EECS 4315 3.0 Mission Critical Systems

Solution of final exam

15:45–17:00 on April 10, 2019

1 (3 marks)

(a) What is a *native method*?

Answer: A method that is implemented in a language other than Java but that is invoked from a Java app.

Marking scheme: 1 mark for mentioning something related to "implemented in another language."

(b) Explain the difference between a *peer* class and a *native peer* class.

Answer: A peer class models the behaviour of a native method as a method in Java code and this method (instead of the native method) is *model checked* by JPF's virtual machine. A native peer class contains a method corresponding to the native method (that may call native code) and this method is *executed* by the host virtual machine.

Marking scheme: 1 mark for mentioning that the method in the peer class is model checked and 1 mark for mentioning that the native peer class is executed.

2 (2 marks)

Consider the following class that tests a JPF listener.

```
import gov.nasa.jpf.util.test.TestJPF;
import org.junit.Test;

public class SampleTest extends TestJPF {
  /**
   * Tests a listener.
   */
  @Test
  public void test() {
    if (!TestJPF.isJPFRun()) {
      ...
    }
    if (this.verifyNoPropertyViolation(...)){
      ...
```

```
15      } else {
16        ...
17      }
18    }
19
20    ...
21  }
```

(a) Of the lines 10–17 of code, which lines are executed by the host virtual machine?

Answer: Lines 10–13 and 16.

Marking scheme: 0.5 mark for line 10–13 and 0.5 mark for lines 16.

(b) Of the lines 10–17 of code, which lines are executed by JPF's virtual machine?

Answer: Lines 10, and 13–14.

Marking scheme: 0.5 mark for line 10 and 0.5 mark for lines 13 and 14.

3 (3 marks)

Two threads share the attribute `a`. Initially, the value of `a` is 0. The first thread executes the following code.

```
a = a + 2;
```

The second thread executes the following code.

```
a = a - 2;
```

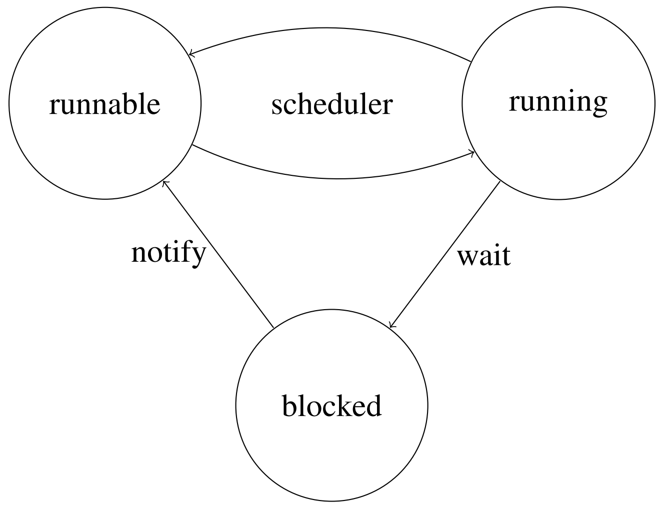
What are the possible final values of `a`? Explain your answer.

Answer: The statement `a = a + 2` consists of reading the value of `a`, incrementing that value by 2, and writing the new value to `a`. The statement `a = a - 2` consists of multiple bytecode instructions as well. Consider all interleavings, the final value of `a` can either, 0, 2, or -2.

Marking scheme: 1 marks for the observation that both statements consist of multiple bytecode instructions. 1 mark for mentioning that there are different interleavings. 1 mark for the correct final values.

4 (2 marks)

A thread can be in different states, as shown in the following diagram.



Consider the solution to the readers-writers problem given on pages 11–12. Using the diagram above, explain why we use while instead of if in line 12 of the `Database` class.

Answer: When line 14 is executed, the state of the `Reader` thread executing the code changes from running to blocked. When that `Reader` thread gets notified, its state changes from blocked to runnable. At that point, the `writing` attribute is false. However, before its state becomes running, `writing` attribute may be set to true. Hence, once its state becomes running, we should check again the value of the `writing` attribute.

Marking scheme: 0.5 mark for explaining the change from running to blocked. 0.5 mark for explaining the change from blocked to runnable. 1 mark for observing that the value of `writing` may change before the change from runnable to running.

5 (1 marks)

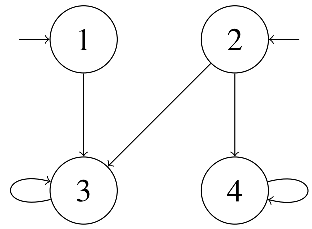
Consider the solution to the readers-writers problem given on pages 11–12. Can `notifyAll` on line 43 of the `Database` class be replaced with `notify`. Explain your answer.

Answer: No. It may cause a deadlock.

Marking scheme: 1 mark for for mentioning deadlock.

6 (5 marks)

Consider the following transition system



and the following labelling function

$$\begin{array}{rcl} \ell(1) & = & \{a\} \\ \ell(2) & = & \{a, b\} \\ \ell(3) & = & \{b, c\} \\ \ell(4) & = & \{c\} \end{array}$$

Note that states 1 and 2 are both initial. For each of the following LTL formulas, determine if that formula holds for the above transition system. A simple yes or no suffices.

(a) a

(b) b

(c) $a \wedge \neg b$

(d) $\bigcirc(b \vee c)$

(e) $\bigcirc \bigcirc \neg a$

(f) $\Diamond b$

(g) $\Box(a \vee c)$

(h) $a \mathbin{\mathrm{U}} b$

(i) $b \mathbin{\mathrm{U}} a$

(j) $a \mathbin{\mathrm{U}} ((b \vee c) \mathbin{\mathrm{U}} c)$

Answer:

(a) Yes.

(b) No.

(c) No.

(d) Yes.

(e) Yes.

(f) Yes.

(g) Yes.

(h) Yes.

(i) Yes.

(j) Yes.

For each of the following CTL formulas, determine if that formula holds for the above transition system. A simple yes or no suffices.

(k) $\exists\Diamond c$

(l) $\forall\Diamond c$

(m) $\exists\Diamond b$

(n) $\forall\Diamond b$

(o) $\exists\Box(\neg a \vee \neg b)$

(p) $\forall\Box(\neg a \vee \neg b)$

(q) $\exists a \text{ U } b$

(r) $\forall a \text{ U } b$

(s) $\exists a \text{ U } (\exists b \text{ U } c)$

(t) $\forall a \text{ U } (\forall b \text{ U } c)$

Answer:

(k) Yes.

(l) Yes.

(m) Yes.

(n) Yes.

(o) No.

(p) No.

(q) Yes.

(r) Yes.

(s) Yes.

(t) Yes.

Marking scheme: 0.25 mark for each correct answer.

7 (3 marks)

(a) Let a and b be atomic propositions. Express "b happens never after a" in LTL.

Answer: $\Box(a \Rightarrow \bigcirc\Box\neg b)$.

Marking scheme: 0.5 mark for the initial $\Box$. 0.5 mark for the $a \Rightarrow$. 0.5 mark for $\bigcirc\Box\neg b$.

(b) Let a be an atomic proposition. Express "If a ever happens, then it won't keep happening forever" in CTL.

Answer: $\forall\Box(a \Rightarrow \forall\Diamond\neg a)$.

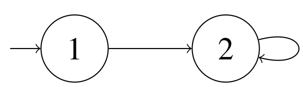
Marking scheme: 0.5 mark for the initial $\forall\Box$. 0.5 mark for the $a \Rightarrow$. 0.5 mark for $\forall\Diamond\neg a$.

8 (4 marks)

Let f and g be arbitrary LTL formulas. Which of the following equivalences hold? If the equivalence holds, give a proof. If the equivalence does not hold, provide a transition system (and a specific choice for f and g) for which one of the two LTL formulas holds and the other LTL formula does not hold.

(a) $\bigcirc(f \text{ U } g) \equiv (f \text{ U } \bigcirc g) \vee g$

Answer: The formulas are not equivalent. For f and g we choose the atomic propostions a and b, respectively. We consider the following transition system.



and the following labelling function

$$\begin{aligned} \ell(1) &= \emptyset \\ \ell(2) &= \{b\} \end{aligned}$$

This transition system satisfies $\bigcirc(a \text{ U } b)$ since state 2 has label b. However, the transition system does not satisfy $(a \text{ U } \bigcirc b) \vee b$ since state 1 is neither labelled a nor b.

Marking scheme: 0.25 mark for choosing appropriate choices for f and g. 0.5 mark for an appropriate transition system. 0.25 mark for arguing why the transition satisfies the one property but not the other.

(b) $\bigcirc\Diamond f \equiv \Diamond \bigcirc f$

Answer: The formulas are equivalent. Let TS be an arbitrary transition system. Let p be a path starting in an initial state. Then

$$\begin{aligned}
&p \models \bigcirc\Diamond f\\
\text{iff } &p[1..] \models \Diamond f\\
\text{iff } &\exists i \geq 0 : p[1..][i..] \models f\\
\text{iff } &\exists i \geq 0 : p[(i+1)..] \models f\\
\text{iff } &\exists i \geq 0 : p[i..][1..] \models f\\
\text{iff } &\exists i \geq 0 : p[i..] \models \bigcirc f\\
\text{iff } &p \models \Diamond \bigcirc f
\end{aligned}$$

Marking scheme: 0.25 mark for expanding the definition of $\bigcirc$. 0.25 mark for expanding the definition of $\Diamond$. 0.5 mark for the remainder of the proof.

Let f and g be arbitrary CTL formulas. Which of the following equivalences hold? If the equivalence holds, give a proof. If the equivalence does not hold, provide a transition system (and a specific choice for f and g) for which one of the two CTL formulas holds and the other CTL formula does not hold.

(c) $\exists(f \text{ U } g) \equiv g \vee (f \wedge \exists \bigcirc \exists(f \text{ U } g))$

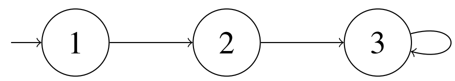
Answer: The formulas are equivalent. Let TS be an arbitrary transition system. Let s be an initial state. Then

$$\begin{aligned}
&s \models \exists(f \text{ U } g)\\
\text{iff } &\exists p \in \mathit{Paths}(s) : p \models f \text{ U } g\\
\text{iff } &\exists p \in \mathit{Paths}(s) : \exists i \geq 0 : p[i] \models g \wedge \forall 0 \leq j < i : p[j] \models f\\
\text{iff } &\exists p \in \mathit{Paths}(s) : p[0] \models g \vee (\exists i \geq 1 : p[i] \models g \wedge (p[0] \models f \wedge \forall 1 \leq j < i : p[j] \models f))\\
\text{iff } &\exists p \in \mathit{Paths}(s) : p[0] \models g \vee (p[0] \models f \wedge (\exists i \geq 1 : p[i] \models g \wedge \forall 1 \leq j < i : p[j] \models f))\\
\text{iff } &s \models g \vee (s \models f \wedge (\exists p \in \mathit{Paths}(s) : \exists i \geq 1 : p[i] \models g \wedge \forall 1 \leq j < i : p[j] \models f))\\
\text{iff } &s \models g \vee (s \models f \wedge (\exists p \in \mathit{Paths}(s) : \exists i \geq 0 : p[i+1] \models g \wedge \forall 0 \leq j < i : p[j+1] \models f))\\
\text{iff } &s \models g \vee (s \models f \wedge (\exists p \in \mathit{Paths}(s) : \exists i \geq 0 : p[1][i] \models g \wedge \forall 0 \leq j < i : p[1][j] \models f))\\
\text{iff } &s \models g \vee (s \models f \wedge (\exists p \in \mathit{Paths}(s) : \exists q \in \mathit{Paths}(p[1]) : \exists i \geq 0 : q[i] \models g \wedge \forall 0 \leq j < i : q[j] \models f))\\
\text{iff } &s \models g \vee (s \models f \wedge (\exists p \in \mathit{Paths}(s) : \exists q \in \mathit{Paths}(p[1]) : q \models f \text{ U } g))\\
\text{iff } &s \models g \vee (s \models f \wedge (\exists p \in \mathit{Paths}(s) : p[1] \models \exists(f \text{ U } g)))\\
\text{iff } &s \models g \vee (s \models f \wedge (\exists p \in \mathit{Paths}(s) : p \models \bigcirc\exists(f \text{ U } g)))\\
\text{iff } &s \models g \vee (s \models f \wedge s \models \exists \bigcirc \exists(f \text{ U } g))\\
\text{iff } &s \models g \vee (f \wedge \exists \bigcirc \exists(f \text{ U } g))
\end{aligned}$$

Marking scheme: 0.25 mark for expanding the definition of $\exists$. 0.25 mark for expanding the definition of U. 0.25 mark for expanding the definition of $\bigcirc$. 0.25 mark for the remainder of the proof.

(d) $\forall(f \text{ U } g) \equiv f \wedge \forall\Diamond g$

Answer: The formulas are not equivalent. For f and g we choose the atomic propostions a and b, respectively. We consider the following transition system.



and the following labelling function

$$\begin{array}{rcl} \ell(1) & = & \{a\} \\ \ell(2) & = & \emptyset \\ \ell(3) & = & \{b\} \end{array}$$

This transition system satisfies $a \wedge \forall\Diamond b$ since state 1 has label a and state 3 has label b. However, the transition system does not satisfy $\forall(a \text{ U } b)$ since state 2 is not labelled b.

Marking scheme: 0.25 mark for choosing appropriate choices for f and g. 0.5 mark for an appropriate transition system. 0.25 mark for arguing why the transition satisfies the one property but not the other.

9 (1 mark)

We extend the syntax of CTL with the operator $\exists^!$. Let g be a path formula. Then the state formula $\exists^! g$ captures that there exists a *unique* path satisfying g. Formally,

$$s \models \exists^! g \text{ iff } \exists p \in \mathit{Paths}(s) : p \models g \wedge \forall q \in \mathit{Paths}(s) : q \neq p \Rightarrow q \not\models g$$

Recall that $\mathit{Sat}(f) = \{\, s \in S \mid s \models f \,\}$. Express $\mathit{Sat}(\exists^! \bigcirc f)$ in terms of $\mathit{Sat}(f)$.

Answer: $\mathit{Sat}(\exists^! \bigcirc f) = \{\, s \in S \mid |\mathit{succ}(s) \cap \mathit{Sat}(f)| = 1 \,\}$.

Marking scheme: 0.5 mark for considering successors. 0.5 mark for limiting to one element.

10 (2 marks)

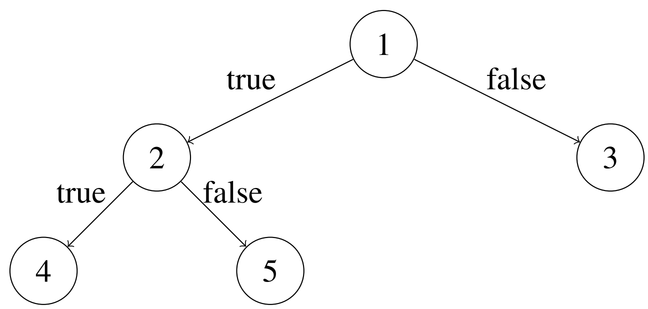
Consider the following method.

```
public void example(int x, int y) {
  if (x >= 0) {
    if (y > x) {
```

```
      System.out.println("y is positive");
    } else {
      System.out.println("x is nonnegative");
    }
  } else {
    System.out.println("x is nonnegative");
  }
}
```

Assume that both x and y are symbolic. Draw the symbolic execution tree and for each node of the tree provide the corresponding path condition.

Answer: The symbolic execution tree looks like the following.



The corresponding path conditions are the following.

state	path condition
1	true
2	$x \geq 0$
3	$x < 0$
4	$x \geq 0 \wedge y > x$
5	$x \geq 0 \wedge \neg(y > x)$

Marking scheme: 1 mark for a tree of the right shape. 0.2 mark for each correct path condition.

11 (1 mark)

Give an estimate of the (yearly world-wide) cost of debugging.

Answer: US $312 billion.

Marking scheme: 1 mark for something in the order of 100's of billions of US $.

12 (1 bonus mark)

(a) From left to right (circle the correct answer):

- Dijkstra, Clarke, Emerson, Sifakis, Lamport, Pnueli
- Sifakis, Clarke, Pnueli, Dijkstra, Lamport, Emerson
- Lamport, Clarke, Emerson, Sifakis, Dijkstra, Pnueli
- Dijkstra, Clarke, Pnueli, Sifakis, Lamport, Emerson
- Lamport, Pnueli, Emerson, Sifakis, Dijkstra, Clarke

Answer: The last combination.

Marking scheme: 0.5 mark for the correct answer.

(b) Which award have they all won?

Answer: The Turing award.

Marking scheme: 0.5 for the correct answer.

Reader class

```
public class Reader extends Thread {
  private Database database;

  public Reader(Database database) {
    super();
    this.database = database;
  }

  public void run() {
    this.database.read();
  }
}
```

Writer class

```
public class Writer extends Thread {
  private Database database;

  public Writer(Database database) {
    super();
    this.database = database;
  }

  public void run() {
    this.database.write();
  }
}
```

Database class

```
public class Database {
  private boolean writing;
  private int readers;

  public Database() {
    this.writing = false;
    this.readers = 0;
```

```
  }

  public void read() {
    synchronized(this) {
      while (this.writing) {
        try {
          this.wait();
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
      }
    }
    // read
    synchronized(this) {
      this.readers--;
      if (this.readers == 0) {
        this.notifyAll();
      }
    }
  }

  public void write() {
    synchronized(this) {
      while (this.writing || this.readers > 0) {
        try {
          this.wait();
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
      }
      this.writing = true;
    }
    // write
    synchronized(this) {
      this.writing = false;
      this.notifyAll();
    }
  }
}
```

Definitions

Definition 1. A transition system is a tuple $\langle S, L, I, \rightarrow, \ell \rangle$ consisting of

- a set S of states,
- a set L of labels,
- a set $I \subseteq S$ of initial states,
- a transition relation $\rightarrow \subseteq S \times S$, and
- a labelling function $\ell : S \rightarrow 2^L$.

Definition 2. Linear temporal logic (LTL) is defined by the grammar

$$f ::= a \mid f \wedge f \mid \neg f \mid \bigcirc f \mid f \ \mathrm{U} \ f$$

where $a \in L$.

We use the following syntactic sugar.

$$\begin{array}{rcll} f \vee g & = & \neg(\neg f \wedge \neg g) & \\ \text{true} & = & a \vee \neg a & \\ \Diamond f & = & \text{true U } f & \text{(eventually } f\text{)} \\ \Box f & = & \neg \Diamond \neg f & \text{(always } f\text{)} \end{array}$$

Definition 3. $\mathit{Paths}(s)$ is the set of (execution) paths starting in state s. Let $p \in \mathit{Paths}(s)$ and $n \geq 0$. Then $p[n]$ is the $(n+1)^{\text{th}}$ state of the path p and $p[n..]$ is the suffix of p starting with the $(n+1)^{\text{th}}$ state.

Definition 4. The relation $\models$ is defined by

$$\begin{array}{rll} p \models a & \text{iff} & a \in \ell(p[0]) \\ p \models f \wedge g & \text{iff} & p \models f \text{ and } p \models g \\ p \models \neg f & \text{iff} & \text{not}(p \models f) \\ p \models \bigcirc f & \text{iff} & p[1..] \models f \\ p \models f \ \mathrm{U} \ g & \text{iff} & \exists i \geq 0 : p[i..] \models g \text{ and } \forall 0 \leq j < i : p[j..] \models f \end{array}$$

and

$$\langle S, L, I, \rightarrow, \ell \rangle \models f \text{ iff } \forall s \in I : \forall p \in \mathit{Paths}(s) : p \models f$$

Definition 5. Computation tree logic (CTL) is defined as follows. The state formulas are defined by

$$f ::= a \mid f \wedge f \mid \neg f \mid \exists g \mid \forall g$$

where $a \in L$. The path formulas are defined by

$$g ::= \bigcirc f \mid f \ \mathrm{U} \ f$$

Definition 6. The relation $\models$ is defined by

$$\begin{array}{rcl} s \models a & \text{iff} & a \in \ell(s) \\ s \models f \wedge g & \text{iff} & s \models f \text{ and } s \models g \\ s \models \neg f & \text{iff} & \text{not}(s \models f) \\ s \models \exists g & \text{iff} & \exists p \in \mathit{Paths}(s) : p \models g \\ s \models \forall g & \text{iff} & \forall p \in \mathit{Paths}(s) : p \models g \end{array}$$

and

$$\begin{array}{rcl} p \models \bigcirc f & \text{iff} & p[1] \models f \\ p \models f \mathrm{~U~} g & \text{iff} & \exists i \geq 0 : p[i] \models g \text{ and } \forall 0 \leq j < i : p[j] \models f \end{array}$$

and

$$\langle S, L, I, \rightarrow, \ell\rangle \models f \text{ iff } \forall s \in I : s \models f$$

Definition 7. The satisfaction set $\mathit{Sat}(f)$ is defined by

$$\mathit{Sat}(f) = \{\, s \in S \mid s \models f \,\}.$$

Definition 8. LTL/CTL formulas f and g are equivalent, denoted $f \equiv g$, if $\langle S, L, I, \rightarrow, \ell\rangle \models f$ iff $\langle S, L, I, \rightarrow, \ell\rangle \models g$ for all transition systems $\langle S, L, I, \rightarrow, \ell\rangle$.