THE AUSTRALIAN NATIONAL UNIVERSITY

Mid-Semester Quiz
Second Semester, 2014

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

Answer questions Q1, Q2, (Q3 exor Q4) Q5, Q6
Total marks: 45

That is, either do Q3 or do Q4. If you do parts of Q3 and parts of Q4 then we will choose the one that gives you most marks, but you will not get marks for both parts.

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:

Q1 Mark | Q2 Mark | Q3 Mark | Q4 Mark | Q5 Mark | Q6 Mark | Total Mark
QUESTION 1 [10 marks]

Structural Induction

Here is the usual Haskell definition of a binary tree:

\[
\text{data Tree } a = \text{Nil} \mid \text{Node } a \ (\text{Tree } a) \ (\text{Tree } a)
\]

Given these function definitions:

\[
\begin{align*}
\text{sumT Nil} &= 0 & \text{-- (ST1)} \\
\text{sumT (Node } a \ t1 \ t2) &= a + \text{sumT } t1 + \text{sumT } t2 & \text{-- (ST2)} \\
\text{sumL []} &= 0 & \text{-- (SL1)} \\
\text{sumL } (x:xs) &= x + \text{sumL } xs & \text{-- (SL2)} \\
\text{flatten Nil} &= [] & \text{-- (F1)} \\
\text{flatten (Node } a \ t1 \ t2) &= \text{flatten } t1 ++ (a : \text{flatten } t2) & \text{-- (F2)} \\
[] \ + + \ ys &= ys & \text{-- (A1)} \\
(x:xs) \ + + \ ys &= x : (xs ++ ys) & \text{-- (A2)}
\end{align*}
\]

and the following lemma:

\[
\text{sumL } (xs ++ ys) = \text{sumL } xs + \text{sumL } ys \quad \text{-- (L1)}
\]

Prove the following property using structural induction:

\[
\text{sumT } t = \text{sumL } (\text{flatten } t)
\]

(a) State and prove the base case goal.

**Base case:** \( t = \text{Nil} \)

Show that \( \text{sumT } \text{Nil} = \text{sumL } (\text{flatten } \text{Nil}) \)

**Proof:**

\[
\begin{align*}
\text{sumT } \text{Nil} &= 0 \quad \text{-- by (ST1)} \\
&= \text{sumL } [] \quad \text{-- by (SL1)} \\
&= \text{sumL } (\text{flatten } \text{Nil}) \quad \text{-- by (F1)}
\end{align*}
\]
(b) State the induction hypotheses.

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

For proving the step case for \( t = (\text{Node} \ a \ t_1 \ t_2) \),  
the induction hypotheses are

\[
\begin{align*}
\text{sumT} \ t_1 &= \text{sumL} \ (\text{flatten} \ t_1) \quad \text{-- (IH1)} \\
\text{sumT} \ t_2 &= \text{sumL} \ (\text{flatten} \ t_2) \quad \text{-- (IH2)}
\end{align*}
\]

(c) State and prove the step case goal.

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

**Step case: \( t = (\text{Node} \ a \ t_1 \ t_2) \)**  
Show that if the inductive hypotheses (as above) holds, then  
\[
\text{sumT} \ (\text{Node} \ a \ t_1 \ t_2) = \text{sumL} \ (\text{flatten} \ (\text{Node} \ a \ t_1 \ t_2))
\]

**Proof:**

\[
\begin{align*}
\text{sumT} \ (\text{Node} \ a \ t_1 \ t_2) &= a + \text{sumT} \ t_1 + \text{sumT} \ t_2 \quad \text{-- by (ST2)} \\
&= a + \text{sumL} \ (\text{flatten} \ t_1) + \text{sumT} \ t_2 \quad \text{-- by (IH1)} \\
&= a + \text{sumL} \ (\text{flatten} \ t_1) + \text{sumL} \ (\text{flatten} \ t_2) \quad \text{-- by (IH2)} \\
&= \text{sumL} \ (\text{flatten} \ t_1) + a + \text{sumL} \ (\text{flatten} \ t_2) \quad \text{-- by arith} \\
&= \text{sumL} \ (\text{flatten} \ t_1) + \text{sumL} \ (a : (\text{flatten} \ t_2)) \quad \text{-- by (SL2)} \\
&= \text{sumL} \ (\text{flatten} \ t_1 ++ (a : \text{flatten} \ t_2)) \quad \text{-- by (L1)} \\
&= \text{sumL} \ (\text{flatten} \ (\text{Node} \ a \ t_1 \ t_2)) \quad \text{-- by (F2)}
\end{align*}
\]
QUESTION 2 [10 marks]

Logic

(a) Use truth tables to determine whether the following proposition is valid, or is a contradiction, or is a contingency:

\[(p \rightarrow (q \lor r)) \leftrightarrow (p \rightarrow q) \lor (p \rightarrow r)\]

In addition to giving the truth table, your answer needs to explicitly state whether the proposition is valid, or is a contradiction, or is a contingency.

For brevity, the truth table template below uses the abbreviations:

\[ R_1 \equiv (p \rightarrow q) \lor (p \rightarrow r) \]
\[ R_2 \equiv (p \rightarrow (q \lor r)) \leftrightarrow (p \rightarrow q) \lor (p \rightarrow r) \]

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>q</td>
<td>r</td>
<td>q ∨ r</td>
<td>p → (q ∨ r)</td>
<td>p → q</td>
<td>p → r</td>
<td>R₁</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>------</td>
<td>------------</td>
<td>------</td>
<td>------</td>
<td>----</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</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>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</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>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

The formula \( R_1 \) is true for every assignment of truth values to its variables, therefore \( R_1 \) is valid.
This question discusses a computer system where a number of processes may be running. Each process consists of a number of threads. The following predicates are given:

- \( T(x) \) - \( x \) can be terminated
- \( R(x) \) - \( x \) is running
- \( B(x, y) \) - \( x \) belongs to \( y \)

Translate the following sentence into First-Order Logic: Any process can be terminated unless some thread belonging to the process is running.

\[
\forall p. \neg (\exists t. B(t, p) \land R(t)) \rightarrow T(p)
\]

Give a natural deduction proof of \( \forall x. P(x) \land (\exists y. P(y) \rightarrow Q(y)) \rightarrow \exists z. Q(z) \)

You may only use the introduction and elimination rules given in Appendix 1. Number each line and include justifications for each step.

\[
\begin{align*}
1 &\quad (\forall x. P(x)) \land (\exists y. P(y) \rightarrow Q(y)) \\
2 &\quad \forall x. P(x) & \land\text{-E, 1} \\
3 &\quad \exists y. P(y) \rightarrow Q(y) & \land\text{-E, 1} \\
4 &\quad a \quad P(a) \rightarrow Q(a) \\
5 &\quad P(a) & \forall\text{-E, 2} \\
6 &\quad Q(a) & \rightarrow\text{-E, 4, 5} \\
7 &\quad \exists z. Q(z) & \exists\text{-I, 6} \\
8 &\quad \exists z. Q(z) & \exists\text{-E, 3, 4–7}
\end{align*}
\]
QUESTION 3 [11 marks]

Hoare Logic

Consider the following code fragment \textit{Square}, in which all variables are typed integer:

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

This code takes an integer \( n \), and is intended to calculate \( n^2 \) and assign that value to \( s \). To confirm this, we wish to use the rules of Hoare Logic (Appendix 3) to show that

\[
\{ \text{True} \} \ \text{Square} \ \{ s = n^2 \}.
\]

In the questions below (and your answers), we may refer to the loop code as \textit{Loop}, and the body of the loop as \textit{Body}. \textit{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) We will need an invariant for \textit{Loop}. We suggest

\[ Inv \equiv (s = i \times n). \]

Prove that

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

QUESTION 3(a) [4 marks]

1. \( \{s = (i + 1) \times n\} \ i := i + 1 \ \{Inv\} \) (Asst.)

2. \( \{s + n = (i + 1) \times n\} \ s := s + n \ \{s = (i + 1) \times n\} \) (Asst.)

Now

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

So

3. \( \{Inv\} \ s := s + n \ \{s = (i + 1) \times n\} \) (2, Precond. Equiv.)

4. \( \{Inv\} \ \text{Body} \ \{Inv\} \) (1,3, Seq.)
(b) Using the result of part (a), prove that

\{ Inv \} \text{Loop} \ \{ s = n^2 \}.

<table>
<thead>
<tr>
<th>QUESTION 3(b)</th>
<th>[3 marks]</th>
</tr>
</thead>
<tbody>
<tr>
<td>5. ((s = i \times n \land i \neq n) \rightarrow s = i \times n) (Basic Logic)</td>
<td></td>
</tr>
<tr>
<td>6. {s = i \times n \land i \neq n} Body {Inv} \quad ((a),5, Precond. Strength.)</td>
<td></td>
</tr>
<tr>
<td>7. {Inv} \text{Loop} \ { s = i \times n \land \neg (i \neq n) } \quad (6, While)</td>
<td></td>
</tr>
<tr>
<td>8. (s = i \times n \land \neg (i \neq n) \rightarrow s = n^2) (Basic Math.)</td>
<td></td>
</tr>
<tr>
<td>9. {Inv} \text{Loop} \ { s = n^2 } \quad (7,8, Postcond. Weak.)</td>
<td></td>
</tr>
</tbody>
</table>

(c) Using the result of part (b), prove that

\(\{ \text{True} \} \text{Square} \ \{ s = n^2 \} \).

<table>
<thead>
<tr>
<th>QUESTION 3(c)</th>
<th>[3 marks]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10. {0 = i \times n} s := 0 \ {Inv} \quad (Asst.)</td>
<td></td>
</tr>
<tr>
<td>11. {0 = 0 \times n} i := 0 \ {0 = i \times n} \quad (Asst.)</td>
<td></td>
</tr>
<tr>
<td>12. {\text{True}} i := 0 \ {0 = i \times n} \quad (11, Precond. Equiv.)</td>
<td></td>
</tr>
<tr>
<td>13. {\text{True}} i := 0; \ s := 0 \ {Inv} \quad (10,12, Seq.)</td>
<td></td>
</tr>
<tr>
<td>14. {\text{True}} \text{Square} \ { s = n^2 } \quad (13,(b), Seq.)</td>
<td></td>
</tr>
</tbody>
</table>

(d) The code fragment \text{Square} would get stuck in an infinite loop for some initial values of \(n\). Explain why it is not necessary to consider this possibility when choosing a precondition for this code.

<table>
<thead>
<tr>
<th>QUESTION 3(d)</th>
<th>[1 mark]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hoare logic is concerned with \textbf{partial} correctness, meaning that we do not worry about memory states for which the code fails to terminate.</td>
<td></td>
</tr>
</tbody>
</table>
QUESTION 4 [11 marks]
Weakest Precondition Calculus

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

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

\[
\text{Body} \quad \text{Loop} \quad \text{Square}
\]

We will use the rules of the weakest precondition calculus (Appendix 4) to calculate \(wp(Square, s = n^2)\).

As in the previous question we will use the abbreviations \(Loop\) and \(Body\) for the indicated parts of the code. Remember to simplify your answers wherever possible, and show all your working when you do so.

(a) We will need to calculate

\[wp(Loop, s = n^2).\]

First, state \(P_0\) (the predicate expressing success for this weakest precondition after zero loop iterations).

**QUESTION 4(a) [1 mark]**

\[P_0 \equiv (i = n \land s = n^2)\]
(b) We claim that the general format for $P_k$ (expressing success after $k$ loop iterations for $k \geq 0$) is

$$P_k \equiv (i + k = n \land s = i^2 + k \ast i)$$

Suppose that this holds for some $k$. Then prove that

$$P_{k+1} \equiv (i + (k + 1) = n \land s = i^2 + (k + 1) \ast i)$$

**QUESTION 4(b)** [6 marks]

\[
P_{k+1} \equiv i \neq n \land \text{wp}(\text{Body}, P_k) \\
\equiv i \neq n \land \text{wp}(s := s + n, \text{wp}(i := i + 1, P_k)) \\
\equiv i \neq n \land \text{wp}(s := s + n, i + 1 + k = n \land s = (i + 1)^2 + k \ast (i + 1)) \\
\equiv i \neq n \land i + 1 + k = n \land s + n = (i + 1)^2 + k \ast (i + 1) \\
\equiv i + 1 + k = n \land s + n = (i + 1)^2 + k \ast (i + 1) \\
\text{(because } k > 0) \\
\equiv i + 1 + k = n \land s + i + 1 + k = (i + 1)^2 + k \ast (i + 1) \\
\text{(substituting } i + 1 + k \text{ for } n) \\
\equiv i + 1 + k = n \land s + i + 1 + k = i^2 + 2i + 1 + ki + k \\
\equiv i + 1 + k = n \land s = i^2 + ki \\
\equiv i + (k + 1) = n \land s = i^2 + (k + 1) \ast i
\]
(c) Given parts (a) and (b), state

\[ wp(\text{Loop}, s = n^2). \]

Do not attempt any simplification at this stage.

**QUESTION 4(c) [1 mark]**

\[ \exists k. ( k \geq 0 \land i + k = n \land s = i^2 + k \cdot i ) \]

(d) Hence find

\[ wp(\text{Square}, s = n^2). \]

State this result in the simplest form possible.

**QUESTION 4(d) [3 marks]**

\[ \equiv wp(i := 0, wp(s := 0, wp(\text{Loop}, s = n^2))) \]
\[ \equiv wp(i := 0, \exists k. ( k \geq 0 \land i + k = n \land 0 = i^2 + k \cdot i )) \]
\[ \equiv \exists k. ( k \geq 0 \land 0 + k = n \land 0 = 0^2 + k \cdot 0 ) \]
\[ \equiv \exists k. ( k \geq 0 \land k = n ) \]
\[ \equiv n \geq 0 \]
QUESTION 5 [10 marks]

Separation Logic

Prove the following Separation Logic triples by instantiating $P$ and $Q$ using the appropriate axiom from the Separation Logic rules given in the appendix. As your solution, write out the whole triple on one line. You do not need any other rules of Separation Logic or Hoare Logic. You can use multiple lines if you want to do it piece by piece, but justify each step. Please use an abbreviation which uses the first three letters from each word in the name of the axiom, but ending in Axm as shown in the example. Please use the axioms as they are shown in the Appendix as this will make it easier to mark such questions in the quiz, assignment and exam: that is, do not use re-namings of these axioms.

(a) $\{P\} \ x := \text{cons}(y + x) \ \{Q\}$

QUESTION 5(a) [2 marks]

(1) $\{(x = v) \land \text{emp}\} \ x := \text{cons}(y + x) \ \{x \mapsto (y + x)(v/x)\}$ (AllAssAxm)

(2) $\{(x = v) \land \text{emp}\} \ x := \text{cons}(x + y) \ \{x \mapsto (y + v)\}$ (Subst)

(b) $\{P\} \ \text{dispose}(42) \ \{Q\}$

QUESTION 5(b) [2 marks]

(1) $\{42 \mapsto -\} \ \text{dispose}(42) \ \{\text{emp}\}$ (DisAxm)

(c) $\{P\} \ [x - 1] := x + 1 \ \{Q\}$

QUESTION 5(c) [2 marks]

(1) $\{(x - 1) \mapsto -\} \ [x - 1] := x + 1 \ \{(x - 1) \mapsto x + 1\}$ (HeaAssAxm)
(d) \{ P \} x := [x - 1] \{ Q \}

\text{QUESTION 5(d)} \hspace{2cm} [2 \text{ marks}]

(1) \{(x = v_1) \land (x - 1) \mapsto v_2\} x := [x - 1] \{(x = v_2) \land (x - 1)(v_1/x) \mapsto v_2\} \text{ (FetAssAxm)}

(2) \{(x = v_1) \land (x - 1) \mapsto v_2\} x := [x - 1] \{(x = v_2) \land (v_1 - 1) \mapsto v_2\} \text{ (Subst)}

(e) \{ P \} x := x + x \{ Q \}

\text{QUESTION 5(e)} \hspace{2cm} [2 \text{ marks}]

(1) \{x = v \land \text{emp}\} x := x + x \{x = (x + x)(v/x) \land \text{emp}\} \text{ (FloStoAxm)}

(2) \{x = v \land \text{emp}\} x := x + x \{x = v + v \land \text{emp}\} \text{ (Substitution)}
QUESTION 6 [4 marks]

Suppose that we are given some fixed store $St$ which maps $x$ and $y$ to $St(x)$ and $St(y)$, and that these are different locations: that is, $St(x) \neq St(y)$.

Suppose that $\text{dom}(Hp1) = St(x)$ and $Hp1(St(x)) = 1$ and $\text{dom}(Hp2) = St(y)$ and $Hp2(St(y)) = 2$. That is, each of $Hp1$ and $Hp2$ are singleton heaps, they are disjoint, and $Hp1$ maps the location $St(x)$ to the value 1 and $Hp2$ maps the location $St(y)$ to the value 2.

The notation $Hp1 \bullet Hp2$ means the heap that is formed by combining heaps $Hp1$ and $Hp2$. Let $Hp = (Hp1 \bullet Hp2)$: that is, the heap $Hp$ is the combination of the heaps $Hp1$ and $Hp2$.

For each part, write down all of the heaps from $Hp1$, $Hp2$, $Hp$ that make the condition true, or else write “none” if you think none of these three heaps makes the condition true, and give a reason of two or three lines in plain English.

(a) $x \mapsto 1 \land x \mapsto 1$

$$
\text{QUESTION 6(a) [2 marks]}
$$

$Hp1$ because it makes $x \mapsto 1$ true

(b) $\neg (x \mapsto 1)$

$$
\text{QUESTION 6(b) [2 marks]}
$$

$Hp$ because it is not of size 1; and

$Hp2$ because it’s domain does not contain $St(x)$
Appendix 1 — Natural Deduction Rules

Propositional Calculus

\[(\& I)\] \[\frac{p \quad q}{p \& q}\] \n \n\[(\& E)\] \[\frac{p \& q}{p}\] \n \n\[(\& E)\] \[\frac{p \& q}{q}\] \n \n\[(\lor I)\] \[\frac{p}{p \lor q}\] \n \n\[(\lor I)\] \[\frac{p}{q \lor p}\] \n \n\[(\lor E)\] \[\frac{p \lor q \quad r}{r}\] \n \n\[(\rightarrow I)\] \[\frac{q}{p \rightarrow q}\] \n \n\[(\rightarrow I)\] \[\frac{q}{q \lor p}\] \n \n\[(\rightarrow E)\] \[\frac{p}{p \rightarrow q}\] \n \n\[(\rightarrow E)\] \[\frac{p}{q}\] \n \n\[(\neg I)\] \[\frac{q \land \neg q}{\neg p}\] \n \n\[(\neg I)\] \[\frac{q \land \neg q}{q \land \neg q}\] \n \n\[(\exists I)\] \[\frac{P(a)}{\exists x. P(x)} \quad (a \text{ arbitrary})\] \n \n\[(\exists I)\] \[\frac{P(a)}{\exists x. P(x)} \quad (a \text{ arbitrary})\] \n \n\[(\exists E)\] \[\frac{\exists x. P(x) \quad q}{q} \quad (a \text{ is not free in } q)\] \n
Predicate Calculus

\[(\forall I)\] \[\frac{P(a)}{\forall x. P(x)} \quad (a \text{ arbitrary})\] \n \n\[(\forall I)\] \[\frac{P(a)}{\forall x. P(x)} \quad (a \text{ arbitrary})\] \n \n\[(\forall E)\] \[\frac{\forall x. P(x)}{P(a)}\] \n \n\[(\forall E)\] \[\frac{\forall x. P(x)}{P(a)}\] \n \n\[(\exists E)\] \[\frac{\exists x. P(x) \quad q}{q} \quad (a \text{ is not free in } q)\] \n
COMP2600 (Formal Methods for Software Engineering) Page 16 of 20
### Appendix 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>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Appendix 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:
  \[ \{Q(e)\} x := e \{Q(x)\} \]

- 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:
  \[ \{P \land b\} S \{P\} \]
  \[ \{P\} \text{ while } b \text{ do } S \{P \land \neg b\} \]
Appendix 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 \( Q \).

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

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

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

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

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

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

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

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

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

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

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