

THE AUSTRALIAN NATIONAL UNIVERSITY

Mid-Semester Quiz  
**SOLUTIONS**  
Second Semester, 2011

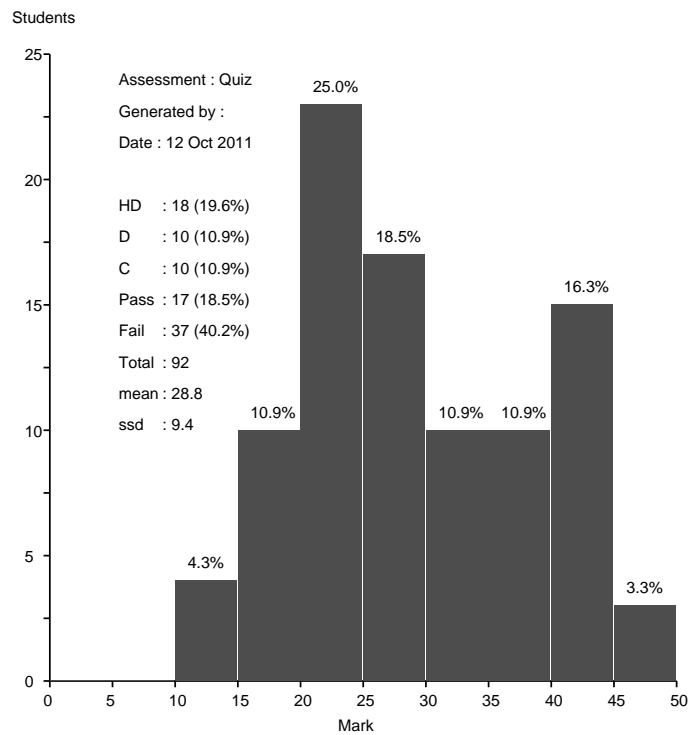
**COMP2600**  
**(Formal Methods for Software Engineering)**

*Writing Period: 1 hour duration*

*Study Period: 10 minutes duration*

*Permitted Materials: One A4 page with hand-written notes on both sides*

The histogram shows class performance on the quiz, and the table shows the average scores for each question:



	Q1	Q2	Q3	Q4	Q5	Total
Marks	11	11	11	7	11	50
Mean	6.21	8.28	6.15	1.98	6.22	28.83

## QUESTION 1 [11 marks]

### Natural Deduction

- (a) Use truth tables to determine whether the following inference is (always) valid or not; if not, give a counterexample (values of  $p, q, r$  for which it is not valid).

$$\frac{(p \rightarrow q) \vee (q \rightarrow r)}{(p \wedge q) \rightarrow r}$$

QUESTION 1(a)							[3 marks]
$p$	$q$	$r$	$p \rightarrow q$	$q \rightarrow r$	$(p \rightarrow q) \vee (q \rightarrow r)$	$p \wedge q$	$(p \wedge q) \rightarrow r$
T	T	T	T	T	T	T	T
T	T	F	T	F	T	T	F
T	F	T	F	T	T	F	T
T	F	F	F	T	T	F	T
F	T	T	T	T	T	F	T
F	T	F	T	F	T	F	T
F	F	T	T	T	T	F	T
F	F	F	T	T	T	F	T

The inference is not valid, with the counterexample  $p$  true,  $q$  true,  $r$  false.

- (b) Give a natural deduction proof of  $\frac{(p \rightarrow r) \vee (q \rightarrow r)}{(p \wedge q) \rightarrow r}$

QUESTION 1(b)			[4 marks]
1	$(p \rightarrow r) \vee (q \rightarrow r)$		
2	$p \wedge q$		
3	$p \rightarrow r$		
4	$p$	$\wedge$ -E, 2	
5	$r$	$\rightarrow$ -E, 3, 4	

QUESTION 1(b), continued

6		$q \rightarrow r$	
7		$q$	$\wedge$ -E, 2
8		$r$	$\rightarrow$ -E, 6, 7
9		$r$	$\vee$ -E, 1, 3–5, 6–8
10	$(p \wedge q) \rightarrow r$		$\rightarrow$ -I, 2–9

- (c) Give a natural deduction proof of  $\frac{(\exists x. P(x)) \rightarrow Q}{\forall x. P(x) \rightarrow Q}$ , that is,  $\frac{(\exists x. P(x)) \rightarrow Q}{\forall x. (P(x) \rightarrow Q)}$  (where  $x$  does not appear free in  $Q$ )

QUESTION 1(c)

[4 marks]

1	$(\exists x. P(x)) \rightarrow Q$		
2	$a$	$P(a)$	
3		$\exists x. P(x)$	$\exists$ -I, 2
4		$Q$	$\rightarrow$ -E, 1, 3
5		$P(a) \rightarrow Q$	$\rightarrow$ -I, 2–4
6	$\forall x. P(x) \rightarrow Q$		$\forall$ -I, 5

## QUESTION 2 [10 marks]

### Structural Induction

Given these function definitions:

$$\text{sum } [] = 0 \quad \text{-- (S1)}$$

$$\text{sum } (x:xs) = x + \text{sum } xs \quad \text{-- (S2)}$$

$$[] ++ ys = ys \quad \text{-- (A1)}$$

$$(x:xs) ++ ys = x : (xs ++ ys) \quad \text{-- (A2)}$$

We would like to prove the following property using structural induction.

$$\text{sum } (xs ++ ys) = \text{sum } xs + \text{sum } ys$$

You may need to include an explicit  $\forall$  in the goals and the inductive hypothesis. If so, in your answers indicate where you need to make use of this.

- (i) State and prove the base case goal.

**QUESTION 2** **[3 marks]**

We need to prove this by induction on the list  $xs$  (not  $ys$ , the definition clause (A2) for  $++$  is the clue here).

**Base case:**  $xs = []$   
Show that  $\text{sum } ([] ++ ys) = \text{sum } [] + \text{sum } ys$

**Proof:**

$$\begin{aligned} \text{sum } ([] ++ ys) &= \text{sum } ys && \text{-- by (A1)} \\ &= 0 + \text{sum } ys \\ &= \text{sum } [] + \text{sum } ys && \text{-- by (S1)} \end{aligned}$$

- (ii) State the induction hypothesis.

**QUESTION 2** **[2 marks]**

For proving the step case for  $xs = a:as$ , the induction hypothesis is

$$\text{sum } (as ++ ys) = \text{sum } as + \text{sum } ys \quad \text{-- (IH)}$$

We will see that we don't need to quantify over  $ys$ .

(iii) State and prove the step case goal.

QUESTION 2

[5 marks]

**Step case:**  $xs = a:as$

Show that if the inductive hypothesis (as above) holds, then

$\text{sum } ((a:as) ++ ys) = \text{sum } (a:as) + \text{sum } ys$

**Proof:**

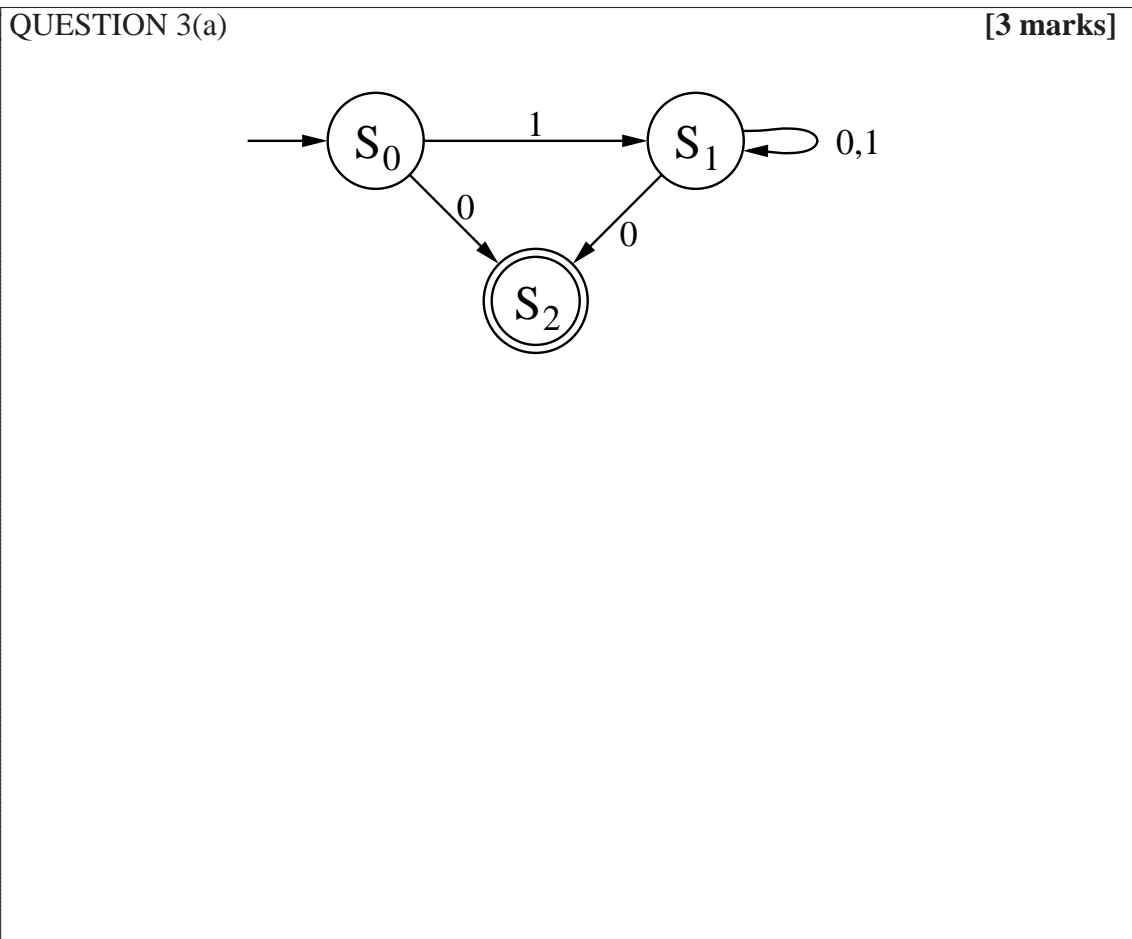
```
sum ((a:as) ++ ys)
  = sum (a : (as ++ ys))      -- by (A2)
  = a + sum (as ++ ys)       -- by (S2)
  = a + (sum as + sum ys)    -- by (IH)
  = (a + sum as) + sum ys    -- by arith
  = sum (a:as) + sum ys      -- by (S2)
```

### QUESTION 3 [11 marks]

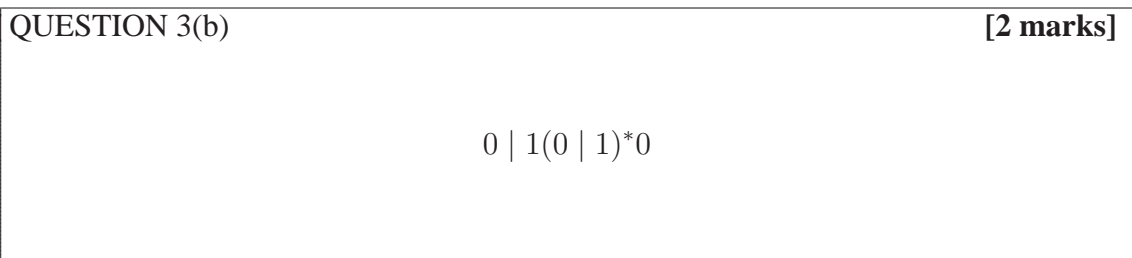
#### Finite State Machines

- (a) Give a Non-deterministic Finite State Automaton (NFA) on the alphabet  $\Sigma = \{0, 1\}$  which accepts the language  $L$  of even binary numerals written without superfluous leading 0, that is,  $L$  consists of non-empty strings which either
- consist of a single 0, or
  - start with 1 and end with 0

If possible, find an NFA with only three states.



- (b) Give a regular expression for the language  $L$ .



- (c) Find a Deterministic Finite State Automaton (DFA) for  $L$ . (Hint: this requires five states).

QUESTION 3(c) [3 marks]

```

graph LR
    S0((S0)) -- 0 --> S2(((S2)))
    S0 -- 1 --> S1((S1))
    S2 -- "0,1" --> S_(((S_)))
    S1 -- 0 --> S12(((S12)))
    S12 -- 1 --> S1
    S1 -- 1 --> S1
    S_ -- "0,1" --> S_
  
```

This DFA is obtained from the NFA using the algorithm described in lectures: when both FSAs have read the same input string,

- the DFA is in state  $S_i$  when the NFA can only be in state  $S_i$  ( $i = 0, 1, 2$ )
- the DFA is in state  $S_{12}$  when the NFA could be in states  $S_1$  or  $S_2$
- the DFA is in state  $S_-$  when the NFA could not be in any state (ie it must have got “stuck” trying to read that input)

- (d) Explain why a Deterministic Finite State Automaton for  $L$  must have more than one final state.

QUESTION 3(d) [3 marks]

Where  $S_0$  is the start state, and  $F$  is the set of final states,  $N(S_0, 0)$  is a final state because  $0 \in L$ . But no other string starting with 0 is in  $L$ , that is, for  $\alpha \neq \epsilon$ ,  $N^*(S_0, 0\alpha) \notin F$ . That is, if you call  $N(S_0, 0)$   $S_2$  (as in the diagram in (c) above), then  $N^*(S_2, \alpha) \notin F$  for every non-empty  $\alpha$ .

On the other hand  $N^*(S_0, 10) \in F$  because  $10 \in L$ , and there are many strings  $\beta$  for which  $N^*(S_0, 10\beta) \in F$  (to be exact, where  $\beta$  ends in 0). That is, if you call  $N^*(S_0, 10)$   $S_1$  (as in the diagram in (c) above), then  $N^*(S_1, \beta) \in F$  for some non-empty  $\beta$ .

Therefore  $S_1$  and  $S_2$  (defined as  $N^*(S_0, 10)$  and  $N(S_0, 0)$ ) must be different. Similarly you can show that there must be three non-final states. Showing it more informally:

- $N^*(S_0, 00) \notin F$ , and from this state you can *never* get to a final state.
- $N^*(S_0, 1) \notin F$ , and from this state you can get to a final state; moreover wherever you can get to from this state you can get to a final state from there (just by reading input 0) — so you *can't* get to a state like  $S_-$
- $S_0 \notin F$ , and neither of the above is true of this state

## QUESTION 4 [7 marks]

### Context-free Languages and Parsing

Consider the following grammar with non-terminal (and start symbol)  $S$ , and terminals  $a, b$ .

$$S \rightarrow a b \qquad S \rightarrow a S b$$

- (a) Describe the language  $L$  generated by this grammar.

QUESTION 4(a)

[2 marks]

$$L = \{a^n b^n \mid n \geq 1\}$$

that is, the language consisting of strings which contain, first, some number (at least 1) of 'a's, then the same number of 'b's

- (b) The PDA derived naturally from this grammar has the following transitions:  
With  $q_0$  as start state, and  $q_2$  as final state:

$$\begin{array}{ll} \delta(q_0, \epsilon, Z) \mapsto q_1/SZ & \delta(q_1, a, a) \mapsto q_1/\epsilon \\ \delta(q_1, \epsilon, Z) \mapsto q_2/\epsilon & \delta(q_1, b, b) \mapsto q_1/\epsilon \end{array}$$

plus two more transitions omitted from the above. What are they?

QUESTION 4(b)

[1 mark]

$$\begin{array}{l} \delta(q_1, \epsilon, S) \mapsto q_1/a b \\ \delta(q_1, \epsilon, S) \mapsto q_1/a S b \end{array}$$

- (c) We want to do top-down parsing of the language using this PDA. But allowing one symbol lookahead in the input is not sufficient to make this parser deterministic. Explain why.

QUESTION 4(c)

[2 marks]

When you have the non-terminal  $S$  on top of the stack, that is, you are trying to match the next part of the input to the  $S$ , the next step is to choose a transition which replaces  $S$  on the top of the stack by either  $ab$  or  $aSb$  (that is, choosing between the two transitions in the answer to (b)). But since the first symbol of both these possibilities is  $a$ , looking at the next input symbol (which should be  $a$ ) does not tell you which of the two transitions to choose.

- (d) We can change the grammar, to avoid this problem. Give a grammar such that

- it generates the same language  $L$ , and
- looking ahead one symbol of the input is sufficient to determine the next transition of the corresponding PDA

Hint: one possible solution has, as one of its productions,  $S \rightarrow aT$ .

QUESTION 4(d)

[2 marks]

The hint suggests we have the non-terminal  $T$  generate all the strings of the form  $\{a^m b^{m+1} \mid m \geq 0\}$ .

This gives the following grammar.

$$S \rightarrow aT$$

$$T \rightarrow aTb \quad (\text{first terminal must be } a)$$

$$T \rightarrow b \quad (\text{first terminal must be } b)$$

There are two productions for  $T$ , but they can only generate sentences which begin with  $a$  and  $b$  respectively, so looking ahead to the next symbol of the input is enough to decide which production (or transition of the corresponding PDA) to use.

## QUESTION 5 [11 marks]

### Specification using Z

A technical college has students; they enrol, take courses and, if their collection of completed units satisfies certain rules, they graduate.

The bare bones of the system is captured in the following Z specification.

<p>[Person] [Unit]</p> <p><math>Status ::= Incomplete \mid Graduated</math></p> <p><math>preRequisites : Unit \leftrightarrow Unit</math> <math>advancedUnits : \mathbb{P} Unit</math></p> <p><math>\forall u : Unit \bullet u \in advancedUnits \Rightarrow</math> <math>\exists p : Unit \bullet (p \mapsto u) \in preRequisites</math></p>	<p><math>UnitEnrolment_o</math></p> <p><math>\Delta College</math> <math>p? : Person</math> <math>u? : Unit</math></p> <p><math>status(p?) = Incomplete</math> <math>u? \notin courses(p?)</math> <math>\forall c : Unit \bullet ((c \mapsto u?) \in preRequisites)</math> <math>\Rightarrow ((p? \mapsto c) \in unitsCompleted)</math> <math>status' = status</math> <math>courses' = courses \setminus \{p? \mapsto courses(p?)\}</math> <math>\cup \{p? \mapsto (courses(p?) \cup \{u?\})\}</math> <math>unitsCompleted' = unitsCompleted</math></p>
<p><math>College</math></p> <p><math>status : Person \leftrightarrow Status</math> <math>courses : Person \leftrightarrow \mathbb{P} Unit</math> <math>unitsCompleted : Person \leftrightarrow Unit</math></p> <p><math>dom unitsCompleted \subseteq dom courses</math> <math>dom courses \subseteq dom status</math> <math>\forall p : Person, u : Unit \bullet</math> <math>(p \mapsto u) \in unitsCompleted \Rightarrow</math> <math>u \in courses(p)</math></p>	<p><math>Graduation_o</math></p> <p><math>\Delta College</math> <math>p? : Person</math></p> <p><math>(p? \mapsto Incomplete) \in status</math> <math>\#(\{p?\} \triangleleft unitsCompleted) \geq 8</math> <math>\#(ran(\{p?\} \triangleleft unitsCompleted)</math> <math>\cap advancedUnits) \geq 4</math> <math>status' = status \setminus (p? \mapsto Incomplete)</math> <math>\cup (p? \mapsto Graduated)</math> <math>courses' = courses</math> <math>unitsCompleted' = unitsCompleted</math></p>
<p><math>StudentAdmission_o</math></p> <p><math>\Delta College</math> <math>p? : Person</math></p> <p><math>p? \notin dom status</math> <math>status' = status \cup \{p? \mapsto Incomplete\}</math> <math>courses' = courses \cup \{p? \mapsto \emptyset\}</math> <math>unitsCompleted' = unitsCompleted</math></p>	

- (a) Give, in plain English, the constraint on the constant set *advancedUnits*.

QUESTION 5(a)	<b>[2 marks]</b>
Every advanced unit has at least one pre-requisite.	

- (b) Describe, in English the global variable (state variable) *unitsCompleted*. How is it different from the state variable *courses*?

QUESTION 5(b)	<b>[2 marks]</b>
Each element of the relation <i>unitsCompleted</i> indicates that some particular person has completed some particular subject. The maplets of the <i>courses</i> function indicate that a person has merely enrolled in a subject, perhaps having completed it.	

- (c) Give (in English) the preconditions for the operation *UnitEnrolment<sub>o</sub>*.

QUESTION 5(c)	<b>[1 mark]</b>
<ol style="list-style-type: none"> <li>(1) The student must not have graduated.</li> <li>(2) The student must not already be enrolled in the unit.</li> <li>(3) The student must have completed all the pre-requisites.</li> </ol>	

- (d) Give (in English) the postconditions for the operation *UnitEnrolment<sub>o</sub>*.

QUESTION 5(d)	<b>[2 marks]</b>
<ol style="list-style-type: none"> <li>(1) The state variables <i>status</i> and <i>unitsCompleted</i> are unchanged.</li> <li>(2) The set of courses that <i>p</i>? is enrolled in has the course <i>u</i>? added.</li> <li>(3) The set of courses of other students is not changed.</li> </ol>	

- (e) Give (in English) the three preconditions for the operation *Graduation<sub>o</sub>*.

QUESTION 5(e)	<b>[2 marks]</b>
<ol style="list-style-type: none"> <li>(1) The student must be enrolled but not graduated already.</li> <li>(2) The student must have completed at least 8 courses overall.</li> <li>(3) The student must have completed at least 4 advanced courses.</li> </ol>	

- (f) Write a schema for a query called *Graduates<sub>o</sub>* which has an output variable *graduates!* which will be the set of students who have graduated.

QUESTION 5(f)	<b>[2 marks]</b>								
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="border-bottom: 1px solid black; padding: 5px;"><i>Graduates<sub>o</sub></i></td> <td style="border-bottom: 1px solid black; padding: 5px;">_____</td> </tr> <tr> <td style="padding: 5px;"><math>\exists \text{College}</math></td> <td style="padding: 5px;"></td> </tr> <tr> <td style="padding: 5px;"><i>graduates!</i> : <math>\mathbb{P} \text{Person}</math></td> <td style="padding: 5px;"></td> </tr> <tr> <td style="border-bottom: 1px solid black; padding: 5px;"><i>graduates!</i> = <math>\text{dom}(\text{status} \triangleright \{\text{Graduated}\})</math></td> <td style="border-bottom: 1px solid black; padding: 5px;">_____</td> </tr> </table>		<i>Graduates<sub>o</sub></i>	_____	$\exists \text{College}$		<i>graduates!</i> : $\mathbb{P} \text{Person}$		<i>graduates!</i> = $\text{dom}(\text{status} \triangleright \{\text{Graduated}\})$	_____
<i>Graduates<sub>o</sub></i>	_____								
$\exists \text{College}$									
<i>graduates!</i> : $\mathbb{P} \text{Person}$									
<i>graduates!</i> = $\text{dom}(\text{status} \triangleright \{\text{Graduated}\})$	_____								

# Appendix 1 — Natural Deduction Rules

## Propositional Calculus

$(\wedge I) \quad \frac{p \quad q}{p \wedge q}$	$(\wedge E) \quad \frac{p \wedge q}{p} \quad \frac{p \wedge q}{q}$
$(\vee I) \quad \frac{p}{p \vee q} \quad \frac{q}{q \vee p}$	$(\vee E) \quad \frac{p \vee q \quad \begin{array}{c} [p] \quad [q] \\ \vdots \quad \vdots \end{array} \quad r \quad r}{r}$
$(\rightarrow I) \quad \frac{\begin{array}{c} [p] \\ \vdots \\ q \end{array}}{p \rightarrow q}$	$(\rightarrow E) \quad \frac{p \quad p \rightarrow q}{q}$
$(\neg I) \quad \frac{\begin{array}{c} [p] \\ \vdots \\ q \wedge \neg q \end{array}}{\neg p}$	$(\neg E) \quad \frac{\begin{array}{c} [\neg p] \\ \vdots \\ q \wedge \neg q \end{array}}{p}$

## Predicate Calculus

$(\forall I) \quad \frac{P(a) \quad (a \text{ arbitrary})}{\forall x. P(x)}$	$(\forall E) \quad \frac{\forall x. P(x)}{P(a)}$
$(\exists I) \quad \frac{P(a)}{\exists x. P(x)}$	$(\exists E) \quad \frac{\begin{array}{c} [P(a)] \\ \vdots \\ \exists x. P(x) \quad q \quad (a \text{ arbitrary}) \end{array}}{q \quad (a \text{ is not free in } q)}$

## Appendix 2 — Truth Table Values

$p$	$q$	$p \vee q$	$p \wedge q$	$p \rightarrow q$	$\neg p$	$p \leftrightarrow q$
T	T	T	T	T	F	T
T	F	T	F	F	F	F
F	T	T	F	T	T	F
F	F	F	F	T	T	T

---

---