THE AUSTRALIAN NATIONAL UNIVERSITY
Second Semester 2014

COMP2600
( Formal Methods for Software Engineering )

Writing Period: 3 hours duration
Study Period: 15 minutes duration
Permitted Materials: One A4 page with hand-written notes on both sides
Answer ALL questions
Total marks: 100

WITH SOME SAMPLE SOLUTIONS

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. If you use an additional panel, be sure to indicate clearly the question and part to which it is linked.

The following spaces are for use by the examiners.

<table>
<thead>
<tr>
<th>Q1 (StrInd)</th>
<th>Q2 (FOL)</th>
<th>Q3 (NatDed)</th>
<th>Q4 (Hoare)</th>
<th>Q5 (WP)</th>
<th>Q6 (SL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q7 (FSA)</td>
<td>Q8 (CFL)</td>
<td>Q9 (TM)</td>
<td>Total</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
QUESTION 1 [10 marks]  

Structural Induction

Here is the usual Haskell definition of a binary tree:

\[ \text{data Tree } a = \text{Nul} | \text{Node } a \ (\text{Tree } a) \ (\text{Tree } a) \]

We define the following functions \( \text{depth} \) and \( \text{rev} \) that operate on trees:

\[
\begin{align*}
\text{depth Nul} &= 0 \quad -- (D1) \\
\text{depth } (\text{Node } a \ t1 \ t2) &= 1 + \max(\text{depth } t1, \text{depth } t2) \quad -- (D2) \\
\text{rev Nul} &= \text{Nul} \quad -- (R1) \\
\text{rev } (\text{Node } a \ t1 \ t2) &= \text{Node } a \ (\text{rev } t2) \ (\text{rev } t1) \quad -- (R2)
\end{align*}
\]

We also define the auxiliary function \( \text{max} \) that operates on pairs of values of any \textit{Num} type:

\[
\begin{align*}
\text{max } (a, b) &= \quad -- (M1) \\
| a > b &= a \\
| \text{otherwise} &= b
\end{align*}
\]

Prove the following property by structural induction:

\[ \text{depth } (\text{rev } t) = \text{depth } t \quad -- (P1) \]

In all proofs indicate the justification (eg, the line of a definition used) for each step.

(a) Using plain English, describe the property P1 you are about to prove:

<table>
<thead>
<tr>
<th>QUESTION 1(a)</th>
<th>[2 marks]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reversing a tree preserves its depth.</td>
<td></td>
</tr>
</tbody>
</table>
(b) State and prove the base case of the proof of P1:

**QUESTION 1(b)**

Show $\text{depth (rev Nul)} = \text{depth Nul}$

**Proof:**

$\text{depth (rev Nul)}$

$= \text{depth Nul} \quad \text{-- by (R1)}$

(c) State the inductive hypotheses of the proof of P1:

**QUESTION 1(c)**

$\text{depth (rev t1)} = \text{depth t1} \quad \text{-- (IH1)}$

$\text{depth (rev t2)} = \text{depth t2} \quad \text{-- (IH2)}$
(d) State and prove the step case goal of the proof of P1. Note: you must give justification for every step of the proof. For example, if you wish to use the fact that the function max is commutative, you must prove this fact first. A correct solution that does not use the commutativity of max is also acceptable.

QUESTION 1(d) [6 marks]

Show that if the inductive hypotheses (as above) hold, then
\[ \text{depth (rev (Node a t1 t2))} = \text{depth (Node a t1 t2)} \]

Proof (verbose/explicit version):

Case when depth t2 > depth t1:

\[
\begin{align*}
\text{depth (rev (Node a t1 t2))} &= \text{depth (Node a (rev t2) (rev t1))} -- \text{by (R2)} \\
&= 1 + \max (\text{depth (rev t2)}, \text{depth (rev t1)}) -- \text{by (D2)} \\
&= 1 + \max (\text{depth t2}, \text{depth t1}) -- \text{by (IH1, IH2)} \\
&= 1 + \text{depth t2} -- \text{by (M1)} \\
&= 1 + \max (\text{depth t1}, \text{depth t2}) -- \text{by (M1)} \\
&= \text{depth (Node a t1 t2)} -- \text{by (D2)} \\
\end{align*}
\]

Case when depth t2 ≤ depth t1:

\[
\begin{align*}
\text{depth (rev (Node a t1 t2))} &= \text{depth (Node a (rev t2) (rev t1))} -- \text{by (R2)} \\
&= 1 + \max (\text{depth (rev t2)}, \text{depth (rev t1)}) -- \text{by (D2)} \\
&= 1 + \max (\text{depth t2}, \text{depth t1}) -- \text{by (IH1, IH2)} \\
&= 1 + \text{depth t1} -- \text{by (M1)} \\
&= 1 + \max (\text{depth t1}, \text{depth t2}) -- \text{by (M1)} \\
&= \text{depth (Node a t1 t2)} -- \text{by (D2)} \\
\end{align*}
\]

Alternative proof (concise version):

- First prove a simple lemma that \( \max (x, y) = \max (y, x) \). This proof should be by cases.
- Use lemma. Then both cases above effectively collapse into one case.
QUESTION 2 [6 marks] Specification in First-Order Logic

This question discusses the production of grapes at a winery. A winery consists of a number of vines, and each vine grows a number of grapes.

The following predicates are given:

- $B(x, y)$: $x$ belongs to $y$
- $G(x, y)$: $x$ produces grapes in time period $y$
- $F(x)$: there is frost in time period $x$
- $P(x, y)$: the time period $x$ is profitable for $y$
- $PP(x)$: $x$ is profitable

Translate each of the following sentences into First-Order Logic:

(a) The vine produces grapes in a season if there is no frost in that season. Hint: use a free variable like $v$ to refer to the vine.

\[
\forall x. \neg F(x) \rightarrow G(v, x)
\]

(b) The season is profitable for a winery if every vine in the winery produces grapes in that season. Hint: use a free variable like $s$ to refer to the season.

\[
\forall w. (\forall v. (B(v, w) \rightarrow G(v, s))) \rightarrow P(s, w)
\]

(c) A winery is profitable if every season is profitable for the winery. Hint: this statement applies to all wineries.

\[
\forall w. (\forall s. P(s, w)) \rightarrow PP(w)
\]
QUESTION 3 [12 marks]  

The following questions ask for proofs using natural deduction. Present your proofs in the Fitch style as used in lectures. You may only use the introduction and elimination rules given in Appendix 1. Number each line and include justifications for each step in your proofs.

(a) Give a natural deduction proof of \( \neg a \lor \neg b \rightarrow \neg (a \land b) \)

\[
\begin{array}{ccc}
\text{QUESTION 3(a)} & \text{[6 marks]} \\
1 & \neg a \lor \neg b \\
2 & \neg a \\
3 & a \land b \\
4 & a & \land \text{-E, 3} \\
5 & a \land \neg a & \land \text{-I, 4, 2} \\
6 & \neg (a \land b) & \neg \text{I, 3–5} \\
7 & \neg b \\
8 & a \land b \\
9 & b & \land \text{-E, 8} \\
10 & b \land \neg b & \land \text{-I, 9, 7} \\
11 & \neg (a \land b) & \neg \text{I, 8–10} \\
12 & \neg (a \land b) & \lor \text{-E, 1, 2–6, 7–11} \\
13 & \neg a \lor \neg b \rightarrow \neg (a \land b) & \rightarrow \text{I, 1–12} \\
\end{array}
\]
(b) Give a natural deduction proof of \((\forall z. C(z) \rightarrow B(z)) \land (\exists w. C(w)) \rightarrow \exists x. B(x)\)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((\forall z. C(z) \rightarrow B(z)) \land \exists w. C(w))</td>
</tr>
<tr>
<td>2</td>
<td>(\forall z. C(z) \rightarrow B(z)) \quad \land\text{-E, 1}</td>
</tr>
<tr>
<td>3</td>
<td>(\exists w. C(w)) \quad \land\text{-E, 1}</td>
</tr>
<tr>
<td>4</td>
<td>(a) \quad C(a)</td>
</tr>
<tr>
<td>5</td>
<td>C(a) \rightarrow B(a) \quad \forall\text{-E, 2}</td>
</tr>
<tr>
<td>6</td>
<td>B(a) \quad \rightarrow\text{-E, 4, 5}</td>
</tr>
<tr>
<td>7</td>
<td>(\exists x. B(x)) \quad \exists\text{-I, 6}</td>
</tr>
<tr>
<td>8</td>
<td>(\exists x. B(x)) \quad \exists\text{-E, 3, 4-7}</td>
</tr>
</tbody>
</table>
QUESTION 4 [12 marks]  

Consider the following code fragment:

\[
\begin{aligned}
i &:= 0; \\
s &:= 0; \\
\textbf{while} (i \neq n) \textbf{do} \\
& \quad i := i + 1; \\
& \quad s := s + 2*i
\end{aligned}
\]

Your task is to use the rules of Hoare Logic (Appendix 3) to show that the value of \( s \), if and when this code terminates, will be the sum of the first \( n \) even numbers where 2 is the first even number. That is, you will try to prove

\[
\{ \text{True} \} \text{ SumEvens } \{ \text{Post} \}
\]

where

\[
\text{Post} \equiv ( s = n \times (n + 1) )
\]

The proof will have this structure where \( \text{Inv} \) is the loop-invariant:

\[
\begin{align*}
\vdots \\
\{ (i \neq n) \land \text{ Inv} \} \text{ Body } \{ \text{Inv} \} \\
\{ \text{Inv} \} \textbf{ while} (i \neq n) \textbf{ do} \text{ Body } \{ \text{Inv} \land \neg (i \neq n) \} \\
\vdots \\
\{ \text{True} \} \text{ Init } \{ \text{Inv} \} \\
\{ \text{Inv} \} \textbf{ while} (i \neq n) \textbf{ do} \text{ Body } \{ \text{Post} \} \\
\vdots \\
\{ \text{True} \} \text{ SumEvens } \{ \text{Post} \}
\end{align*}
\]

You may assume that all variables are typed integer. In the questions below we will refer to the code fragment as \text{SumEvens}, to the loop code as \text{Loop}, and to the body of the loop as \text{Body}. Make sure that every step of your proof is numbered, and is justified by citing the rule, and any previous proof steps, that you are using.
(a) The following algebraic equivalence will be useful in both this question, and the next question on Weakest Precondition Calculus. You may use it at any time, regardless of whether you successfully answer this question, so long as you explicitly refer to it as Lemma (a).

Prove, using standard algebraic manipulations, that

\[(s + 2 \times i = i \times (i + 1)) \iff (s = (i - 1) \times i)\]

Lemma (a)

**QUESTION 4(a)**  [2 marks]

\[
\begin{align*}
s + 2 \times i &= i \times (i + 1) \\
\iff s + 2 \times i &= i^2 + i \\
\iff s &= i^2 + i - 2 \times i \\
&= i^2 - i \\
&= (i - 1) \times i
\end{align*}
\]

(b) We will need an invariant for Loop. We suggest

\[\text{Inv} \equiv (s = i \times (i + 1)).\]

Prove that

\[
\{\text{Inv}\} \text{ Body } \{\text{Inv}\}.
\]

**QUESTION 4(b)**  [4 marks]

(1). \[
\{s = (i - 1 + 1) \times (i + 1)\} i := i + 1 \{s = (i - 1) \times i\} \quad \text{(Asst.)}
\]

(2). \[
s = (i - 1 + 1) \times (i + 1) \iff s = i \times (i + 1) \quad \text{(Basic Arithmetic)}
\]

(3). \[
\{s = i \times (i + 1)\} i := i + 1 \{s = (i - 1) \times i\} \quad \text{(PreConEq.)}
\]

(4). \[
\{s + 2 \times i = i \times (i + 1)\} s := s + 2 \times i \{s = i \times (i + 1)\} \quad \text{(Asst.)}
\]

(5). \[
(s + 2 \times i = i \times (i + 1)) \iff (s = (i - 1) \times i) \quad \text{(Lemma (a))}
\]

(6). \[
\{s = (i - 1) \times i\} s := s + 2 \times i \{s = i \times (i + 1)\} \quad \text{(PreConEq)}
\]

(7). \[
\{s = i \times (i + 1)\} i := i + 1; \ s := s + 2 \times i \{s = i \times (i + 1)\} \quad \text{((3),(6), Seq.)}
\]

(8). \[
\{\text{Inv}\} \text{ Body } \{\text{Inv}\} \quad \text{(i.e.)}
\]
(e) Using part (b), prove that

\{ \text{Inv} \} \text{ Loop } \{ \text{Post} \}.

**QUESTION 1(c) [3 marks]**

(1). \( (s = i \times (i + 1) \land (i \neq n)) \Rightarrow (s = i \times (i + 1)) \)  
   (Basic Logic)

(2). \{ s = i \times (i + 1) \land (i \neq n) \} \text{ Body } \{ s = i \times (i + 1) \}  
   (b),(1), PreConStr

(3). \{ s = i \times (i + 1) \} \text{ Loop } \{(s = i \times (i + 1)) \land \neg (i \neq n)\}  
   (2), While

(4). \{(s = i \times (i + 1)) \land \neg (i \neq n) \} \Rightarrow (s = n \times (n + 1))  
   (Basic Maths)

(5). \{ s = i \times (i + 1) \} \text{ Loop } \{ s = n \times (n + 1) \}  
   ((3),(4), PostConWk.)

(6). \{ \text{Inv} \} \text{ Loop } \{ \text{Post} \}  
   (i.e.)

(d) Using part (c), prove that

\{ \text{True} \} \text{ SumEvens } \{ \text{Post} \}.

**QUESTION 1(d) [3 marks]**

(1). \{ 0 = 0 \times (0 + 1) \} \ i := 0 \{ 0 = i \times (i + 1) \}  
   (Asst)

(2). \{ 0 = 0 \times (0 + 1) \} \leftrightarrow \text{True}  
   (Basic Arith)

(3). \{ \text{True} \} \ i := 0 \{ 0 = i \times (i + 1) \}  
   ((1),(2), PreConEq)

(4). \{ 0 = i \times (i + 1) \} \ s := 0 \{ s = i \times (i + 1) \}  
   (Asst)

(5). \{ \text{True} \} \ i := 0; \ s := 0 \{ s = i \times (i + 1) \}  
   ((3), (4) Seq)

(6). \{ \text{True} \} \ i := 0; \ s := 0; \text{ Loop } \{ s = n \times (n + 1) \}  
   ((5), c, Seq)

(7). \{ \text{True} \} \text{ SumEvens } \{ \text{Post} \}  
   (i.e.)
QUESTION 5  [12 marks]  
Weakest Precondition Calculus

As with the previous question, we will consider the code fragment `SumEvens`:

\[
\begin{align*}
i &:= 0; \\
s &:= 0; \\
\text{while } (i \neq n) \text{ do} & \quad \{ \text{Init} \} \\
i &:= i + 1; \\
s &:= s + 2*i & \quad \{ \text{Body} \} \\
\end{align*}
\]

\[
\text{Loop} \quad \{ \text{SumEvens} \}
\]

We will use the rules of the Weakest Precondition Calculus (Appendix 4) to calculate

\[wp(SumEvens, s = n \cdot (n + 1)).\]

As in the previous question we will use the abbreviations `Loop` and `Body` for the indicated parts of the code. You may continue to make use of the lemma from part (a) of the previous question by referring to it explicitly as Lemma (a). Remember show all your working when you do so.

(a) We will want to calculate

\[wp(\text{Loop}, s = n \cdot (n + 1)).\]

\(P_0\) is the predicate expressing success for this weakest precondition after zero loop iterations. Calculate and show that \(P_0\) is equal to

\[(i + 0 = n) \land (s = i \cdot (i + 1))\]

\[
\begin{align*}
\text{QUESTION 5(a)}
\end{align*}
\]

\[
\begin{align*}
P_0 &\equiv \neg b \land Q \\
P_0 &\equiv \neg (i \neq n) \land s = n \cdot (n + 1) \\
&\equiv (i = n) \land (s = i \cdot (i + 1)) \\
&\equiv (i + 0 = n) \land (s = i \cdot (i + 1))
\end{align*}
\]
(b) $P_1$ is the predicate expressing success after one loop iteration. Calculate and show that $P_1 \equiv (i + 1 = n) \land (s = i \times (i + 1))$

**QUESTION 5(b)**

$P_1 \equiv (i \neq n) \land \text{wp}(\text{Body}, P_0)$

$\equiv (i \neq n) \land \text{wp}(i := i + 1; s := s + 2 \times i, P_0)$

$\equiv (i \neq n) \land \text{wp}(i := i + 1, \text{wp}(s := s + 2 \times i, P_0))$  

$\equiv (i \neq n) \land \text{wp}(i := i + 1, \text{wp}(s := s + 2 \times i, (i = n) \land (s = i \times (i + 1))))$  

$\equiv (i \neq n) \land \text{wp}(i := i + 1, (i = n) \land (s + 2 \times i = i \times (i + 1)))$  

$\equiv (i \neq n) \land \text{wp}(i := i + 1, (i = n) \land (s = (i - 1) \times i))$  

Lemma (a)

$\equiv (i \neq n) \land (i + 1 = n) \land (s = ((i + 1) - 1) \times (i + 1))$  

$\equiv (i \neq n) \land (i + 1 = n) \land (s = i \times (i + 1))$  

$\equiv (i + 1 = n) \land (s = i \times (i + 1))$
(c) $P_k$ is the predicate expressing success after $k$ loop iterations that holds for all $k \geq 0$. Using induction, prove that

$$\forall k \geq 0. \ P_k \equiv (i + k = n) \land (s = i \ast (i + 1))$$

**QUESTION 5(c)**

[5 marks]

**Base Case:** We already have $P_0 \equiv (i + 0 = n) \land (s = i \ast (i + 1))$

**I.H.:** Assume for some $j > 0$ that $\forall 0 \leq k \leq j. \ P_k \equiv (i + k = n) \land (s = i \ast (i + 1))$

**Inductive Step:** We have to show that $P_{j+1} \equiv (i + (j + 1) = n) \land (s = i \ast (i + 1))$

$$P_{j+1} \equiv (i \neq n) \land wp(i := i + 1; \ s := s + 2 \ast i, P_j)
\equiv (i \neq n) \land wp(i := i + 1, \ wp(s := s + 2 \ast i, P_j))
\equiv (i \neq n) \land wp(i := i + 1, \ wp(s := s + 2 \ast i,
(i + j = n) \land (s = i \ast (i + 1))))
\equiv (i \neq n) \land wp(i := i + 1, (i + j = n) \land (s + 2 \ast i = i \ast (i + 1))))
\equiv (i \neq n) \land (i + 1 + j = n) \land (s + 2 \ast (i + 1) = (i + 1) \ast ((i + 1) + 1))
\equiv (i \neq n) \land (i + (j + 1) = n) \land (s = i^2 + i)
\equiv (i + (j + 1) = n) \land (s = i \ast (i + 1))$$

(d) Given your answer to part (c), state

$$wp(\text{Loop}, \ s = n \ast (n + 1)).$$

Do not attempt any simplification at this stage.

**QUESTION 5(d)**

[1 mark]

$$\exists k. \ k \geq 0 \land P_k
\equiv \exists k. \ k \geq 0 \land (i + k = n) \land (s = i \ast (i + 1))$$
(e) Hence calculate

\[ wp(SumEvens, s = n \ast (n + 1)). \]

and state this result in the simplest form possible. Stating an answer without showing your working out is not sufficient.

\[
\begin{align*}
wp(SumEvens, s = n \ast (n + 1)) \\
\equiv wp(i := 0, wp(s := 0, wp(Loop, s = n \ast (n + 1))))
\end{align*}
\]

\[
\begin{align*}
\equiv wp(i := 0, wp(s := 0, \exists k. k \geq 0 \land (i + k = n) \land (s = i \ast (i + 1))))
\end{align*}
\]

\[
\begin{align*}
\equiv wp(i := 0, \exists k. k \geq 0 \land (0 + k = n) \land (0 = i \ast (i + 1)))
\end{align*}
\]

\[
\begin{align*}
\equiv \exists k. k \geq 0 \land (0 + k = n) \land (0 = 0 \ast (0 + 1)))
\end{align*}
\]

\[
\begin{align*}
\equiv \exists k. k \geq 0 \land k = n \land True
\end{align*}
\]

\[
\begin{align*}
\equiv n \geq 0
\end{align*}
\]
QUESTION 6 [12 marks]  

Separation Logic.

Let $\Pi$ be the following program:

$$
\begin{align*}
x & := \text{cons}(1, 2) \\
[x+1] & := 3
\end{align*}
$$

Our goal is to prove

$$\{\text{emp}\} \Pi \{ x \mapsto 1, 3 \}$$

But we are going to do it bit by bit.

You may need the following properties of separation logic:

Commutativity: $(A * B) \iff (B * A)$

Abbreviation: $(x \mapsto e_1, e_2) \equiv (x \mapsto e_1 * (x + 1) \mapsto e_2)$

Given: $((x + 1) \mapsto 2) \Rightarrow ((x + 1) \mapsto -)$

(a) Prove $\{\text{emp}\} x := \text{cons}(1, 2) \{ x \mapsto 1 * (x + 1) \mapsto 2 \}$

Set out the proof as a linear sequence of numbered lines with justifications on the right hand side as usual. Do *not* use the frame rule. My solution is three lines long but any proper proof without the frame rule will do.

<table>
<thead>
<tr>
<th>QUESTION 6(a)</th>
<th>[3 marks]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1). ${\text{emp}} x := \text{cons}(1, 2) { x \mapsto 1, 2 }$</td>
<td>(DerAllAssAxm)</td>
</tr>
<tr>
<td>(2). $x \mapsto 1, 2 \equiv (x \mapsto 1 * (x + 1) \mapsto 2)$</td>
<td>(Abbreviation)</td>
</tr>
<tr>
<td>(3). ${\text{emp}} x := \text{cons}(1, 2) { x \mapsto 1 * (x + 1) \mapsto 2 }$</td>
<td>((1), (2) PostConEq)</td>
</tr>
</tbody>
</table>
(b) Prove \( \{(x + 1) \mapsto 2\} [x + 1] := 3 \{ (x + 1) \mapsto 3 \} \)

Set out the proof as a linear sequence of numbered lines with justifications on the right hand side as usual. Do *not* use the frame rule. My solution is three lines long but any proper proof will do.

<table>
<thead>
<tr>
<th>QUESTION 6(b)</th>
<th>3 marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1). ( {(x + 1) \mapsto _} [x + 1] := 3 { (x + 1) \mapsto 3 } ) (HeaAssAxm)</td>
<td></td>
</tr>
<tr>
<td>(2). ( (x + 1) \mapsto 2 \Rightarrow ((x + 1) \mapsto _) ) (Given)</td>
<td></td>
</tr>
<tr>
<td>(3). ( {(x + 1) \mapsto 2} [x + 1] := 3 { (x + 1) \mapsto 3 } ) (PreConStr)</td>
<td></td>
</tr>
</tbody>
</table>

(c) Using (a) and (b) prove \( \{ \text{emp} \} x := \text{cons}(1, 2) ; [x + 1] := 3 \{ x \mapsto 1, 3 \} \)

Set out the proof as a linear sequence of numbered lines with justifications on the right hand side as usual. My solution has first line (a) and second line (b), and then seven more lines, but any proper proof will do. You will need the frame rule.

<table>
<thead>
<tr>
<th>QUESTION 6(c)</th>
<th>6 marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1). ( { \text{emp} } x := \text{cons}(1, 2) { x \mapsto 1 \ast (x + 1) \mapsto 2 } ) (a)</td>
<td></td>
</tr>
<tr>
<td>(2). ( {(x + 1) \mapsto 2} [x + 1] := 3 { (x + 1) \mapsto 3 } ) (b)</td>
<td></td>
</tr>
<tr>
<td>(3). ( {(x + 1) \mapsto 2 \ast x \mapsto 1} [x + 1] := 3 { (x + 1) \mapsto 3 \ast x \mapsto 1 } ) (FrameRule)</td>
<td></td>
</tr>
<tr>
<td>(4). ( (A \ast B) \Leftrightarrow (B \ast A) ) (Commutativity)</td>
<td></td>
</tr>
<tr>
<td>(5). ( {(x + 1) \mapsto 2 \ast x \mapsto 1} [x + 1] := 3 { x \mapsto 1 \ast (x + 1) \mapsto 3 } ) (PostConEqv)</td>
<td></td>
</tr>
<tr>
<td>(6). ( x \mapsto 1 \ast (x + 1) \mapsto 2 } [x + 1] := 3 { x \mapsto 1 \ast (x + 1) \mapsto 3 } ) (PreConEqv)</td>
<td></td>
</tr>
<tr>
<td>(7). ( { \text{emp} } x := \text{cons}(1, 2) ; [x + 1] := 3 { x \mapsto 1 \ast (x + 1) \mapsto 3 } ) (Seq)</td>
<td></td>
</tr>
<tr>
<td>(8). ( (x \mapsto 1, 3) \equiv (x \mapsto 1 \ast (x + 1) \mapsto 3) ) (Abbreviation)</td>
<td></td>
</tr>
<tr>
<td>(9). ( { \text{emp} } x := \text{cons}(1, 2) ; [x + 1] := 3 { x \mapsto 1, 3 } ) (PostConEq)</td>
<td></td>
</tr>
</tbody>
</table>
(a) The following is a right linear grammar where $S$ is the start symbol:

\[
\begin{align*}
S & \rightarrow aA \\
A & \rightarrow bT \\
B & \rightarrow aT \\
T & \rightarrow \epsilon \\
S & \rightarrow bB \\
A & \rightarrow a \\
B & \rightarrow b
\end{align*}
\]

Convert this grammar to a non-deterministic finite automaton (NFA).
(b) The following is a non-deterministic finite automaton (NFA):

Convert this NFA to a deterministic finite automaton (DFA).

QUESTION 7(b) [3 marks]

COMP2600 (Formal Methods for Software Engineering) Page 18 of 28
(e) The following is a deterministic finite automaton (DFA):

Minimise this DFA. Show all the steps of your work.

**QUESTION 7(c) [4 marks]**

We first detect equivalent states by running the algorithm as below:

- $[[S_0, S_1, S_2, S_3], [S_3, S_4]]$: initial split
- $[[S_0, S_1, S_2], [S_3], [S_3, S_4]]$: testing with $a$
- $[[S_0], [S_1, S_2], [S_3], [S_3, S_4]]$: testing with $b$

Subsequent tests with $a$ or $b$ produce no further split.

Therefore $S_1, S_2$ are equivalent, and $S_3, S_4$ are equivalent.

The state $S_5$ is inaccessible, so it is deleted. The minimised DFA is as below:
(d) The following is a deterministic finite automaton (DFA):

State (1) the language this DFA recognises; and (2) the subgoals for proving that this DFA indeed accepts the language you describe.

For (1), you do not need to give a mathematical description, an informal description suffices. For (2), you can abbreviate your description of the language as $P$. That is, any string that satisfies $P$ also satisfies your description. You do not need to give the full proof.

**QUESTION 7(d)**  
[2 marks]

This DFA recognises the language over \{a, b\} where every occurrence of $a$ is proceeded by **AND** followed by an occurrence of $b$.

Subgoal 1: any string $w$ that satisfies $P$ is accepted by the DFA.

Subgoal 2: any string $w$ accepted by the DFA satisfies $P$. 
QUESTION 8 [12 marks]  Context Free Languages and Pushdown Automata

(a) The following context free grammar, where $S$ is the start symbol, generates a sub-language of abstract separation logic:

$$S \rightarrow S \ast S \mid S \ast S \mid \neg S \mid A$$
$$A \rightarrow a \mid b \mid c$$

Give two parse trees for $a * b \ast c$ to show that the above grammar is ambiguous.

(b) Give an unambiguous grammar that generates the same language as the grammar given in sub question (a).

We assume that the precedence of logical connectives is as follows: $\neg$ binds tighter than $\ast$, which in turn binds tighter than $\ast \ast$.

$$S \rightarrow S \ast T \mid T$$
$$T \rightarrow T \ast U \mid U$$
$$U \rightarrow \neg U \mid a \mid b \mid c$$
(c) Convert the context free grammar given in sub question (a) to a (non-deterministic) PDA.

**QUESTION 8(c)**

<table>
<thead>
<tr>
<th>Initialisation:</th>
<th>δ(q₀, ϵ, Z) ⟷ q₁/SZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-terminals:</td>
<td>δ(q₁, ϵ, S) ⟷ q₁/S S</td>
</tr>
<tr>
<td></td>
<td>δ(q₁, ϵ, S) ⟷ q₁/S ¬ S</td>
</tr>
<tr>
<td></td>
<td>δ(q₁, ϵ, S) ⟷ q₁/A</td>
</tr>
<tr>
<td></td>
<td>δ(q₁, ϵ, A) ⟷ q₁/a</td>
</tr>
<tr>
<td></td>
<td>δ(q₁, ϵ, A) ⟷ q₁/b</td>
</tr>
<tr>
<td></td>
<td>δ(q₁, ϵ, A) ⟷ q₁/c</td>
</tr>
<tr>
<td>Terminals:</td>
<td>δ(q₁, x, x) ⟷ q₁/ε</td>
</tr>
<tr>
<td>Termination:</td>
<td>δ(q₁, ϵ, Z) ⟷ q₂/ε</td>
</tr>
</tbody>
</table>

Where x ∈ {*, ¬*, ¬, a, b, c}.

(d) Consider the following PDA defined by its transitions:

δ(p, ϵ, Z) ⟷ q/XZ

δ(q, i, X) ⟷ q/XX

δ(q, e, X) ⟷ q/ε

δ(q, ϵ, Z) ⟷ ε/ε

where the initial state is p. Write a trace for this PDA when processing the string iei. State explicitly whether the string is accepted or rejected.

**QUESTION 8(d)**

(p, iei, Z) ⇒ (q, iei, XZ)

⇒ (q, ei, XXZ)

⇒ (q, i, XZ)

⇒ (q, ϵ, XXZ)

reject
(a) Assume the input alphabet is \{0, 1\}. Design a Turing machine that replaces all 0s in the input with 1s and replaces all 1s in the input with 0s. Assume that the pointer head points at the beginning of the input initially. You only need to draw a diagram of states and transitions for your Turing machine.

(b) What is the class of languages recognised by Turing machines?

(c) If a language $L$ is recursively enumerable but not recursive, is the problem $P_L$ of $L$ decidable?
(d) Are decidable problems easy to solve? If not, give an example that is hard to solve.

**QUESTION 9(d)**

No. Any example of NP problems, exp-time problems will be fine.

(e) If a non-deterministic Turing machine can solve a problem in polynomial time, can a deterministic Turing machine solve the same problem in polynomial time?

**QUESTION 9(e)**

I don’t know. No one knows.
Additional answers: deliberately left like this for use in landscape mode. Clearly indicate the corresponding question and part.
Additional answers: deliberately left like this for use in landscape mode. Clearly indicate the corresponding question and part.
This material to be given to the students as a separate handout, not an appendix to the exam paper!

**Handout 1 — Natural Deduction Rules**

### Propositional Calculus

<table>
<thead>
<tr>
<th>Rule</th>
<th>Premises</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\land I)$</td>
<td>$p$ $q$</td>
<td>$p \land q$</td>
</tr>
<tr>
<td>$(\land E)$</td>
<td>$p \land q$</td>
<td>$p$ $q$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$[p]$ $[q]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\vdots$ $\vdots$</td>
</tr>
<tr>
<td>$(\lor I)$</td>
<td>$p$</td>
<td>$p \lor q$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$q \lor p$</td>
</tr>
<tr>
<td>$(\lor E)$</td>
<td>$p \lor q$ $r$</td>
<td>$r$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$[p]$ $[\neg p]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\vdots$ $\vdots$</td>
</tr>
<tr>
<td>$(\Rightarrow I)$</td>
<td>$q$</td>
<td>$p \Rightarrow q$</td>
</tr>
<tr>
<td>$(\Rightarrow E)$</td>
<td>$p \Rightarrow q$</td>
<td>$q$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$[p]$ $[\neg p]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\vdots$ $\vdots$</td>
</tr>
<tr>
<td>$(\neg I)$</td>
<td>$q \land \neg q$</td>
<td>$\neg p$</td>
</tr>
<tr>
<td>$(\neg E)$</td>
<td>$q \land \neg q$</td>
<td>$p$</td>
</tr>
</tbody>
</table>

### Predicate Calculus

<table>
<thead>
<tr>
<th>Rule</th>
<th>Premises</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\forall I)$</td>
<td>$P(a)$ $(a$ arbitrary$)$</td>
<td>$\forall x. P(x)$</td>
</tr>
<tr>
<td>$(\forall E)$</td>
<td>$\forall x. P(x)$</td>
<td>$P(a)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$[P(a)]$ $[\neg P(a)]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\vdots$ $\vdots$</td>
</tr>
<tr>
<td>$(\exists I)$</td>
<td>$P(a)$</td>
<td>$\exists x. P(x)$</td>
</tr>
<tr>
<td>$(\exists E)$</td>
<td>$\exists x. P(x)$ $q$ $(a$ arbitrary$)$</td>
<td>$q$ $(a$ is not free in $q$)</td>
</tr>
</tbody>
</table>
Handout 2 — Truth Table Values

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \lor q$</th>
<th>$p \land q$</th>
<th>$p \Rightarrow q$</th>
<th>$\neg p$</th>
<th>$p \Leftrightarrow q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Handout 3 — Hoare Logic Rules

- **Precondition Strengthening:**
  \[
  P_s \Rightarrow P_w \quad \{P_w\} S \{Q\} \\
  \{P_s\} S \{Q\}
  \]

- **Postcondition Weakening:**
  \[
  \{P\} S \{Q_s\} \quad Q_s \Rightarrow Q_w \\
  \{P\} S \{Q_w\}
  \]

- **Assignment:**
  \[
  \{Q(e)\} x := e \quad \{Q(x)\}
  \]

- **Sequence:**
  \[
  \{P\} S_1 \{Q\} \quad \{Q\} S_2 \{R\} \\
  \{P\} S_1; S_2 \{R\}
  \]

- **Conditional:**
  \[
  \{P \land b\} S_1 \{Q\} \quad \{P \land \neg b\} S_2 \{Q\} \\
  \{P\} \text{ if } b \text{ then } S_1 \text{ else } S_2 \{Q\}
  \]

- **While Loop:**
  \[
  \{P \land b\} S \{P\} \\
  \{P\} \text{ while } b \text{ do } S \{P \land \neg b\}
  \]
Handout 4 — Weakest Precondition Rules

\[ wp(x := e, Q(x)) \equiv Q(e) \]
\[ wp(S_1; S_2, Q) \equiv wp(S_1, wp(S_2, Q)) \]
\[ wp(\text{if } b \text{ then } S_1 \text{ else } S_2, Q) \equiv (b \Rightarrow wp(S_1, Q)) \land (\neg b \Rightarrow wp(S_2, Q)) \]
\[ \equiv (b \land wp(S_1, Q)) \lor (\neg b \land wp(S_2, Q)) \]

\( P_k \) is the weakest predicate that must be true before while \( b \) do \( S \) executes, in order for the loop to terminate after exactly \( k \) iterations in a state that satisfies \( Q \).

\[ P_0 \equiv \neg b \land Q \]
\[ P_{k+1} \equiv b \land wp(S, P_k) \]
\[ wp(\text{while } b \text{ do } S, Q) \equiv \exists k. (k \geq 0 \land P_k) \]
Handout 5: Separation Logic Rules

Floyd Store Axiom for Separation Logic: replaces Hoare (Store) Axiom
\[
\{ x = v \land \text{emp} \} \quad x := e \quad \{ x = e(v/x) \land \text{emp} \}
\]
where \( v \) is an auxiliary variable which does not occur in \( e \)

Derived Floyd Store Axiom for Separation Logic:
\[
\{ \text{emp} \} \quad x := e \quad \{ x = e \land \text{emp} \}
\]
where \( x \) does not occur in \( e \)

Fetch Assignment Axiom
\[
\{(x = v_1) \land (e \mapsto v_2)\} \quad x := [e] \quad \{(x = v_2) \land (e(v_1/x) \mapsto v_2)\}
\]
where \( v_1 \) and \( v_2 \) are auxiliary variables which do not occur in \( e \)

Derived Fetch Assignment Axiom
\[
\{(e \mapsto v_2)\} \quad x := [e] \quad \{(x = v_2) \land (e \mapsto v_2)\}
\]
where \( v_2 \) and \( x \) do not occur in \( e \)

Heap Assignment Axiom
\[
\{ e \mapsto - \} \quad [e] := e_1 \quad \{ e \mapsto e_1 \}
\]
where \( e \mapsto - \) abbreviates \( \exists z. e \mapsto z \) and \( z \) does not occur in \( e \)

Allocation Assignment Axiom
\[
\{ x = v \land \text{emp} \} \quad x := \text{cons}(e_1, e_2, \cdots, e_n) \quad \{ x \mapsto e_1(v/x), e_2(v/x), \cdots, e_n(v/x) \}
\]
where \( v \) is an auxiliary variable different from \( x \) and not appearing in \( e_1, e_2, \cdots, e_n \)

Derived Allocation Assignment Axiom
\[
\{ \text{emp} \} \quad x := \text{cons}(e_1, e_2, \cdots, e_n) \quad \{ x \mapsto e_1, e_2, \cdots, e_n \}
\]
where \( x \) does not appear in \( e_1, e_2, \cdots, e_n \)

Dispose Axiom
\[
\{ e \mapsto - \} \quad \text{dispose}(e) \quad \{ \text{emp} \}
\]
where \( e \mapsto - \) abbreviates \( \exists z. e \mapsto z \) and \( z \) does not occur in \( e \)

The Frame Rule:
\[
\{ P \} \text{ S } \{ Q \} \quad \frac{\{ P \text{ S } R \} \text{ S } \{ Q \text{ S } R \}}{\{ P \text{ R } \} \text{ S } \{ Q \text{ R } \}}
\]
where no variable modified by \( S \) appears free in \( R \)

Other Rules: the Hoare Logic rules from Handout 3, except the Assignment rule, are also in the calculus for Separation Logic.