HAND IN EXAM
Answers recorded on examination paper

Queen's University
Faculty of Arts and Science
School of Computing
CISC422
Final Examination
Tuesday, Dec 10, 2013
Instructor: S. Grant

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. Extra pages are 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	/12
2	/36
3	/18
4	/16
5	/27
Total	/109

Id: ____________________

Question 1: (Predicate Logic, 12 points)

Consider the following two predicate logic formulas

$$\begin{aligned} \varphi_1 &= \forall x.\big(P(x) \to \neg\exists y.Q(f(x,y))\big) \\ \varphi_2 &= \forall x.\neg\exists y.\big(P(x) \to Q(f(x,y))\big) \end{aligned}$$

where x and y are variables, P and Q are predicate symbols of arity 1, and f is a function symbol of arity 2.

φ_1 and φ_2 are not equivalent. More precisely, φ_1 does not imply φ_2. Find a model $\mathcal{M}$, such φ_1 holds in $\mathcal{M}$, but φ_2 does not hold in $\mathcal{M}$. Make sure you provide a **complete** definition of your model $\mathcal{M}$.

Id: ____________________

Question 2: (Alloy, 36 points)

Consider the following Alloy specification of a source code repository (roughly similar to cvs, subversion, or git).

```
module CodeRepository
   sig Repo {
     files:  set File,
     commits:  set Commit,
     users:  set Person
   }
   sig Commit {
     id:  Int,
     files:  set File
     submitter:  Person
   }
   sig File { }
   sig Person { }
   pred exists[c:  Commit, r:  Repo] {
     c in r.commits
   }
```

```
pred addCommit[r, r':  Repo, c:  Commit] {
  r'.commits = r.commits + c
}
pred showAddCommit {
  some r, r':  Repo | some c:  Commit |
        !exists[c, r] && addCommit[r, r', c]
}
pred Show {
  #Repo = 1
  #Commit = 2
  #Person = 2
}
fact RepoFacts {
  all c:  Commit | some c.files
  all c1, c2:  Commit | c1.id=c2.id => c1=c2
  all p:  Person | some r:  Repo | p in r.users
}
```

1. (10 points) In the space below, draw *one* state generated by the model corresponding to the Alloy specification above after the command **run Show for 3** is executed. Make sure your diagram names nodes and edges appropriately.

2. (6 points) Briefly and informally, but nonetheless precisely, explain the properties enforced by each line in *RepoFacts* (three in total).

```
module CodeRepository
   sig Repo {
     files: set File,
     commits: set Commit,
     users: set Person
   }
   sig Commit {
     id: Int,
     files: set File
     submitter: Person
   }
   sig File { }
   sig Person { }
   pred exists[c: Commit, r: Repo] {
     c in r.commits
   }
```

```
pred addCommit[r, r': Repo, c: Commit] {
  r'.commits = r.commits + c
}
pred showAddCommit {
  some r, r': Repo | some c: Commit |
        !exists[c, r] && addCommit[r, r', c]
}
pred Show {
  #Repo = 1
  #Commit = 2
  #Person = 2
}
fact RepoFacts {
  all c: Commit | some c.files
  all c1, c2: Commit | c1.id=c2.id => c1=c2
  all p: Person | some r: Repo | p in r.users
}
```

3. (5 points) Complete the fact below such that it expresses the statement: *"Each file in the model is associated with at least one commit"*.

```
fact Q2.3 {
```

4. (5 points) Briefly and informally, but nonetheless precisely, explain whether or not the *addCommit* predicate appropriately simulates the operation in which a commit is added to the code repository. If the operation is correctly simulated, why is the code sufficient? And if not, what is missing from either *addCommit* or *showAddCommit*?

```
module CodeRepository
   sig Repo {
     files: set File,
     commits: set Commit,
     users: set Person
   }
   sig Commit {
     id: Int,
     files: set File
     submitter: Person
   }
   sig File { }
   sig Person { }
   pred exists[c: Commit, r: Repo] {
     c in r.commits
   }
```

```
pred addCommit[r, r': Repo, c: Commit] {
  r'.commits = r.commits + c
}
pred showAddCommit {
  some r, r': Repo | some c: Commit |
        !exists[c, r] && addCommit[r, r', c]
}
pred Show {
  #Repo = 1
  #Commit = 2
  #Person = 2
}
fact RepoFacts {
  all c: Commit | some c.files
  all c1, c2: Commit | c1.id=c2.id => c1=c2
  all p: Person | some r: Repo | p in r.users
}
```

5. (10 points) Complete the assertion below such that it expresses: *"Every commit associated with a particular source code repository must be submitted by a person in that repository's user list".*

   `assert Q2.6 {`

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"** and draw a Kripke structure (i.e., finite state machine) M such that one formula holds in M, but not the other. When drawing M, clearly indicate the initial state and which atomic propositions hold in which states.

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

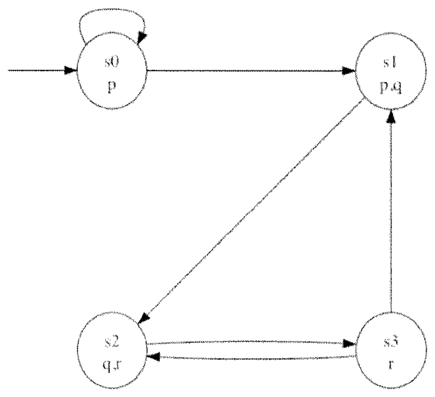
2. (6 pts) $\varphi = (\mathbf{AF}\ p) \wedge (\mathbf{AF}\ q)$ and $\varphi' = (\mathbf{AF}\ p \wedge q)$

3. (6 pts) $\varphi = \mathbf{E}[p\ \mathbf{U}\ q] \wedge \mathbf{E}[p\ \mathbf{U}\ r]$ and $\varphi' = \mathbf{E}[p\ \mathbf{U}\ q \wedge r]$

Id: ____________________

Question 4: (Model checking, 16 points)

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

$M =$ 

1. (4 pts) Define M textually as a 4-tuple $M = (S, S_0, R, L)$.

2. (4 pts) Beginning from state s_0, unwind M into its corresponding computation tree T. Draw T to a depth of 5. More precisely, you must show all computation paths of M starting at s_0 up to length 5, where the length of a path is the number of edges on it.

Id: ____________________

Question 4: (Model checking, continued)

3. (8 pts) For each of the following CTL formulas φ decide whether the formula holds in machine M defined on the previous page. 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. Remember that paths are always infinite. Use periods "..." and parentheses to indicate that a sequence of states is repeated infinitely often.

   (a) $\mathbf{AG}\big(p \to (q \vee r)\big)$

   (b) $\mathbf{AG}(r \to \mathbf{AF}q)$

   (c) $\mathbf{AG}\big(p \to \mathbf{AX}\ \mathbf{AX}(p \vee r)\big)$

   (d) $\mathbf{A}\big[(p \vee q)\ \mathbf{U}\ \mathbf{EG}r\big]$

Question 5: (SMV, 27 pts)

Id: ____________________

Consider the following SMV code.

```
MODULE main
VAR
  state:  {s0, s1, s2};
ASSIGN
  init(state) := s0;
  next(state) :=
    case
      state = s0:  s1;
      state = s1:  {s0, s2};
      state = s2:  {s0, s2};
    esac
DEFINE
  p := (state = s1) | (state = s2);
  q := state = s2;
  r := state = s2;
```

1. (4 pts) Provide a 4-tuple $M = (S, S_0, R, L)$ that corresponds to the SMV code above.

Question 5: (SMV, continued)

Id: ______________________

2. (4 pts) Using the 4-tuple M from the previous question as a guide, draw the Kripke structure M that is defined by the SMV code above. Your drawing should clearly indicate the initial states of M, the reachable states of M, the transition relation of M, and the state labels.

3. (4 pts) Beginning from state s_0, unwind M into its corresponding computation tree T. Draw T to a depth of 4. More precisely, you must show all computation paths of M starting at s_0 up to length 4, where the length of a path is the number of edges on it.

Id: ____________________

Question 5: (SMV, continued)

4. (15 pts) For each of the following properties φ_1 through φ_5, express it formally in CTL and decide whether or not it is true in your finite state machine M given in Part 1 of this question by writing **"Yes"** or **"No"**. No justification necessary.

   1. (5 pts) φ_1: "Along every path it is always the case that if p is true, then in some later state, r will be true".
      - φ_1 in CTL:

      - φ_1 true in M?:

   2. (5 pts) φ_2: "For all states s along every path, if p is true, then s has a successor s' in which q is true, and s' has a successor in which r is true".
      - φ_2 in CTL:

      - φ_2 true in M?:

   3. (5 pts) φ_3: "There exists at least one path in which q is false until r is true".
      - φ_3 in CTL:

      - φ_3 true in M?:

Id: ____________________

Scratch sheet: