Formal Methods for Java

Lecture 25: Java Pathfinder

Jochen Hoenicke

Software Engineering
Albert-Ludwigs-University Freiburg

Feb 01, 2012

Bridging the Gap

JML Tools	JPF	ESC/Java2 KeY Jahob
Runtime Checking	Model Checking	Verification

Java Pathfinder (JPF)

JPF *.. the swiss army knife of Java™ verification*

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

- Developed at NASA Ames Research Center
- One tool – many different usage patterns
- Highly extensible core
- Core implements explicit state model checking on top of a Java VM
- Key concepts:
  - Execution choices as transition breakers
  - State matching
  - Backtracking (restoring previous state)
  - Listeners, Properties, and Publishers

History of JPF

1999 Start as front end for the Spin model checker.

2000 Reimplementation as virtual machine

2003 Extension interfaces

2005 Open sourced on Sourceforge

since 2008 Participation in Google Summer of Code

since 2009 Project, extensions, and wiki hosted on NASA servers (still open source)

Obtaining and Building JPF

- Download from `http://babelfish.arc.nasa.gov/trac/jpf`
- Binary builds not recommended since tool still evolves
- Recommendation: use Mercurial repositories

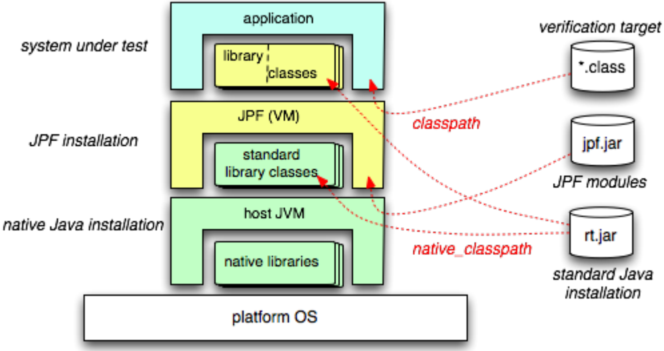
```
> hg clone http://babelfish.arc.nasa.gov/hg/jpf/jpf-core
```

- Repository contains everything needed to build jpf-core

```
> bin/ant
```

- Instructions for Eclipse or NetBeans can be found in the JPF wiki

What We Got



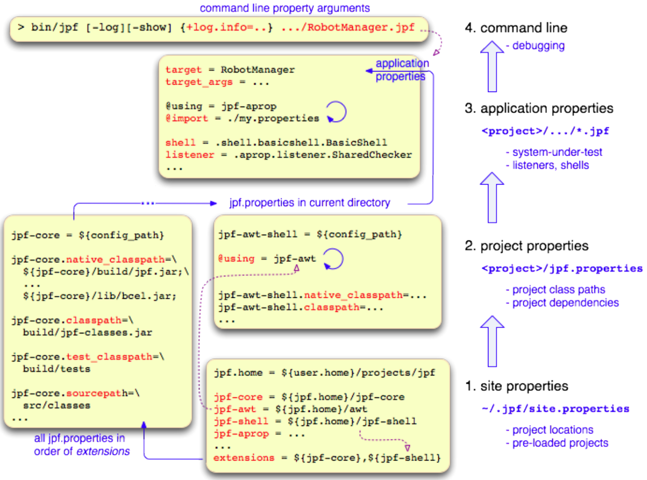
http://babelfish.arc.nasa.gov/trac/jpf/wiki

VM Inside a VM?

- JPF is written in Java $\Longrightarrow$ runs on a JVM
- JPF interprets Java Bytecode $\Longrightarrow$ acts as a JVM
- JPF operates differently:
  - Bytecode of System under Test (SUT) and
  - SUT-specific Configuration produce
  - a report and (possibly) some other artefacts (e.g., test cases)
- JPF might terminate the application if a property is violated

How to Configure JPF

JPF Configuration



command line property arguments

```
> bin/jpf [-log][-show] {+log.info=..} .../RobotManager.jpf
```

application properties

```
target = RobotManager
target_args = ...

@using = jpf-aprop
@import = ./my.properties

shell = .shell.basicshell.BasicShell
listener = .aprop.listener.SharedChecker
...
```

... jpf.properties in current directory

```
jpf-core = ${config_path}

jpf-core.native_classpath=\
  ${jpf-core}/build/jpf.jar;\
  ...
  ${jpf-core}/lib/bcel.jar;

jpf-core.classpath=\
  build/jpf-classes.jar

jpf-core.test_classpath=\
  build/tests

jpf-core.sourcepath=\
  src/classes
...
```

```
jpf-awt-shell = ${config_path}

@using = jpf-awt


jpf-awt-shell.native_classpath=...
jpf-awt-shell.classpath=...
...
```

```
jpf.home = ${user.home}/projects/jpf

jpf-core = ${jpf.home}/jpf-core
jpf-awt = ${jpf.home}/awt
jpf-shell = ${jpf.home}/jpf-shell
jpf-aprop = ...
...
extensions = ${jpf-core},${jpf-shell}
```

all jpf.properties in order of *extensions*

4. command line
   - debugging

3. application properties

   <project>/.../*.jpf

   - system-under-test
   - listeners, shells

2. project properties

   <project>/jpf.properties

   - project class paths
   - project dependencies

1. site properties

   ~/.jpf/site.properties

   - project locations
   - pre-loaded projects

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

JPF Configuration Files

- Basically Java properties files:
  - `key=value` assigns `value` to `key`
  - `# This is a comment`
- Extensions:
  - `${x}` expands to current value of variable `x`
  - `key+=value` appends `value` to the value of `key`
    (No space between `key` and `+=`)
  - `+key=value` prepend `value` to the value of `key`
  - `${config_path}` expands to the directory of the currently parsed file
  - `${config}` expands to the filename of the currently parsed file
  - `@using=<project-name>` loads project `project-name` from location defined in `site.properties` with line `<project-name>=<project-path>`
  - ...
- Shortcut for class names: package prefix *gov.nasa.jpf* can be omitted
- Configuration of JPF can be difficult

Configuring Our Compiled Version

- Switch to your home directory
- Create folder `.jpf`
- Create file `.jpf/site.properties`

  ```
  jpf.home = <Path where you downloaded jpf>

  jpf-core = ${jpf.home}/jpf-core

  extensions = ${jpf-core}
  ```
- This creates the basic configuration
- Add line `jpf-proj = path` to `site.properties` for every additional project you download

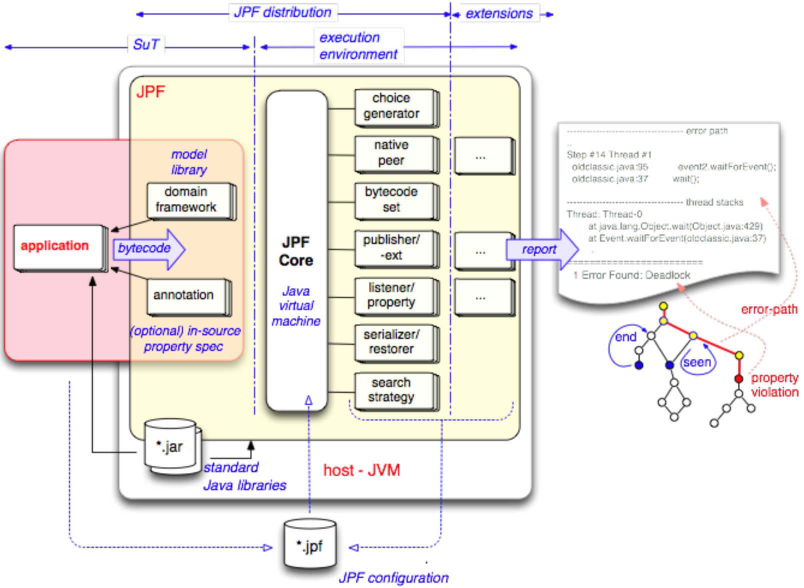
Configuring SuTs

- Create configuration file (typically ends with `.jpf`)
- Content:
  - Some `@using` directives (optionally)
  - One line `target = <SuT>`
  - Optional arguments in a line `target_args = <args>`
  - Additional JPF and related project configuration (optional)
  - Optional `classpath` entry to locate the `.class` file
  - Optional `sourcepath` entry to locate the `.java` file

Demo

Insights into JPF

JPF Components

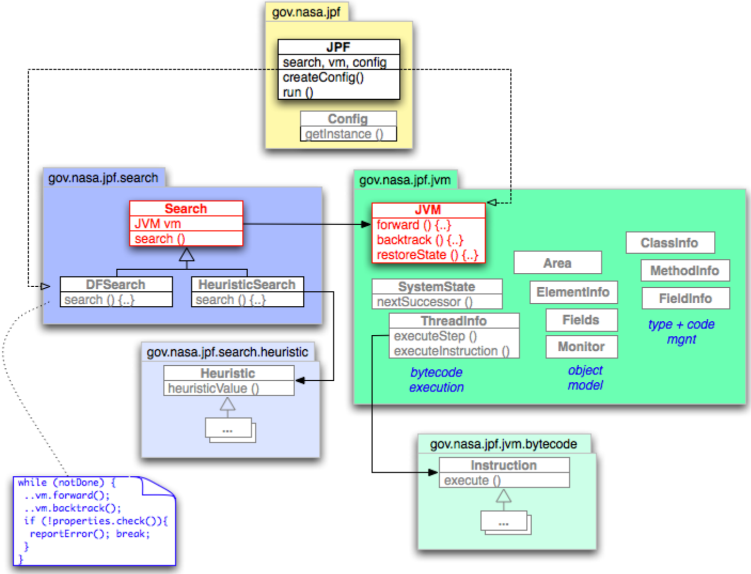


```
------------------------------------ error path
..
Step #14 Thread #1
  oldclassic.java:35          event2.waitForEvent();
  oldclassic.java:37        wait();

------------------------------------ thread stacks
Thread: Thread-0
        at java.lang.Object.wait(Object.java:429)
        at Event.waitForEvent(oldclassic.java:37)
         ..
==========================
  1 Error Found: Deadlock
```

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

JPF Core Architecture



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

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

Explicit State Model Checking and JPF (1/3)

JVM

Unifies states, produces successor states, backtracking.
Configurations:

vm.class	VM implementation
vm.insn_factory	instruction factory
vm.por	apply partial order reduction
vm.por.sync_detection	detect fields protected by locks
vm.gc	run garbage collection
vm.max_alloc_gc	maximal number of allocations before garbage collection
vm.tree_output	generate output for all explored paths
vm.path_output	generate program trace output
...	and many, many more

Explicit State Model Checking and JPF (2/3)

Search

Selects next state to explore.

Configurations:

search.class	search implementation
search.depth_limit	maximal path length
search.match_depth	only unify if depth for revisit is lower than known depth
search.multiple_errors	do not stop searching at first property violation
search.properties	which properties to check during search
...	further options for each search

Explicit State Model Checking and JPF (3/3)

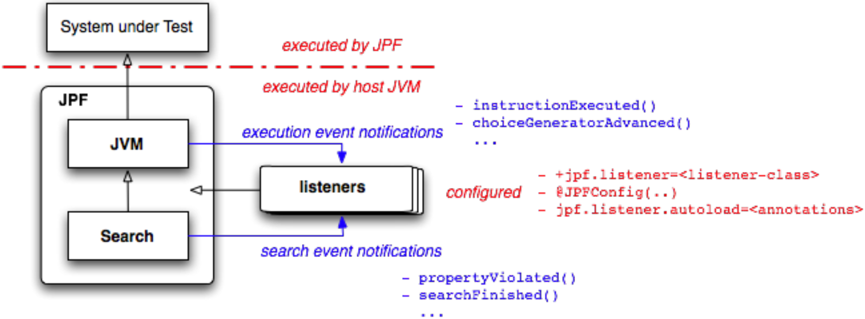
Listener

Evaluate states against properties.

Listeners can influence current transition while properties cannot.

Listener can monitor search and instruction execution.

Own listener can be set with the `listener` configuration option.



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

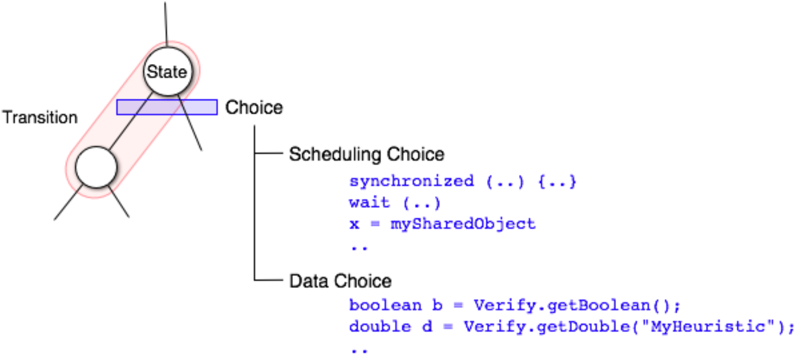
States

Collection of

- thread state (current instruction, stack),
- global variables,
- heap references, and
- trail (path to the state)

Transitions

- Sequence of instructions
- End of transition determined by
  - Multiple successor states (choices)
  - Enforced by listeners (*vm.breakTransition();*)
  - Reached maximal length (configuration `vm.max_transition_length`)
  - End or blocking of current thread



Scheduling Choice

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

Data Choice

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

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

Choices

Scheduling Choices

Which other thread is runnable?

Partial Order Reduction: Is this thread affected by the current transition?

Controlled by search and VM

Data Choices

Which concrete value to choose for the variables?

Mostly configured by the user

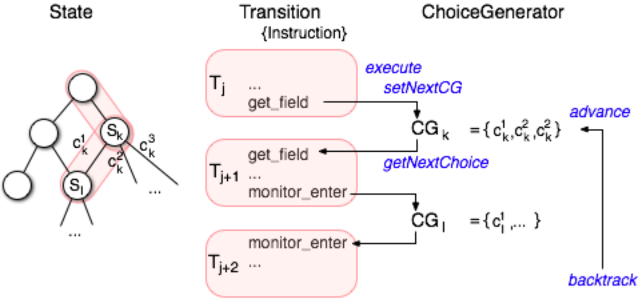
Control Choices

Which branch in the program to take?

Explicit invocation schedule by extensions

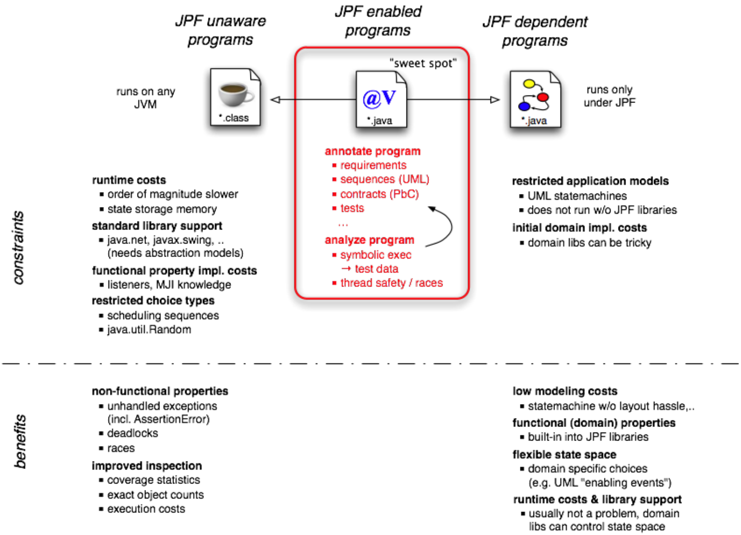
Implementing Choices

- choices encapsulated in ChoiceGenerators (CGs)
- registered by VM, instructions, extensions, or listeners
- `cg.randomize_choices` configures JPF to randomly explore choices



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

Applications, JPF, and JPF-Applications



JPF unaware programs

runs on any JVM

*.class

JPF enabled programs

"sweet spot"

@V

*.java

annotate program
- requirements
- sequences (UML)
- contracts (PbC)
- tests

...

analyze program
- symbolic exec
  → test data
- thread safety / races

JPF dependent programs

*.java

runs only under JPF

constraints

runtime costs
- order of magnitude slower
- state storage memory

standard library support
- java.net, javax.swing, ..
  (needs abstraction models)

functional property impl. costs
- listeners, MJI knowledge

restricted choice types
- scheduling sequences
- java.util.Random

restricted application models
- UML statemachines
- does not run w/o JPF libraries

initial domain impl. costs
- domain libs can be tricky

benefits

non-functional properties
- unhandled exceptions
  (incl. AssertionError)
- deadlocks
- races

improved inspection
- coverage statistics
- exact object counts
- execution costs

low modeling costs
- statemachine w/o layout hassle,..

functional (domain) properties
- built-in into JPF libraries

flexible state space
- domain specific choices
  (e.g. UML "enabling events")

runtime costs & library support
- usually not a problem, domain libs can control state space

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

Interfering with the Search (1/2)

gov.nasa.jpf.jvm.Verify for choices

getBoolean	Get a Boolean CG
getInt	Get a named integer CG
getIntFromList	Get an integer CG initialized from a list
getObject	Get a named object CG
getDouble	Get a named double CG
getDoubleFromList	Get a double CG initialized from a list
getLongFromList	Get a long CG initialized from a list
getFloatFromList	Get a float CG initialized from a list
random	Get a CG for random values
randomBool	Get a Boolean CG

Interfering with the Search (2/2)

gov.nasa.jpf.jvm.Verify for transitions and states

addComment	Add a comment to a state
instrumentPoint	Add a label to a state
atLabel	Check for a label
boring	Hint an uninteresting state
interesting	Conditionally hint an interesting state
ignoreIf	Conditionally prune the search space
beginAtomic	Start an atomic block
endAtomic	End an atomic block
breakTransition	End the current transition