Exam System Validation (214012) **8:45-12:15, 25-06-2010**

- The exercises are worth a total of 90 points. The final grade is $\frac{10+pts}{10}$.
- The exam is open book: copies of slides/papers/notes/etc. are allowed.

Exercise 1 (15 points)

Suppose that you are working as a verification engineer at a small software company. The company is working on a project to develop software for the storm surge barrier in the Nieuwe Waterweg. The software consists of three components:

- the decision component that decides whether the barrier should be opened or closed
- the control component that controls the movements of the barrier doors
- the communication component that passes information between the different components.

The decision component has to make complex calculations. It has to function 100% correctly. The barrier should always be closed when the weather conditions require this. However, the barrier should not be closed unnecessarily. Because of the high amount of ship traffic on the Nieuwe Waterweg, unnecessary closing can lead to millions of euros damage.

The control component may fail, but whenever it fails, it should be stopped and reset immediately.

The communication component passes for example sensor information and weather forecasts (provided as user input) on to the decision component, and it passes results from the decision component on to the control component.

Your company should not only provide the software (written in Java), it should also provide a formal argument why the software is safe to be used.

You are asked to give a recommendation how different formal methods and tools should be used to provide this formal argument. The recommendation should be similar in length to the question (200-250 words).

Exercise 2 (10 points)

Write formal specifications for the following properties in an appropriate specification formalism (2 points per item).

A Method `doubleBetween` takes a non-empty array `arr` of integers and two appropriate indices i, j as input. All values in the closed interval $[i, j]$ are doubled, all others remain unchanged.

B In an elevator system, whatever happens, the system can always reach a state where the lift door can open.

C If a method changes the value of variable `x`, then it can only double it.

D When the lift has arrived on floor 1, the door of the lift is open as long as the alarm does not sound. The alarm has to sound at some point.

E The method `m` terminates normally when the integer array `arr` is sorted, otherwise it throws an exception IllegalParameterFormatException. This is the only exception that can be thrown by the method.

Exercise 3 (15 points).

Below are several Java classes. Predict the behaviour of the JML run-time checker, when applied to these classes. Argue why the predicted behaviour will occur. (3 points each, 1 for the correct answer, 2 for the correct argument)

problem A.

(Also used for question 4A.)

```
1  class RtQuestion1 {
2
3          //@ invariant a != null;
4          //@ constraint a.length == \old(a.length);
5          int [] a = new int[5];
6
7          /*@ requires x >= 0;
8              ensures \result == a[x];
9              signals (Exception) false;
10          */
11         int lookup(int x) {
12                 return a[x];
13         }
14 }
15 public class RtChecking1 {
16
17         public static void main(String [] args){
18                 RtQuestion1 r = new RtQuestion1();
19                 for (int i = 0; i < 4; i++) {
20                         System.out.println(r.lookup(2*i));
21                 }
22         }
23
24 }
```

problem B.

(Also used for question 4B.)

```
1  class RtQuestion2 {
2      boolean open;
3      //@ public final ghost int state0 = 0;
4      //@ public final ghost int state1 = 1;
5      //@ public final ghost int state2 = 2;
6      //@ public final ghost int state3 = 3;
7      //@ public ghost int state = state0;
8
9      //@ constraint \old(state) == state0 ==> state == state1 || state == state2;
10     //@ constraint \old(state) == state1 ==> state == state1 || state == state2;
11     //@ constraint \old(state) == state2 ==> state == state1 || state == state2;
12
13     void initDoor(boolean open) {
14         if (open) {
15             this.open = true;
16             //@ set state = state1;
17         } else {
18             this.open = false;
19             //@ set state = state2;
20         }
21     }
22
23     /*@ requires state == state1;
24       @ ensures state == state2;
25      */
26     void closeDoor() {
27         open = false;
28         //@ set state = state2;
29     }
30
31     /*@ requires state == state2;
32       @ ensures state == state1;
33      */
34     void openDoor() {
35         open = true;
36     }
37
38     /*@ requires state == state1 || state == state2;
39         ensures state == \old(state);
40      */
41     void doSomethingElse() {
42         System.out.println("pomtiedom");
43     }
44 }
45 public class RtChecking2 {
46     public static void main (String [] args) {
47         RtQuestion2 door = new RtQuestion2();
48         door.initDoor(true);
49         door.closeDoor();
50         while (true) {
51             door.doSomethingElse();
52         }
53     }
54 }
```

problem C.

```
1 class RtQuestion3 {
2
3         int arr [];
4         int count;
5
6         //@ invariant count >= 0 && count < arr.length;
7
8         RtQuestion3(int initCount, int initSize){
9                 arr = new int[initSize];
10                count = initCount;
11        }
12
13        void add(int elem) {
14                count++;
15                if (count == arr.length) {
16                        resize(count);
17                }
18                arr[count] = elem;
19        }
20
21        void resize(int count) {
22                int [] new_arr = new int [2 * count];
23                for (int i = 0; i < count; i++) {
24                        new_arr[i] = arr[i];
25                };
26                arr = new_arr;
27        }
28
29 }
30
31 public class RtChecking3 {
32
33        public static void main (String [] args) {
34                RtQuestion3 list = new RtQuestion3(0, 2);
35                list.add(2);
36                list.add(4);
37                list.add(6);
38                list.add(8);
39        }
40
41 }
```

problem D.

```
1 class RtQuestion4 {
2
3         //@ ensures \result == a*a - b*b;
4         int compute(int a, int b){
5                 int res1 = a + b;
6                 int res2 = a - b;
7                 return res1 + res2;
8         }
9 }
10
11 public class RtChecking4 {
12
13         public static void main (String [] args) {
14                 RtQuestion4 c = new RtQuestion4();
15                 c.compute(2, 0);
16         }
17
18 }
```

problem E.

```
1 class RtQuestion5 {
2
3         /*@ spec_public */ private int x = 24;
4         private int y = 45;
5         private int z = 88;
6
7         //@ constraint x > \old(x);
8
9         void addXtoX () { x = x + x; }
10        void addYtoX () { x = x + y; }
11        void addZtoX () { x = x + z; }
12        void addXtoY () { y = y + x; }
13        void addYtoY () { y = y + y; }
14        void addZtoY () { y = y + z; }
15        void addXtoZ () { z = z + x; }
16        void addYtoZ () { z = z + y; }
17        void addZtoZ () { z = z + z; }
18 }
19 public class RtChecking5 {
20                public static void main (String [] args) {
21                RtQuestion5 val = new RtQuestion5();
22                val.addXtoX();
23                val.addYtoX();
24                val.addZtoX();
25                val.addZtoZ();
26                val.addYtoY();
27        }
28 }
```

Exercise 4 (15 points).

Below are several (references to) Java classes. Predict the behaviour of ESC/Java, when applied to these classes. Argue why the predicted behaviour will occur. (3 points each, 1 for the correct answer, 2 for the correct argument)

problem A.

Class `RtQuestion1` from problem 3A, page 3.

problem B.

Class `RtQuestion2` from problem 3B, page 4.

problem C.

```
1 public class StaticChecking3 {
2
3         //@ public model int surface;
4
5         private int height;
6         private int width;
7
8         //@ represents surface = height * width;
9
10
11        //@ ensures surface == h * w;
12        StaticChecking3(int h, int w){
13                height = h;
14                width = w;
15        }
16
17        //@ ensures surface == \old(surface) * factor;
18        void enlarge(int factor){
19                height = height * factor;
20                width = width * factor;
21        }
22
23        //@ ensures surface == \old(surface);
24        //@ ensures height == \old(width) && width == \old(height);
25        void rotate() {
26                int temp = height;
27                height = width;
28                width = temp;
29        }
30
31 }
```

problem D.

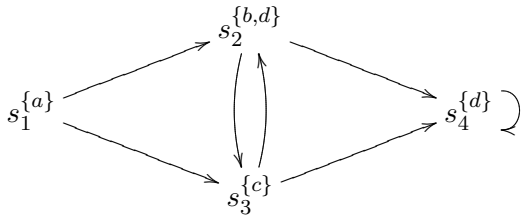
```
1 public class StaticChecking4 {
2
3 int [] arr;
4
5 /*@ requires arr != null;
6     ensures (\forall int i; 0 <= i && i < arr.length; arr[i] == \old(arr[i]* i));
7  */
8 void update() {
9   int i = 0;
10  //@ loop_invariant (\forall int j; 0 <= j && j < i; arr[j] == \old(arr[j] * j));
11  //@ loop_invariant 0 <= i && i <= arr.length;
12  while (i < arr.length) {
13    arr[i] = arr[i] * i;
14    i++;
15  }
16 }
17 }
```

problem E.

```
1 public class StaticChecking5 {
2
3         //@ invariant 0 <= x && x < max;
4
5         /*@ spec_public */ private int x;
6         /*@ spec_public */ private int y;
7         public static final int max = 24;
8
9         /*@ requires 0 <= a && a < max;
10            ensures x == a;
11         */
12        StaticChecking5(int a, int b){
13                x = a;
14                y = b;
15        }
16
17        /*@ assignable x;
18            ensures x == (\old(x) + \old(x))%max;
19         */
20        void addXtoX() {
21                x = x + x;
22                x = x % max;
23        }
24
25        /*@ assignable x;
26            ensures x == (\old(x) + y)%max;
27         */
28        void addYtoX() {
29                x = x + y;
30                x = x % max;
31        }
32 }
```

Exercise 5 (10 points)

Consider the model:



For each of the following formula's, is the formula true or false for the initial state s_1?

A $\mathsf{G}(c \Rightarrow (c \,\mathsf{U}\, b))$

B $\mathsf{AG}\,\mathsf{EF}\,\mathsf{AG}\, d$

C $\mathsf{G}(\mathsf{F}\, b)$

D $\mathsf{G}((\mathsf{G}(b \vee c)) \Rightarrow (\mathsf{G}\,\mathsf{F}\, b))$

E $\mathsf{AX}\,\mathsf{E}[b \,\mathsf{U}\, c]$.

You only have to answer yes or no. For each correct answer: +2 points. For each incorrect answer: −2 points. For each unanswered question, nothing changes. A negative total becomes 0.

Exercise 6 (10 points)

An airlock is used to prevent too much warmth being lost from a building. (E.g. the entrance of the Hogekamp building.) The control software has been modeled in SMV and uses the following atomic propositions:

`inner_open` - The inner door is standing still in open position.
`inner_closing` - The inner door is moving towards the closed position.
`inner_opening` - The inner door is moving towards the open position.
`inner_closed` - The inner door is standing still in closed position.
`inner_detect` - A person is detected in the proximity of the inner door.

`outer_open` - The outer door is standing still in open position.
`outer_closing` - The outer door is moving towards the closed position.
`outer_opening` - The outer door is moving towards the open position.
`outer_closed` - The outer door is standing still in closed position.
`outer_detect` - A person is detected in the proximity of the outer door.

The manager, who has an MBA rather than a Computer Science degree, came up with the following requirements:

(i) The doors shall not squash people by closing at the wrong time.

(ii) The doors are never open at the same time.

(iii) If someone waits in front of a door it will eventually open.

Write a set of LTL and CTL properties that verify that the model satisfies all these requirements.

Note that the definitions of the atomic properties and the requirements were written by two different persons. This means that you will need to translate the intended meaning of the requirements rather than what's actually written.

Exercise 7 (15 points)

This exercise presents three cases of model checking problems and ask for interpretation of the counter-example output. Each case is worth 5 points.

problem A.

The eight queens puzzle is the problem of placing eight chess queens on an 8×8 chessboard such that none of them are able to capture any other using the standard chess queen's moves. The queens must be placed in such a way that no two queens attack each other. Thus, a solution requires that no two queens share the same row, column, or diagonal. The eight queens puzzle is an instance of the more general n-queens problem of placing n queens on an $n \times n$ chessboard.

Someone has modeled the n-queens problem in SMV. The solution is based on putting one queen in each column. The row of each queen is kept in the array `board`. The safety of each queen (no other queen on same row,column or diagonal) is kept in the boolean array `safe`. The transitions of the system model the fact that continuously one of the queens is chosen.

The next state of a board is determined by choosing one of the queens non-deterministically. If she is is safe, she must stay where she is. Otherwise she can move to any row in her column.

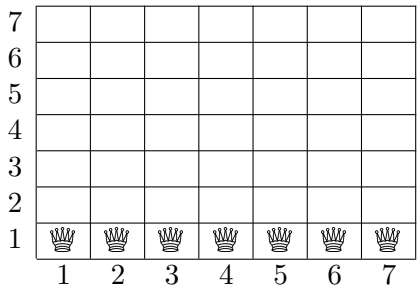
When NuSMV is run on the 7-queens problem, the output is

```
-- specification AG !((((((safe[1] & safe[2]) & safe[3]) & safe[4]) &
                           safe[5]) & safe[6]) & safe[7])  is false
-- as demonstrated by the following execution sequence
Trace Description: AG Only counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  board[1] = 1
  board[2] = 1
  board[3] = 1
  board[4] = 1
  board[5] = 1
  board[6] = 1
  board[7] = 1
  safe[1] = 0
  safe[2] = 0
  safe[3] = 0
  safe[4] = 0
  safe[5] = 0
  safe[6] = 0
  safe[7] = 0
-> Input: 1.2 <-
  col = 1
-> State: 1.2 <-
  board[1] = 4
-> Input: 1.3 <-
  col = 1
-> State: 1.3 <-
  board[2] = 6
-> Input: 1.4 <-
  col = 4
-> State: 1.4 <-
  board[4] = 3
  safe[1] = 1
-> Input: 1.5 <-
```

```
  col = 4
-> State: 1.5 <-
  board[5] = 5
  safe[5] = 1
-> Input: 1.6 <-
  col = 2
-> State: 1.6 <-
  board[6] = 7
  safe[4] = 1
  safe[6] = 1
-> Input: 1.7 <-
  col = 2
-> State: 1.7 <-
  board[7] = 2
  safe[2] = 1
  safe[3] = 1
  safe[7] = 1
```

(i) What (in plain English) is the meaning of the formula?

(ii) The initial position is:



What are the positions of the queens on the board at the end of the counter-example trace?

problem B.

Given a small randomized Java program.

```
1 import java.util.Random;
2 public class JPFrandom {
3
4   public static void main(String args[]){
5     int N=Integer.parseInt(args[0]);
6     Random random = new Random();
7     int total=0;
8     int i=0;
9     while(i<N) {
10        total=(total + i*i) % N;
11        i+=random.nextInt(2)+1;
12    }
13    assert total>0;
14  }
15 }
```

The input for JPF was

```
target=JPFrandom
target_args=6
classpath=.
sourcepath=.
cg.enumerate_random=true
report.console.property_violation=error,trace
```

The result was the trace below, apparently it is possible for the variable `total` to reach 0.

Explain how this happend by reconstructing the values of the variables `i` and `total` during the run of the program from the trace.

```
====================================================== error #1
gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty
java.lang.AssertionError
at JPFrandom.main(JPFrandom.java:13)


====================================================== trace #1
------------------------------------------------------ transition #0 thread: 0
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {>main}
      [2844 insn w/o sources]
  JPFrandom.java:2               : public class JPFrandom {
      [1 insn w/o sources]
  JPFrandom.java:5               : int N=Integer.parseInt(args[0]);
  JPFrandom.java:6               : Random random = new Random();
      [2 insn w/o sources]
  JPFrandom.java:6               : Random random = new Random();
      [57 insn w/o sources]
  JPFrandom.java:6               : Random random = new Random();
  JPFrandom.java:7               : int total=0;
  JPFrandom.java:8               : int i=0;
  JPFrandom.java:9               : while(i<N) {
  JPFrandom.java:10              : total=(total + i*i) % N;
  JPFrandom.java:11              : i+=random.nextInt(2)+1;
------------------------------------------------------ transition #1 thread: 0
gov.nasa.jpf.jvm.choice.IntIntervalGenerator[0..1,delta=+1,cur=0]
```

```
  JPFrandom.java:11              : i+=random.nextInt(2)+1;
  JPFrandom.java:9               : while(i<N) {
  JPFrandom.java:10              : total=(total + i*i) % N;
  JPFrandom.java:11              : i+=random.nextInt(2)+1;
------------------------------------------------------ transition #2 thread: 0
gov.nasa.jpf.jvm.choice.IntIntervalGenerator[0..1,delta=+1,cur=0]
  JPFrandom.java:11              : i+=random.nextInt(2)+1;
  JPFrandom.java:9               : while(i<N) {
  JPFrandom.java:10              : total=(total + i*i) % N;
  JPFrandom.java:11              : i+=random.nextInt(2)+1;
------------------------------------------------------ transition #3 thread: 0
gov.nasa.jpf.jvm.choice.IntIntervalGenerator[0..1,delta=+1,cur=0]
  JPFrandom.java:11              : i+=random.nextInt(2)+1;
  JPFrandom.java:9               : while(i<N) {
  JPFrandom.java:10              : total=(total + i*i) % N;
  JPFrandom.java:11              : i+=random.nextInt(2)+1;
------------------------------------------------------ transition #4 thread: 0
gov.nasa.jpf.jvm.choice.IntIntervalGenerator[0..1,delta=+1,cur=0]
  JPFrandom.java:11              : i+=random.nextInt(2)+1;
  JPFrandom.java:9               : while(i<N) {
  JPFrandom.java:10              : total=(total + i*i) % N;
  JPFrandom.java:11              : i+=random.nextInt(2)+1;
------------------------------------------------------ transition #5 thread: 0
gov.nasa.jpf.jvm.choice.IntIntervalGenerator[0..1,delta=+1,cur=1]
  JPFrandom.java:11              : i+=random.nextInt(2)+1;
  JPFrandom.java:9               : while(i<N) {
  JPFrandom.java:13              : assert total>0;
      [13 insn w/o sources]

====================================================== results
error #1: gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty "java.lang.AssertionError
   at JPFrandom.main(JPFran..."
```

problem C.

Given is a small two threaded Java program:

```
1  class IntVal{
2    private int val;
3    public IntVal(int val){this.val=val;}
4    public void set(int val){this.val=val;}
5    public int get(){return val;}
6  }
7  public class TwoThreads {
8    public static void main(String args[]){
9      final IntVal ival = new IntVal(0);
10     Thread t1 = new Thread(new Runnable(){
11       public void run(){
12         for(int i=0;i<2;i++) {
13           ival.set(ival.get()+1);
14         }
15       }
16     });
17     Thread t2 = new Thread(new Runnable(){
18       public void run(){
19         for(int i=0;i<2;i++) {
20           ival.set(ival.get()+1);
21         }
22      }
23     });
24     t1.start();
25     t2.start();
26     try {
27       t1.join();
28       t2.join();
29     } catch (InterruptedException e) {}
30     assert ival.get()>2;
31   }
32 }
```

When verified with JPF, the assertion (line 30) was triggered. Extract the scenario that leads to the error from the JPF output below.

In the scenario threads take turns running for a while. When you describe the scenario, it is important to know which thread is running and precisely what the first and/or last action of the running thread during its turn is. (E.g. thread 37 starts by reading 7 from variable x and stops just before reading variable y.) The description of what a thread does during its turn can and should be much less detailed.

```
====================================================== error #1
gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty
java.lang.AssertionError
at TwoThreads.main(TwoThreads.java:30)

====================================================== trace #1
------------------------------------------------------ transition #0 thread: 0
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {>main}
      [2844 insn w/o sources]
  TwoThreads.java:7             : public class TwoThreads {
      [1 insn w/o sources]
```

```
  TwoThreads.java:9              : final IntVal ival = new IntVal(0);
  TwoThreads.java:3              : public IntVal(int val){this.val=val;}
      [1 insn w/o sources]
  TwoThreads.java:3              : public IntVal(int val){this.val=val;}
  TwoThreads.java:9              : final IntVal ival = new IntVal(0);
  TwoThreads.java:10             : Thread t1 = new Thread(new Runnable(){
      [1 insn w/o sources]
  TwoThreads.java:10             : Thread t1 = new Thread(new Runnable(){
      [180 insn w/o sources]
  TwoThreads.java:10             : Thread t1 = new Thread(new Runnable(){
  TwoThreads.java:17             : Thread t2 = new Thread(new Runnable(){
      [1 insn w/o sources]
  TwoThreads.java:17             : Thread t2 = new Thread(new Runnable(){
      [133 insn w/o sources]
  TwoThreads.java:17             : Thread t2 = new Thread(new Runnable(){
  TwoThreads.java:24             : t1.start();
------------------------------------------------------ transition #1 thread: 0
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {>main,Thread-0}
  TwoThreads.java:24             : t1.start();
  TwoThreads.java:25             : t2.start();
------------------------------------------------------ transition #2 thread: 0
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {>main,Thread-0,Thread-1}
  TwoThreads.java:25             : t2.start();
  TwoThreads.java:27             : t1.join();
------------------------------------------------------ transition #3 thread: 0
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {>main,Thread-0,Thread-1}
  TwoThreads.java:27             : t1.join();
      [5 insn w/o sources]
------------------------------------------------------ transition #4 thread: 1
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {>Thread-0,Thread-1}
  TwoThreads.java:12             : for(int i=0;i<2;i++) {
  TwoThreads.java:13             : ival.set(ival.get()+1);
  TwoThreads.java:5              : public int get(){return val;}
------------------------------------------------------ transition #5 thread: 1
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {>Thread-0,Thread-1}
  TwoThreads.java:5              : public int get(){return val;}
  TwoThreads.java:13             : ival.set(ival.get()+1);
  TwoThreads.java:4              : public void set(int val){this.val=val;}
------------------------------------------------------ transition #6 thread: 1
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {>Thread-0,Thread-1}
  TwoThreads.java:4              : public void set(int val){this.val=val;}
  TwoThreads.java:12             : for(int i=0;i<2;i++) {
  TwoThreads.java:13             : ival.set(ival.get()+1);
  TwoThreads.java:5              : public int get(){return val;}
------------------------------------------------------ transition #7 thread: 1
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {>Thread-0,Thread-1}
  TwoThreads.java:5              : public int get(){return val;}
  TwoThreads.java:13             : ival.set(ival.get()+1);
  TwoThreads.java:4              : public void set(int val){this.val=val;}
------------------------------------------------------ transition #8 thread: 2
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {Thread-0,>Thread-1}
  TwoThreads.java:19             : for(int i=0;i<2;i++) {
  TwoThreads.java:20             : ival.set(ival.get()+1);
  TwoThreads.java:5              : public int get(){return val;}
------------------------------------------------------ transition #9 thread: 2
```

```
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {Thread-0,>Thread-1}
  TwoThreads.java:5              : public int get(){return val;}
  TwoThreads.java:20             : ival.set(ival.get()+1);
  TwoThreads.java:4              : public void set(int val){this.val=val;}
------------------------------------------------------ transition #10 thread: 2
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {Thread-0,>Thread-1}
  TwoThreads.java:4              : public void set(int val){this.val=val;}
  TwoThreads.java:19             : for(int i=0;i<2;i++) {
  TwoThreads.java:20             : ival.set(ival.get()+1);
  TwoThreads.java:5              : public int get(){return val;}
------------------------------------------------------ transition #11 thread: 2
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {Thread-0,>Thread-1}
  TwoThreads.java:5              : public int get(){return val;}
  TwoThreads.java:20             : ival.set(ival.get()+1);
  TwoThreads.java:4              : public void set(int val){this.val=val;}
------------------------------------------------------ transition #12 thread: 2
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {Thread-0,>Thread-1}
  TwoThreads.java:4              : public void set(int val){this.val=val;}
  TwoThreads.java:19             : for(int i=0;i<2;i++) {
  TwoThreads.java:22             : }
------------------------------------------------------ transition #13 thread: 1
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {>Thread-0}
  TwoThreads.java:4              : public void set(int val){this.val=val;}
  TwoThreads.java:12             : for(int i=0;i<2;i++) {
  TwoThreads.java:15             : }
------------------------------------------------------ transition #14 thread: 0
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {>main}
      [6 insn w/o sources]
  TwoThreads.java:28             : t2.join();
      [4 insn w/o sources]
  TwoThreads.java:29             : } catch (InterruptedException e) {}
  TwoThreads.java:30             : assert ival.get()>2;
  TwoThreads.java:5              : public int get(){return val;}
  TwoThreads.java:30             : assert ival.get()>2;
      [13 insn w/o sources]

====================================================== results
error #1: gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty "java.lang.AssertionError
   at TwoThreads.main(TwoTh..."
```