JPF Lecture 3
Fun With Extensions

Neha Rungta
SGT / NASA Ames Research Center
<neha.s.rungta@nasa.gov>

Peter C. Mehlitz
SGT / NASA Ames Research Center
<peter.c.mehlitz@nasa.gov>
Roadmap

- concrete value execution examples:
  1. **DeadlockAnalyzer** - problem specific traces (*Listener*)
  2. **Catastrophic Cancellation** - deep inspection (*InstructionFactory*)
  3. **GUI Model Checking** - domain specific ChoiceGenerators (*NativePeer*)
  4. **Net-IO-Cache** - distributed (native) applications (*NativePeer, Listener*)
Example 1 - DeadlockAnalyzer

- scalability is not just an input issue - length of counter examples
- use listener to create a property specific trace

DeadlockAnalyzer Listener

deadlock encountered:
thread index=1,name=Thread-0,status=WAITING,...
thread index=2,name=Thread-1,status=WAITING,...
...

------------------------ transition #0 thread: 0
oldclassic.java:47   : Event new_event1 = new Event();
oldclassic.java:59   : class Event {
                  [1 insn w/o sources]
oldclassic.java:60   : int count = 0;
oldclassic.java:47   : Event new_event1 = new Event();
oldclassic.java:48   : Event new_event2 = new Event();
...

------------------------ transition #32 thread: 2
oldclassic.java:127  : event1.signal_event();
oldclassic.java:71   : count = (count + 1) % 3;
oldclassic.java:74   : notifyAll();
oldclassic.java:75   : }
oldclassic.java:129  : if (count == event2.count)

------------------------ transition #33 thread: 1
oldclassic.java:103  : event1.wait_for_event();
oldclassic.java:79   : wait();

jpf-core/src/examples/oldclassic-da.jpf

Thursday, May 26, 2011
Example 1 - DeadlockAnalyzer

- use VMLListener notifications to record synchronization events
- use SearchListener notifications to update trace

```java
public class DeadlockAnalyzer extends ListenerAdapter {
    //--- VMLListener interface
    @Override public void objectWait (JVM vm) {
        addOp(vm,WAIT);
    }
    ...
    //--- SearchListener interface
    @Override public void stateAdvanced (Search search){
        if (search.isNewState())
            storeLastTransition();
    }
    @Override public void stateBacktracked (Search search){
        int stateId = search.getStateId();
        while ((lastTransition != null) && (lastTransition.stateId > stateId))
            lastTransition = lastTransition.prevTransition;
        ...
    }
    ...
    //--- PublisherExtension interface
    @Override public void publishPropertyViolation (Publisher publisher) {
        ...
    }
}
```
deadlock with 18 threads, 70MB trace file reduced to 20 lines

--- error #1
deadlock encountered:
  thread index=0, name=main, status=TERMINATED
  thread index=1, name=Thread-0, status=WAITING
  thread index=2, name=Thread-1, status=TERMINATED
  thread index=3, name=Thread-2, status=WAITING

--- statistics
elapsed time:       3:53:23
states:             new=123010, visited=235893, backtracked=357879, end=81378
search:             maxDepth=1023, constraints=0
choice generators:  thread=94664, data=0
heap:               gc=599818, new=7687542, free=12860670
instructions:       276556150
max memory:         9838MB

too large to analyze manually

Thursday, May 26, 2011
Example 2 - Catastrophic Cancellation

✦ errors and required inspection can be a lot more intricate
✦ e.g.: amplification of previous operand errors due to cancellation of identical leading bits in significands

Annie Cuyt et al. “A remarkable example of catastrophic cancellation unraveled”

double a = 77617.0;
double b = 33096.0;
res = 333.75*\text{pow}(b,6) + \text{pow}(a,2)*(11*\text{pow}(a,2)*\text{pow}(b,2) - \text{pow}(b,6) - 121*\text{pow}(b,4) - 2) + 5.5*\text{pow}(b,8) + a/(2*b);

res=-1.1805916207174113e21 (should be -0.827396...)

java.lang.ArithmeticException: cancellation of:
-7.917111340668963E36 + 7.917111340668962E36
= -1.1805916207174113E21

---

.gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty

jpf-numeric/src/examples/CatastrophicCancellation.jpf
Example 2 - Catastrophic Cancellation

- InstructionFactory with specialized FP instructions

```java
package gov.nasa.jpf.numeric.bytecode;

public class NumericInstructionFactory extends InstructionFactory {
    ...
    @Override public Instruction dsub() { return new DSUB(); }
}

public class DSUB extends Instruction {
    @Override public Instruction execute(ThreadInfo ti) {
        long l1 = ti.longPop();
        double d1 = Double.longBitsToDouble(l1);

        long l2 = ti.longPop();
        double d2 = Double.longBitsToDouble(l1);

        double r = d2 - d1;
        ti.longPush(Double.doubleToLongBits(r);

        if (checkCancellation(l1,l2)){
            reportCancellation(l1,l2);
        }

        return getNextInsn(ti);
    }
}
```
Example 3 - GUI Model Checking

✦ GUI application that uses concurrent data acquisition
✦ thread structure not obvious because of large amount of framework / library code (javax.swing)
✦ application logic mostly implemented as callback actions driven by framework (user input)
✦ two overlaid non-determinisms: user input and scheduling sequence
✦ “impossible” to test
Example 3 - GUI Model Checking

- modeled libraries + NativePeers
- user simulated with input script supporting alternatives

```
ANY { $<FORCED|QUEUED>.doClick() }
$Send.doClick()
...
FORCED.doClick()
Send.doClick()
...
QUEUED.doClick()
Send.doClick()
...
```
Example 3 - GUI Model Checking

- JPF finds NullPointerException that depends on scheduling sequences and user input

~1500 insn/transition, error at depth 36 ⇒ trace too long
Example 3 - Mixed Choice Types

✦ error too general

    java.lang.NullPointerException:
    Calling 'processSequence(Ljava/lang/String;Ljava/lang/String;)' on null object
    at RobotManager.sendSequence(RobotManager.java:269)
    at RobotManagerView.sendSequence(RobotManager.java:567)
    at RobotManagerView$3.actionPerformed(RobotManager.java:341)
    at java.awt.event.ActionListener.actionPerformed(ActionListener.java:120)
    
✦ generic trace analysis listener can help

✦ still too time consuming in many cases

+ OverlappingMethodAnalyzer Listener

...
Example 3 - Mixed Choice Types

✦ better approach - get specific properties from SUT annotations
✦ more efficient to check
✦ self explaining - no need for further analysis

```java
// RobotManager instances are not thread-safe, don’t use them concurrently
@NonShared
public class RobotManager {
    ...

    + NonSharedChecker Listener

++++++ error #1
gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty
java.lang.AssertionError: NonShared object: RobotManager@13d
    accessed in live thread cycle: AWT-EventQueue-0,Thread-2,AWT-EventQueue-0,main
    at RobotManager$ListModel.getSize(RobotManager.java:173)
    at javax.swing.JList.setSelectedIndex(JList.java:280)
    at java.awt.EventQueue.processEventQueue(JPF_java_awt_EventQueue)
    at java.awt.EventQueue.run(EventQueue.java:65)

++++++ statistics
elapsed time: 0:00:02
states: new=531, visited=0, backtracked=512, end=0
search: maxDepth=19, constraints hit=0
choice generators: thread=13 (signal=0, lock=8, shared ref=2), data=604
```
Example 4: Distributed Applications

- example: ChatServer
- 1 ChatServer process
- N Client processes
- server creates one worker object/thread per connection

- model checking problem:
  - state explosion for known clients:
  - what to do if clients are native or unknown?
Example 4: Distributed Applications

✦ no stubs required - uses NativePeers to intercept Socket calls
✦ environment executes normally
✦ protocol structure preserved

![Diagram showing process flow]

- **new state**: store I/O data streams "globally"
  - cache I/O data & message size
  - map program state to stream position

- **backtracking**: restore I/O state "locally"
  - reset streams back to old state

- **continue**: replay previous I/O
  - duplicate sends: ignore
  - duplicate reads: previous peer response

Thursday, May 26, 2011
Example 4: Distributed Applications

- jpf-net-iocache finds server defect (deadlock) without analyzing clients

CacheNotifier Listener
+ java.net.Socket NativePeer

... [SEVERE] unprotected field access of ChatServer.workers in thread: Thread-1..

Client quit, 1 client(s) left.
Client quit, 1 client(s) left.

============================= error #1
gov.nasa.jpf.jvm.NotDeadlockedProperty
deadlock encountered:
  thread index=0,name=main,status=WAITING..
  thread index=1,name=Thread-0,status=TERMINATED..
  thread index=2,name=Thread-1,status=TERMINATED..

============================= snapshot #1
thread index=0,name=main,status=WAITING...
  waiting on: gov.nasa.jpf.network.chat.ChatServer@294
call stack:
  at gov.nasa.jpf.network.chat.ChatServer.accept(ChatServer.java:85)
  ...