THE AUSTRALIAN NATIONAL UNIVERSITY

Mid-Semester Quiz

SOLUTION

Second Semester, 2010

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 questions are followed by labelled blank spaces into which your answers are to be written. Additional answer panels are provided at the end of the paper should you wish to use more space for an answer than is provided in the associated labelled panels.

Student Number:

<table>
<thead>
<tr>
<th>Q1 Mark</th>
<th>Q2 Mark</th>
<th>Q3 Mark</th>
<th>Q4 Mark</th>
<th>Q5 Mark</th>
<th>Q6 Mark</th>
<th>Total Mark</th>
</tr>
</thead>
</table>
QUESTION 1 [10 marks]

Natural Deduction

(a) Give a natural deduction proof of

\[
\frac{(p \rightarrow q) \land (q \rightarrow r)}{(p \lor q) \rightarrow r}
\]

QUESTION 1(a) [5 marks]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((p \rightarrow q) \land (q \rightarrow r))</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(p \rightarrow q)</td>
<td>(\land)-E, 1</td>
</tr>
<tr>
<td>3</td>
<td>(q \rightarrow r)</td>
<td>(\land)-E, 1</td>
</tr>
<tr>
<td>4</td>
<td>(p \lor q)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(p)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>(q)</td>
<td>(\rightarrow)-E, 2, 5</td>
</tr>
<tr>
<td>7</td>
<td>(r)</td>
<td>(\rightarrow)-E, 3, 6</td>
</tr>
<tr>
<td>8</td>
<td>(q)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>(r)</td>
<td>(\rightarrow)-E, 3, 8</td>
</tr>
<tr>
<td>10</td>
<td>(r)</td>
<td>(\lor)-E, 4, 5–7, 8–9</td>
</tr>
<tr>
<td>11</td>
<td>((p \lor q) \rightarrow r)</td>
<td>(\rightarrow)-I, 4–10</td>
</tr>
</tbody>
</table>
(b) Give a natural deduction proof of $\forall x. Q \rightarrow P(x)$ (where $x$ does not appear free in $Q$)

\[ Q \rightarrow (\forall x. P(x)) \]

**QUESTION 1(b)** [5 marks]

1. $\forall x. Q \rightarrow P(x)$
2. $Q$
3. $a \quad Q \rightarrow P(a) \quad \forall$-E, 1
4. $P(a) \quad \rightarrow$-E, 3, 2
5. $\forall x. P(x) \quad \forall$-I, 4
6. $Q \rightarrow (\forall x. P(x)) \quad \rightarrow$-I, 2–5
QUESTION 2 [10 marks]

Structural Induction

Given these function definitions:
\[
\begin{align*}
\text{sum2} \ [] \ acc &= acc \quad -- \ T1 \\
\text{sum2} \ (x:xs) \ acc &= \text{sum2} \ xs \ (x + acc) \quad -- \ T2 \\
\text{sum} \ [] &= 0 \quad -- \ S1 \\
\text{sum} \ (x:xs) &= x + \text{sum} \ xs \quad -- \ S2
\end{align*}
\]

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

\[
\text{sum2} \ xs \ acc = \text{sum} \ xs + acc
\]

You may need to include an explicit \( \forall \) in the goals and the inductive hypothesis. In your answers indicate where you need to make use of this.

(i) State and prove the base case goal.

**QUESTION 2**

Note: Universal quantifiers are often left implicit when writing out mathematical theorems, and so it is here. The result to be proved is actually:

\[\forall xs. \forall acc. \text{sum2} \ xs \ acc = \text{sum} \ xs + acc\]

The structural induction is done using the variable \( xs \).

**Base case:** \( (xs = []) \)

Show that (for all \( acc \)) \( \text{sum2} \ [] \ acc = \text{sum} \ [] + acc \)

**Proof:**

\[
\begin{align*}
\text{sum2} \ [] \ acc &= acc \quad -- \ by \ (T1) \\
&= 0 + acc \quad -- \ arithmetic \\
&= \text{sum} \ [] + acc \quad -- \ by \ (S1)
\end{align*}
\]

To be complete, we should now apply the \( \forall \)-I rule to get:

\[\forall acc. \text{sum2} \ [] \ acc = \text{sum} \ [] + acc\]

(ii) State the induction hypothesis.

**QUESTION 2**

The induction hypothesis is:

\[\forall acc. \text{sum2} \ xs \ acc = \text{sum} \ xs + acc \quad -- \ (IH)\]

Note: The universal quantifier will prove necessary here.
(iii) State and prove the step case goal.

**QUESTION 2**

The step goal is, assuming the induction hypothesis, *for all* `acc`,

\[ \text{sum2} \ (x:xs) \ acc = \text{sum} \ (x:xs) + acc \]

**Proof:**

\[
\begin{align*}
\text{sum2} \ (x:xs) \ acc &= \text{sum2} \ xs \ (x + acc) \quad \text{-- by (T2)} \\
&= \text{sum} \ xs + (x + acc) \quad \text{-- by (IH) **} \\
&= (x + \text{sum} \ xs) + acc \quad \text{-- arithmetic} \\
&= \text{sum} \ (x:xs) + acc \quad \text{-- by (S2)}
\end{align*}
\]

(**) Note: It is here that it is important that that IH is universally quantified

– the bound variable `acc` is instantiated to `(x + acc)`

Again, to be complete, we should now apply the `∀`-I rule to get:

\[ ∀acc. \ \text{sum2} \ xs \ acc = \text{sum} \ xs + acc \]

Further comment – not part of the answer to the question

Having proved the base case:

\[ ∀acc. \ \text{sum2} \ [\] \ acc = \text{sum} \ [\] + acc \]

and the step case:

\[ ∀acc. \ \text{sum2} \ xs \ acc = \text{sum} \ xs + acc \]

\[ \text{sum2} \ (x:xs) \ acc = \text{sum} \ (x:xs) + acc \]

we can infer the *fully quantified* version of the theorem:

\[ ∀xs. \ ∀acc. \ \text{sum2} \ xs \ acc = \text{sum} \ xs + acc \]

This is an application of the structural induction principle for lists.
QUESTION 3  [10 marks]

Hoare Logic

(a) Clearly explain, in terms of the meaning of the notation  \( \{P\} S \{Q\} \) why the following assertion of Hoare Logic is true.

\[
\{n < 10\} \text{ while } (n < 11) \text{ do } n:=n*2 \ {n > 10}\]

Note: this part does not involve using the proof rules.

QUESTION 3(a)  [3 marks]

The assertion is true according to the meaning of the notation because the statement either does not terminate or it terminates with \(n > 10\).

- if \(n < 0\) initially, it loops forever with \(n\) decreasing
- if \(n = 0\) initially, it loops forever with \(n\) remaining 0
- if \(0 < n < 10\) initially, \(n\) doubles on each iteration until \(n > 10\)

(b) Using the rules of Hoare Logic (given in Appendix 2) justify the following assertion (a loop invariant for part (c) below). Justify each step with the name of the rule used.

\[
\{p = m \times (n - i)\} \ i:=i-1; \ p:=p+m \ \{p = m \times (n - i)\}
\]

QUESTION 3(b)  [4 marks]

1. \(\{p + m = m \times (n - i)\} \ p:=p+m \ \{p = m \times (n - i)\}\) (Asst.Rule)
2. \(\{p + m = m \times (n - (i - 1))\} \ i:=i-1 \ \{p + m = m \times (n - i)\}\) (Asst.Rule)
3. \(\{p = m \times (n - i)\} \ i:=i-1 \ \{p + m = m \times (n - i)\}\) (Simplify 2)
4. \(\{p = m \times (n - i)\} \ i:=i-1; \ p:=p+m \ \{p = m \times (n - i)\}\) (Seq.Rule 3,1)
(e) Using the rules of Hoare Logic justify the following assertion using the loop invariant proved in the previous part. Make explicit any use of precondition strengthening or post-condition weakening that is needed.

\[ \{ p = m \times (n - i) \} \text{ while } i \neq 0 \text{ do } i := i - 1; \ p := p + m \ \{ p = m \times n \} \]

In your answer you may use the abbreviations indicated

\[
\begin{aligned}
\text{Inv} & \quad \{ p = m \times (n - i) \} \\
\text{Body} & \quad \text{while } i \neq 0 \text{ do } i := i - 1; \ p := p + m \ \{ p = m \times n \} \\
\text{Loop} & \\
\end{aligned}
\]

**QUESTION 3(c) [3 marks]**

5. \{Inv\} \ i := i - 1; \ p := p + m \ \{Inv\} \quad \text{(Abbreviate 4)}
6. Inv \land (i \neq 0) \Rightarrow Inv \quad \text{(Algebra)}
7. \{Inv \land (i \neq 0)\} \ \text{Body} \ \{Inv\} \quad \text{(Prec.Str. 6,5)}
8. \{Inv\} \ \text{Loop} \ \{Inv \land \neg (i \neq 0)\} \quad \text{(WhileRule 7)}
9. Inv \land \neg (i \neq 0) \Rightarrow p = m \times n \quad \text{(Algebra)}
10. \{Inv\} \ \text{Loop} \ \{p = m \times n\} \quad \text{(PostWkng 8,9)}
QUESTION 4 [10 marks]

Weakest Preconditions

The following is a program for computing the product of two integers.

\[
\begin{align*}
  &i := n; \\
  &p := 0; \\
  \text{while} \ (i /= 0) \ 	ext{do} \\
  &i := i - 1; \\
  &p := p + m; \\
\end{align*}
\]

Again, you may use the abbreviations indicated for parts of the program.

The desired postcondition, \( Post \), is that \( p \) is the product of \( m \) and \( n \):

\[ Post \equiv p = m \ast n \]

We will be concerned with the subproblem of computing the weakest precondition that must apply just before the while statement (to achieve that result).

The rules for calculating weakest preconditions are in Appendix 3.

(a) For while loop and postcondition \( Post \), give expressions for \( P_0 \) and \( P_1 \) in as simple a form as you can.

\[
\begin{align*}
  P_0 &\equiv \neg (i \neq 0) \land (p = m \ast n) \\
        &\equiv (i = 0) \land (p = m \ast n) \\

  P_1 &\equiv (i \neq 0) \land \text{wp}(\text{Body}, \ (i = 0 \land p = m \ast n)) \\
        &\equiv (i \neq 0) \land \text{wp}(i := i - 1, \ wp(p := p + m, \ (i = 0 \land p = m \ast n))) \\
        &\equiv (i \neq 0) \land (i - 1 = 0) \land (p + m = m \ast n) \\
        &\equiv i = 1 \land p = m \ast (n - 1)
\end{align*}
\]
(b) Calculate a similar simple expression for $P_2$. (You should be mindful of the loop invariant.)

$$P_2 \equiv (i \neq 0) \land wp(Body, (i = 1 \land p = m \ast (n - 1)))$$

$$\equiv (i \neq 0) \land wp(i := i - 1, wp(p := p + m, (i = 1 \land p = m \ast (n - 1))))$$

$$\equiv (i \neq 0) \land (i - 1 = 1) \land (p + m = m \ast (n - 1))$$

$$\equiv i = 2 \land p = m \ast (n - 2)$$

(c) Infer a general expression for $P_k$. (Don’t worry about the inductive proof.)

$$P_k \equiv i = k \land p = m \ast (n - i)$$

Note that the 2nd conjunct is what we know to be the loop invariant.

(d) Compute $wp(while i /= 0 do i:=i-1; p:=p+m, Post)$, explaining briefly how the existential quantifier was eliminated.

$$wp(Loop, p = m \ast n) \equiv \exists k. (k \geq 0) \land P_k$$

$$\equiv \exists k. (k \geq 0) \land i = k \land p = m \ast (n - i)$$

$$\equiv p = m \ast (n - i) \land \exists k. (k \geq 0) \land i = k$$

$$\equiv p = m \ast (n - i) \land i \geq 0$$

We first simplified the existential term using the fact that

$$\exists x. (P(x) \land Q) \equiv Q \land \exists x. P(x)$$

provided that $x$ is not free in $Q$.

Then it is clear that there is a non-negative $k$ which is equal to $i$ (and only if) $i$ is non-negative.
QUESTION 5  [5 marks]

Finite State Machines

Give a Deterministic Finite State Automation (DFA) on the language \( \Sigma = \{a, b, c\} \) which accepts the language \( L = \{ab\alpha \mid \alpha \in \Sigma^*\} \), that is, it accepts strings which start with \( ab \).
QUESTION 6 [5 marks]

Specification using Z

In the recent tutorial for Z, the system being modelled was the inventory of a rich man’s art collection. The state of this system is captured by two sets of artworks and an amount of money available for making purchases; one of those sets is the collection currently owned and the other is the set of pictures previously owned.

The state schema, ArtInventory, follows. The schema Buy specifies the error-free case of buying a new picture.

\[
\begin{align*}
\text{ArtInventory} & \quad \Delta\text{ArtInventory} \\
\text{current} : \mathbb{P} \text{Picture} & \quad p? : \text{Picture} \\
\text{previous} : \mathbb{P} \text{Picture} & \quad \text{price}? : \mathbb{N} \\
\text{capital} : \mathbb{N} & \quad \text{capital}' = \text{capital} - \text{price}? \quad \text{current}' = \text{current} \cup \{p?\} \\
\text{current} \cap \text{previous} = \emptyset & \quad \text{previous}' = \text{previous} \cup \{p?\}
\end{align*}
\]

(a) Give, in plain English, the preconditions for buying pictures.

QUESTION 6(a) [1 mark]

i) The picture to be purchased must not already be in the collection and must not have been previously owned by the collector.

ii) The collector must have sufficient funds allocated.

(b) Write a corresponding schema, Sell, that specifies the selling of pictures.

QUESTION 6(b) [4 marks]

\[
\begin{align*}
\text{Sell} & \quad \Delta\text{ArtInventory} \\
\text{current} : \mathbb{P} \text{Picture} & \quad p? : \text{Picture} \\
\text{capital} : \mathbb{N} & \quad \text{price}? : \mathbb{N} \\
p? \in \text{current} & \quad \text{capital}' = \text{capital} + \text{price}? \\
\text{current}' = \text{current} \setminus \{p?\} & \quad \text{previous}' = \text{previous} \cup \{p?\}
\end{align*}
\]