All Rules on One Slide

The Assignment Rule

$$\frac{}{\{P\ [e/x]\}\ x := e\ \{P\}}$$

Sequencing

$$\frac{\{P\}\ C_1\ \{Q\} \quad \{Q\}\ C_2\ \{R\}}{\{P\}\ C_1;\ C_2\ \{R\}}$$

If Then Else

$$\frac{\{P \text{ and } B\}\ C_1\ \{Q\} \quad \{P \text{ and (not } B)\}\ C_2\ \{Q\}}{\{P\}\ \textbf{if}\ B\ \textbf{then}\ C_1\ \textbf{else}\ C_2\ \textbf{fi}\ \{Q\}}$$

Precondition Strengthening

$$\frac{P \rightarrow P' \quad \{P'\}\ C\ \{Q\}}{\{P\}\ C\ \{Q\}}$$

Postcondition Weakening

$$\frac{\{P\}\ C\ \{Q'\} \quad Q' \rightarrow Q}{\{P\}\ C\ \{Q\}}$$

Rule of Consequence

$$\frac{P \rightarrow P' \quad \{P'\}\ C\ \{Q'\} \quad Q' \rightarrow Q}{\{P\}\ C\ \{Q\}}$$

While

$$\frac{\{\ P \text{ and } B\ \}\ \ C\ \ \{\ P\ \}}{\{\ P\ \}\ \textbf{while}\ \ B\ \ \textbf{do}\ \ C\ \textbf{od}\ \ \{\ P \text{ and not } B\ \}}$$

Hoare Logic (40 pts)

Determine the truth value of the following Hoare triples. All values are integers. Show your work and explain your reasoning. The above slide contains some convenient rules you can use; however, we encourage you to read the full lecture slides.

1. `{i = j} i:=j-2*i {i + j = 0}` **[5 pts]**

{ i = j } i := j - 2 * i { i + j = 0 } (Postcondition Weakening) =>
{ i = j } i := j - 2 * i { i = - j }

Since i = j, i := j - 2 * i <==> i := j - 2 * j <==> i := - j
Thus, the Hoare triple is correct.

2. `{i = a} i:= j + i ; i:=j-i {i =  a}` **[5 pts]**

{ i = a } i := j + i ; i := j - i { i = a } (Sequencing) =>
{ i = a } i := j + i { i = j + a }, { i = j + a } i := j - i { i = a }

In the first Hoare triple, since i = a, i := j + i <==> i := j + a
The first Hoare triple is correct.

In the second Hoare triple, since i = j + a, i := j - i <==> i := j - (j + a) <==> i := j - j - a
<==> i := -a
The second Hoare triple is not correct.

Thus, this Hoare triple is not correct.

*The blank space in the post-condition suggests there was a typo and the post-condition should have been { i = -a }. If that were the case then the Hoare triple is correct. However, since it's missing, the Hoare triple is not correct.

3. `{i != j} if i<j then m:=j-i else m:=j+i fi {m > 0}` **[5 pts]**

{ i != j } if i < j then m := j - 1 else m := j + 1 fi { m > 0 }

Let i = -2, j = -1

This satisfies the pre-condition where i != j

Then applying small-step semantics to the equation with the specified variables gives us:
(if i < j then m := j - 1 else m := j + 1 fi, { i -> -2, j -> -1 })
-> (if -2 < j then m := j - 1 else m := j + 1 fi, { i -> -2, j -> -1 })
-> (if -2 < -1 then m := j - 1 else m := j + 1 fi, { i -> -2, j -> -1 })
-> (if True then m := j - 1 else m := j + 1 fi, { i -> -2, j -> -1 })
-> (m := j - 1, { i -> -2, j -> -1 })
-> (m := -1 - 1, { i -> -2, j -> -1 })
-> (m := -2, { i -> -2, j -> -1 })
-> ({ i -> -2, j -> -1, m := -2 })

However, this does not satisfy the post-condition of the Hoare triple, which states m > 0.

Thus, the Hoare triple is not correct.

4. {x < a} while x<a do x:=x+1 {x = a} **[5 pts]**

{ x < a } while x < a do x := x + 1 { x := a }

We need to find a condition P that is true before and after the loop is executed, such that
(P and not x < a) => x := a

Let P = x ≤ a

Then (P and not x < a) <==> (x ≤ a and not x < a) <==> (x ≤ a and x ≥ a) => x := a

We know that x < a => x ≤ a.

Thus:
{ x < a } while x < a do x := x + 1 { x := a } (Precondition Strengthening) =>
{ x ≤ a } while x < a do x := x + 1 { x := a } (Postcondition Weakening) =>
{ x ≤ a } while x < a do x := x + 1 { x ≤ a and x ≥ a } <==>
{ x ≤ a } while x < a do x := x + 1 { x ≤ a and not x < a } (While) =>
{ x ≤ a and x < a } x := x + 1 { x ≤ a } <==>
{ x < a } x := x + 1 { x ≤ a } (Assignment Rule) ==>
{ x + 1 ≤ a } x := x + 1 { x ≤ a }

Since all values must be integers, we know x < a <==> 0 < a - x <==> 1 ≤ a - x <==> x + 1 ≤ a.

Thus, the Hoare triple is correct.

5. `{x = a} while x <= a do x:= x {x < a}` **[5 pts]**

The while loop will never terminate so long as x ≤ a. Thus:
{ x = a } while x ≤ a do x := x { x < a } <==>
{ x = a } while x ≤ a do x := x { x < a and not x ≤ a } <==>
{ x = a } while x ≤ a do x := x { x < a and x > a }
{ x = a } while x ≤ a do x := x { False }

Thus, the Hoare triple is not correct.

6. Prove the given statements about the following programs using Hoare logic. **[15 pts]**

```
1  {0 ≤ s and u < |a|}
2  bool LinearSearch(int[] a, int s, int u, int e) {
3      int i=s;
4      while(i<=u){
5          if(a[i]==e) return true;
6          i++;
7      }
8      return false;
9  }
10 {ret_LinearSearch ⇔ ∃i.s ≤ i ≤ u and a[i] = e}
```

Listing 1: (a)

{ 0 ≤ s and u < |a| }
i := s;
while i ≤ u do
    (if a[i] = e then Return := True else i += 1 fi)
od;
Return := False
{ Return <==> ∃ i s.t. s ≤ i ≤ u and a[i] = e } (Sequence) =>
{ 0 ≤ s and u < |a| } i := s { 0 ≤ s and u < |a| and i := s }
{ 0 ≤ s and u < |a| and i := s }
while i ≤ u do
    (if a[i] = e then Return True else i += 1 fi)
od
Return False
{ Return <==> ∃ i s.t. s ≤ i ≤ u and a[i] = e }

The Hoare triple, { 0 ≤ s and u < |a| } i := s { 0 ≤ s and u < |a| and i := s }, is trivially correct.

We now need to show:
{ $0 \leq s$ and $u < |a|$ and $i := s$ }
while $i \leq u$ do
    (if $a[i] = e$ then Return True else i += 1 fi)
od
Return False
{ Return <==> $\exists$ i s.t. $s \leq i \leq u$ and $a[i] = e$ }

Return can be either True or False. If it is True, then $\exists$ i s.t. $s \leq i \leq u$ and $a[i] = e$. If it is False, then $\neg(\exists$ i s.t. $s \leq i \leq u$ and $a[i] = e)$

Notice that the function ends and Returns when $a[i] = e$, or when $i = u + 1$. In the former case, Return = True, and in the latter case, Return = False.

Then: $a[i] = e$ => $\exists$ i s.t. $s \leq i \leq u$ and $a[i] = e$
And: $i = u + 1$ => $\neg(\exists$ i s.t. $s \leq i \leq u$ and $a[i] = e)$ <==> $\forall$ i s.t. $s \leq i \leq u$, $a[i]$!= e

Using Postcondition Weakening, we obtain a new Postcondition:
{ (Return and $\exists$ i s.t. $s \leq i \leq u$ and $a[i] = e$) or (Not Return and $\forall$ i s.t. $s \leq i \leq u$, $a[i]$!= e) }

This gives us:
{ $0 \leq s$ and $u < |a|$ and $i := s$ }
while $i \leq u$ do
    (if $a[i] = e$ then Return True else i += 1 fi)
od
Return False
{ (Return and $\exists$ i s.t. $s \leq i \leq u$ and $a[i] = e$) or (Not Return and $\forall$ i s.t. $s \leq i \leq u$, $a[i]$!= e) }

Next, $0 \leq s$ and $u < |a|$ and $i := s$ => $s \leq i \leq u$.
Thus, we can use Precondition Strengthening to obtain a new Precondition:
{ $s \leq i \leq u$ }

This gives us:
{ $s \leq i \leq u$ }
while $i \leq u$ do
    (if $a[i] = e$ then Return True else i += 1 fi)
od
Return False
{ (Return and $\exists$ i s.t. $s \leq i \leq u$ and $a[i] = e$) or (Not Return and $\forall$ i s.t. $s \leq i \leq u$, $a[i]$!= e) }

Next, we know that if a[i] != e, then i is incremented by i. Thus, it follows that:
$\forall$ x, s.t. $s \leq x < i$, a[i] != e.

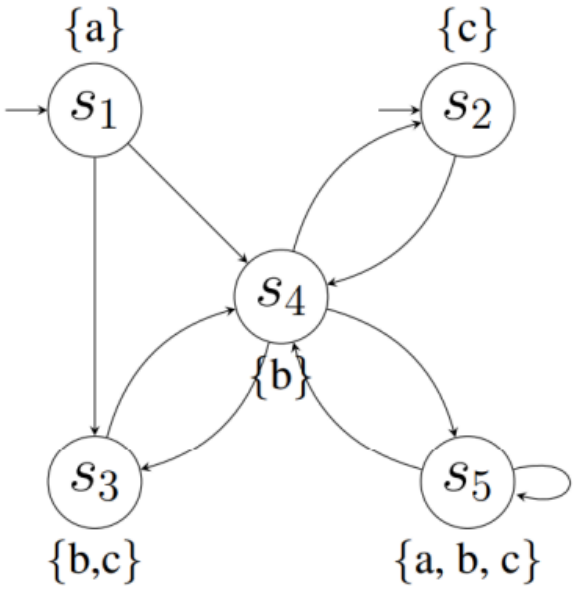
This gives us:
{ $s \leq i \leq u$ and $\forall$ x, s.t. $s \leq x < i$, a[i] != e}
while $i \leq u$ do
    (if a[i] = e then Return True else i += 1 fi)
od
Return False
{ (Return and $\exists$ i s.t. $s \leq i \leq u$ and a[i] = e) or (Not Return and $\forall$ i s.t. $s \leq i \leq u$, a[i] != e) }

We can see that if Return is True, then $s \leq i \leq u$, and a[i] = e.
Additionally, if Return is False, then $i > u$, and $\forall$ x, s.t. $s \leq x < i$, a[i] != e.

Thus, the given statements are correct.

LTL - Theory (20 pts)

For the next few questions, we will be working with the following transition system TS over the set of atomic propositions {a, b, c} and starting states s_1, s_2.



Decide for each of the following LTL formulae φ_i whether TS $\models \varphi_i$ holds. Provide a brief explanation for your answer in words. There is no requirement to provide a formal proof. If TS $\not\models \varphi_i$, provide a path $\pi \in Paths(TS)$ such that $\pi \not\models \varphi_i$

7. $\varphi_1 = \circ \neg c \Rightarrow \circ \circ c$ **[5 pts]**

This formula asserts that if the next state from the starting state is ¬c, then it implies the next state after that must have c.

From s_1 and s_4, the only next state with ¬c is s_4. However, every other state that s_4 can go to has c. Thus, the formula holds.

8. $\varphi_2 = \Box a$ **[5 pts]**

This formula asserts that a is always True.

Let π = s_2. s_2 has ¬a. Thus, the formula doesn't hold.

9. $\varphi_3 = a \cup \Box(b \vee c)$ **[5 pts]**

This formula asserts that a will be True until either b or c is always True.

If you start at s_1, you are forced to go to either s_4 or s_3. Once you do, you will only be able to travel to either s_2, s_3, s_4, or s_5, where (b $\vee$ c) holds for all 4 states. A similar result occurs if you start in s_2.

Thus, the formula holds.

10. $\varphi_4 = (^{\circ\circ}b) \cup (b \vee c)$ **[5 pts]**

This formula asserts that moving two steps at a time will always place you in a state where b is True, and when that is no longer true, you will reach a state where b or c is True.

Moving two steps from either starting state will always put you on either s_2, s_3, s_4, or s_5. Of those states, only s_2 doesn't contain b. Since, s_2 contains c, landing on s_2 will stop the until condition and satisfy the statement (b $\vee$ c).

Thus, the formula holds.