Quiz 2

When: Monday January 21, 9:15-10:15
Where: Stong College, room 214
Type: written test
Material: all the material covered before January 18, that is, material covered in the lectures and in the labs, as well as the posted reading material
Note: Students may bring one letter-sized piece of paper with notes (both sides of the piece of paper may be used)

Check models

EECS 4315

wiki.eecs.yorku.ca/course/4315/

The state space explosion problem

Problem

The size of the state space, that is, the number of states, may become very large.

The state space explosion problem

Problem

The size of the state space, that is, the number of states, may become very large.

This is one of the major challenges in model checking.

Model checking

Develop a model (states connected by transitions) of the code and check properties of the model.

Model checking

Model checking was developed independently by Clarke and Emerson and by Queille and Sifakis in early 1980s.

Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In, Dexter Kozen, editor, *Proceedings of Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52-71. Yorktown Heights, NY, USA, May 1981. Springer-Verlag.

Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In, Mariangiola Dezani-Ciancaglini and Ugo Montanari, editors, *Proceedings of the 5th International Symposium on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 337-351. Torino, Italy, April 1982. Springer-Verlag.

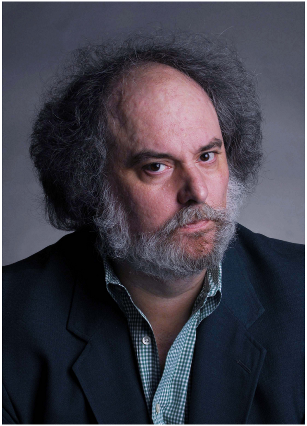
Edmund Clarke

- Recipient of the Turing Award (2007)
- Recipient of the ACM Paris Kanellakis Award (1999)
- Member of the National Academy of Engineering (2005)
- Member of the American Academy of Arts and Sciences (2011)

source: Dennis Hamilton

Allen Emerson

- Recipient of the Turing Award (2007)
- Recipient of the ACM Paris Kanellakis Award (1999)
- Recipient of the CMU Newell Medal (1999)



source: Marsha Miller

Joseph Sifakis

- Recipient of the Turing Award (2007)
- Grand officer of France's national order of merit (2008)
- Commander in France's legion of honour (2011)

source: David Monniaux

Jean-Pierre Queille

source: unknown

Model of a system

A model of a system is an abstraction of the system.

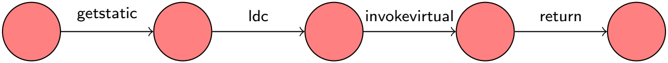
THE PROJECT MANAGEMENT FRAMEWORK EMBODIES A PROJECT LIFE CYCLE AND FIVE MAJOR PROJECT MANAGEMENT PROCESS GROUPS.
Dilbert.com DilbertCartoonist@gmail.com
OH NO! THE EXTREME LEVEL OF ABSTRACTION HAS MADE US WEIGHTLESS!
1-7-12 ©2012 Scott Adams, Inc./Dist. by Universal Uclick
THAT DOESN'T EVEN MAKE SENSE.

Source: Toronto Star

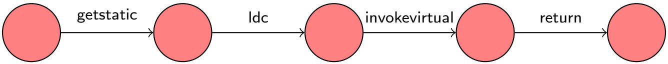
There are many levels of abstraction and, hence, a system can be modelled in many different ways.

Model of a system

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```



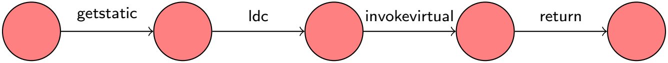
Model of a system



Question

What are the three entities that make up the above model?

Model of a system



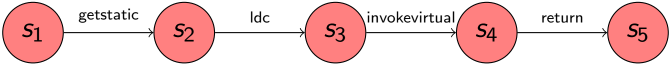
Question

What are the three entities that make up the above model?

Answer

1. ○
2. →
3. getstatic, ldc, . . .

Model of a system

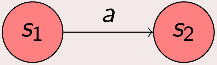


Question

Such a model consists of

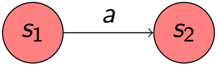
- a set S of *states*,
- a set A of *actions*, and
- a set of *transitions*.

Which three ingredients specify the following transition?



Model of a system

The transition



is specified by

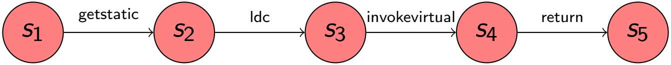
1. the source state s_1,
2. the label a, and
3. the target state s_2.

Hence, the transition can be captured by the triple (s_1, a, s_2). This is an *element* of $S \times A \times S$, that is,

$$(s_1, a, s_2) \in S \times A \times S.$$

Therefore, the set of transitions is a *subset* of $S \times A \times S$.

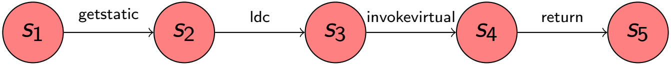
Model of a system



Question

How can we model all the labelled transitions?

Model of a system



Question

How can we model all the labelled transitions?

Answer

{ (s_1, getstatic, s_2), (s_2, lcd, s_3), (s_3, invokevirtual, s_4),
(s_4, return, s_5) }.

Labelled transition system

Systems can be modelled by means of labelled transition systems.

Definition

A labelled transition system is a tuple $\langle S, A, \rightarrow, s \rangle$ consisting of

- a set S of states,
- a set A of actions,
- a set of transitions $\rightarrow \subseteq S \times A \times S$, and
- a start state $s \in S$.

Labelled transition system

Systems can be modelled by means of labelled transition systems.

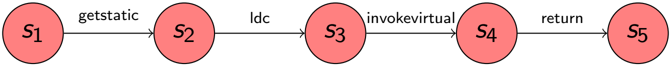
Definition

A labelled transition system is a tuple $\langle S, A, \rightarrow, s \rangle$ consisting of

- a set S of states,
- a set A of actions,
- a set of transitions $\rightarrow \subseteq S \times A \times S$, and
- a start state $s \in S$.

Instead of $(s_1, a, s_2) \in \rightarrow$, we usually write $s_1 \xrightarrow{a} s_2$.

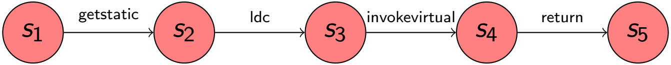
Model of a system



Question

Give the corresponding labelled transition system.

Model of a system



Question

Give the corresponding labelled transition system.

Answer

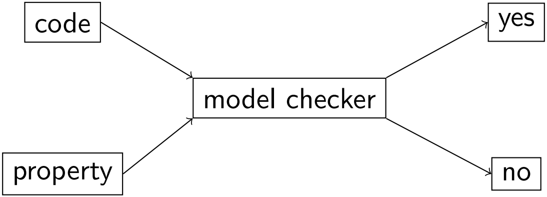
$\langle\{s_1, s_2, s_3, s_4, s_5\},$
$\{\text{getstatic}, \text{lcd}, \text{invokevirtual}, \text{return}\},$
$\{(s_1, \text{getstatic}, s_2), (s_2, \text{lcd}, s_3), (s_3, \text{invokevirtual}, s_4), (s_4, \text{return}, s_5)\},$
$s_1\rangle.$

Introduction to Java PathFinder

EECS 4315

www.eecs.yorku.ca/course/4315/

Model checker



Clarke and Emerson used the term *model checking* because they wanted to determine if the property (expressed as a temporal formula) f was true in the Kripke structure M, that is, whether the structure M was a *model* for the formula f.

Some people believe erroneously that the use of the term *model* refers to the dictionary meaning of this word and indicates that an abstraction of the actual system is considered.

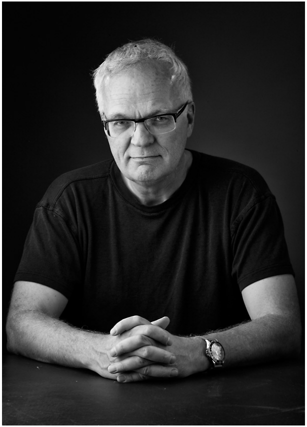
Some history

In 1999, Klaus Havelund introduced Java PathFinder (JPF).

Klaus Havelund. Java PathFinder – A Translator from Java to Promela. In, Dennis Dams, Rob Gerth, Stefan Leue and Mieke Massink, editors, *Proceedings of the 5th and 6th International SPIN Workshops*, volume 1680 of *Lecture Notes in Computer Science*, page 152. Springer-Verlag.

Klaus Havelund

- PhD in Computer Science from the University of Copenhagen.
- Senior Research Scientist at NASA's Jet Propulsion Laboratory.
- ASE 2014 most influential paper award.



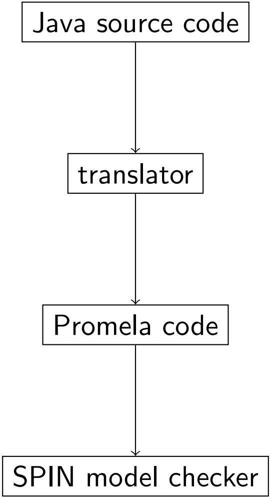
source: Klaus Havelund

Some history

Others who initially worked on JPF:

- Michael Lowry (NASA)
- John Penix (NASA, now Google)
- Thomas Pressburger (NASA)
- Jens Ulrik Skakkebaek (Stanford, now Google)
- Willem Visser (NASA, now Stellenbosch University)

First version of JPF

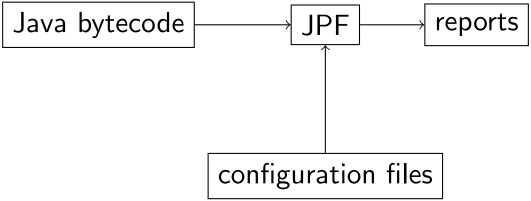


First version of JPF

Major limitations:

- Representing all features of Java in Promela is impossible;
- Mapping bugs found by SPIN in the Promela code back to the Java code is challenging.

Second version of JPF



The second version of JPF is a Java virtual machine (JVM).

Second version of JPF

Willem Visser, Klaus Havelund, Guillaume Brat, Seungjoon Park. Model Checking Programs. In *Proceedings of the 15th IEEE International Conference on Automated Software Engineering*, pages 3–12, Grenoble, France, September 2000. IEEE

The Automated Software Engineering conference series has a rich history of good contributions to the area of research and development. The ASE most influential paper award is an effort to identity the most influential ASE paper 14 years after being published. In 2014, the above paper won this award.

A simple example

```
import java.util.Random;

public class PrintRandom {
  public static void main(String[] args) {
    Random random = new Random();
    final int MAX = 4;
    System.out.println(random.nextInt(MAX + 1));
  }
}
```

A simple example

```
target=PrintRandom
classpath=<path to directory that contains PrintRandom.clas
```

A simple example

```
JavaPathfinder core system v8.0 (rev 2+) - (C) 2005-2014 Un
====================================================== syst
PrintRandom.main()
====================================================== sear
0
====================================================== resu
no errors detected
====================================================== stat
elapsed time:     00:00:00
states:           new=1,visited=0,backtracked=1,end=1
search:           maxDepth=1,constraints=0
choice generators: thread=1 (signal=0,lock=1,sharedRef=0,th
heap:             new=350,released=12,maxLive=0,gcCycles=1
instructions:     3176
max memory:       61MB
loaded code:      classes=57,methods=1232
```

A simple example

Question

To how many different executions may the Java code give rise?

A simple example

Question

To how many different executions may the Java code give rise?

Answer

5.

A simple example

Question

To how many different executions may the Java code give rise?

Answer

5.

Question

How many different executions does JPF check?

A simple example

Question

To how many different executions may the Java code give rise?

Answer

5.

Question

How many different executions does JPF check?

Answer

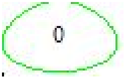
1.

A simple example

Let's have a look at the state space diagram.

```
target=PrintRandom
classpath=<path to directory that contains PrintRandom.clas
listener=gov.nasa.jpf.listener.StateSpaceDot
```

A simple example



A simple example

Configure JPF so that it explores all random choices.

```
target=PrintRandom
classpath=<path to directory that contains PrintRandom.clas
cg.enumerate_random=true
```

A simple example

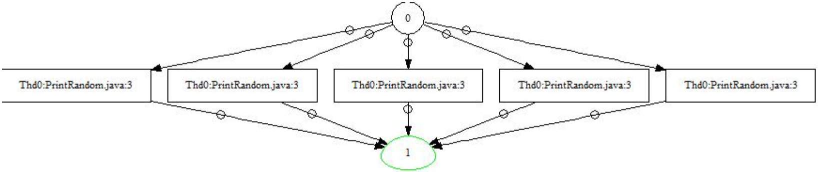
```
JavaPathfinder core system v8.0 (rev 2+) - (C) 2005-2014 Un
====================================================== syst
PrintRandom.main()
====================================================== sear
0
1
2
3
4
====================================================== resu
no errors detected
====================================================== stat
elapsed time:     00:00:00
states:           new=2,visited=9,backtracked=11,end=10
search:           maxDepth=2,constraints=0
choice generators: thread=1 (signal=0,lock=1,sharedRef=0,th
heap:             new=350,released=102,maxLive=348,gcCyc
```

A simple example

Let's have a look at the state space diagram.

```
target=PrintRandom
classpath=<path to directory that contains PrintRandom.clas
cg.enumerate_random=true
listener=gov.nasa.jpf.listener.StateSpaceDot
```

A simple example



The ColorTest revisited

In Lab 1, we wrote a JUnit test case to test the `Color` class.

- JPF can only be run on apps, that is, classes that contain a main method.
- By default JPF checks for uncaught exceptions.

The ColorTest revisited

In Lab 1, we wrote a JUnit test case to test the `Color` class.

- JPF can only be run on apps, that is, classes that contain a main method.
- By default JPF checks for uncaught exceptions.

Question

Write an app that invokes the JUnit test case methods.

The ColorTest revisited

```
package lab;

public class RunTest {
  public static void main(String[] args) {
    ColorTest tester = new ColorTest();
    tester.testConstructorAndAccessors();
    tester.testBLACK();
    tester.testEquals();
    tester.testEqualsRandomly();
  }
}
```

The ColorTest revisited

```
target=lab.RunTest
classpath=<path to directory that contains lab>; \
  <paths to several jar files>
cg.enumerate_random=true
```

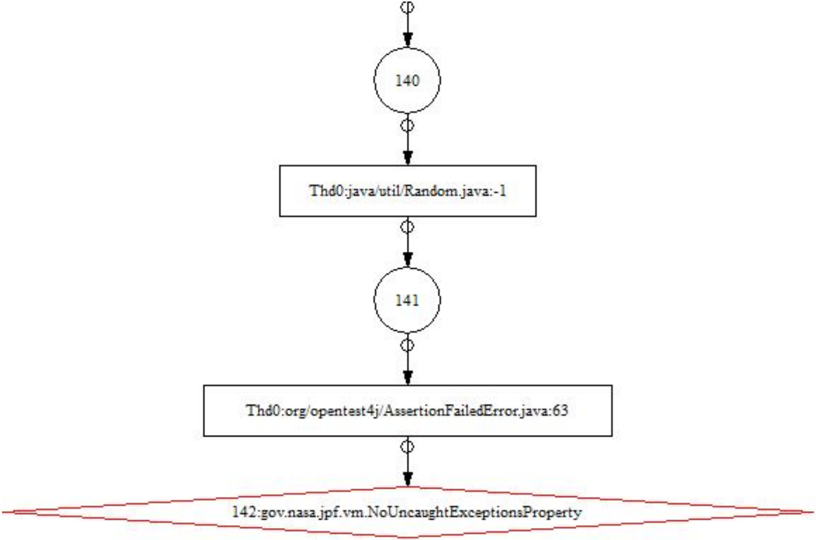
- target contains both the class name and the package name.
- Some of the JUnit jars needs to be added to the classpath.

The ColorTest revisited

```
====================================================== erro
gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
org.opentest4j.AssertionFailedError: expected: <13> but was
at org.junit.jupiter.api.AssertionUtils.fail(org/junit/jupi
at org.junit.jupiter.api.AssertEquals.failNotEqual(org/juni
at org.junit.jupiter.api.AssertEquals.assertEquals(org/juni
at org.junit.jupiter.api.AssertEquals.assertEquals(org/juni
at org.junit.jupiter.api.Assertions.assertEquals(org/junit/
at lab.ColorTest.testConstructorAndAccessors(lab/ColorTest.
at lab.RunTest.main(lab/RunTest.java:11)


====================================================== snap
thread java.lang.Thread:{id:0,name:main,status:RUNNING,prio
call stack:
at org.junit.jupiter.api.AssertionUtils.fail(AssertionUtils
at org.junit.jupiter.api.AssertEquals.failNotEqual(Asser
```

The ColorTest revisited


140
Thd0:java/util/Random.java:-1
141
Thd0:org/opentest4j/AssertionFailedError.java:63
142:gov.nasa.jpf.vm.NoUncaughtExceptionsProperty

The ColorTest revisited

By default, JPF stops after detecting a bug.

The ColorTest revisited

By default, JPF stops after detecting a bug.

To find multiple bugs . . .

```
target=lab.RunTest
classpath=<path to directory that contains lab>; \
  <paths to several jar files>
cg.enumerate_random=true
search.multiple_errors=true
```

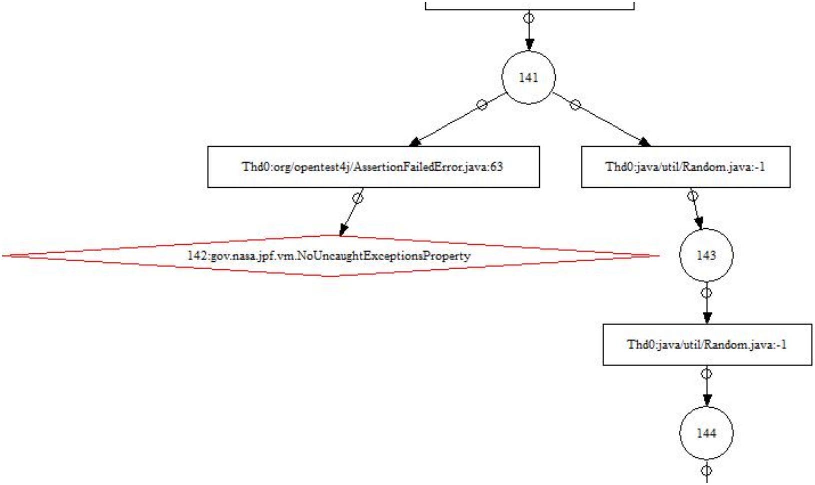
The ColorTest revisited

```
====================================================== erro
gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
org.opentest4j.AssertionFailedError: expected: <13> but was
at org.junit.jupiter.api.AssertionUtils.fail(org/junit/jupi
...
at lab.RunTest.main(lab/RunTest.java:11)

====================================================== snap
thread java.lang.Thread:{id:0,name:main,status:RUNNING,prio
call stack:
...
at lab.RunTest.main(RunTest.java:11)

====================================================== erro
gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
org.opentest4j.AssertionFailedError: expected: <13> but was
...
```

The ColorTest revisited


141
Thd0:org/opentest4j/AssertionFailedError.java:63
Thd0:java/util/Random.java:-1
142:gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
143
Thd0:java/util/Random.java:-1
144