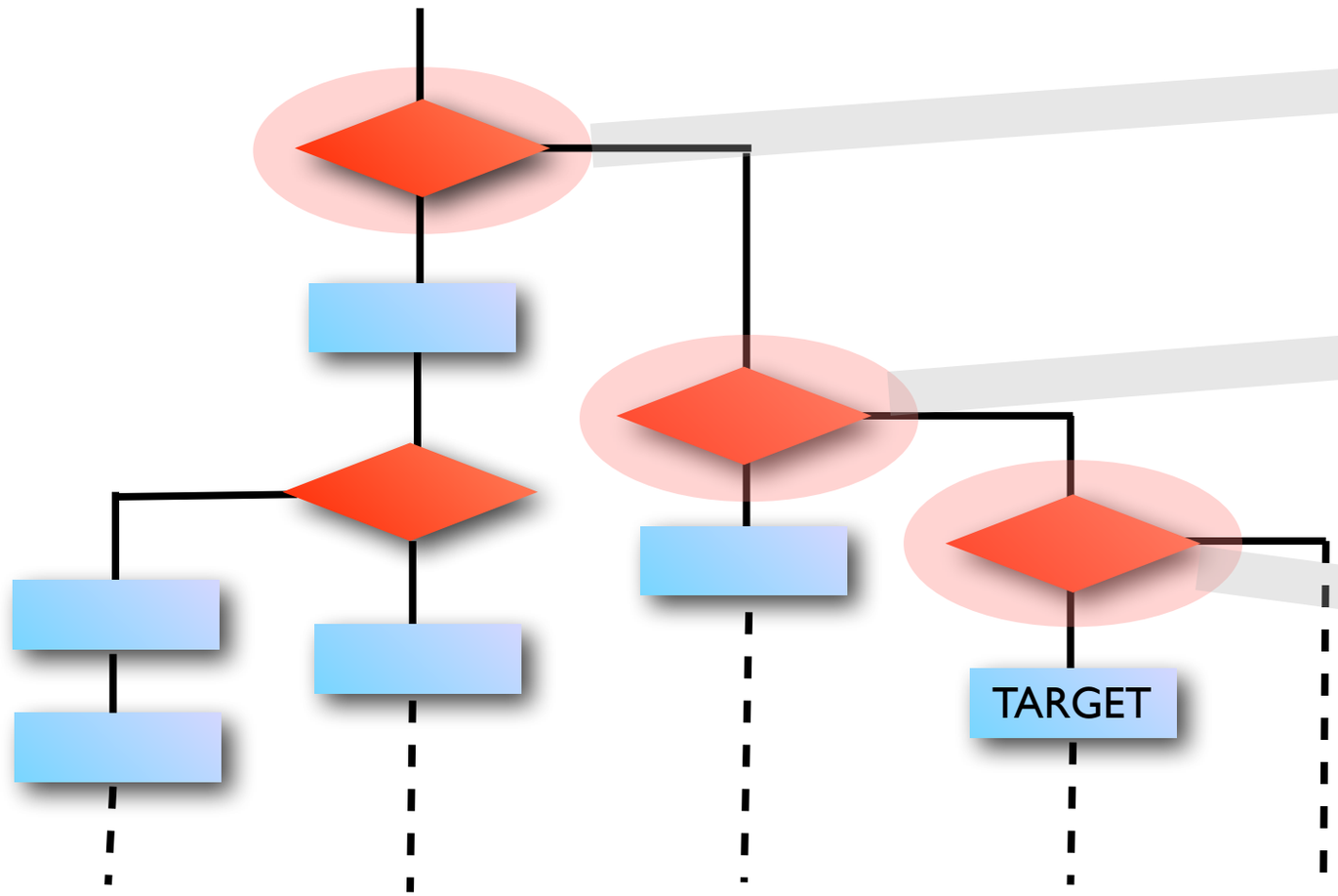
Analysing control flow



The outcomes at key decision statements matter.

These are the decisions on which the target is control dependent

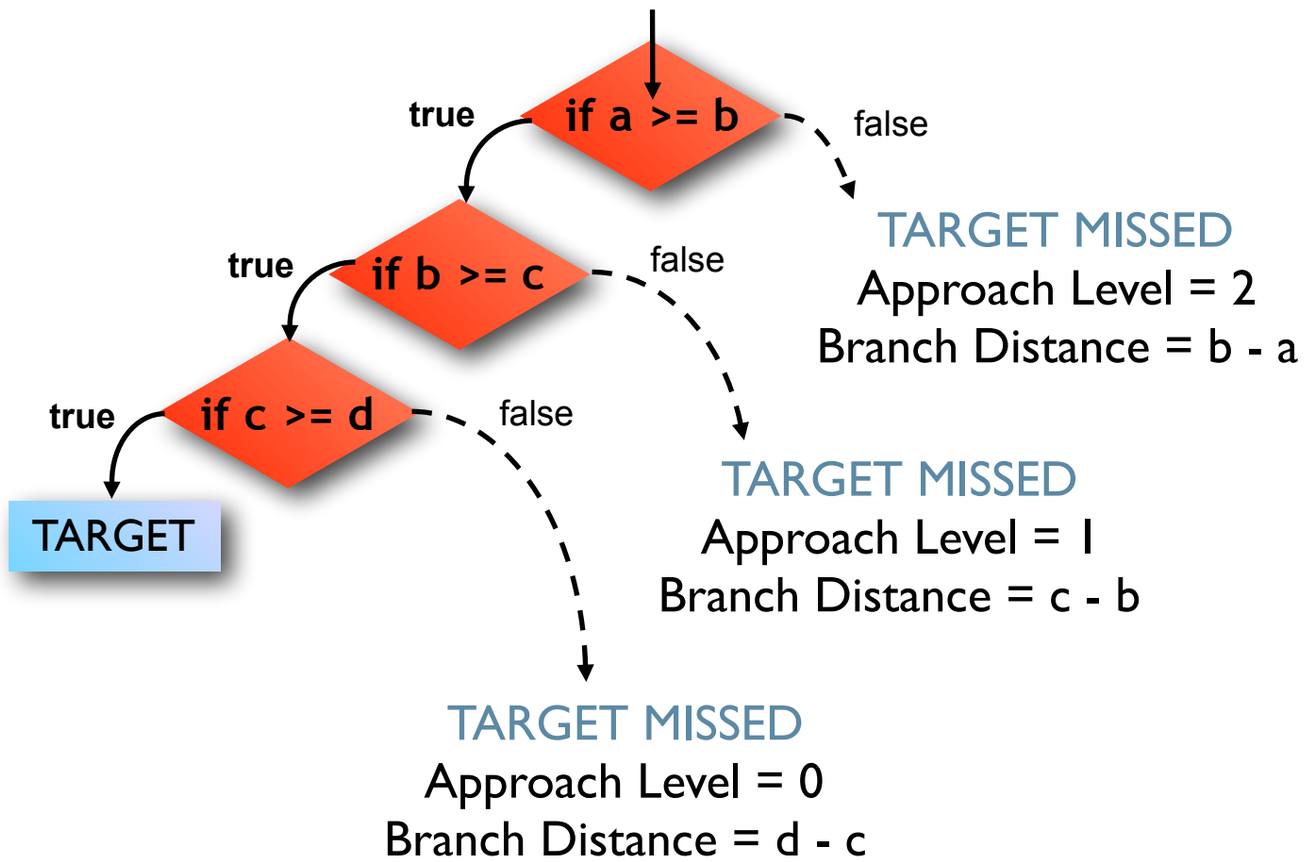
Approach Level



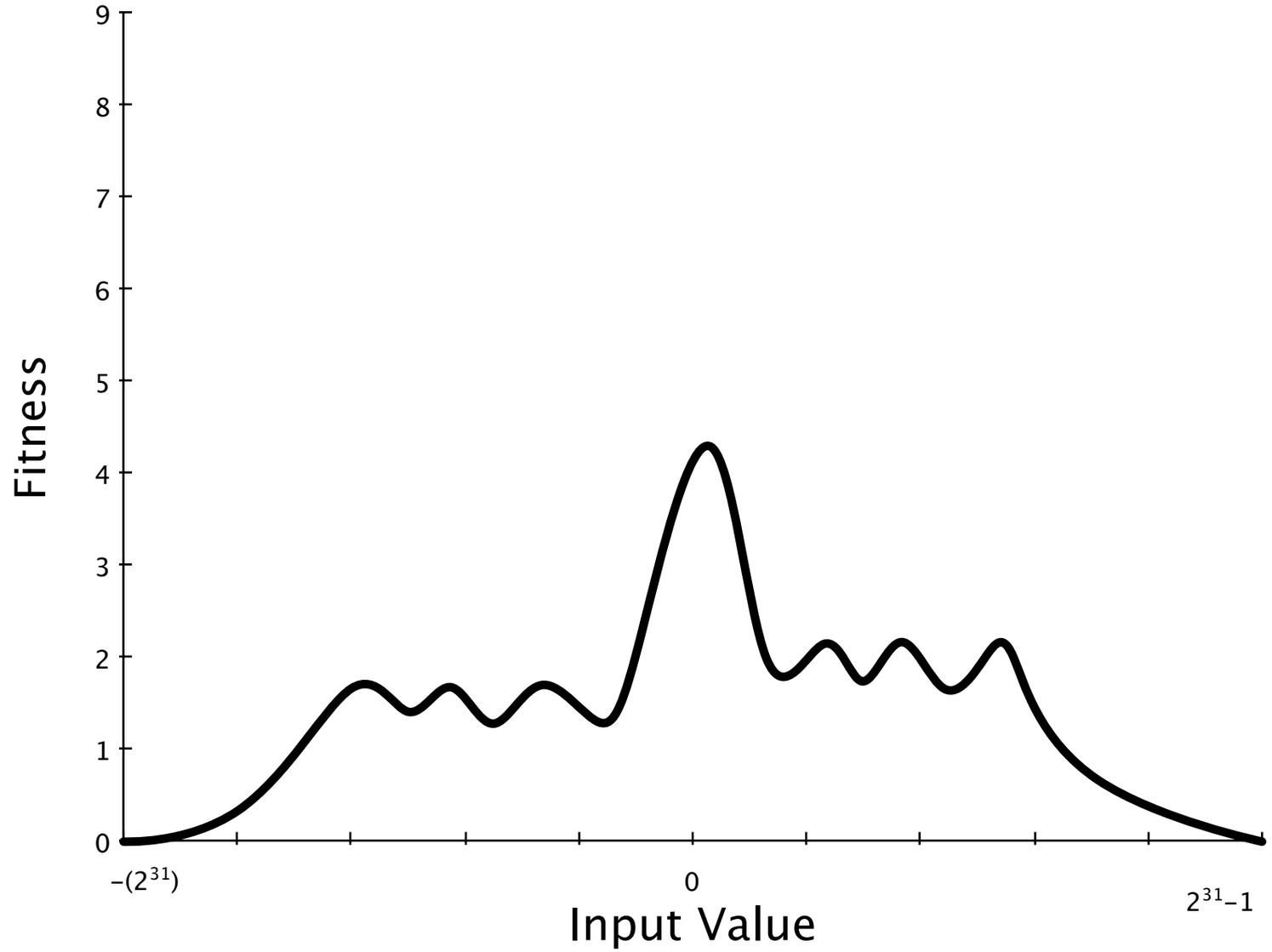
Putting it all together

Fitness = approach Level + *normalised* branch distance

```
void f1(int a, int b, int c, int d)
{
    if (a > b)
    {
        if (b > c)
        {
            if (c > d)
            {
                // target
. . .
```

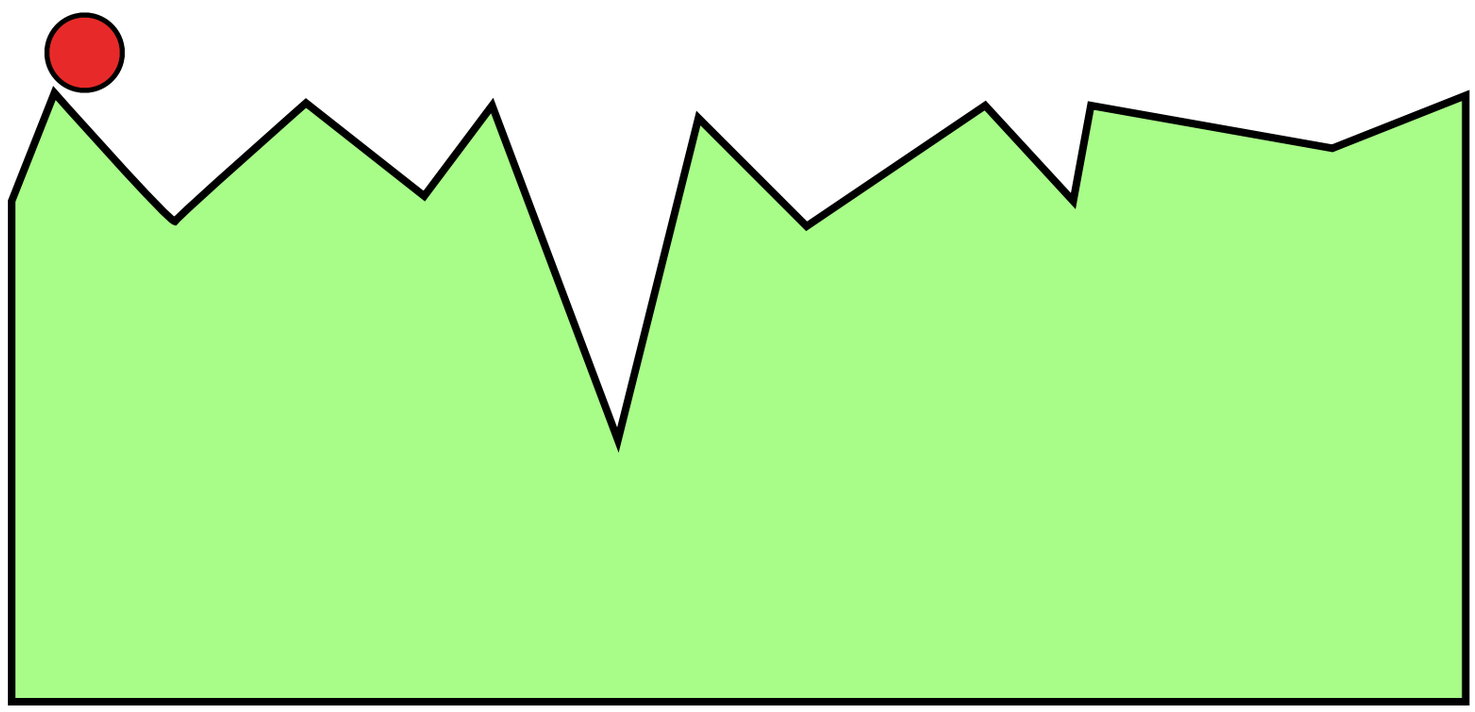


normalised branch distance between 0 and 1
indicates how close approach level is to being penetrated


Fitness
9
8
7
6
5
4
3
2
1
0
-(2^31)
0
2^31-1
Input Value

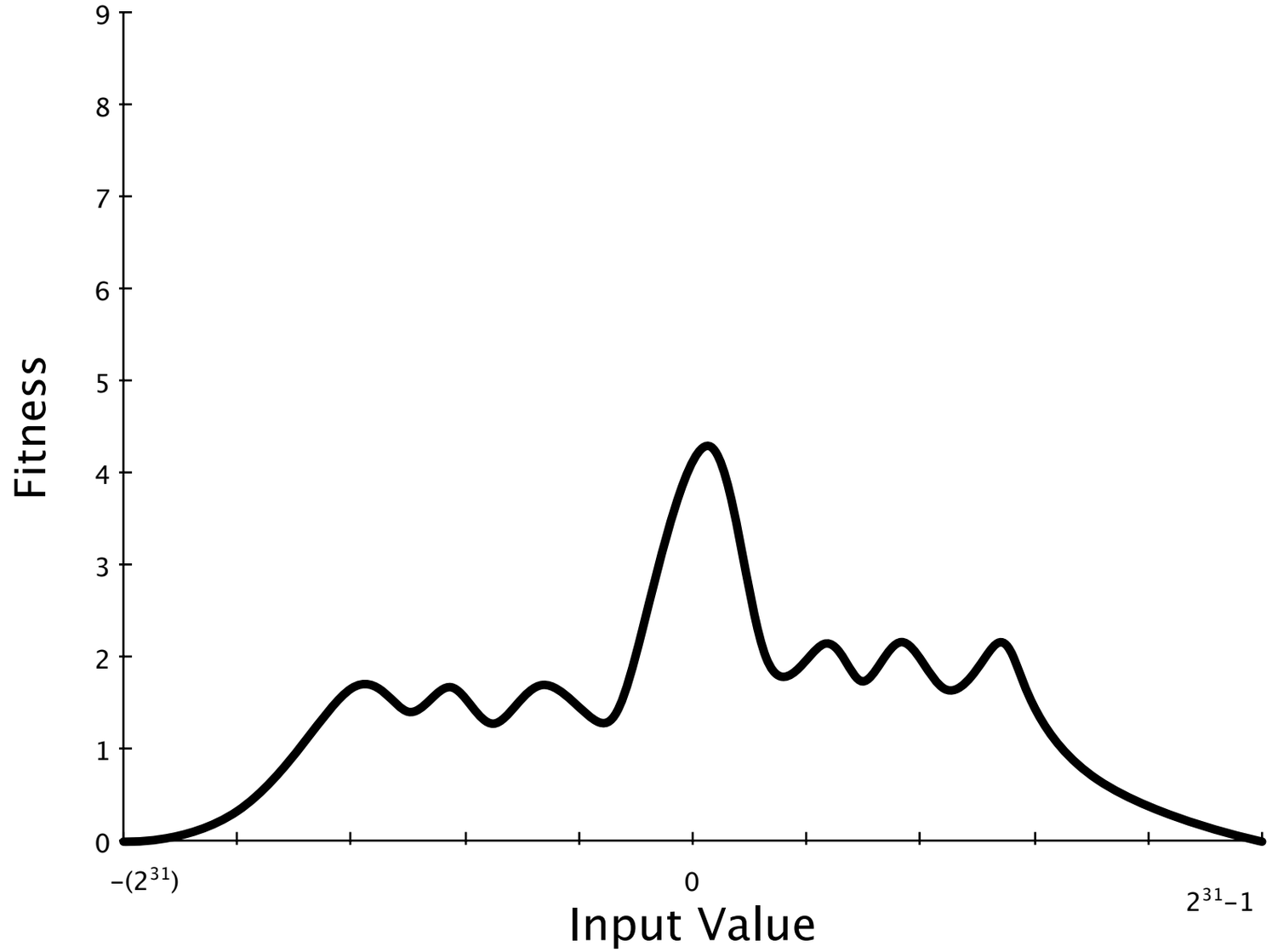
Simulated Annealing

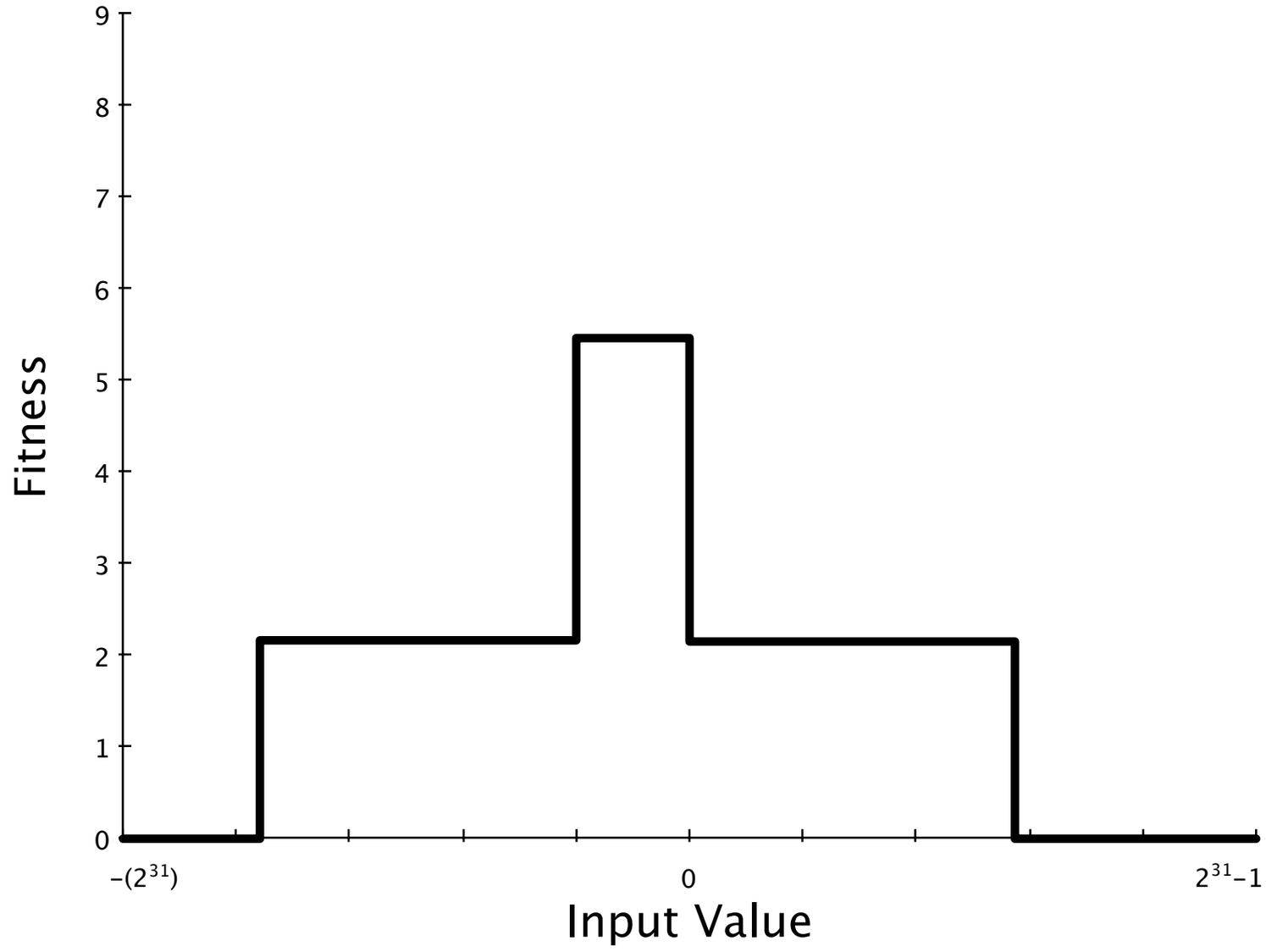
- Accept also worse solutions with $p = e^{\frac{\delta}{T(t)}}$
- Temperature $T(t) = \frac{d}{log(t)}$
- δ = difference in objective value



Components of an SBST Tool

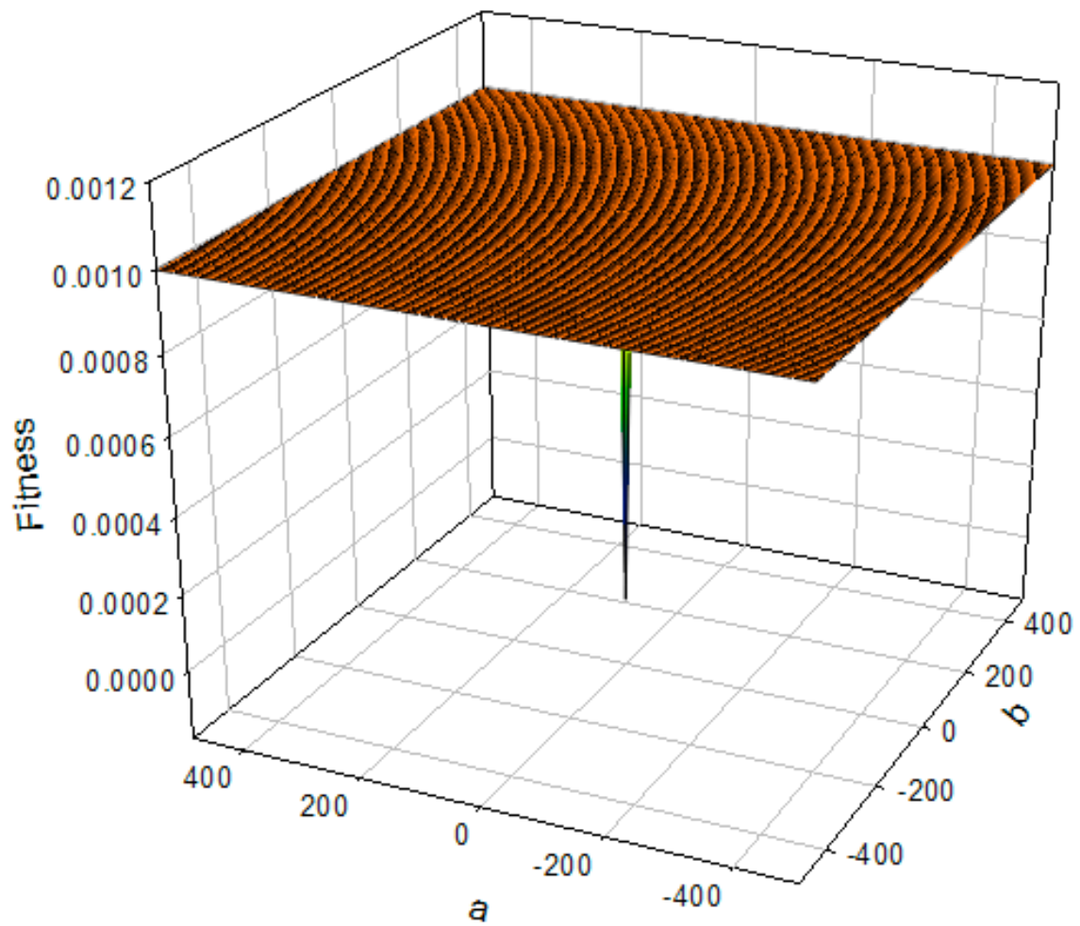
Search Algorithm	Simulated Annealing
Representation	Tuple (x, y)
Search Operators	Neighbourhood of (x, y)
Fitness Function	Branch distance
Test Execution	Call method
Instrumentation	BranchDistance instrumentation


Fitness
9
8
7
6
5
4
3
2
1
0
-(2^31)
0
2^31-1
Input Value


Fitness
9
8
7
6
5
4
3
2
1
0
-(2^31)
0
2^31-1
Input Value

The ‘Flag’ Problem

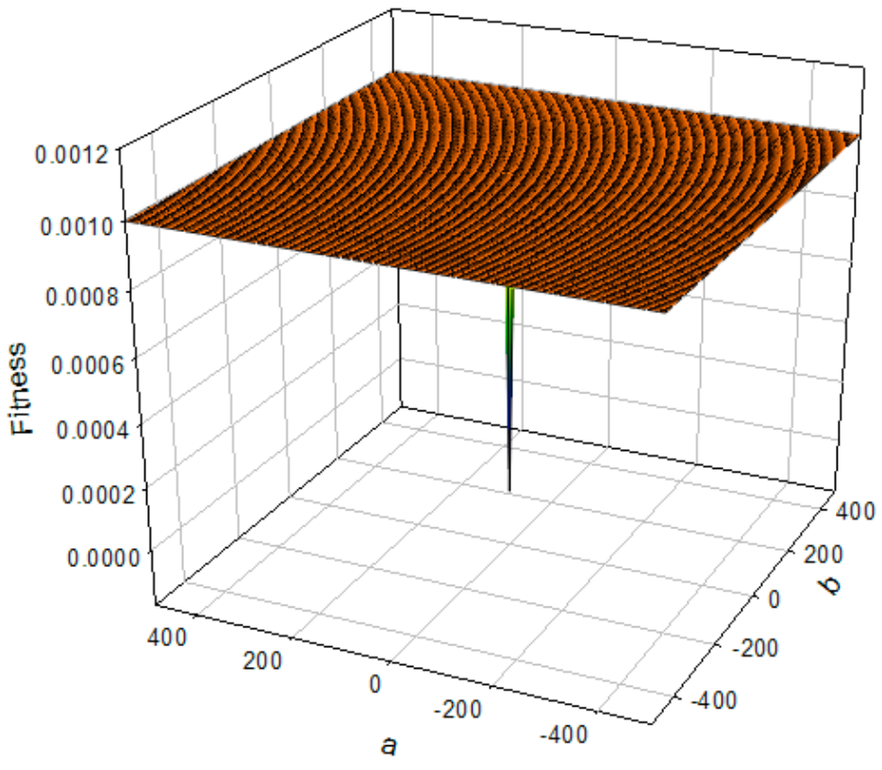
```
void testme(int a, int b, ....)
{
     ...

     flag = (a == 0 && b == 0);

     if (flag) {
          ....
     }
}
```

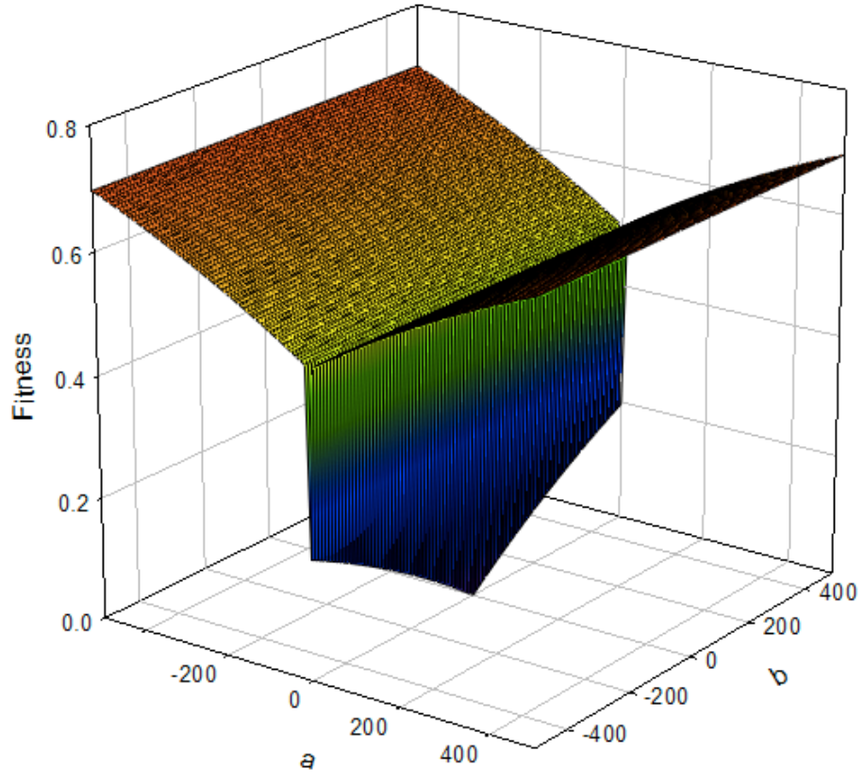


Program Transformation

```
void testme(int a, int b, ....)
{
    ...

    flag = (a == 0 && b == 0);

    if (flag) {
        ....
    }
}
```

```
void testme(int a, int b, ....)
{
    ...

    flag = (a == 0 && b == 0);

    if (a == 0 && b == 0) {
        ....
    }
}
```





Programs will inevitably have features that heuristic searches handle less well

Testability transformation:

change the program to improve test data generation

... whilst preserving test adequacy

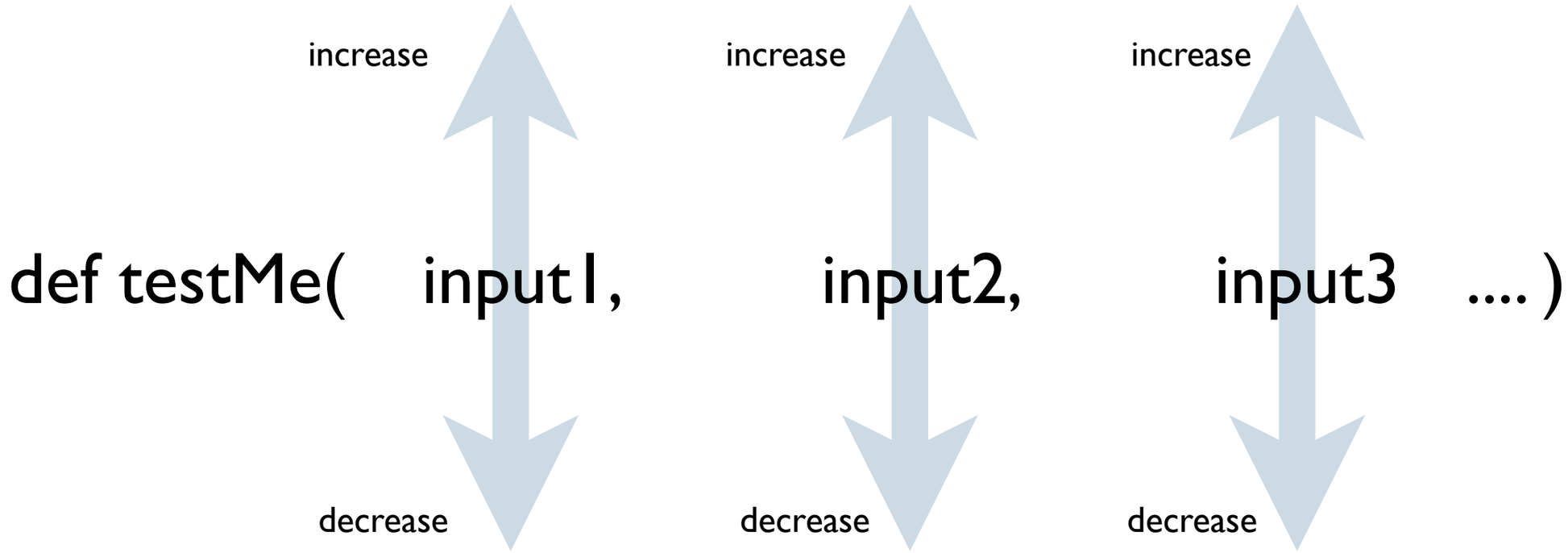
Mark Harman, Lin Hu, Rob Hierons, Joachim Wegener, Harmen Sthamer, Andre Baresel and Marc Roper.
Testability Transformation.
IEEE Transactions on Software Engineering. 30(1): 3-16, 2004.

```
def testMe(x, y, z):
    if x * z == 2 * (y + 1):
        return True
    else:
        return False
```

http://goo.gl/8mj8HC - 4.zip

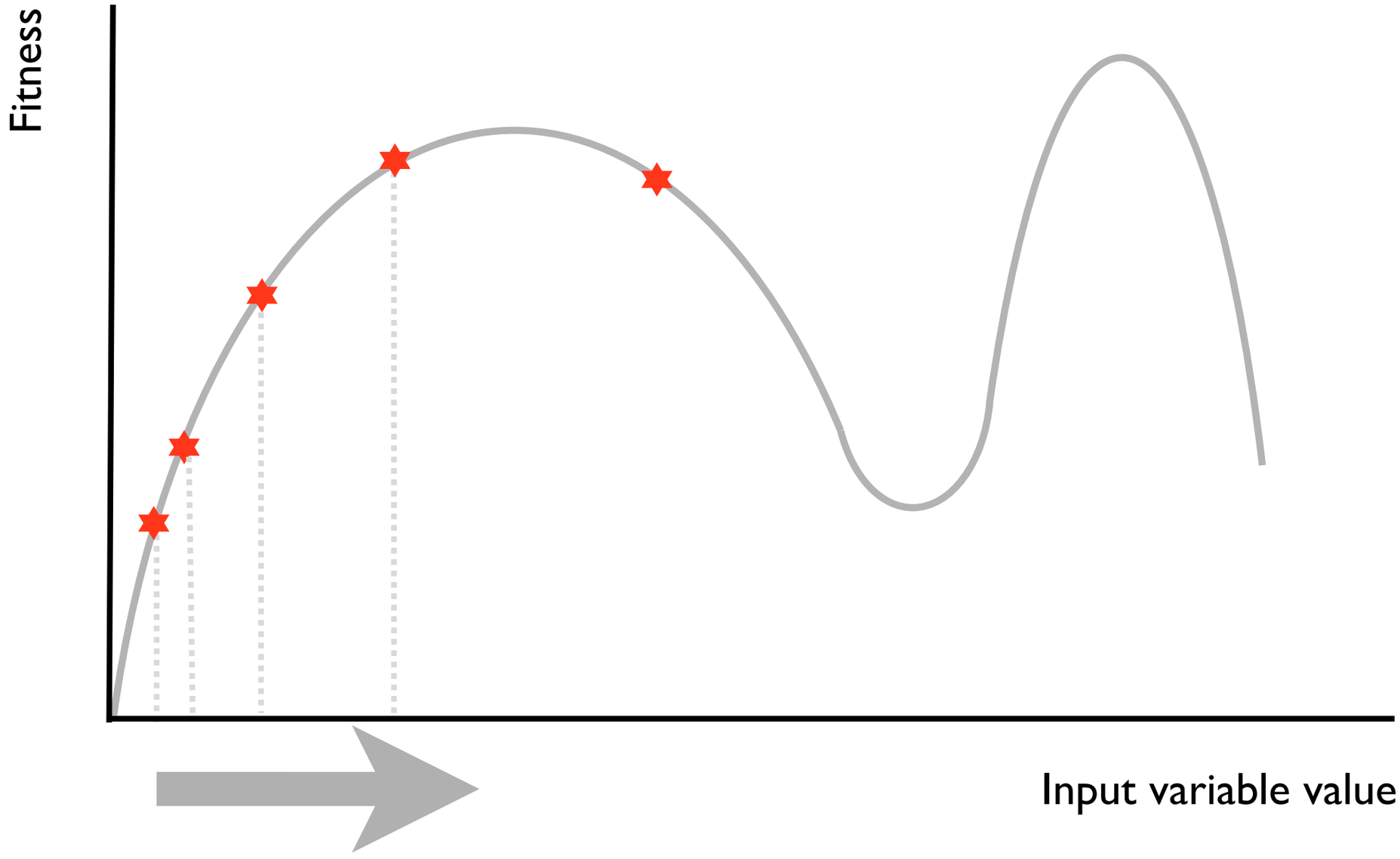
Alternating Variable Method

‘Probe’ moves



Alternating Variable Method

Accelerated hill climb



Alternating Variable Method

1. Randomly generate start point

   a=10, b=20, c=30

2. 'Probe' moves on a

   a=9, b=20, c=30

   a=11, b=20, c=30

   no effect

3. 'Probe' moves on b

   a=10, b=19, c=30

   improved branch distance

4. Accelerated moves in direction of improvement

```
void example(int a, int b, ...) {

    if (a == 0) {
        ...
    }

    if (b == 0) {
        // target
    }

    ...
}
```

```
def testMe(x, y, z):
    if x * z == 2 * (y + 1):
        return True
    else:
        return False
```

http://goo.gl/8mj8HC - 5.zip

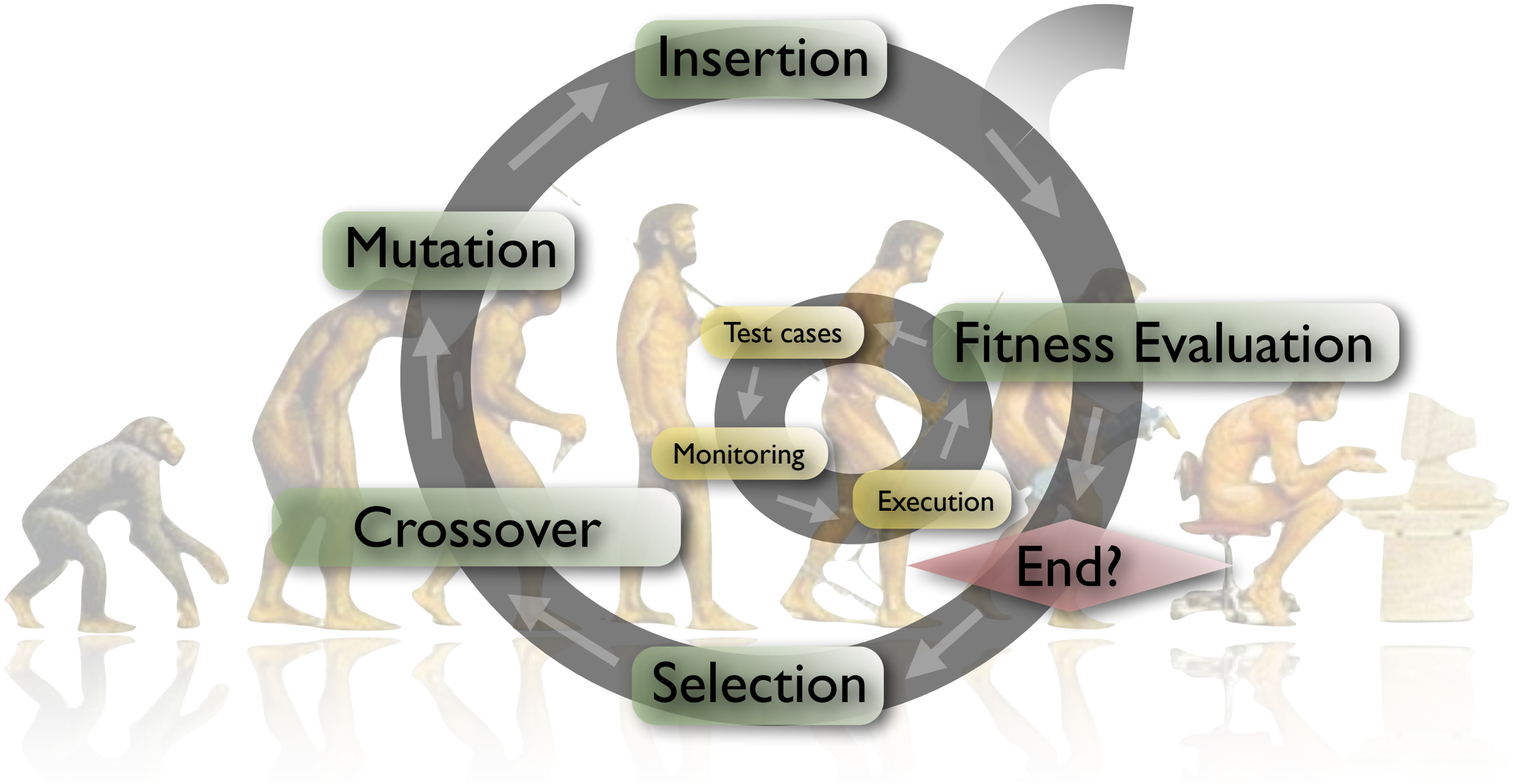
Components of an SBST Tool

Search Algorithm	Alternating Variable Method
Representation	Tuple (x, y, z)
Search Operators	Increment/decrement of x, y, and z
Fitness Function	Branch distance
Test Execution	Call method
Instrumentation	BranchDistance instrumentation

Contents

Contents

Evolutionary Testing



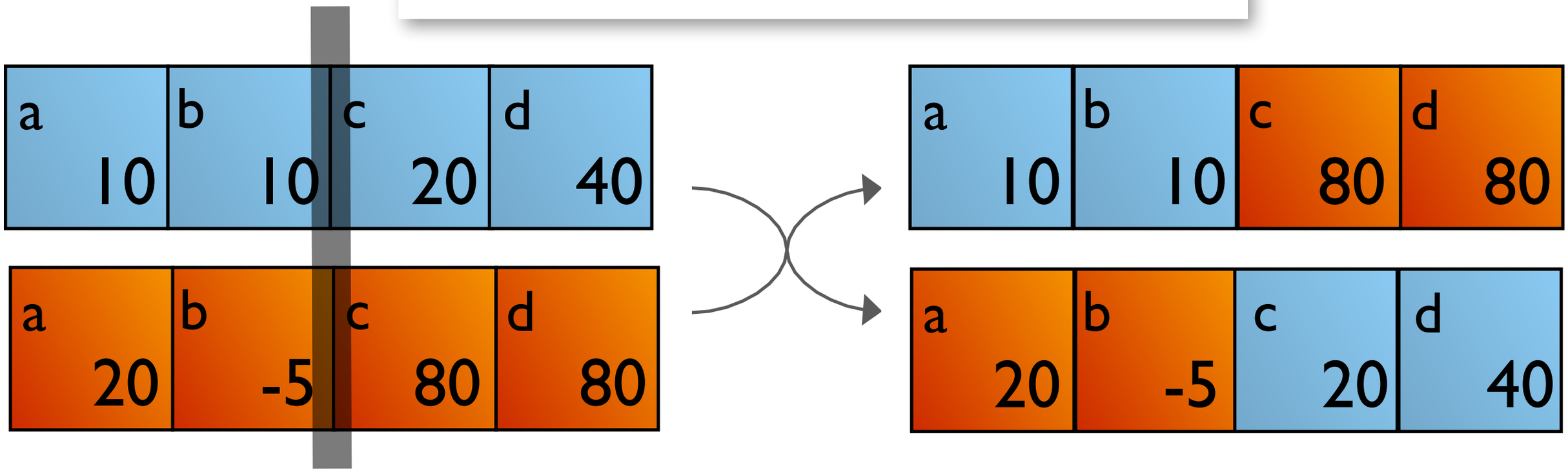
```
void test_me(int a, int b, int c, int d) {

   if (a == b) {

       if (c == d) {

           // branch we want to execute

       }
   }

   ...
```

Components of an SBST Tool

Search Algorithm	Genetic Algorithm
Representation	Tuple (a, b, c, d)
Search Operators	Crossover, mutation, selection
Fitness Function	Branch distance
Test Execution	Call method
Instrumentation	BranchDistance instrumentation

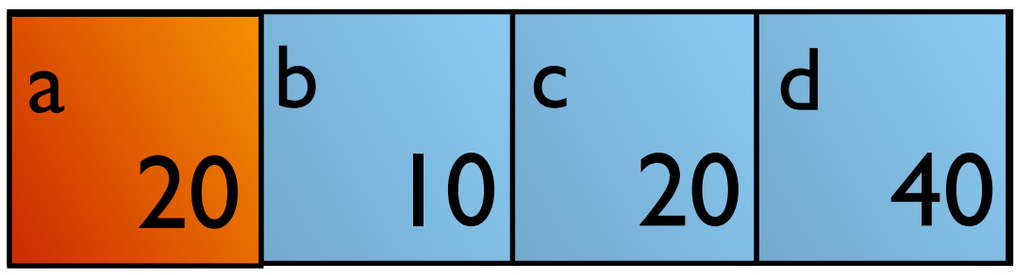
Crossover

```
void test_me(int a, int b, int c, int d) {

   if (a == b) {

       if (c == d) {

           // branch we want to execute

       }
   }

   ...
```



Mutation

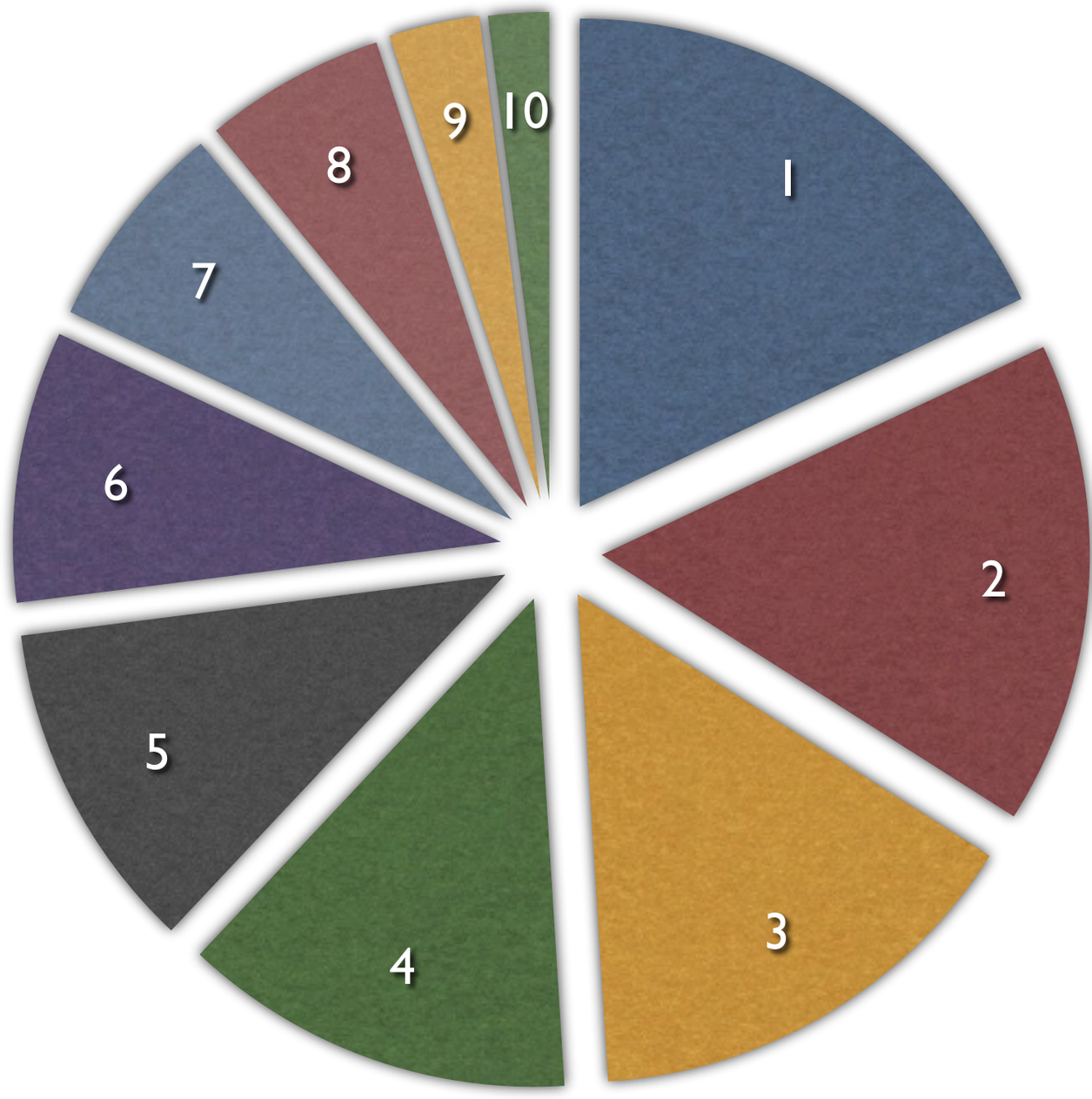
```
void test_me(int a, int b, int c, int d) {

   if (a == b) {

       if (c == d) {

           // branch we want to execute

       }
   }

   ...
```



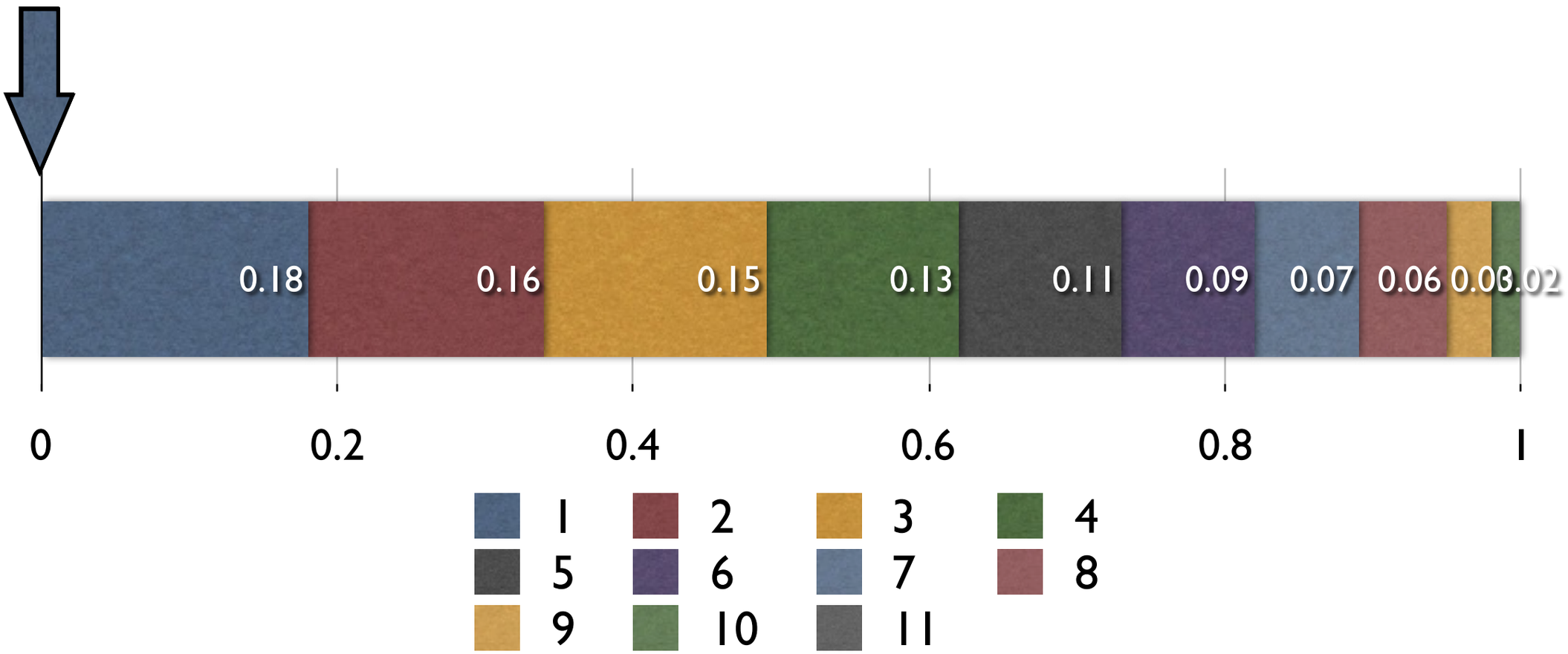
Selection

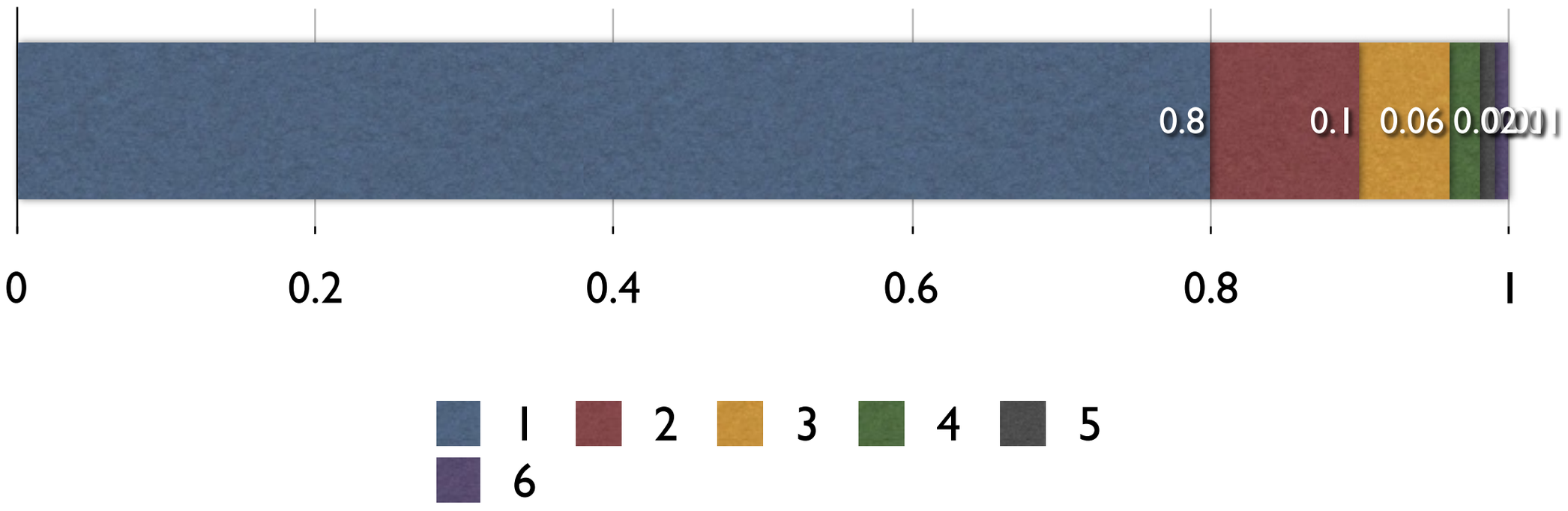
- Selective pressure:
  The higher, the more likely the fittest are chosen
- Stagnation:
  Selective pressure too small
- Premature convergence:
  Selective pressure too high

Individual	Fitness
1	2
2	1.8
3	1.6
4	1.4
5	1.2
6	1
7	0.8
8	0.6
9	0.4
10	0.2
11	0

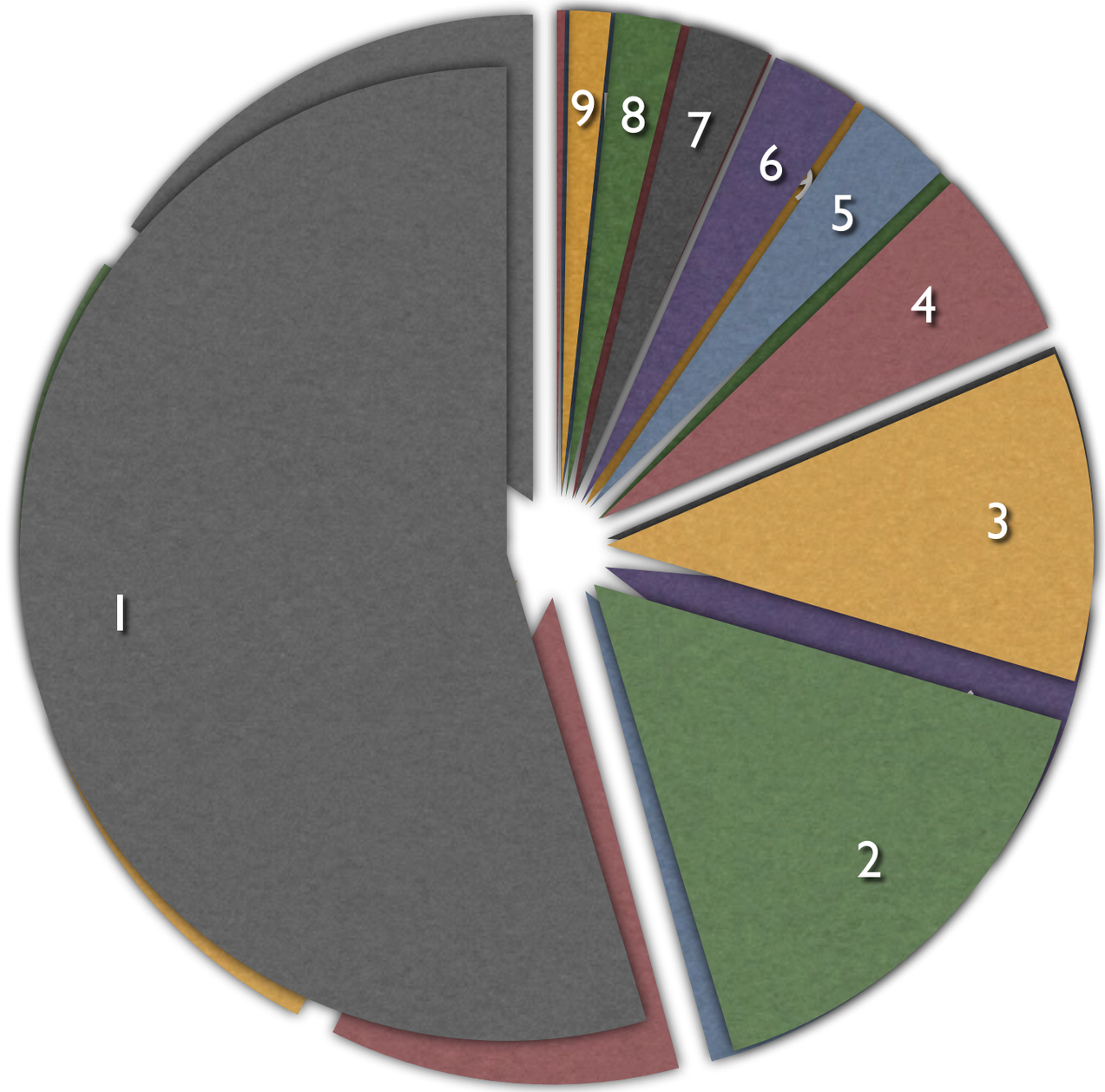


- Chosen value: 0,81
- Chosen value: 0,32
- Chosen value: 0,01




0.8
0.1
0.06
0.02
0
0.2
0.4
0.6
0.8
1
1
2
3
4
5
6

Individual	Fitness
1	2
2	0.58
3	0.4
4	0.21
5	0.12
6	0.11
7	0.1
8	0.08
9	0.05
10	0.01
11	0



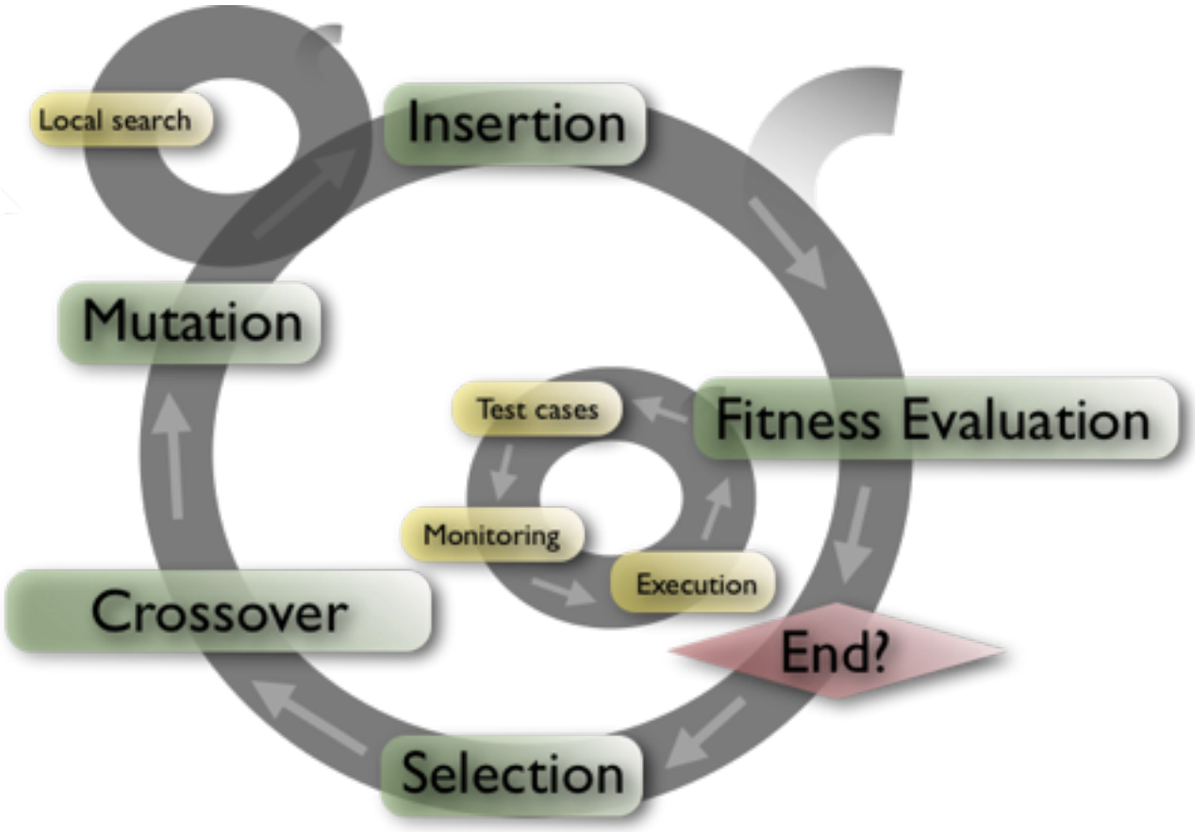
```
def testMe(a, b, c, d):
    if a == b and c == d:
        return True
    else:
        return False
```

http://goo.gl/8mj8HC - 6.zip

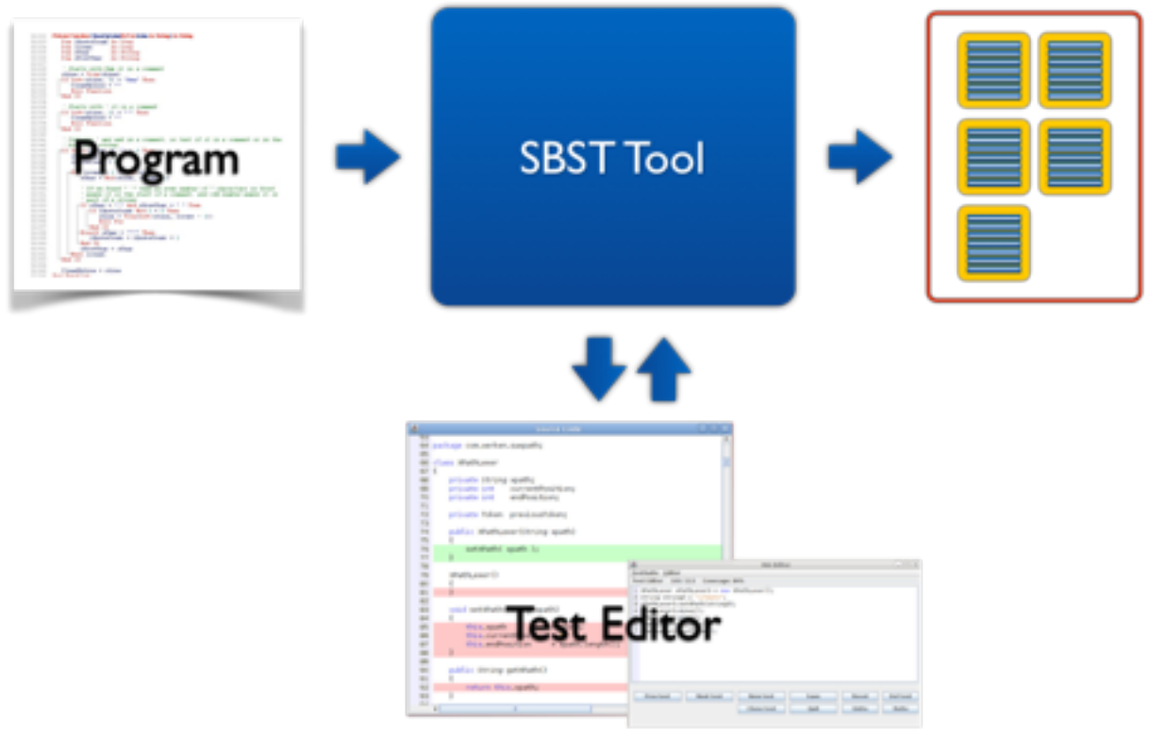
Other Heuristics in SBST

- Particle Swarm Optimisation
- Chemical Reaction Optimisation
- Ant Colony Optimisation
- Estimation of Distribution Algorithms
- Novelty Search

Hybrid SBST Approaches



Memetic Algorithms
SBST+DSE



Interactive GA

No Free Lunch Theorem

- In the entire domain of search problem, all search algorithms perform equally on average
- Each algorithm makes assumptions on the underlying problem
- Choose / adapt algorithms to specific domain

Contents

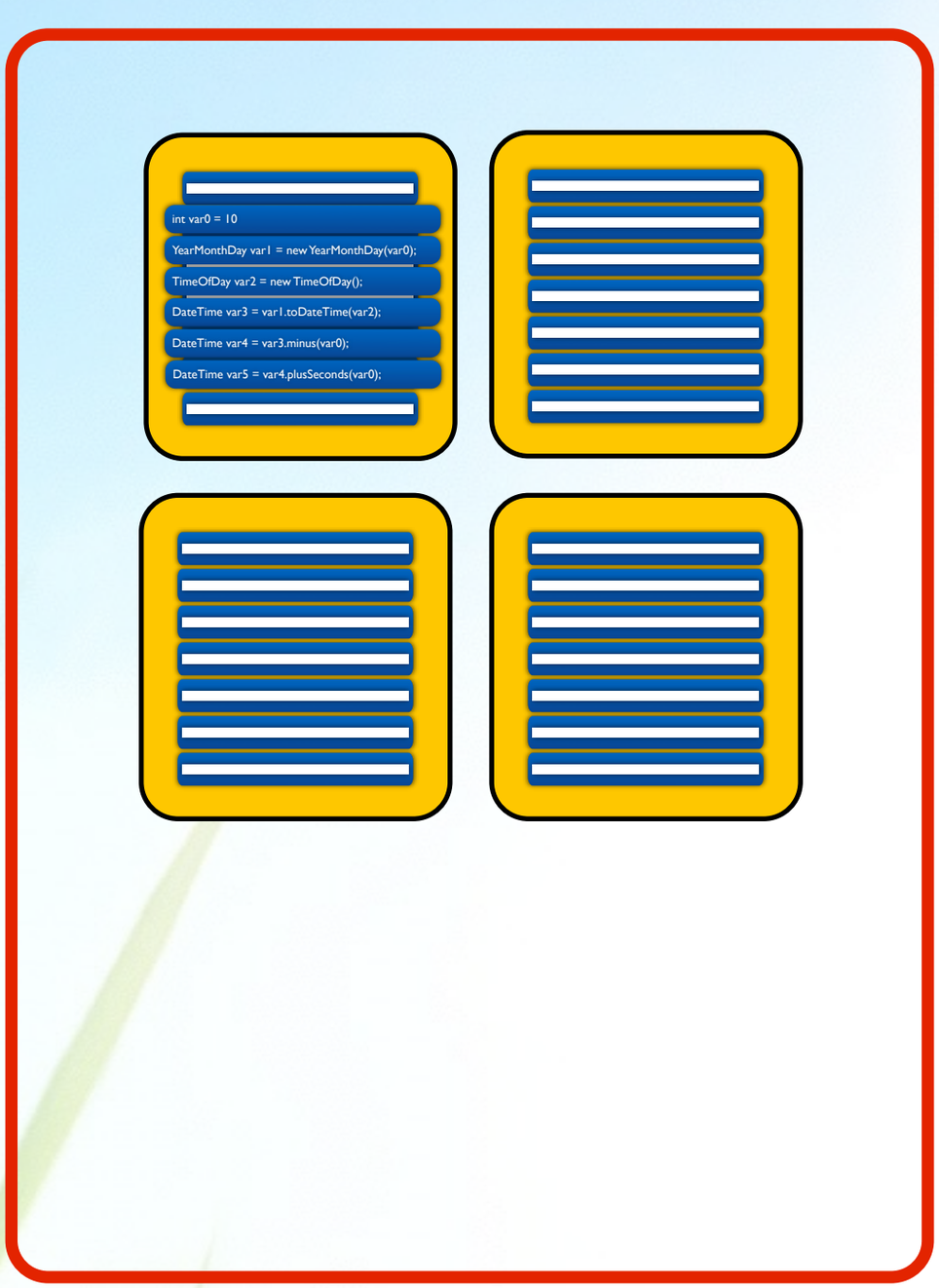
Contents

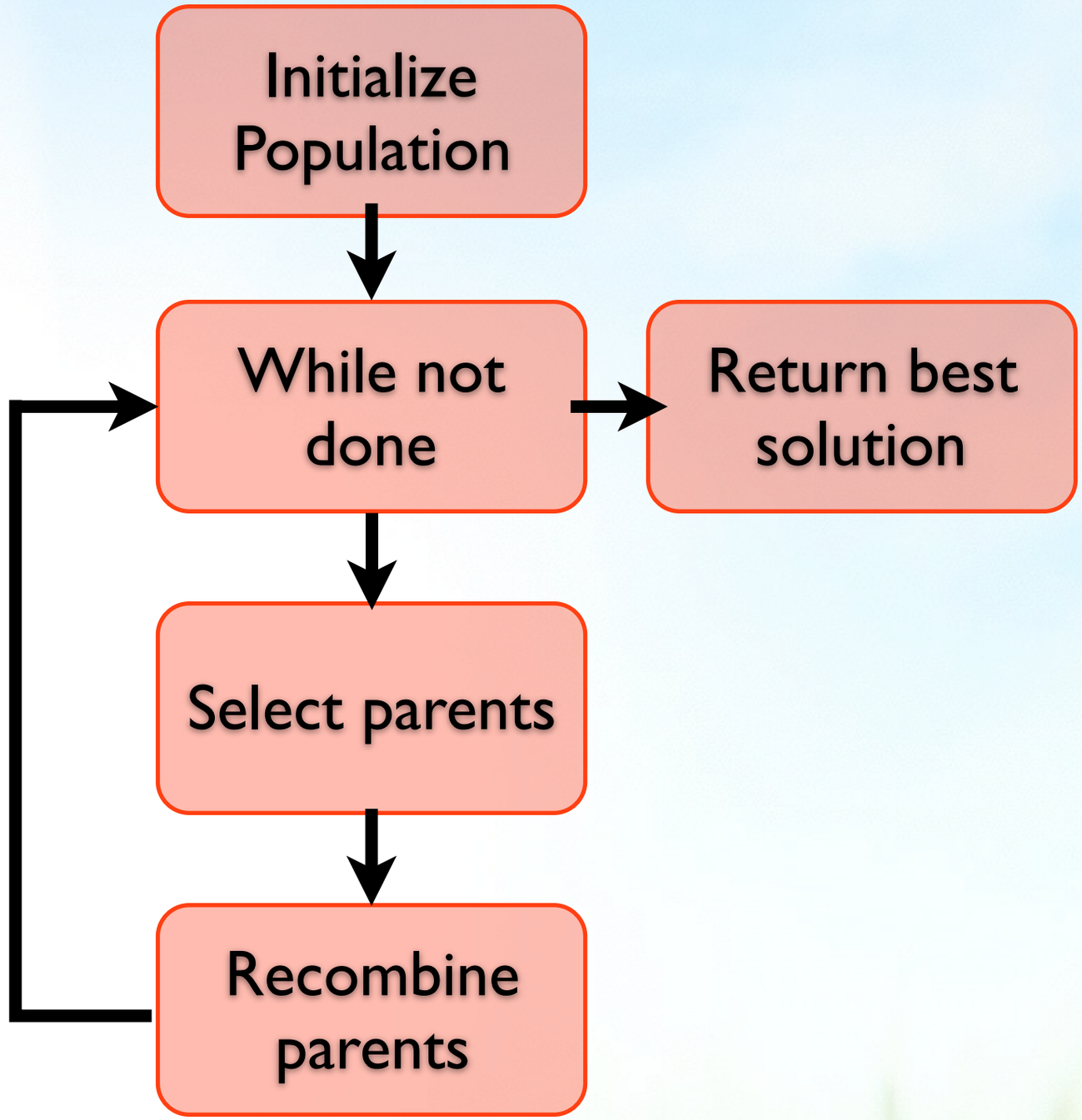
```
@Test
public void test()
{
   int x = 2;
   int y = 2;
   int result = x + y;
   assertEquals(4, result);
}
```

EVOSUITE

```
@Test
public void test()
{
    int var0 = 10
    YearMonthDay var1 = new YearMonthDay(var0);
    TimeOfDay var2 = new TimeOfDay();
    DateTime var3 = var1.toDateTime(var2);
    DateTime var4 = var3.minus(var0);
    DateTime var5 = var4.plusSeconds(var0);
}
```

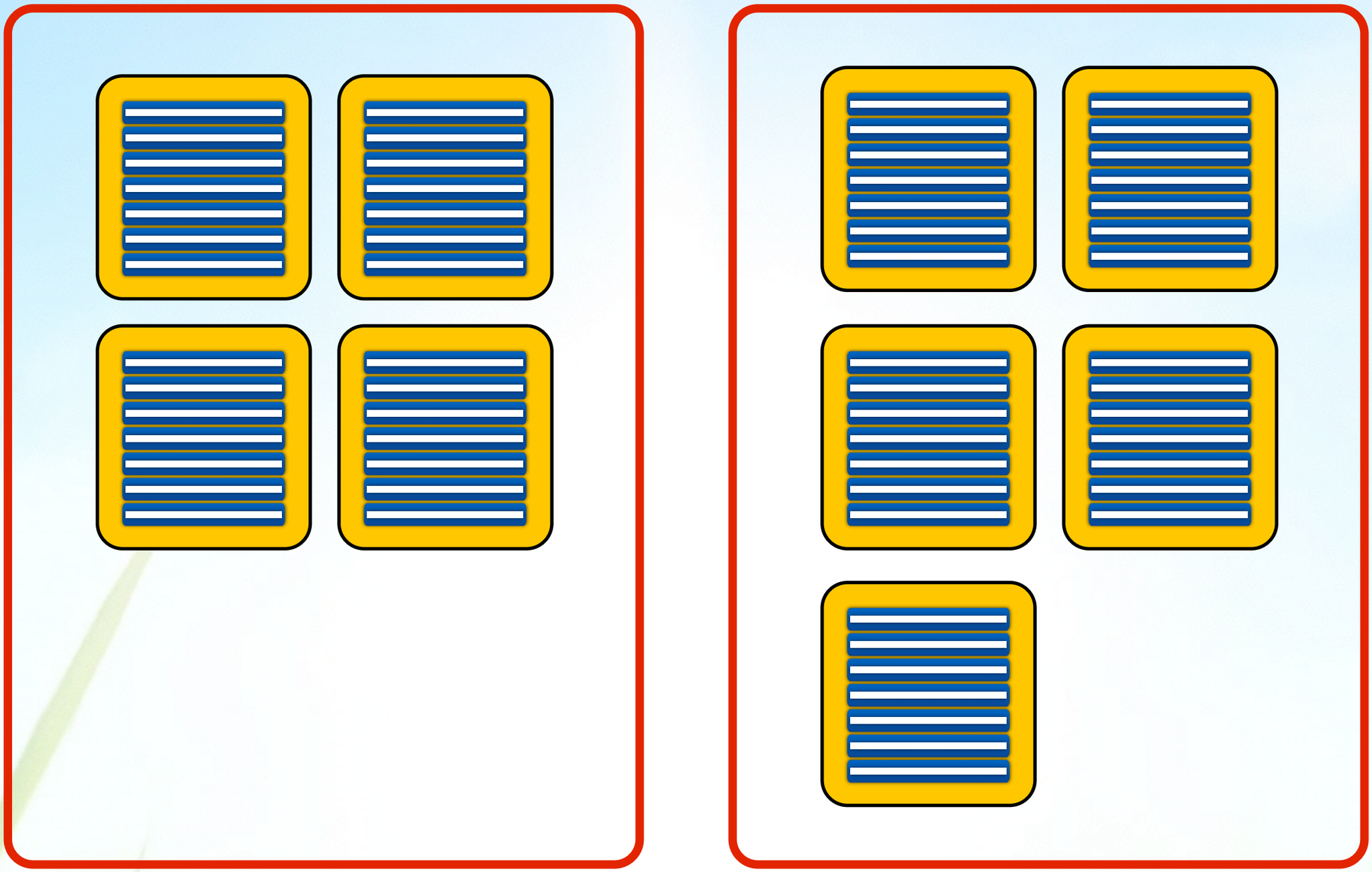
Test Suite Generation



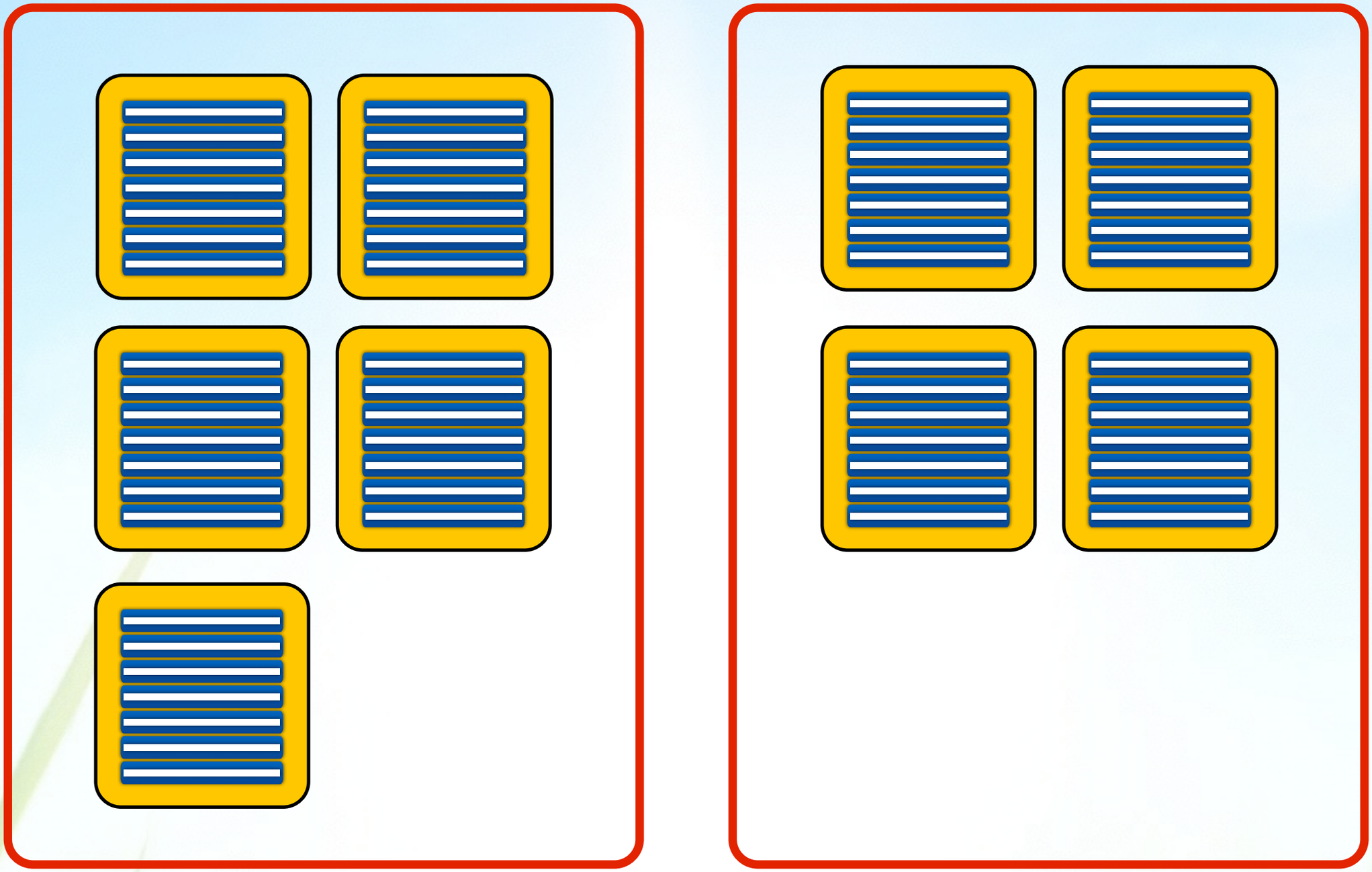


EVOSUITE

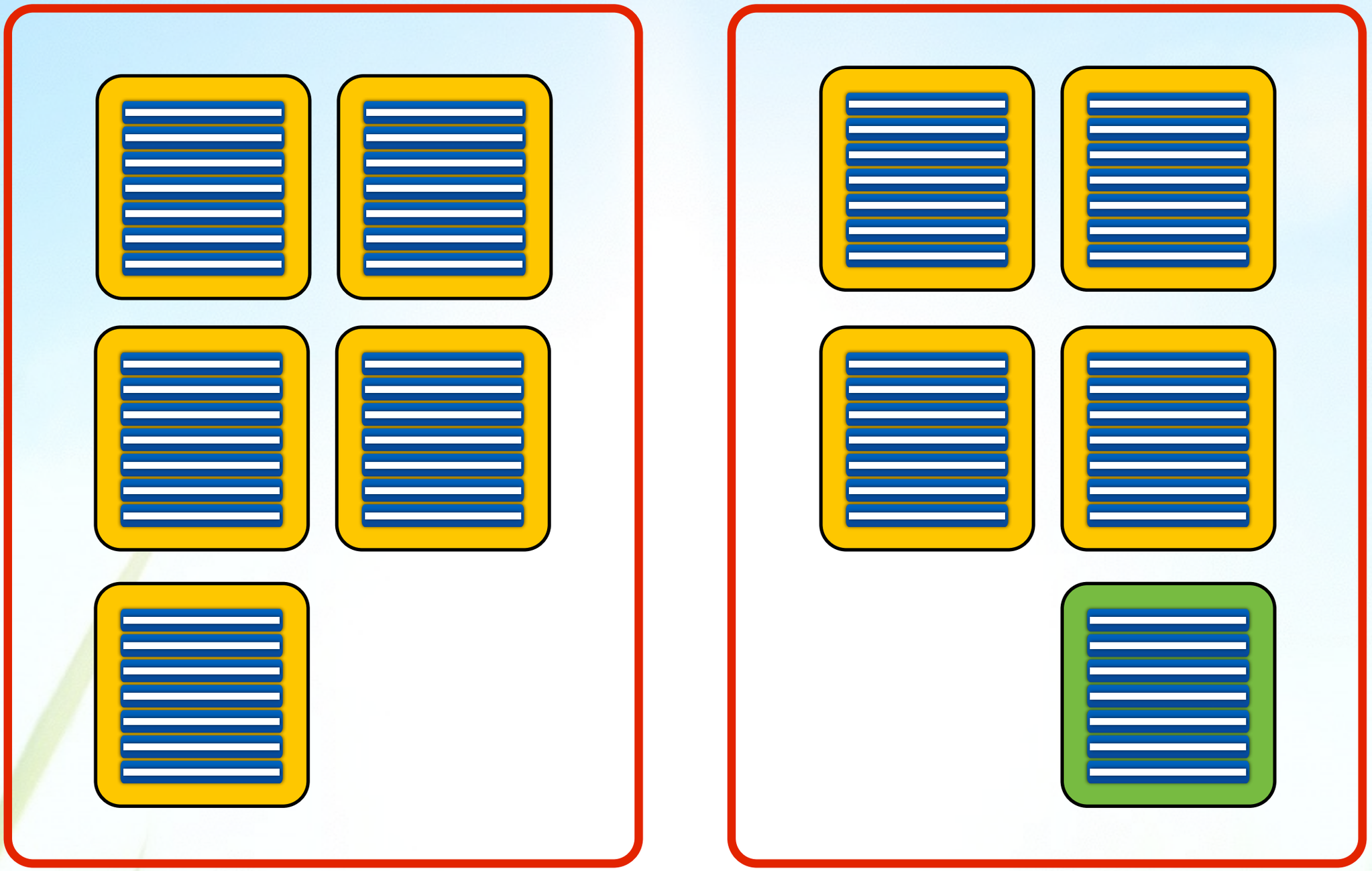
Test Suite Generation



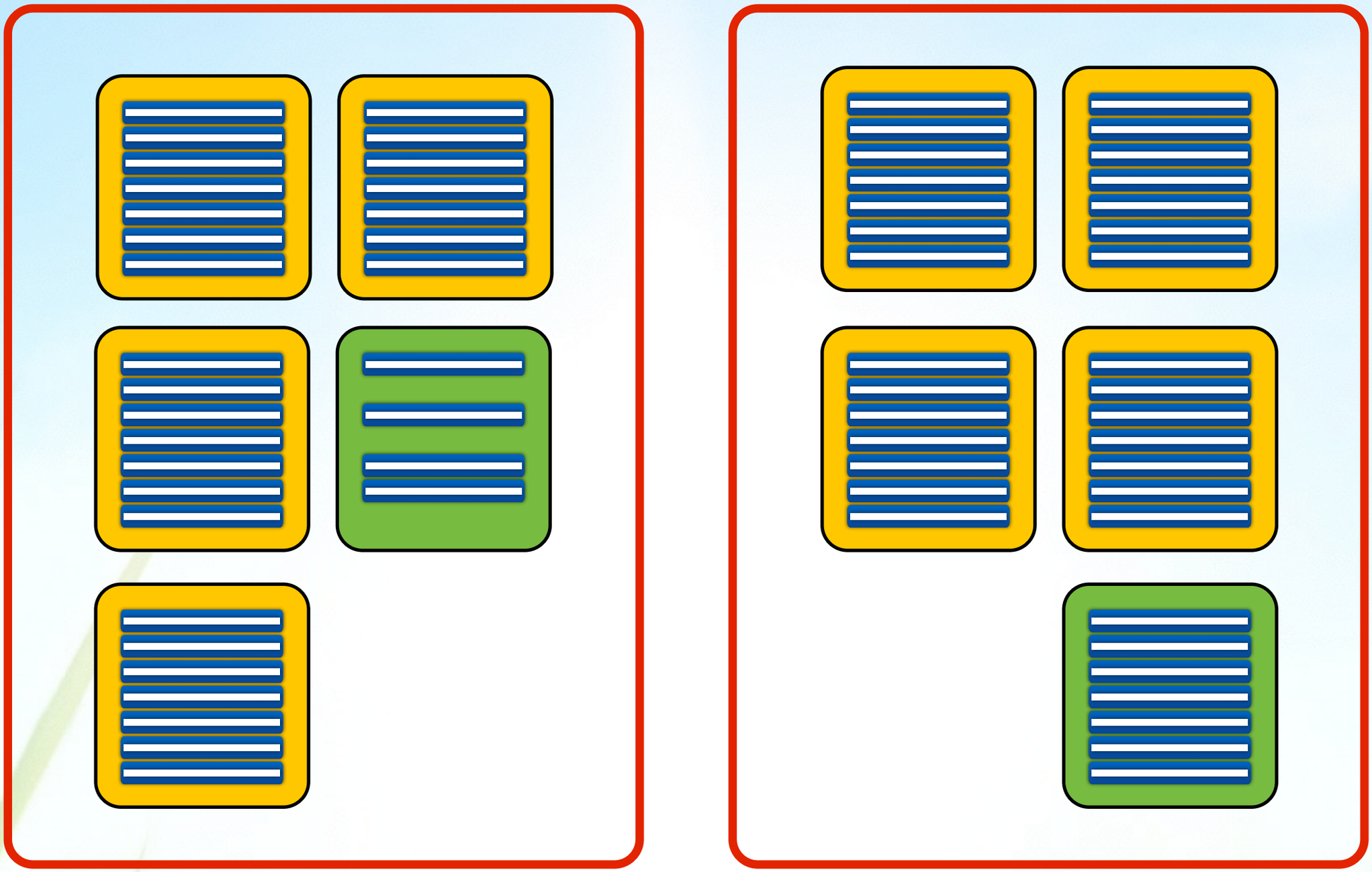
Crossover



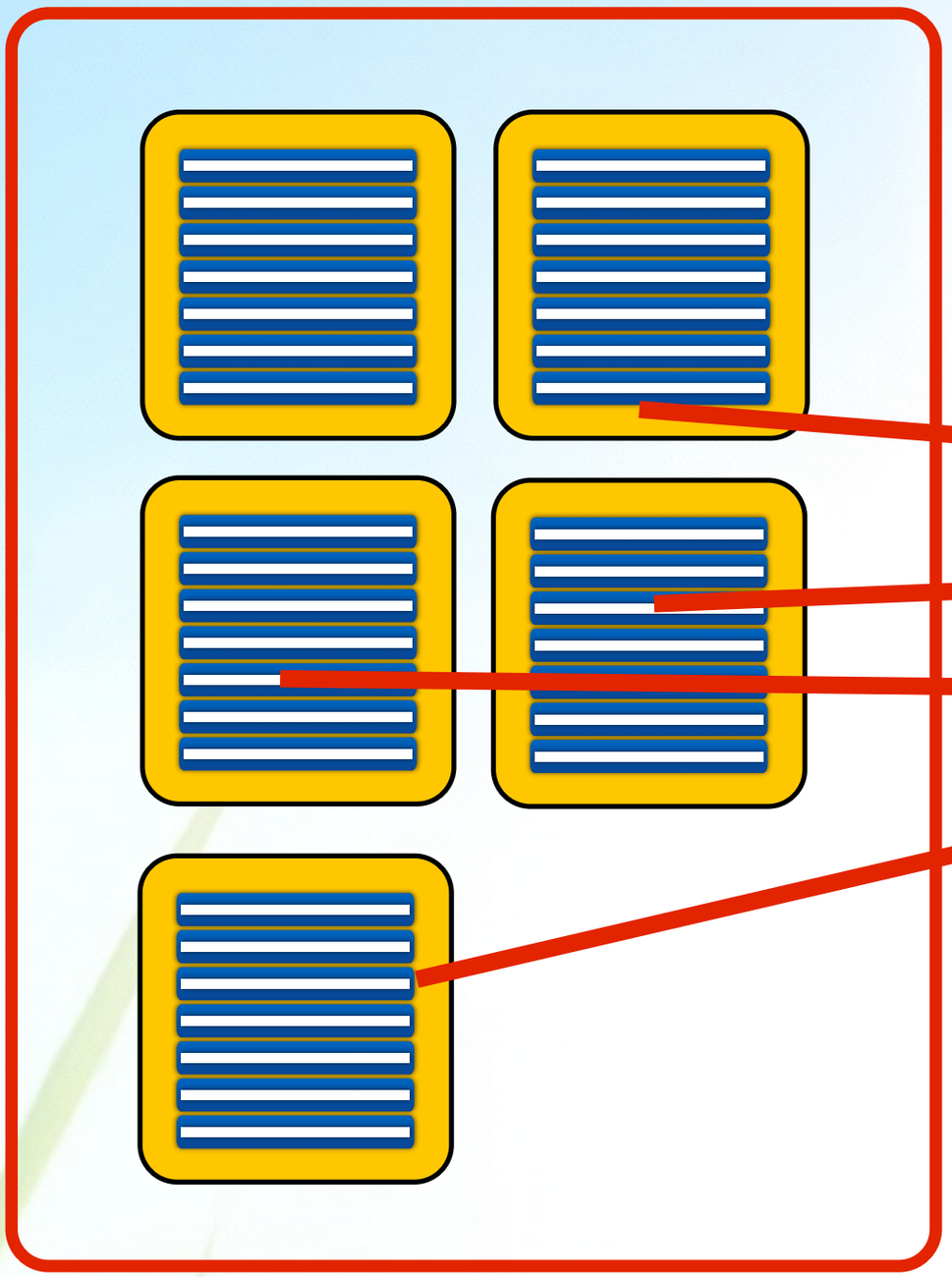
Mutation



Mutation



Fitness



```
public int gcd(int x, int y) {
  int tmp;
  while (y != 0) {
    tmp = x % y;
    x = y;
    y = tmp;
  }
  return x;
}
```

Getting EvoSuite

`http://www.evosuite.org/downloads`

- Jar release - for command line usage
- Maven plugin
- IntelliJ plugin
- Eclipse plugin
- Jenkins plugin

Testing a Class

- Demo - command line
- Main options:
  `-projectCP`
  `-class`
  `-criterion`

Properties

- `-Dproperty=value`
- Search budget (s)
  `-Dsearch_budget=60`
- Assertion generation
  `-Dassertions=false`
  `-Dassertion_strategy=all`
- Minimisation (length and values)
  `-Dminimize=false`
- Inlining
  `-Dinline=false`

EvoSuite Sandbox

- Demo - Nondeterministic class
- Runtime library to execute tests

EVOSUITE

Testing multiple classes

Demo:

- Target / prefix
- Continuous
- Maven
- Jenkins
- IntelliJ

Experiment Exercise

- EvoSuite by default uses a combination of different coverage criteria.
- RQ1: Does the combination lead to larger test suites than just using branch coverage?
- RQ2: Does the combination lead to better test suites than just using branch coverage?

Experiment Exercise

- Get evosuite-1.0.3.jar from http://evosuite.org/downloads
- Get http://evosuite.org/files/tutorial/Tutorial_Experiments.zip
- Common Options:

```
-projectCP target/classes -prefix tutorial -Dsearch_budget=20
-Doutput_variables=configuration_id,TARGET_CLASS,
 Size,Length,MutationScore
```

- Treatment 1:

```
java -jar evosuite-1.0.3.jar –Dconfiguration_id=Default
<common options>
```

- Treatment 2:

```
java -jar evosuite-1.0.3.jar -Dconfiguration_id=Branch
-criterion branch <common options>
```

- Resulting data:

```
evosuite-report/statistics.csv
```

Building EvoSuite

- Git repository:
  `git clone https://github.com/EvoSuite/evosuite.git`
- Maven
  `mvn package`
  `(mvn -DskipTests package)`
- Where is EvoSuite now?
  `master/target/evosuite-master-1.0.4-SNAPSHOT.jar`
- Why is the jar file so huge?

Module Structure

- master
- client
- runtime
- standalone-runtime
- plugins
- generated
- shaded

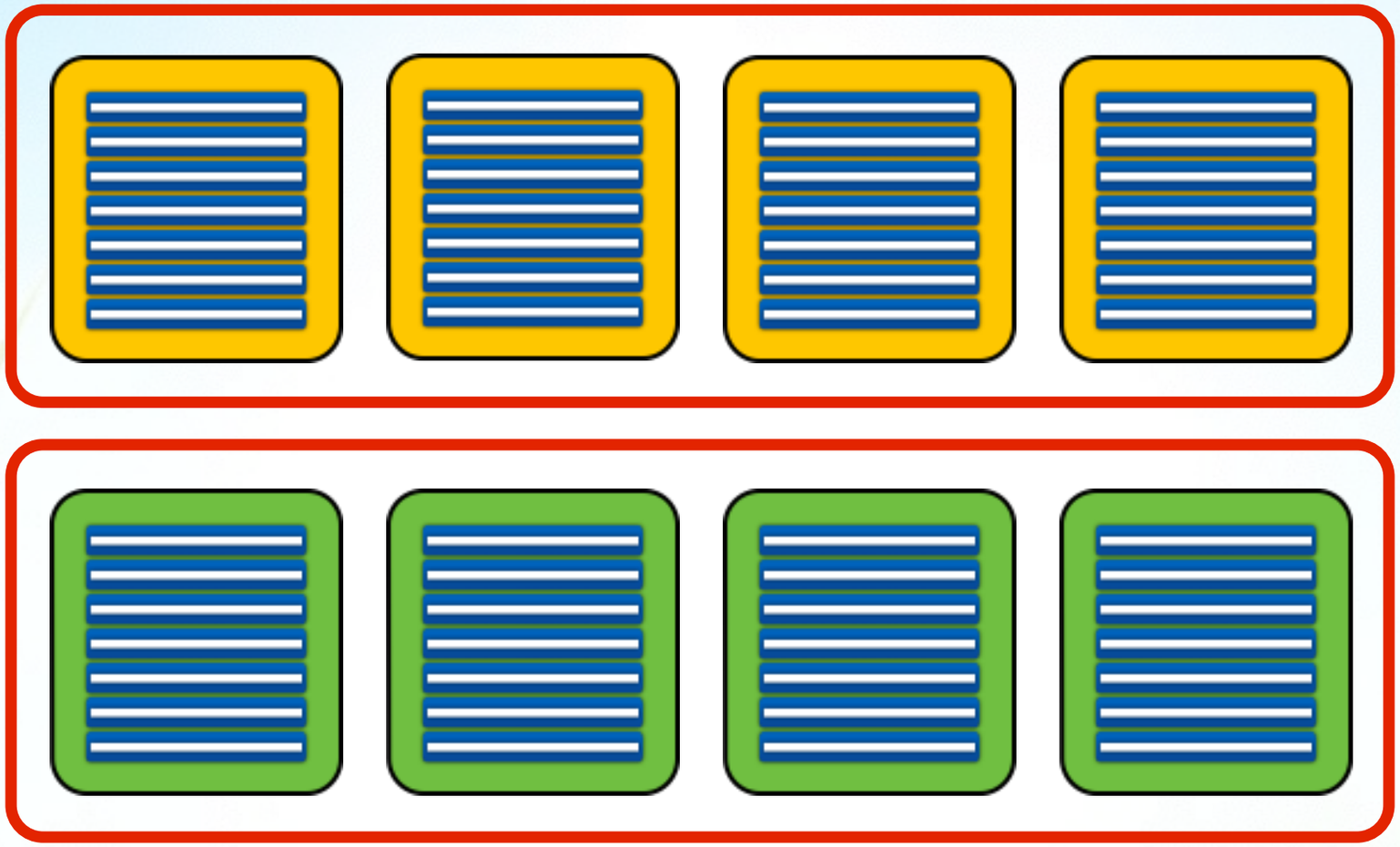
Testing EvoSuite

Example Test:

```
public class NullStringSystemTest extends SystemTestBase {

	@Test
	public void testNullString() {
		EvoSuite evosuite = new EvoSuite();

		String targetClass = NullString.class.getCanonicalName();
		Properties.TARGET_CLASS = targetClass;

		String[] command = new String[] { "-generateSuite", "-class", targetClass };

		Object result = evosuite.parseCommandLine(command);
		GeneticAlgorithm<?> ga = getGAFromResult(result);
		TestSuiteChromosome best = (TestSuiteChromosome) ga.getBestIndividual();

		int goals = TestGenerationStrategy.getFitnessFactories().
			get(0).getCoverageGoals().size();
		Assert.assertEquals("Wrong number of goals: ", 3, goals);
		Assert.assertEquals("Non-optimal coverage: ", 1d, best.getCoverage(), 0.001);
	}
}
```

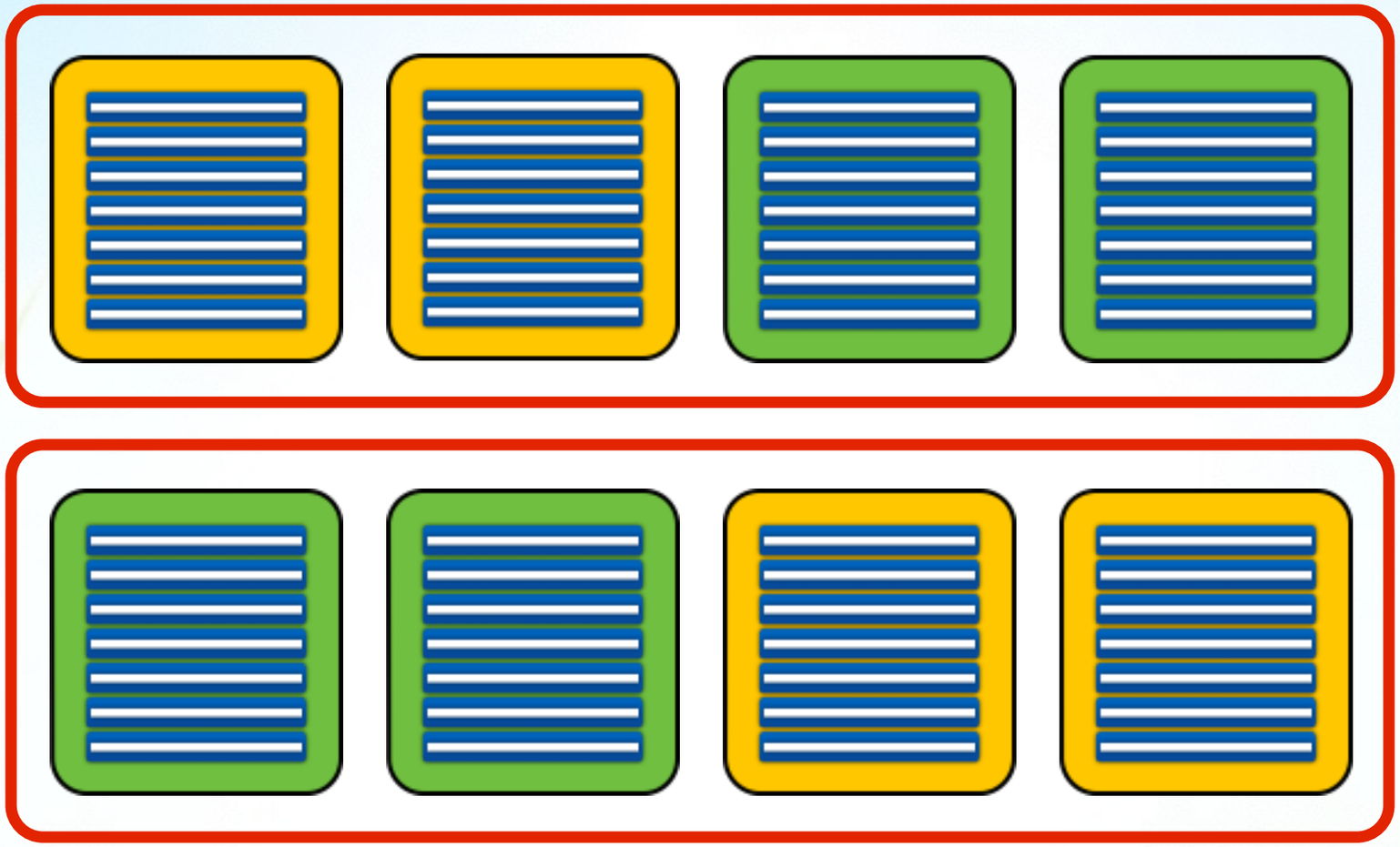
Extending EvoSuite

- (Artificial) Example: Middle point crossover



Extending EvoSuite

- (Artificial) Example: Middle point crossover



Extending EvoSuite

- Demo:
  MiddleCrossOver class
  Additional property
  Test case

Contents

Contents

EVOSUITE

I . Java

…is a weird language
and never ceases to surprise me

My personal enemy: Java Generics

Bytecode over sourcecode - yes!

2. Corner Cases

The more corner cases you cover

...the more can go wrong

...the more new corner cases you will find

...the slower EvoSuite becomes

2. Corner Cases

- Constant Seeding: +5%
- Virtual FS: +1.4%
- Mocking +4.7%
- JEE support: +3%
- DSE: +1.2%

EVOSUITE

3. Developers

…some really care only about coverage

…others don’t care about coverage:

"I wouldn’t normally in real life be aiming for 100% coverage. I’d probably end up with fewer tests without this tool but I couldn’t tell you if they would be all the right tests.”

…do not want their tests to be generated

…hate ugly tests

…don’t like waiting

Talk to them!

3. Developers

```
public class Example {

    private Example() {}

    // …
}
```

4. Testing

Testing randomised algorithms is difficult

Make the implementation deterministic

Always use LinkedHashSet over HashSet, LinkedHashMap over HashMap

Java reflection is not deterministic

Avoid static state (e.g. singletons)

EVOSUITE

4. Testing

EvoSuite uses one central random number generator

Any change will affect something at a completely different part of the program

Change seeds frequently during testing to find flaky tests

EVOSUITE

5. Documentation

I don’t comment my code

Students struggle

I spend more time explaining things than it would take me to implement them

6. Tool Comparisons

Reviewers want to see them

I don't like doing them

It's impossible to make them fair

Contact tool authors

Report bugs

Make your own tools *usable*

7. Open Source

"The source code will be released under an open source library (most likely GPL2) at a later point, as soon as a number of refactorings are completed." — FSE'11 tool paper appendix

Public GitHub repo: 2015

It will never be clean enough, just release it!

8. Licensing

License matters

Google will not touch GPL

BSD, MIT - do you want others to become rich with your idea?

Gnu Lesser Public License, Apache

9. Tool Papers

The first one will be cited

The rest no one will cite

It shouldn't be this way

10. Robustness

Creating a robust tool…

…is a huge effort

…it will never be finished

EvoSuite is a black hole swallowing all my time!

Contents

Contents

1. SBST is Slow

- Fitness evaluation means executing tests
- Executing tests is slow
- How to reduce the number of fitness evaluations?
- How to improve search operators?
- Can we use ML to predict test execution results?

2. OO Guidance

- Object oriented code has a terrible search landscape
- Complex dependency objects are a problem
- Include dependency objects in fitness functions?
- Better testability transformations?
- Better fitness functions?

3. Technical Challenges

- Integration testing
- Concurrent code
- GUI handling code
- Database dependent code
- Prioritising tests
- Dynamically typed languages

4. SBST Usability

- Assertion/contract testing code?
- Coverage isn't a great objective
- Usability as optimisation goal
- Study developers using SBST tools

5. Anything but Coverage

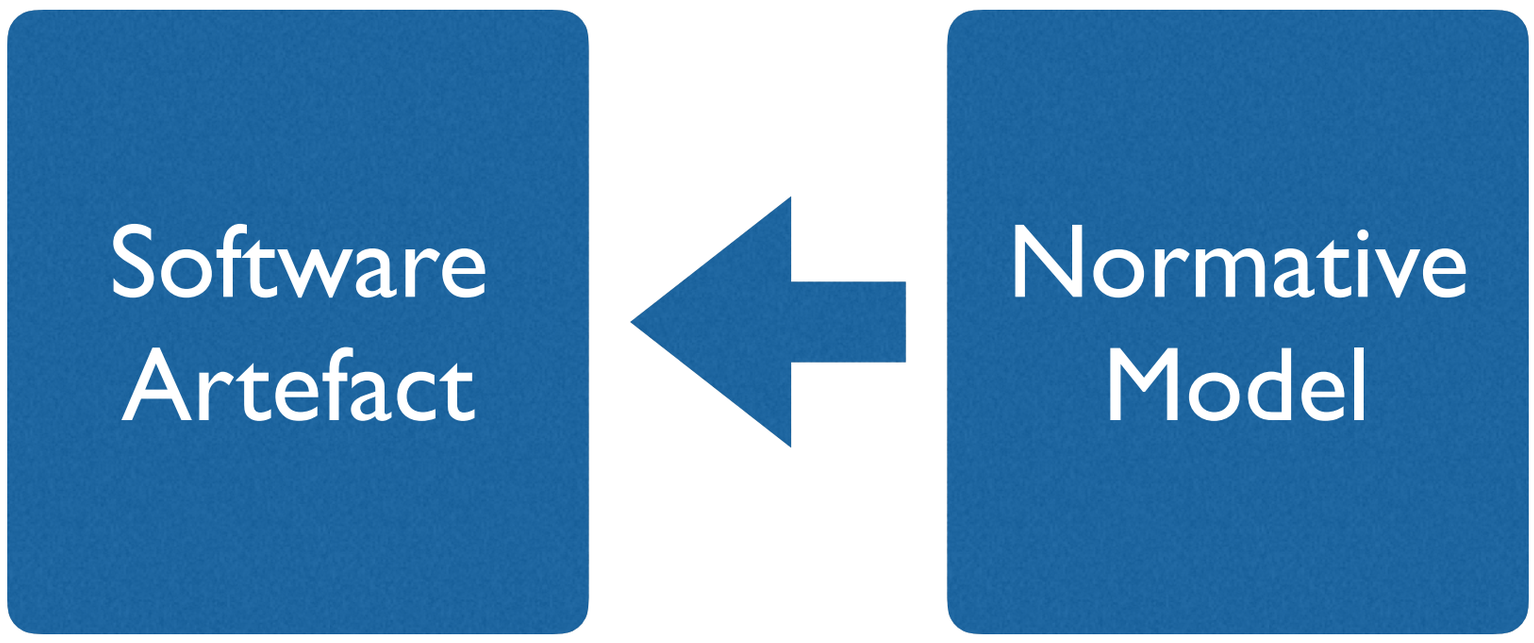
- Energy consumption
- Performance
- Tests for diagnosis and repair

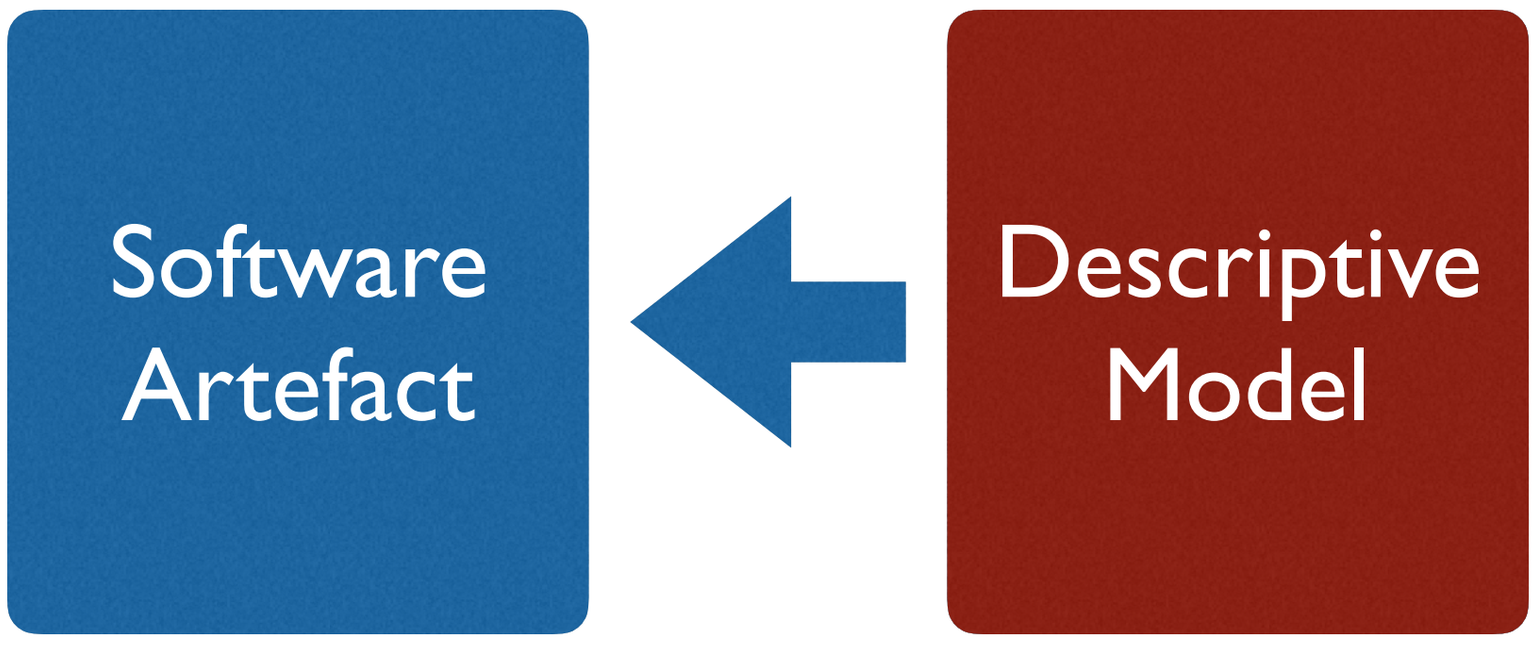

Software Artefact
Quality Attribute
Measurement


Software Artefact
Normative Model
Measurement

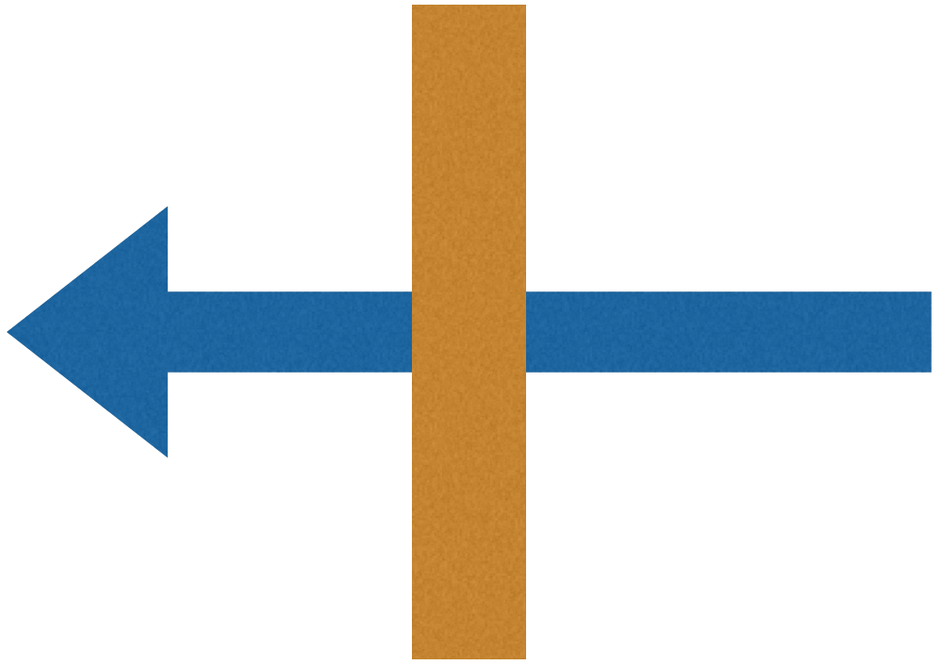

Software Artefact
Normative Model
Measurement Objective


Measurement Objective
Normative Model
Software Artefact


Software Artefact
Normative Model


Software Artefact
Descriptive Model

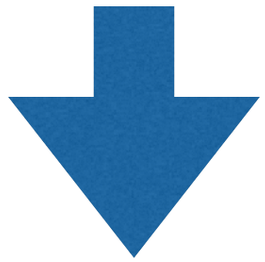
Software Artefact



Normative Model

Method Names

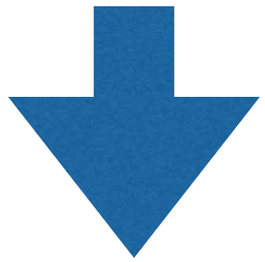
```
@Test(timeout = 4000)
public void test3()  throws Throwable  {
    StringExample stringExample0 = new StringExample();
    boolean boolean0 = stringExample0.foo("");
    assertFalse(boolean0);
}
```



```
@Test(timeout = 4000)
public void testFooReturningFalse()  throws Throwable  {
    StringExample stringExample0 = new StringExample();
    boolean boolean0 = stringExample0.foo("");
    assertFalse(boolean0);
}
```

Variable Names

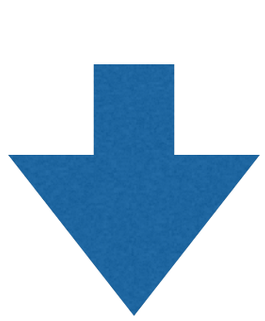
```
@Test(timeout = 4000)
public void testFooReturningFalse()  throws Throwable  {
    StringExample stringExample0 = new StringExample();
    boolean boolean0 = stringExample0.foo("");
    assertFalse(boolean0);
}
```



```
@Test(timeout = 4000)
public void testFooReturningFalse()  throws Throwable  {
    StringExample invokesFoo = new StringExample();
    boolean resultFromFoo = invokesFoo.foo("");
    assertFalse(resultFromFoo);
}
```

Variable Names

```
public class Foo {
   public void foo() {
      StringExample sx = new StringExample();
      boolean bar = sx.foo("");
   }
}
```



```
@Test(timeout = 4000)
public void testFooReturningFalse()  throws Throwable  {
    StringExample sx = new StringExample();
    boolean bar = sx.foo("");
    assertFalse(bar);
}
```

Software Artefact

Further attributes

Normative Model

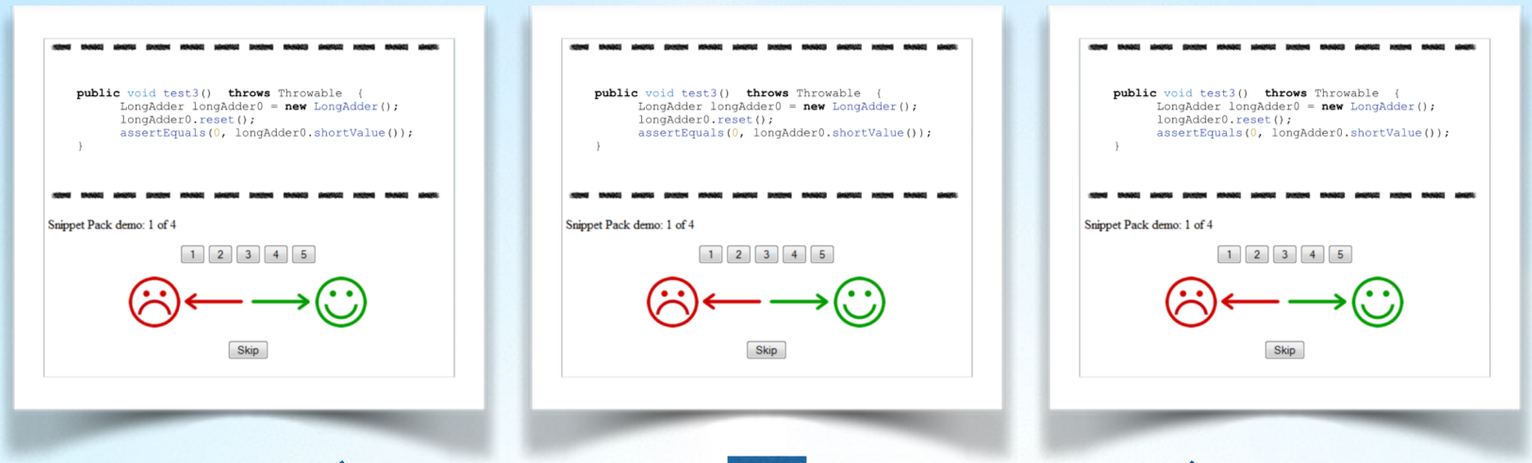
EVOSUITE

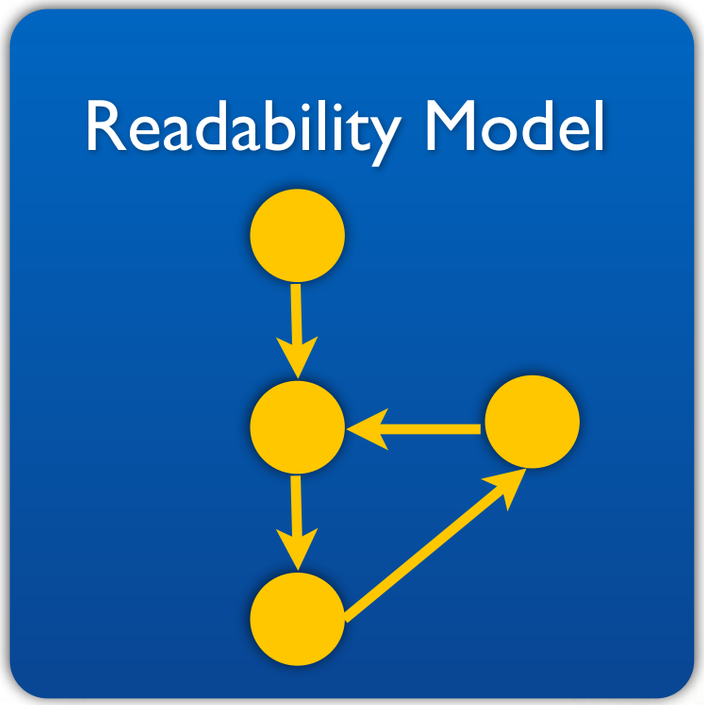
```
public void test3()  throws Throwable  {
        LongAdder longAdder0 = new LongAdder();
        longAdder0.reset();
        assertEquals(0, longAdder0.shortValue());
}
```

Snippet Pack demo: 1 of 4

1 2 3 4 5

Skip


public void test3() throws Throwable {
LongAdder longAdder0 = new LongAdder();
longAdder0.reset();
assertEquals(0, longAdder0.shortValue());
}
Snippet Pack demo: 1 of 4
1 2 3 4 5
Skip


Readability Model

EVOSUITE

Java Unit Testing Study

Gordon

readability.evosuite.org/pair/study/question/0

0%

Test Case A

```
package org.apache.commons.cli;

import static org.junit.Assert.*;
import org.junit.Test;
import org.apache.commons.cli.CommandLine;
import org.apache.commons.cli.Option;

public class CommandLine_ESTest {

  @Test
  public void test0()  throws Throwable  {
      CommandLine commandLine0 = new CommandLine();
      boolean boolean0 = commandLine0.hasOption("!VW
      String string0 = commandLine0.getOptionValue('
      Option option0 = new Option((String) null, "!V
      commandLine0.addOption(option0);
      boolean boolean1 = commandLine0.hasOption("!VW
      assertFalse(boolean1 == boolean0);
      assertTrue(boolean1);
  }
}
```

Test Case B

```
package org.apache.commons.cli;

import static org.junit.Assert.*;
import org.junit.Test;
import org.apache.commons.cli.CommandLine;
import org.apache.commons.cli.Option;

public class CommandLine_ESTest {

  @Test
  public void test0()  throws Throwable  {
      CommandLine commandLine0 = new CommandLine();
      Option option0 = new Option("", false, "");
      commandLine0.addOption(option0);
      boolean boolean0 = commandLine0.hasOption('-')
      assertTrue(boolean0);
  }
}
```

Test A

Test B

Next »

Contents

Contents

Submit your SBST papers here:

http://aster.or.jp/conference/icst2017

http://goo.gl/kyO2Jq