Network Security

Weakest Preconditions

Weakest Preconditions

- The Hoare Logic Formalism
- The WP Formalism
- WP Definitions for Protocol Operations
- Concatenating statements
- Proving Protocol Properties using WP

Hoare Logic

1. Precondition
2. Program statement or segment
3. Postcondition

\{P\} \gamma \{Q\}
Hoare Logic

- The program segment is defined in terms of the precondition that must exist immediately before execution of the segment in order to ensure that the postcondition is true after execution of the segment.

A Simple Example

1. P: \( X == 3 \)
2. \( \gamma: Y := X + 7 \)
3. Q: \( Y == 10 \)

\( \{P\} \gamma \{Q\} \)
\( \{X == 3\} Y := X + 7 \{Y == 10\} \)

To be Usable, we define the meaning of the Assignment Operator
\( \{Q[x|t]\} x := t \{Q\} \)
Select \( S \) as \( x := y + 4 \) and \( Q \) as \( x == 7 \)
\( \{y + 4 == 7\} x := y + 4 \{x == 7\} \)
\( \{y == 3\} x := y + 4 \{x == 7\} \)

Any proposition dependent on \( x \) will be true after an assignment statement, if the same proposition with all instances of \( x \) replaced by \( t \) were true before the assignment statement.
Weakest Preconditions

- We define the weakest precondition to be:
  - The condition that characterizes the set of ALL initial states such that activation will certainly result in a proper termination leaving the system in a final state satisfying a given post-condition.
- We call this the "weakest precondition corresponding to the given postcondition".
- It is weakest, because the weaker a condition, the more states satisfy it. We are aimed at characterizing all possible starting states.

R = \text{wp}(S,Q)

Find R = \text{wp}(\{y:= x/k\}, (x == ky)) where x,y,k \in \text{Real } \#s

By guessing, select R = (k == 3)

Is Hoare triple (k == 3){y:= x/k}(x == ky) true?
  - (sure) (y == x/3) \implies (x == 3y)

Is (k==3) the weakest precondition?

How about (k > 0)? Precondition yes; wp no.

How about (k \neq 0)? THAT's IT!

Weakest Precondition of the Assignment Operator

R = \text{wp}(\{x := t \},Q(x))
R = Q(t) or
Q(t) = \text{wp}(\{x := t \},Q(x))

Any proposition dependent on x will be true after an assignment statement, if the same proposition with all instances of x replaced by t were true before the assignment statement.
WP for the Assignment Operator

\[ R = \text{wp}(x := t, Q(x)) \]
\[ = \text{wp}(x := y + 4, \{x == 7\}) \]
\[ = (y + 4 == 7) \]
\[ = (y == 3) \]

WP for Catenation

• The Hoare Logic Definition, roughly:

\[ \neg R \{S1; S2\} Q \iff R \{S1\} Q' \text{ and } Q' \{S2\} Q \]

• Notice the intermediate condition Q'. R guarantees Q after S1:S2 if and only if R guarantees Q' after S1 and Q' guarantees Q after S2.

• \[ R = \text{wp}(S1; S2, Q) = \text{wp}(s1, \text{wp}(s2, Q)) \]

• Notice the recursive nature of this definition

WP Catenation Exercise

• Here’s the program:
  
  S1 is \{x := 5x;\}
  S2 is \{x := x + 3\}
  Q is \{x == 28\}

• Find R = \text{wp}(s1; s2, Q) where R = \text{wp}(S1; S2, Q)

\[ = \text{wp}(x := 5x; x := x + 3, x == 28) \text{ Given} \]
\[ = \text{wp}(x := 5x, x := x + 3, x == 28) \text{def catenation} \]
\[ = \text{wp}(x := 5x, x + 3 == 28) \text{ wp}(S2, Q) \]
\[ = \text{wp}(x := 5x, x == 25) \text{ Math} \]
\[ = (5x == 25) \text{ wp}(S1, P) \]
\[ = (x == 5) \text{ Math} \]
WP Catenation Exercise

- For a statement that will compute two weighted grades to come up with a final grade, write a test program that will ensure that two grades of 100 result in a grade of 100.

- Here's the program:
  
g1 := 100;
g2 := 100;
grade := g1*w1 + g2 * w2;

- Find R = wp (s1;s2;s3, Q) where
  - s1 is g1 := 100; and
  - s2 is g2 := 100; and
  - s3 is grade := g1*w1 + g2 * w2; and
  - Q is (grade == 100)

WP Catenation Exercise

- Find R = wp (s1;s2;s3, Q) where
  s1 is {g1 := 100;} and s2 is {g2 := 100;} and
  s3 is {grade := g1*w1 + g2 * w2;} and
  Q is (grade == 100)

\[ R = wp(s1,(wp(s2,wp(s3,Q)))) = wp(s1,wp(g2:=100;,g1*w1+g2*w2 == 100)) (wp s3 & subs) = wp(g1:=100;,g1*w1+100*w2 == 100) (wp s2 & subs) = (100w1+100w2 == 100) (wp s1 & subs) = (w1 + w2) == 1 (arithmetic) \]

If the weights add up to 1, the pgm meets it's goal!

The Formality of Semantics

\[ (100w1+100w2 == 100) = (w1 + w2) == 1 \quad \text{(arithmetic)} \]

Notice that we have no formal rule to translate (100w1+100w2 == 100) into (w1 + w2) == 1

So, unless we have a semantic representation for arithmetic in our automated prover system, this must be manual.
WP For Conditionals

R = wp (if P s1 else skip, Q)
= (P => wp(s1;Q)) Λ (¬P => wp(skip,Q))

In order for R to guarantee Q after execution of an ifnoelse statement the conditional P must imply the weakest precondition of s1 and Q, and ¬P must imply Q.

A Helpful Simplification

• P => Q ≣ ¬P V Q

• (P => wp(s1;Q)) Λ (¬P => wp(skip,Q))

is equivalent to:

• (¬P V wp(s1;Q)) Λ (P V wp(skip,Q)) (DeMorgan)

WP Conditional Exercise

• Here’s the program (ifthenelse):
  if (x == a) then y := b;  else y := b;
  Goal:  (y == b)
• Find R = wp (if P then s1 else s1, Q)
  = wp (if (x == a) then y := b else y:=b, (y == b))
  = (x == a) => wp(y:=b, y==b) Λ ¬(x == a) => wp(y:=b,(y == b))
  = (x == a) => b==b Λ ¬(x == a) => (b == b) wp asgn
  = ¬(x == a) or b==b Λ (x == a) or (b == b) DeMorgan
  = ¬(x == a) or T  Λ (x == a) or T  Simp
  =  T  Λ  T  Simp
  =  T  Simp
  =  T  Simp
WP Conditional Exercise

- Here's another program (ifnoelse):
  ```
  if (x == a) then y := b;
  Goal:  (y == b)
  ```
- Find \( R = \text{wp} \left( \text{if } P \text{ then } s_1, Q \right) \)
  
  \[
  \begin{align*}
  \text{wp (if } x == a \text{ then } y := b, \ (y == b)) \\
  &= (x == a) \Rightarrow \text{wp (y := b, y == b)} \land \neg(x == a) \Rightarrow \text{wp (skip, y == b)} \\
  &= (x == a) \Rightarrow \text{wp (y := b, y == b)} \land \neg(x == a) \Rightarrow (y == b)) \quad \text{wp skip} \\
  &= (x == a) \Rightarrow b == b \land \neg(x == a) \Rightarrow (y == b)) \quad \text{wp asgn} \\
  &= \neg(x == a) \lor b == b \land (x == a) \lor (y == b)) \quad \text{DeMorgan} \\
  &= \neg(x == a) \lor T \land (x == a) \lor (y == b)) \quad \text{Simp(b == b)} \\
  &= T \land (x == a) \lor (y == b)) \quad \text{Simp PVT} \\
  &= (x == a) \lor (y == b)) \quad \text{Simp TAP}
  \end{align*}
\]

Four Properties of WPs

1. For any \( S \), \( \text{wp}(S,F) = F \)
2. For any segment \( S \) and conditions \( Q,R \):
   \( \text{if } Q \Rightarrow R, \text{then } \text{wp}(S,Q) \Rightarrow \text{wp}(S,R) \)
3. For any segment \( S \) and conditions \( Q,R \):
   \( (\text{wp}(S,Q) \text{ and wp}(S,R)) = \text{wp}(S,Q \text{ and } R) \)
4. For any segment \( S \) and conditions \( Q, R \)
   \( (\text{wp}(S,Q) \text{ or wp}(S,R)) = \text{wp}(S,Q \text{ or } R) \)

WP Exercise

Show that the weakest precondition definition of catenation
\( (\text{wp}(S_1;S_2, Q) = \text{wp}(S_1, \text{wp}(S_2, Q)) \)

has property #1

(For any \( S \), \( \text{wp}(S,F) = F \))
Proof that Prop 1 holds for Catenation

Show that $wp(S, F) = F$ holds for
\[ wp(S_1; S_2, Q) = wp(S_1, wp(S_2, Q) \]
\[ wp(S_1; S_2, F) = wp(S_1, wp(S_2, F)) \] Subst
\[ = wp(S_1, F) \] Prop 1
\[ = F \] Prop 1 QED

Weakest Preconditions and Security Protocols

What semantic meaning do we want "SEND" statements to have?
1. Copy a value from one user to another user
2. Copy a value to another user's queue
   • This would mean a user would need to execute a RECEIVE statement in order to read their messages
3. Copy a value from the sender to a man-in-the-middle

Weakest Preconditions and Security Protocols

Let's use the queue paradigm
\[ wp (A->B A.X, Q(Bq.X)) = Q (A.X) \]
\[ wp (B <- Bq.X, Q(B.X)) = Q (Bq.X) \]
Protocol Exercise

TRUE
(A.na == A.na)
A: -> B(na);  (Bq.na == A.na)
B: <- na;     (B.na == A.na)
B: -> A (na);  (Aq.na == A.na)
A: <- na';    (A.na' == A.na)

Review

- Preconditions & Postconditions
- Assignment, Catenation, Conditionals
- Some WP Properties
- Examples