Estudo e Desenvolvimento de novas funcionalidades do Java Pathfinder

Orientadora: Ana Cristina Vieira de Melo (MAC-IME)
Aluno: Rodrigo Della Vittoria Duarte

Universidade de São Paulo
Instituto de Matemática e Estatística

Este trabalho faz parte de um projeto maior da área de Testes e Validação. Para aproveitarmos todas funcionalidades disponíveis no Java PathfinderX, necessitamos operá-lo em linha com outros software:

OcongraX → **Java PathfinderX**

Ferramenta responsável pela análise dos dados gerados

Java Pathfinder

O Java Pathfinder é um verificador de modelo desenvolvido no centro de pesquisas da NASA. Verifica um programa alvo, em Java, de todas as maneiras possíveis para checar se este viola alguma propriedade(ocorrência de deadlock, existência de exceções não tratadas, etc.);

- Entrada: Programa em Java;
- Saída: Resultados na Tela.

Java PathfinderX

Além de manter as funcionalidades do Java Pathfinder, o Java PathfinderX apresenta novas funcionalidades:

- Geração de arquivos com todas instruções executadas;
- Leitura do arquivo XML da OcongraX;
- Geração dos arquivos XML que contém os pares de definição e uso cobertos e não cobertos pelo Java PathfinderX.

- Entrada: Programa em Java + Arquivo XML gerado pela OcongraX;
- Saída: Resultados na Tela + Arquivos com instruções executadas + Arquivos XML com os pares definição-uso cobertos e não cobertos.

Programa em Java:

```
public class DeadlockException {
    static Lock lock1;
    static Lock lock2;
    static int  state;

    public static void main(String[] args) {
        lock1 = new Lock();
        lock2 = new Lock();
        Process1 p1 = new Process1();
        Process2 p2 = new Process2();
        p1.start();
        p2.start();
    }

    public static void geraException() throws Exception{
        int valor = (int) (Math.random()*10);
        if(valor%2 == 0)
            throw new Exception();
    }
}

class Process1 extends Thread {
    public void run() {
        DeadlockException.state++;
        synchronized (DeadlockException.lock2) {
            synchronized (DeadlockException.lock1) {
                DeadlockException.state++;
            }
        }
    }
}
```

```
class Process2 extends Thread {
    public void run() {
        DeadlockException.state++;
        try {
            DeadlockException.geraException();
        } catch(Exception e) {
            synchronized (DeadlockException.lock2) {
                synchronized (DeadlockException.lock1) {
                    DeadlockException.state++;
                }
            }
        } finally {
            synchronized (DeadlockException.lock1) {
                synchronized (DeadlockException.lock2) {
                    DeadlockException.state++;
                }
            }
        }
    }
}

class Lock {
}
```

Arquivo da OcongraX(somente para Java PathfinderX):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<elements>
    <object>
        <classname>name1</classname>
        <name>teste</name>
        <pair>
            <def>23</def>
            <use>24</use>
        </pair>
        <pair>
            <def>25</def>
            <use>37</use>
        </pair>
        <pair>
            <def>23</def>
            <use>100</use>
        </pair>
        <def>1</def>
        <use>2</use>
    </object>
    <object>
      <classname>name2</classname>
        <name>teste2</name>
        <pair>
            <def>24</def>
            <use>23</use>
        </pair>
        <pair>
            <def>26</def>
            <use>25</use>
        </pair>
        <pair>
            <def>100</def>
            <use>99</use>
        </pair>
    </object>
```

```
    <object>
      <classname>name3</classname>
        <name>teste5</name>
        <pair>
            <def>24</def>
            <use>23</use>
        </pair>
        <pair>
            <def>26</def>
            <use>25</use>
        </pair>
        <pair>
            <def>100</def>
            <use>99</use>
        </pair>
    </object>
    <exception>
      <classname>name4</classname>
        <name>teste3</name>
        <pair>
            <def>23</def>
            <use>24</use>
        </pair>
        <pair>
            <def>25</def>
            <use>26</use>
        </pair>
        <pair>
            <def>99</def>
            <use>100</use>
        </pair>
    </exception>
```

```
    <exception>
      <classname>name5</classname>
        <name>teste4</name>
        <pair>
            <def>24</def>
            <use>23</use>
        </pair>
        <pair>
            <def>26</def>
            <use>25</use>
        </pair>
        <pair>
            <def>100</def>
            <use>99</use>
        </pair>
    </exception>
</elements>
```

Resultados na Tela:

```
JavaPathfinder v4.1 - (C) 1999-2007 RIACS/NASA Ames Research Center

====================================================== system under test
application: C:\Documents and Settings\Kiko\Meus
documentos\tcc\projeto\TestesJPF\src\DeadlockException.java

====================================================== search started: 08/11/07
.
.
.
====================================================== snapshot #1
thread index=1,name=Thread-
0,status=BLOCKED,this=Process1@223,priority=5,lockCount=0
  owned locks:Lock@219
  blocked on: Lock@218
  call stack:
       at Process1.run(DeadlockException.java:39)

thread index=2,name=Thread-
1,status=BLOCKED,this=Process2@248,priority=5,lockCount=0
  owned locks:Lock@218
  blocked on: Lock@219
  call stack:
       at Process2.run(DeadlockException.java:59)


====================================================== results
error #1: gov.nasa.jpf.jvm.NotDeadlockedProperty "deadlock encountered:   thread
inde..."

====================================================== search finished: 08/11/07
```

Arquivo(s) com todas instruções executadas(somente para Java PathfinderX):

```
15      DeadlockException.state++;
16      synchronized (DeadlockException.lock2) {
17          synchronized (DeadlockException.lock1) {
18              DeadlockException.state++;
19          synchronized (DeadlockException.lock1) {
20      synchronized (DeadlockException.lock2) {
21  }
38              DeadlockException.state++;
39          synchronized (DeadlockException.lock1) {
40      synchronized (DeadlockException.lock2) {
41  }
58              DeadlockException.state++;
59          synchronized (DeadlockException.lock1) {
60      synchronized (DeadlockException.lock2) {
61  }
80              DeadlockException.state++;
81          synchronized (DeadlockException.lock1) {
82      synchronized (DeadlockException.lock2) {
83  }
102             DeadlockException.state++;
103         synchronized (DeadlockException.lock1) {
```

```
22      DeadlockException.state++;
23          DeadlockException.geraException();
24      int valor = (int) (Math.random()*10);
25      if(valor%2 == 0)
26  }
27          DeadlockException.geraException();
28      }
29      } finally {
30          synchronized (DeadlockException.lock1) {
31              synchronized (DeadlockException.lock2) {
32                  DeadlockException.state++;
33              synchronized (DeadlockException.lock2) {
34          synchronized (DeadlockException.lock1) {
35      }
36  }
37      DeadlockException.state++;
42      DeadlockException.state++;
43          DeadlockException.geraException();
44      int valor = (int) (Math.random()*10);
45      if(valor%2 == 0)
46  }
```

Arquivo XML com os pares cobertos(somente para Java PathfinderX):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<elements>
    <object>
        <classname>name1</classname>
        <name>teste</name>
        <pair>
            <def>23</def>
            <use>24</use>
        </pair>
        <pair>
            <def>25</def>
            <use>37</use>
        </pair>
    </object>
    <object>
      <classname>name2</classname>
        <name>teste2</name>
        <pair>
            <def>24</def>
            <use>23</use>
        </pair>
        <pair>
            <def>26</def>
            <use>25</use>
        </pair>
    </object>
```

```
    <object>
      <classname>name3</classname>
        <name>teste5</name>
        <pair>
            <def>24</def>
            <use>23</use>
        </pair>
        <pair>
            <def>26</def>
            <use>25</use>
        </pair>
    </object>
    <exception>
      <classname>name4</classname>
        <name>teste3</name>
        <pair>
            <def>23</def>
            <use>24</use>
        </pair>
        <pair>
            <def>25</def>
            <use>26</use>
        </pair>
    </exception>
```

```
    <exception>
      <classname>name5</classname>
        <name>teste4</name>
        <pair>
            <def>24</def>
            <use>23</use>
        </pair>
        <pair>
            <def>26</def>
            <use>25</use>
        </pair>
    </exception>
</elements>
```

Arquivo XML com os pares não cobertos(somente para Java PathfinderX):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<elements>
    <object>
        <classname>name1</classname>
        <name>teste</name>
        <pair>
            <def>23</def>
            <use>100</use>
        </pair>
    </object>
    <object>
      <classname>name2</classname>
        <name>teste2</name>
        <pair>
            <def>100</def>
            <use>99</use>
        </pair>
    </object>
```

```
    <object>
      <classname>name3</classname>
        <name>teste5</name>
        <pair>
            <def>100</def>
            <use>99</use>
        </pair>
    </object>
    <exception>
      <classname>name4</classname>
        <name>teste3</name>
        <pair>
            <def>99</def>
            <use>100</use>
        </pair>
    </exception>
```

```
    <exception>
      <classname>name5</classname>
        <name>teste4</name>
        <pair>
            <def>100</def>
            <use>99</use>
        </pair>
    </exception>
</elements>
```

- Agora podemos testar somente os pares não cobertos pelo Java PathfinderX, evitando testar todos os pares contidos no arquivo gerado pela OcongraX. Além disso, podemos rastrear a execução do programa através da análise dos arquivos com todas instruções executadas.