Name: _____

Student Number: _____

Signature: _____

THE UNIVERSITY OF NEW SOUTH WALES

COMP3153/9153
Algorithmic Verification

SAMPLE Final Exam

Term 1, 2020

Time Allowed: 2 Hours, plus 10 minutes reading time.

Total Number of Questions: 7

Answer **all** questions.

The questions **are not** of equal value.

You are permitted **two** hand-written, double-sided A4 sheets of notes.

Only write your answers on the provided booklets.

Your notes will be collected at the end of the exam.

Answers must be written in ink, with the exception of diagrams.

Drawing instruments or rules may be used.

Excessively verbose answers may lose marks.

There is a 3% penalty if your name and student number are not filled in correctly.

Papers may not be retained by students.

Question 1 (10 marks)

(a) (6 marks) Determine the truth value of the following statements, no justifications required. One mark is awarded for correct answers and *one mark is subtracted* for incorrect answers. No marks are awarded or subtracted when no answer is given.

i. The only property that is both safety and liveness is the trivial property $(2^{\mathcal{P}})^{\omega}$.

ii. In static analysis, a False Positive is a situation where an alarm is raised, even though there is no bug.

iii. Symbolic CTL model checking is based on the fix-point semantics of CTL.

iv. The CEGAR loop for predicate abstraction always terminates.

v. Two timed automata that are timed-abstract equivalent are also timed-equivalent.

vi. The diameter of an automaton is the greatest lower bound of the shortest distances between any connected states.

(b) (4 marks) Answer the following questions with short answers (1-2 sentences):

i. Name one advantage and one disadvantage of the *abstraction refinement* method.

ii. Give a reason why "classical" LTL model checking using Büchi automata cannot be used for timed systems.

Question 2 (16 marks)

(a) (7 marks) Let p and q be atomic propositions. Consider the following two pairs of LTL/CTL-formulae. For each pair, determine whether they are equivalent. If your answer is 'Yes' provide a proof. If the answer is 'No' give a counterexample model.

i. $\mathbf{A}(\mathbf{A}\, p\, \mathbf{Until}\, q)\, \mathbf{Until}\, (\mathbf{A}\, q\, \mathbf{Until}\, p)$ and $\mathbf{A}\, p\, \mathbf{Until}\, q$;

ii. $(\mathbf{X}\, p)\, \mathbf{R}\, p$ and $p \wedge \mathbf{X}\, p$.

(b) (9 marks) Let **AtNext** be a binary temporal operator. Informally, $\varphi\, \mathbf{AtNext}\, \psi$ means that "φ will be true the next time ψ is true, not assuming that φ or ψ will be true at all". Its formal semantics is given by

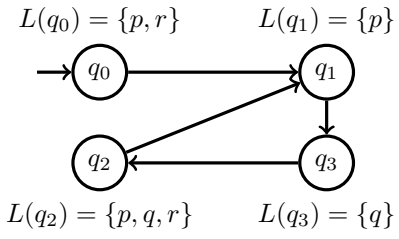
$$\rho \models \varphi\, \mathbf{AtNext}\, \psi \;:\Longleftrightarrow\; \begin{cases} \big(\exists k \geq 1.\ (\rho[k] \models (\varphi \wedge \psi)\ \wedge\ \forall j.\ 1 \leq j < k \Rightarrow \rho[j] \models \neg\psi)\big) \vee \\ \neg(\exists k \geq 1.\ \rho[k] \models \psi) \end{cases}$$

i. Prove that the next-operator $\mathbf{X}$ can be expressed using **AtNext**.

ii. Prove that the operator **AtNext** does not increase the expressiveness of LTL.

Question 3 (18 marks)

Here we have the automaton B:



For atomic propositions p, q, r let $\varphi = (\mathbf{EX}\, p) \wedge \neg(\mathbf{E}(\neg q)\, \mathbf{Until}\, (\neg r))$ be a CTL formula.

(a) Give the parse tree of the formula φ.

(b) Manually run the CTL explicit-state marking algorithm on φ for Automaton B. Does $B \models \varphi$ hold? Explain your answer.

(c) Our marking algorithm only considers the temporal operators $\mathbf{EX}\, \varphi$, $\mathbf{E}\, \varphi\, \mathbf{Until}\, \psi$ and $\mathbf{A}\, \varphi\, \mathbf{Until}\, \psi$. To improve performance one introduces procedures for other temporal operators. Describe a procedure for "exists release", i.e. for $\mathbf{A}\, \varphi\, \mathbf{R}\, \psi$, which is defined as $\neg(\mathbf{E}(\neg\varphi)\, \mathbf{Until}\, (\neg\psi))$. (Pseudocode is not necessary)

Question 4 (10 marks)

Let φ and ψ be LTL formulae. Propose an algorithm to check if φ and ψ are equivalent. The following operations can be used without any explanations:

- *Build a Büchi automaton* A_ϕ that accepts the sequences satisfied by a given LTL-formula ϕ, e.g. by the construction using the local and eventuality automata.
- *Emptiness check* for Büchi automata, i.e. given a Büchi automaton B one can check if $L(B) = \varnothing$.
- *Build the cross product* of two Büchi automata, i.e. given two Büchi automata B_1, B_2, we can compute the product $B_1 \times B_2$.

All other steps used in the algorithm should be explained and justified.

Question 5 (16 marks)

This exercise is about *Live Variables Analysis* of source code. Consider the following pseudo-code in a simple WHILE language:

```
[x:=a+b]1;
while [y > x*a]2 do (
  if [y<365]3
    then [x:=y+2]4;
    else skip5;
  [a:=2*x]6
);
```

(a) (2 marks) Give the control flow graph (CFG) for the above program.

(b) (3 marks) Give the gen_{LV} and $kill_{\mathsf{LV}}$ function for each statement in the program.

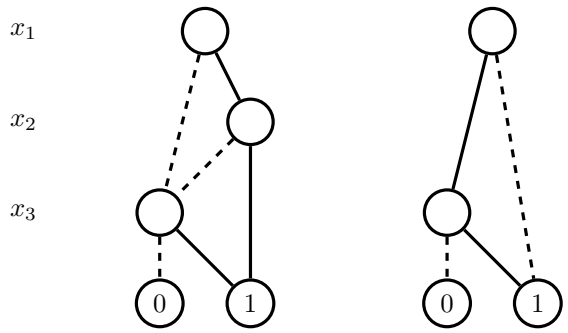
(c) (3 marks) Give the data flow equation for each node's entry (LV_{entry}) and each node's exit(s) (LV_{exit}).

(d) (8 marks) Compute the least fix point of the equation set, i.e., the result of the live variables analysis by giving the resulting LV_{entry}, LV_{exit} for all nodes after resolving the equation set.

Question 6 (15 marks)

(a) (5 marks) Give the reduced binary decision diagram (ORBDD) for $x_1 \Rightarrow (x_3 \wedge (x_2 \vee x_4))$, using the order of the variables $x_1 < x_2 < x_3 < x_4$, i.e., x_1 is at the top of the tree.

(b) (6 marks) Given the following two ORBDDs, encoding the formulas φ and ψ

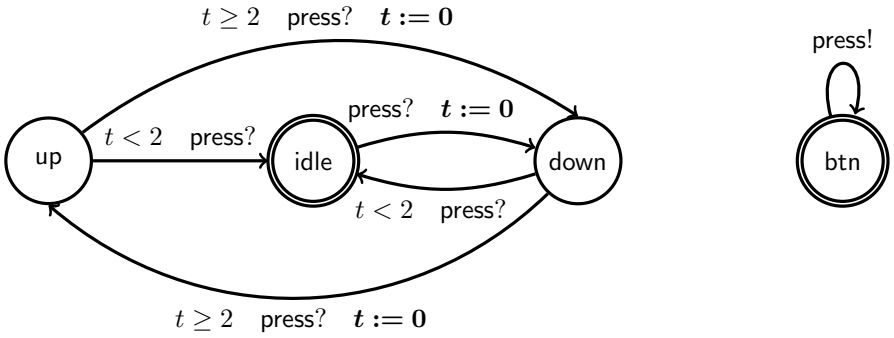


Here a solid line indicates the Boolean value true, and a dashed line the value false.
Determine a BDD for $\neg\varphi \wedge \psi$, using among others the operator $\otimes$ of the lecture.
(The diagram does not need to be a tree, nor need it be reduced.)

(c) (4 marks) Reduce the BDD for part (b).

Question 7 (15 marks)

Here we have two timed automata, Elevator and Button:



Locations are labelled with their names. Double-circled locations are *initial* states. Each transition is labelled with a guard (if present), a communication channel in sans-serif font (if present), and finally an assignment in **boldface** (if present).

The two automata synchronise via the binary handshake channel press. They must synchronise. The model contains only one clock t. The cross-product of the timed automata is denoted by $S = \text{Elevator} \times \text{Button}$.

The model is intended to describe an elevator controlled by one button only. There are three states: the elevator can go up, down or stand still (idle).

(a) (3 marks) Describe the behaviour of the timed automaton Elevator in English words.

(b) (6 marks) The atomic propositions up, down and idle represent the corresponding locations of the automaton Elevator in the product S. Assume the following TCTL specifications:

$$\begin{aligned} \varphi_1 &= \mathbf{AG}\ \big(\mathsf{down} \Rightarrow (\mathbf{AF}_{\geq 0}\ \mathsf{idle})\big) \\ \varphi_2 &= \mathbf{AG}\ \big(\mathsf{up} \Rightarrow (\mathbf{EF}_{\leq 1}\ \mathsf{idle})\big) \\ \varphi_3 &= \mathbf{AG}\ \big(\mathsf{up} \Rightarrow (\mathbf{EF}_{<2}\ \mathsf{down})\big) \end{aligned}$$

For each $\varphi_i, 1 \leq i \leq 3$, does S satisfy φ_i? Justify your answer.

(c) (6 marks) We want to enforce a press! action to occur infinitely often.

The specification for this is $P(k)$: the first press! action has to occur within $(\leq)$ k time units after the system started, and any press! must be followed by another press! action within $(\leq)$ k time units, for a fixed integer $k \geq 0$.

i. What do you need to add to Button to enforce this behaviour?

ii. Does the product Elevator $\times$ Button(k) contain deadlocks, where Button(k) is the modified Button automaton?

END OF EXAM