HAND IN EXAM
Answers recorded on examination paper

Queen's University
Faculty of Arts and Science
School of Computing
CISC/CMPE 422 and CISC 835
Final Examination
Tuesday, Dec 16, 2014
Instructor: J. Dingel

Student id (clearly printed please): _____

Instructions: This examination is **three hours** in length. You are permitted to have one 8.5"x11" data sheet with information on both sides. No other references, calculators, or aids are allowed.

Read all questions carefully. There are 5 questions in this examination, some with several parts. You must answer all questions. Write all your answers in this exam. An extra page is provided at the end for rough work. If you answer a question on a different page, you must indicate where the answer is to be found.

Important notes: Make sure your student id is clearly printed on this page. If the instructor is unavailable in the examination room and doubt exists as to the interpretation of any question, you are urged to submit your answer with a clear statement of any assumptions made.

Good luck!

For marking use only:

Question	Points
1	/15
2	/28
3	/18
4	/12
5	/33
Total	/106

Id: _____

Question 1: Propositional & Predicate Logic, 15 points

Each of the three parts below contains a predicate logic formula. For each formula, say whether it is satisfiable by writing "yes" or "no". If you answered "yes", write down a model in which the formula holds; if you answered "no", write down a model in which the negation of the formula holds. Remember to fully define your models by providing the domain and the interpretation of any function and predicate symbols used.

1. (5 points) $\big(\forall x.\exists y.P(x,y)\big) \ \wedge \ \big(\forall x.\neg P(x,x)\big)$

2. (5 points) $\big(\exists y.\forall x.P(x,y)\big) \ \wedge \ \big(\forall x.\neg P(x,x)\big)$

3. (5 points) $\big(\exists y.\forall x.P(f(x),y)\big) \ \wedge \ \big(\forall x.\neg P(x,x)\big)$

Id: _____

Question 2: (Alloy, 28 points)

To support the development of a scheduler for a concurrent system, the following Alloy model M is constructed:

```
sig CPU {}
sig Process {
   assigned : CPU
}
sig User {
   owns : set Process
}
```

Note how every instance of M represents an assignment of processes to CPUs.

1. (3 points) In the space below, draw the graphical representation of M (i.e., the "metamodel" or "class diagram" corresponding to M). Make sure you also include any multiplicity constraints expressed in M.

2. (4 points) Consider the following predicate

   ```
   pred Show() {
      some Process
      some User
   }
   ```

   Draw an instance of M that also satisfies all the constraints in `Show`, that is, draw an instance that could be created by Alloy in response to executing the command `run show for 3` on M.

Id: _____

Question 2: (Alloy, continued)

For your convenience, the Alloy model M from the previous page is repeated here.

```
sig CPU {}
sig Process {
   assigned : CPU
}
sig User {
   owns : set Process
}
```

3. (4 points) We say that an assignment of processes to CPUs is *conflict-free*, if it avoids placing two processes on the same CPU if they belong to different users. Complete the predicate `ConflictFree()` below such that it characterizes conflict-free assignments, i.e., `ConflictFree()` must hold in an instance if and only if that instance represents a conflict-free assignment.

   ```
   pred ConflictFree() {
   ```

4. (2 points) Write down the Alloy command that can be used to generate conflict-free instances of M.

5. (2 points) Write down the Alloy command that can be used to generate non-conflict-free instances of M.

6. (4 points) An assignment of processes to CPUs is called *optimal*, if and only if processes only share CPUs with processes that belong to the same user. Complete the predicate `Optimal()` below such that it holds in an instance if and only if that instance is optimal.

   ```
   pred Optimal() {
   ```

Id: _____

Question 2: (Alloy, continued)

For your convenience, the Alloy model M from the previous page is repeated here.

```
sig CPU {}
sig Process {
   assigned : CPU
}
sig User {
   owns : set Process
}
```

7. (5 points) A conflict-free assignment is not necessarily optimal. Write down an instance of M that is conflict-free, but not optimal.

8. (4 points) Complete the fact `ConfictFreedomImpliesOptimality` below such that, when added to M, it ensures that every conflict-free instance of M also is optimal.

```
fact ConflictFreedomImpliesOptimality {
```

Id: _____

Question 3: CTL, 18 points

For each of the following three pairs of formulas φ, φ', decide if they are equivalent. If they are equivalent, write **"Yes"**; if they are not equivalent, write **"No"**, draw a Kripke structure (i.e., finite state machine) M such that one formula holds in M, but not the other, and clearly indicate which formula holds in M.

When drawing M, clearly show the initial state and which atomic propositions hold in which states; also, remember that the transition relation of a Kripke structure is total.

1. (6 points) $\varphi = \mathbf{EG}\ (p \vee q)$ and $\varphi' = (\mathbf{EG}\ p) \vee (\mathbf{EG}\ q)$

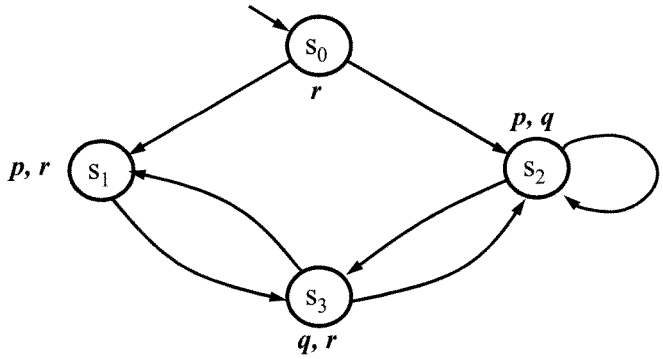
2. (6 points) $\varphi = \mathbf{AX}\ \mathbf{AF}\ p$ and $\varphi' = \mathbf{AF}\ p$

3. (6 points) $\varphi = \mathbf{A}[\neg p\ \mathbf{U}\ p]$ and $\varphi' = \mathbf{AF}\ p$

Id: _____

Question 4: Model checking, 12 points

Consider the following graphical representation of a Kripke structure (i.e., finite state machine) M.



For each of the following four CTL formulas φ decide whether the formula holds in machine M defined above. If your answer is **"No"**, that is, φ does not hold in M, then give a counter example, that is, an execution path in M illustrating the **violation** of φ. Should the counter example be infinite, use periods "..." and parentheses to indicate the sequence of states that is repeated infinitely often.

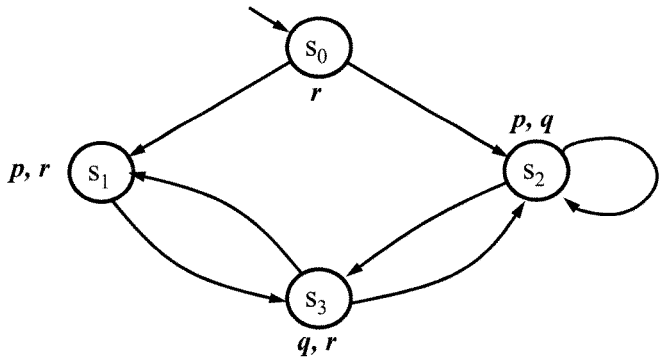
1. (3 points) **AG** $(q \rightarrow p)$

2. (3 points) **AF** r

Id: _____

Question 4 (continued)

The Kripke structure M from the previous page is repeated here for your convenience.



3. (3 points) $\mathbf{E}[p\ \mathbf{U}\ \mathbf{AF}\ q]$

4. (3 points) $\mathbf{AG}(p \rightarrow \mathbf{AF}\ r)$

Id: _____

Question 5: SMV, 33 points

Consider the following code defining an NuSMV program `main`:

```
MODULE P1
VAR
  x :  0..9;
ASSIGN
  init(x) := 0;
  next(x) := case
               x=2 :  0;
               TRUE: x+1;
             esac;
```

```
MODULE P2(x)
VAR
  y :  {a, b, c};
ASSIGN
  init(y) := a;
  next(y) := case
               x=2 & y=a :  b;
               x=2 & y=b :  {a,b};
               TRUE : y;
             esac;
```

```
MODULE main
VAR
  p1 :  P1;
  p2 :  P2(p1.x);
```

1. (9 points) Draw the finite state machine M that is defined by the code above. Represent a single state s of M by a pair (x, y) where x is the value of `x` in process `p1` and y is the value of `y` in process `p2`. For instance, the initial state of M is represented as $(0, a)$. Your drawing should clearly indicate the initial states of M, the reachable states of M, and the transition relation of M. You don't need to show the labelling function.

Question 5 (continued)

Id: _____

```
MODULE P1
VAR
  x :  0..9;
ASSIGN
  init(x) := 0;
  next(x) := case
               x=2 :  0;
               TRUE: x+1;
             esac;
```

```
MODULE P2(x)
VAR
  y :  {a, b, c};
ASSIGN
  init(y) := a;
  next(y) := case
               x=2 & y=a :  b;
               x=2 & y=b :  {a,b};
               TRUE : y;
             esac;
```

```
MODULE main
VAR
  p1 :  P1;
  p2 :  P2(p1.x);
```

2. (4 points) Let M be the Kripke structure (i.e., Finite State Machine) defined by program `main` (repeated above for your convenience). Beginning from the initial state of the machine M, draw the computation tree T of M to a depth of three, i.e., in your tree, there should be three edges on every path from the root to a leaf.

Id: _____

Question 5 (continued)

3. (20 points) For each of the following properties φ_1 through φ_5, express it formally in CTL and decide whether or not it is satisfied by program `main` given in Part 1) of this question by writing **"Yes"** or **"No"**. No justification necessary.

   1. (4 points) φ_1: *"For all states along every path, if* `p1.x` *is* `2`*, then* `p2.y` *is* `b`*."*

      - φ_1 in CTL:

      - φ_1 true in `main`?:

   2. (4 points) φ_2: *"There exists a path along which* `p2.y` *is always equal to* `a`*"*

      - φ_2 in CTL:

      - φ_2 true in `main`?:

   3. (4 points) φ_3: *"Along every path it is always the case that when* `p2.y` *is equal to* `b`*, then* `p2.y` *will always eventually be equal to* `a`*.*

      - φ_3 in CTL:

      - φ_3 true in `main`?:

   4. (4 points) φ_4: *"It is not the case that: along all paths* `p2.y` *is equal to* `b` *eventually and* `p2.y` *is equal to* `a` *until then."*

      - φ_4 in CTL:

      - φ_4 true in `main`?:

   5. (4 points) φ_5: *"For all states along every path, if a state s only has (immediate) successors in which* `p1.x` *is equal to* `0`*, then* `p2.y` *is equal to* `a` *in s".*

      - φ_5 in CTL:

      - φ_5 true in `main`?:

Id: _____

Scratch sheet: