JPF HowTo

Josef Kufner <kufnejos@fel.cvut.cz>

28. listopadu 2013

1 JPF

- Co je to JPF?
  `http://babelfish.arc.nasa.gov/trac/jpf/wiki/intro/what_is_jpf`
- Tutoriál a další dokumentace:
  `http://babelfish.arc.nasa.gov/trac/jpf/wiki/presentations/start`

2 Instalace JPF

Zkoušeno na Debian Linuxu.

1. Nainstalovat si Javu od Oracle:
   - Nefunguje to s OpenJDK, které je v Debianu defaultně.
   - Prý stačí Java 6, zkoušeno s Javou 1.7.0_45.
   - `http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html`
2. Vytvořit si adresář pro JPF:
   - `mkdir ~/projects/jpf`
3. Naklonovat si repositáře (Mercurial):
   - `hg clone http://babelfish.arc.nasa.gov/hg/jpf/jpf-core`
   - `hg clone http://babelfish.arc.nasa.gov/hg/jpf/jpf-symbc`
   - Tím vzniknou adresáře `~/projects/jpf/jpf-core a ~/projects/jpf/jpf-symbc`
4. Vytvořit lokální konfiguraci `~/.jpf/site.properties`:
   - Viz `http://babelfish.arc.nasa.gov/trac/jpf/wiki/install/site-properties`
   - ```
     # JPF site configuration
     jpf-core = ${user.home}/projects/jpf/jpf-core
     jpf-symbc = ${user.home}/projects/jpf/jpf-symbc
     extensions = ${jpf-core},${jpf-symbc}
     ```
5. Spustit Eclipse nebo Netbeans a nainstalovat si plugin:
   - Viz `http://babelfish.arc.nasa.gov/trac/jpf/wiki/install/eclipse-plugin`
   - Viz `http://babelfish.arc.nasa.gov/trac/jpf/wiki/install/netbeans-plugin`
6. Importovat do svého IDE projekty z naklonovaných repositářů (viz bod 3).
7. V IDE najít soubor jpf-core/src/examples/HelloWorld.jpf, pravým myšítkem a zvolit „Verify...“.
   - Měl objevit poněkud delší výpis a někde uprostřed „I won't say it!“.
   - Přibaleno je mnoho dalších příkladů, jak v `jpf-core`, tak v `jpf-symbc`.
     ```

3 Symbolic PathFinder

- http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/jpf-symbc
- http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/jpf-symbc/doc

3.1 Ukázka testovaného programu

```
public class HelloWorld
{
    public static void sayHello(int x, int y) throws RuntimeException
    {

        if (x == 7) {
            throw new RuntimeException("I don't like number 7.");
        }

        if (x > y + 1) {
            System.out.println("Hello world!");
        } else {
                for (int i = 0; i < x && i < 10; i++) {
                        System.out.println("I won't say it.");
                }
        }
    }

    public static void main(String[] args) {
        try {
            sayHello(3, 4);
            sayHello(3, 3);
            sayHello(3, 2);
            sayHello(3, 1);
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
    }

}
```

3.2 Ukázka konfigurace

```
target=HelloWorld
classpath=bin/,${classpath}

#shell=.shell.BasicShell

# You can specify multiple methods to be executed symbolically as follows:
# symbolic.method=<fully qualified name>.test(sym#con)
symbolic.method=HelloWorld.sayHello(sym#con)

# listener to print information (PCs, test cases) about symbolic run
#listener=gov.nasa.jpf.symbc.SymbolicListener

# listener to print test sequences
listener+=,gov.nasa.jpf.symbc.sequences.SymbolicSequenceListener

# specify the search strategy (default is DFS)
#search.class = .search.heuristic.BFSHeuristic

# limit the search depth (number of choices along the path)
search.depth_limit = 10

# You can pick which decision procedure to choose (if unspecified,
# choco is used as default):
symbolic.dp=choco
#symbolic.dp=iasolver
#symbolic.dp=cvc3
#symbolic.dp=cvc3bitvec
#symbolic.dp=no_solver

# New options have been added, to specify min/max values for symbolic
# variables and also to give the default for don't care values.
symbolic.min_int=-100
symbolic.max_int=100
symbolic.min_double=-1000.0
symbolic.max_double=1000.0
symbolic.undefined=0

# An option to increase the time limit until which choco tries
# to solve a particular constraint
# default value is 30000
choco.time_bound=30000

# Print debug information
symbolic.debug=on
```

3.3 Ukázka vygenerovaného unit testu

```
import static org.junit.Assert.*;
import org.junit.Before;
import org.junit.Test;

public class HelloWorldTest {

        private HelloWorld helloworld;

        @Before
        public void setUp() throws Exception {
                helloworld = new HelloWorld();
        }

        @Test
        public void test0() {
                helloworld.sayHello(7,-2147483648);
        }

        @Test
        public void test1() {
                helloworld.sayHello(-98,-100);
                helloworld.sayHello(7,-2147483648);
        }

        @Test
        public void test2() {
                helloworld.sayHello(-98,-100);
                helloworld.sayHello(-98,-100);
                helloworld.sayHello(7,-2147483648);
        }

        @Test
        public void test3() {
                helloworld.sayHello(-98,-100);
                helloworld.sayHello(-98,-100);
                helloworld.sayHello(-98,-100);
        }

        @Test
        public void test4() {
                helloworld.sayHello(-98,-100);
                helloworld.sayHello(-98,-100);
                helloworld.sayHello(-100,-100);
        }

        @Test
        public void test5() {
                helloworld.sayHello(-98,-100);
                helloworld.sayHello(-98,-100);
        }

// ...

        @Test
        public void test39() {
                helloworld.sayHello(-100,-100);
        }
}
```