Java™ PathFinder

Neha Rungta
NASA Ames Research Center
Software Crisis

• Software crisis declared in 1968
• Programs around 100K lines of code
• What has changed?
  – Programs bigger (5M-40M)
  – Processors faster and memory larger
  – Programs in more places (Ubiquitous?)
• Software engineering relatively the same
If 1968 was a crisis then, what is today?
Software Engineering

- Little engineering in software engineering
- Very little modeling and analysis
- Reuse and copy is common
- Trial and error testing
- Struggle to produce reliable software
Data: Standish Group, 1995 survey of 365 companies and 8,380 applications. NIST Report 02-3: The economic impacts of inadequate infrastructure for software testing. (May 2002).
Software Engineering

Most decisions made:
- Detail design and code structure
- High level design and system architecture
- Requirement analysis and system engineering

Bad decisions found:
- Coding
- Testing
- Evolution

Cost increase with late error discovery

Cost decrease with early error discovery
Software Model Checking

Most decisions made
- Detail design and code structure
- High level design and system architecture
- Requirement analysis and system engineering

Bad decisions found
- Evolution
- Testing
- Coding

Reduced need for testing
Software Model Checking

Detailed modeling and analysis

Most decisions made

Detail design and code structure

High level design and system architecture

Requirement analysis and system engineering

Formalize requirements in mathematically precise language

Build logical models of all designs and analyze with requirements

Do not move up until designs provably implement requirements and meet specifications

valid == 0 & ack == 0

data = DATA

valid = 1

ack = 1
So;ware Model Checking

Evolution

Testing

Coding

Bad decisions found
What can software model checking find?

- Errors in deep execution traces
- Deadlock, live-lock, and starvation
- Race conditions
- Priority inversion and locking problems
- Resource allocation errors
- Bounds checking
- Incompleteness and redundancy
- Logic problems
- What ever you ask!
- BTW, don’t ask don’t tell policy
History

• not a new project: around for 10 years and continuously developed:
  – 1999 - project started as front end for Spin model checker
  – 2000 - reimplementaQon as concrete virtual machine for software model checking (concurrency defects)
  – 2003 - introduction of extension interfaces
  – 2005 - open sourced on Sourceforge
  – 2008 - participation in Google Summer of Code
  – 2009 - moved to own server, hosting extension projects and Wiki
JPF’s Home

http://babelfish.arc.nasa.gov/trac/jpf

JPF’s User Forum

http://groups.google.com/group/java-pathfinder
Overall Architecture
Exploring Choices

• model checker needs choices to explore state space
• there are many potential types of choices (scheduling, data, ..)
• choice types should not be hardwired in model checker
Transition -> State

Choice

Scheduling Choice

```java
synchronized (...) {
    wait (...)
    x = mySharedObject
    ..
}
```

Data Choice

```java
boolean b = Verify.getBoolean();
double d = Verify.getDouble("MyHeuristic");
..
```

Control Choice

```java
if (<cond>) ..
INVOKEECG.setInvocations(..)
..
```
Choice Generators

- transitions begin with a choice and extend until the next ChoiceGenerator (CG) is set (by instruction, native peer or listener)
- **advance** positions the CG on the next unprocessed choice (if any)
- **backtrack** goes up to the next CG with unprocessed choices
Choice Generators

State

Transition

\{Instruction\}

ChoiceGenerator

\begin{align*}
T_j & \quad \ldots \\
& \quad \text{get\_field} \\
& \quad \text{setNextCG} \\
CG_k & \quad = \{c^1_k, c^2_k, c^3_k\} \\
T_{j+1} & \quad \ldots \\
& \quad \text{get\_field} \\
& \quad \text{getNextChoice} \\
CG_l & \quad = \{c^1_l, \ldots\} \\
T_{j+2} & \quad \ldots \\
& \quad \text{monitor\_enter} \\
& \quad \text{backtrack} \\
& \quad \text{advance}
\end{align*}
Search Strategies

• state explosion mitigation: search the interesting state space part first ("get to the bug early, before running out of memory")
• Search instances encapsulate (configurable) search policies
Search Strategies

```
while (notDone) {
    ..vm.forward();
    ..vm.backtrack();
    if (!properties.check()){
        reportError(); break;
    }
}
```

depth first traversal optional state matching

```
Search
JVM vm
search ()
```

```
DFS
search ()
```

```
HeuristicSearch
search ()
```

```
SimplePriorityHeuristic
```

```
BFSHeuristic
search ()
```

sorted state queue (bounded)
Bytecode Factory

code[i] = factory.create(..IFEQ);

abstract execution semantics

concrete execution semantics

concrete value execution

symbolic value execution

Instruction execute (...){
    cond = popCondition();
    if (cond)
        return jumpTarget;
    else
        return getNextInsn();
}

Instruction execute (...){
    if (!firstStepInsn()){
        setNextCG(new PCChoiceGenerator());
        return this;
    }
    popCondition(); // not interested
    cond = getCG().getNextChoice();
    if (cond){
        updatePathCondition(., EQ);
        return jumpTarget;
    } else {
        updatePathCondition(., NE);
        return getNextInsn();
    }
}
```java
void notSoObvious(int x){
    int a = x * 50;
    int b = 19437583;
    int c = a;
    for (int k = 0; k < 100; k++){
        c += b;
        System.out.println(c);
    }
}
```

```java
notSoObvious(21474836);
```

```
... iinc
[20] goto 10
[10] iload_4
[12] if_icmpge 22
[13] iload_3
[14] iload_2
[15] iadd
...}
void notSoObvious(int x){
    int a = x * 50;
    int b = 19437583;
    int c = a;
    for (int k = 0; k < 100; k++){
        c += b;
        System.out.println(c);
    }
}
...
notSoObvious(21474836);
```

class IADD extends Instruction {
    Instruction execute (.., ThreadInfo ti) {
        int v1 = ti.pop();
        int v2 = ti.pop();
        int res = v1 + v2;
        if ((v1>0 && v2>0 && res<=0) ... throw ArithmeticException...)
    }
}
```

```java
class IADD extends Instruction {
    Instruction execute (.., ThreadInfo ti) {
        int v1 = ti.pop();
        int v2 = ti.pop();
        int res = v1 + v2;
        if ((v1>0 && v2>0 && res<=0) ... throw ArithmeticException...)
    }
}
```

```java
IADD

```

---

**Example**

**JPF configuration**

```
vm_insn_factory.class =
 .numeric.NumericInstructionFactory.
```

**class loading**

**code execution (by JPF)**
Attributes
Partial Order Reduction

executed bytecode instruction

scheduling relevant insn type

field insn

- GETFIELD
- PUTFIELD
- GETSTATIC
- PUTSTATIC
- xALOAD
- xASTORE

sync insn

- MONITORENTER
- MONITOREXIT

sync mth

- INVOKEVIRTUAL
- INVOKESTATIC

threading call

- Thread. start(), yield()
- sleep(), join()
- Object.wait(), notify()

other runnable threads

recursive locks

shared objects

tracking of access threads

lock protected access

lock distance & statistics

scheduling relevant instruction (registeres a ThreadChoiceGenerator)
refTid always initialized by creator

refTid_x

T_1

x = new X

T_2 start

S_i

get x.f

S_j

terminate T_1

{1}

S_k

terminate T_2

{1,2}

refTid_x

{1}

start() must break transition

scheduling relevant

{1,2}

{1}

{2}

{1}
State Serialization

kernelStateChanged() {
    cache = null
}

processElementInfo() {
    fmask = getFilterMask()
    int[] values = getFieldValues()
    for (i<values.length; i++){
        if (isFiltered(fmask,i)){
            if (isRef(i))
                processRef(values[i])
        else
            intBuffer.add(values[i])
        }
    }
}

processRef(int r) {
    ElementInfo ei=heap.get(r);
    if (ei.getSid()==0)
        ei.setSid(sidCount++)
    intBuffer.add( ei.getSid() )
}

implements Heap Symmetry
(storing canonical order of reference not reference value itself)

JVM(Conf)
   serializer = conf.get("vm.serializer.class")
   stateSet = conf.get("vm.storage.class")

forward() {
    stateSet.addCurrent(); ..
}

addCurrent() {
    add(serializer.getStoringData());
}

getStoringData() {
    if (cache==null){
        cache = computeStoringData()
        ks.pushChangeListener(this)
    }
    return cache()
}

computeStoringData() {
    intBuffer.clear(); refQueue.clear()
    serializeThreads()
    serializeClass()
    serializeStatics()
    processRefQueue()
    return intBuffer.toArray()
}
Heap Symmetry

\[ S_i = f(T_1, T_2, \ldots, T_n) \]
\[ S_j = f(T_1, T_2, \ldots, T_m) \]
package x.y.z;
class MyClass {
    ...
    native String foo (int i, String s);
}

"Model" Class

JPF Class

- method lookup
- parameter conversion
- invocation

MJJ - "Model Java Interface"

NativePeer
MJIEEnv

JPF objects
Java objects

- field access
- object conversion
- JPF intrinsics access

class JPF_x_y_z_MyClass {
    public static
    int foo__ILjava_lang_String__2 (MJIEEnv env, int objRef, int i, int sRef) {
        String s = env.getStringObject(sRef);
        ...
        int ref = env.newString(., .);
        return ref;
    }
}

Java Class

"NativePeer" Class
package x.y.z;

class C {
    native int foo(int p);
}

class JPF_x_y_z_C {
    ...
    public static int foo__I(MJIEnv env, int thisRef, int p) {
        int d = env.getIntField(thisRef, "data");
        ...
    }
    ...
}
Listeners

- classLoaded()
- threadScheduled()
- threadNotified()
  ...
- executeInstruction()
- instructionExecuted()
- objectCreated()
  ...
- exceptionThrown()
  ...
- choiceGeneratorAdvanced()
  ...

- +listener=<listener-class>
- @JPFConfig(..
- listener.autoload=<annotations>
- jpf.addListener(..
  ...

- propertyViolated()
- searchConstraintHit
- searchFinished()
Design Hierarchy

- **interface** JPFLListener
  - **interface** VMLListener
    - executeInstruction(vm)
    - instructionExecuted...
    - threadStarted...
    - objectCreated...
    - choiceGeneratorSet...
    - choiceGeneratorAdvanced...
  - **interface** SearchListener
    - searchStarted(search)
    - stateAdvanced...
    - stateBacktracked...
    - propertyViolated...
    - searchFinished...
    - ...
  - **interface** PublisherExtension
    - publishStart(publisher)
    - publishFinished...
    - ...
  - **interface** Property
    - check(search, vm)
    - getErrorMessage()
    - ...
  - **interface** GenericProperty
    - check(search, vm) {}
    - ...
  - ListenerAdapter
    - instructionExecuted(vm) {}
    - searchStarted(search) {}
    - ...
  - PropertyListenerAdapter
    - instructionExecuted(vm) {}
    - searchStarted(search) {}
    - ...
public class NonnullChecker extends ListenerAdapter {

... 

public void executeInstruction (JVM vm) {
    Instruction insn = vm.getLastInstruction();
    ThreadInfo ti = vm.getLastThreadInfo();

    if (insn instanceof ARETURN) { // check @NonNull method returns
        ARETURN areturn = (ARETURN)insn;
        MethodInfo mi = insn.getMethodInfo();
        if (areturn.getReturnValue(ti) == null) {
            if (mi.getAnnotation("java.annotation.NonNull") != null) {
                Instruction nextPc = ti.createAndThrowException(
                    "java.lang.AssertionError",
                    "null return from @NonNull method: " +
                    mi.getCompleteName());

                ti.setNextPC(nextPc);
                return;
            }
        }
    }

..
JPF and JUnit

- derive your test cases from `gov.nasa.jpf.util.test.TestJPF`
- run normally under JUnit or from Ant `<junit ..>` task
- be aware of that test case is run by JVM and JPF

```java
public class ConstTest extends TestJPF {
    static final String[] JPF_ARGS = {"+listener=.aprop.listener.ConstChecker"};

    //--- standard driver to execute single test methods
    public static void main(String[] args) {
        runTestsOfThisClass(args);
    }

    //--- the test methods
    @Test
    public void testStaticConstOk () {
        if (verifyNoPropertyViolation(JPF_ARGS)){
            ConstTest.checkThis();
        }
    }
    ...
}
```
Obtaining JPF

- Mercurial repositories on [http://babelfish.arc.nasa.gov/hg/jpf/](http://babelfish.arc.nasa.gov/hg/jpf/){jpf-core,jpf-aprop,...}

- Eclipse Steps
  1. Get Mercurial
     2. Get jpf-core
        1. **File — Import — Mercurial - Clone repository using Mercurial - Next**
        2. Specify [http://babelfish.arc.nasa.gov/hg/jpf/jpf-core](http://babelfish.arc.nasa.gov/hg/jpf/jpf-core)
        3. Check the box for 'Search for .project files in clone and use them to create projects'
        4. Finish
  2. Build
     1. **Project — Properties - Select Builders - Ant Builder - Click Edit**
     2. **Click JRE tab - Separate JREs - Installed JREs**
     3. **Pick a JDK 1.6xxx...JRE will not find javac**
Running JPF (1)

- Create `site.properties` in `$(user.home)/.jpf`
  - One line is enough for now:
  - `$(user.home)/My Documents/workspace/jpf-core`
- Install Eclipse Plugin (from the website description)
  - Ensure that you are running Eclipse >= 3.5 (Galileo)
  - In Eclipse go to Help -> Install New Software
  - In the new window selected "Add"
  - The name is up to you but, set "Location" to 
  - From the "Work with:" drop down menu select the update site 
    that you just entered from the previous step
  - Check the "Eclipse-JPF" check box, select "Next" and go through the install process.
Running JPF (2)

- Right click on *.jpf file and pick “Verify”
  - Go to src/examples and right click on oldclassic.jpf
  - Should see a deadlock!
Configuring JPF

• almost nothing in JPF is hardwired ⇒ great flexibility but config can be intimidating

• all of JPFs configuration is done through Java properties (but with some extended property file format)
  – keyword expansion
    jpf-root = ${user.home}/jpf
    • previously defined properties
    • system properties
  – append extensions+=,jpf-aprop no space between key and ‘+’!
  – prepend +peer_packages=jpf-symbc/build/peers,
  – directives
    • dependencies @requires jpf-awt
    • recursive loading @include ../jpf-symbc/jpf.properties

• hierarchical process
  – system defaults (from jpf.jar)
  – site.properties
  – project properties from all site configured projects (<project-dir>/jpf.properties)
  – current project properties ./jpf.properties
  – selected application properties file (*.jpf)
  – command line args (e.g. bin/jpf +listener=.listeners.ExecTracker ...)
Automated Test Case generation

• Symbolic Execution

```c
int m(int y){
  1: if (y>0)
  2:   y++;
  3: else
  4:   y--; 
  5: return y;
}
```

\[
m_{\text{sum}} = \{(\text{Y} > 0), \text{RETURN} = \text{Y} + 1),
\text{!(Y} > 0), \text{RETURN} = \text{Y} - 1)\}\]
Agile Development

```java
4    int old;
5    int[] data;
6
7    public int logicalValue(int t){
8        if (!(currentTime - t >= 100)){
9            return old;
10       }
11    }else{
12        int val = 0;
13        for (int i=0; i<data.length; i++){
14            val = val + data[i];
15        }
16        old = val;
17        return val;
18    }
```

```java
4    int old;
5    int[] data;
6
7    public int logicalValue(int t){
8        int elapsed = currentTime - t;
9        int val = 0;
10       if (elapsed < THRESHOLD){
11            val = old;
12       }
13       else{
14            for (int i=0; i<data.length; i++){
15                val = val + data[i];
16            }
17            old = val;
18       }
19        return val;
20    }
```
Evolution

• Regression analysis technique focused on version differences
• Combines syntactic and semantic analysis techniques
• Identify and characterize effects of program changes

Version Differences ➔ Directed Symbolic Execution
Background

• Abstract Syntax Tree

```
if (a > b)
    a = a + b;
```

□ Control Flow Graph

```
a > b
   true            false
   a = a + b
   ...
```

```
if
   >
      =
          a
          b
          a
          +
          ...
```
Incremental Execution
Incremental Analysis

```
package precise;

public class Example01 {
    public void test(int a, int b, int c, int d, int e) {
        // assignment of b is different
        // based on the branch taken by
        // after the evaluation of
        // (a == 0)
        if(a == 0) {
            b = (c+d);
        } else {
            b = e;
        }
        // modified statement
        if(b >= 10) {
            b = b+1;
        }
    }

    public static void main(String[] args) {
        Example01 ex = new Example01();
        ex.test(0, 0, 8, 0, 0);
    }
}
```
package precise;

public class Example05_mod{
    public void test (int a, int b, int c, int d, int x) {
        // modified statement
        b = b - x;
        int e = (a + b);
        int f = (c - x);
        // conditional branch statement
        // affected by the change
        if((e + f) == (c+d)) {
            e = f;
        }
        // no path conditions should be generated
        // during this set of conditional branch
        // statements
        if (c == d) {
            c = d+1;
        } else if (c < d) {
            c = d+2;
        } else if (c > d) {
            c = d+3;
        }
    }
}
Extensions!