Student id: ____________________

HAND IN
Answers recorded on examination paper

QUEEN'S UNIVERSITY FINAL EXAMINATION

FACULTY OF ARTS AND SCIENCE
SCHOOL OF COMPUTING

CISC/CMPE 422 and CISC 835

Instructor: J. Dingel

Wednesday, Dec 12, 2018

INSTRUCTIONS TO STUDENTS:

This examination is 3 HOURS in length.
Please answer all questions in the exam.

The following aids are allowed:
One 8.5"x 11" data sheet

Put your student number on all pages including this one (see upper right corner).
GOOD LUCK!

PLEASE NOTE:
Proctors are unable to respond to queries about the interpretation of exam questions.
Do your best to answer exam questions as written.

For marking use only:

Q1	/18
Q2	/38
Q3	/18
Q4	/18
Q5	/40
Total	/132

Student id: ____________________

Question 1: Predicate Logic (18 points total)

Let $\mathcal{V}$, $\mathcal{F}$, and $\mathcal{P}$ denote the following sets of variables, function symbols, and predicate symbols, respectively: $\mathcal{V} = \{x, y\}$, $\mathcal{F} = \{\}$, and $\mathcal{P} = \{P\}$ where P has arity one. Ensure that all formulas that you write in this question are well-formed formulas over these variables and symbols.
For parts a), b), and c) below, let ψ be the formula $\forall x.P(x)$.

a) Is it possible to find a formula φ_a such that $\varphi_a \wedge \psi$ is satisfiable? If so, write down that formula φ_a in the space below. If not, write *"Impossible"*.

b) Is it possible to find a formula φ_b such that $\varphi_b \wedge \psi$ is unsatisfiable? If so, write down that formula φ_b in the space below. If not, write *"Impossible"*.

c) Is it possible to find a formula φ_c such that $\varphi_c \wedge \psi$ is valid? If so, write down that formula φ_c in the space below. If not, write *"Impossible"*.

d) Is it possible to find a formula φ_d such that $\forall x.\big(P(x) \to \varphi_d\big)$ is satisfiable? If so, write down that formula φ_d in the space below. If not, write *"Impossible"*.

e) Is it possible to find a formula φ_e such that $\forall x.\big(P(x) \to \varphi_e\big)$ is unsatisfiable? If so, write down that formula φ_e in the space below. If not, write *"Impossible"*.

f) Is it possible to find a formula φ_f such that $\forall x.\big(P(x) \to \varphi_f\big)$ is valid? If so, write down that formula φ_f in the space below. If not, write *"Impossible"*.

Student id: ____________________

Question 2: Alloy (38 points total)

Consider the following partial Alloy specification M for a plugin architecture (akin to the one in Eclipse):

```
module Plugins

sig ExtensionPoint {}

sig Plugin {
   requires : set Plugin,
   exPoints : set ExtensionPoint,
   plugsInto : set ExtensionPoint,
   extends : set Plugin }
```

a) (4 points) In the space below, draw an instance of the Alloy specification above. Your instance should satisfy all the constraints expressed in the specification and contain non-empty (interpretations of the) **requires** and **exPoints** relations. In your instance, clearly label every object with the signature (i.e., type) and every link (i.e., edge) with the relation (i.e., attribute) they belong to.

b) (2 points) What is the smallest scope in which the Alloy analyzer would be able to produce your instance?

c) (4 points) For each of the following Alloy expressions and formulas, determine which value the expression or formula evaluates to in the instance you have drawn above and write down that value.

i) `Plugin & ExtensionPoint` evaluates to:

ii) `^extends.^extends = ^extends` evaluates to:

Student id: ____________________

Question 2: Alloy (continued)

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

```
module Plugins

sig ExtensionPoint {}

sig Plugin {
   requires : set Plugin,
   exPoints : set ExtensionPoint,
   plugsInto : set ExtensionPoint,
   extends : set Plugin }
```

Using the Alloy specification above, express each of the following invariants formally in Alloy.

d) (4 points) *"The* `requires` *relationship is acyclic."*

e) (4 points) *"Every extension point belongs to exactly one plugin."*

f) (4 points) *"A plugin p1 extends plugin p2 if and only if p1 plugs into at least one of the extension points of p2."*

g) (4 points) *"A plugin that requires some other plugin cannot also extend that plugin."*

Student id: ________________

Question 2: Alloy (continued)

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

```
module Plugins

sig ExtensionPoint {}

sig Plugin {
    requires : set Plugin,
    exPoints : set ExtensionPoint,
    plugsInto : set ExtensionPoint,
    extends : set Plugin }
```

h) (12 points) In the table below, the rows and columns are labeled with a sequence of Alloy formulas φ_1 through φ_5. For each cell (φ_i, φ_j) in the table, determine whether or not φ_i and φ_j are equivalent, i.e., whether or not any instance of M satisfies φ_i if and only if it satisfies φ_j. Write *"Yes"* into cell (φ_i, φ_j), if φ_i and φ_j are equivalent. Write *"No"*, otherwise. For all formulas, assume that p denotes a plugin, i.e., that `p : Plugin`.

	`no p.extends`	`no ~extends.p`	`p.^extends in none`	`all q:Plugin | q !in p.extends`	`no q:Plugin | q->p in extends`
`no p.extends`					
`no ~extends.p`					
`p.^extends in none`					
`all q:Plugin | q !in p.extends`					
`no q:Plugin | q->p in extends`					

Student id: ____________________

Question 3: CTL (18 points total)

Consider the three pairs of non-equivalent formulas φ_i, φ'_i below. For each pair, find a Kripke structure that distinguishes them. More precisely, for each pair, draw a Kripke structure M_i such that one formula holds in M_i, but not the other. **Important:** When drawing M_i, make sure that you clearly indicate (1) the initial state of M_i, (2) which atomic propositions occuring φ_i and φ'_i hold in which states of M_i, and (3) which of the two formulas holds in M_i. Also, remember that the transition relation of a Kripke structure is total.

a) (6 points) $\varphi_1 = \mathbf{AG}\ p$ and $\varphi'_1 = \mathbf{AG}\ \mathbf{EX}\ p$

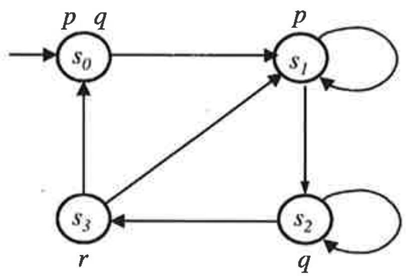
b) (6 points) $\varphi_2 = q \wedge \mathbf{E}[p\ \mathbf{U}\ \mathbf{AG}\ q]$ and $\varphi'_2 = q \wedge \mathbf{E}[p\ \mathbf{U}\ q]$

c) (6 points) $\varphi_3 = \mathbf{EF}\ p$ and $\varphi'_3 = \mathbf{EF}\ \mathbf{EX}\ p$

Student id: ____________________

Question 4: Model checking (18 points total)

Consider the following graphical representation of a Kripke structure M.



For each of the following six CTL formulas φ decide whether the formula holds in M. If your answer is *"No"*, that is, φ does not hold in M, then give a counter example, that is, a sequence of states corresponding to an execution path in M illustrating the *violation* of φ. Remember that some counter examples are infinite. To show infinite counter examples enclose the sequence of states that are repeated in parentheses. E.g., the sequence $s_0(s_1 s_2 s_3)$ represents an execution that starts with s_0 after which states s_1, s_2 and s_3 are repeated forever in this order.

a) $\mathbf{AG}\ \mathbf{EX}\ p$

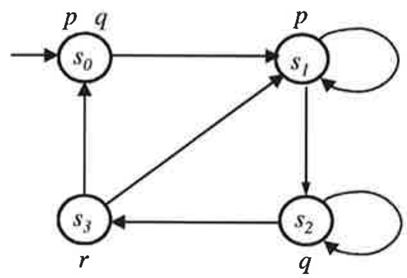
b) $\mathbf{AG}(r \to \mathbf{AX}\ (p \land q))$

c) $\mathbf{EF}\ \mathbf{EG}\ q$

Student id: ____________________

Question 4: Model checking (continued)

For your convenience, the Kripke structure M from the previous page is repeated here.



d) $\mathbf{A}[p\ \mathbf{U}\ (q \wedge \mathbf{AX}\ r)]$

e) $\mathbf{AG}\ \mathbf{AF}\ q$

f) $\neg\mathbf{EF}\ (p \wedge r)$

Student id: ____________________

Question 5: SMV (40 points total)

Consider the following SMV program.

```
MODULE P1
VAR
  x :  {a, b, c};
ASSIGN
  init(x) := a;
  next(x) := case
                 x=a :  {b, c};
                 TRUE : a;
             esac;


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

```
MODULE P2(x)
VAR
  y :  {0, 1, 2};
ASSIGN
  init(y) := 0;
  next(y) := case
                 x=b & y=0 :  1;
                 x=b & y=1 :  2;
                 x=b & y=2 :  0;
                 TRUE : y;
             esac;
```

a) (10 points) Draw the Kripke structure M that is defined by the SMV program 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 $(a, 0)$. 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.

Student id: ______________________

Question 5: SMV (continued)

For your convenience, the SMV program from the previous page is repeated here.

```
MODULE P1
VAR
  x :  {a, b, c};
ASSIGN
  init(x) := a;
  next(x) := case
                 x=a :  {b, c};
                 TRUE : a;
             esac;
```

```
MODULE P2(x)
VAR
  y :  {0, 1, 2};
ASSIGN
  init(y) := 0;
  next(y) := case
                 x=b & y=0 :  1;
                 x=b & y=1 :  2;
                 x=b & y=2 :  0;
                 TRUE : y;
             esac;
```

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

b) (2 points) In the SMV program above, do processes p1 and p2 execute synchronously or asynchronously? Circle the correct answer.

Synchronously   Asynchronously

c) (2 points) Does the SMV program above have any unfair executions? Circle the correct answer.

No   Yes

d) (6 points) Suppose you want to check that in the SMV program above state $(c, 2)$ is reachable.

- Write down a CTL formula φ that you could use for this purpose.

  $\varphi =$

- What could you conclude if the SMV program satisfies φ, i.e., if NuSMV does not find a counter example to the formula φ? Circle the correct answer.

State $(c, 2)$ is reachable   State $(c, 2)$ is not reachable

Student id: ____________________

Question 5: SMV (continued)

e) (20 points) Each of the informally given properties φ_1 through φ_5 refers to the SMV program defined on the previous page. Express each of them formally in CTL. We use `p1.x` to refer to the value of variable `x` in process `p1`. Similarly for `p2.y`. Given a state s, a successor of s is a state that can be reached from s in one transition.

φ_1: *"Along every path it is always the case that if* `p1.x` *is* `b` *and* `p2.y` *is 0, then* `p2.y` *is 1 in all successor states"*

φ_1 in CTL:

φ_2: *"For all states s along every path, if* `p1.x` *is* `a` *in s and* `p2.y` *is 0 in s and* `p2.y` *is 1 in all successor states of the successor states of s, then* `p1.x` *is* `b` *in all successor states of s"*

φ_2 in CTL:

φ_3: *"Along every path,* `p1.x` *will be* `b` *eventually and* `p2.y` *will be 0 until then"*

φ_3 in CTL:

φ_4: *"There exists a path along which* `p2.y` *is always 0"*

φ_4 in CTL:

φ_5: *"In every state, along every path, if* `p2.y` *is 0, then* `p2.y` *is 0 or 1 in all successor states"*

φ_5 in CTL:

Student id: ____________________

Scratch sheet: