

Weakest Precondition Calculus III

COMP2600 — Formal Methods for Software Engineering

Ranald Clouston

Australian National University

Semester 2, 2011

Example 3 — Sum of n Odd Numbers

We proved this program in Hoare Logic:

Program (with specification):

```
{ $n \geq 0$ }  
i:=0;  
s:=0;  
while (i  $\neq$  n) do  
    i:=i+1;  
    s:=s+(2*i-1)  
{ $s = n^2$ }
```

(The sum of the first n odd numbers gives their square.)

Proof Summary

Let W be the while loop.

Let P_k be the precondition that assures termination and $s = n^2$ after k iterations. We will calculate that:

$$P_0 \equiv ((i = n) \wedge (s = n^2))$$

$$P_1 \equiv ((i + 1 = n) \wedge (s = i^2))$$

$$P_2 \equiv ((i + 2 = n) \wedge (s = i^2)) \dots$$

By induction, $P_k \equiv ((i + k = n) \wedge (s = i^2))$

This will allow us to define $wp(W, (s = n^2))$.

We will then push that precondition back through the initialisation code to get the *program's* weakest precondition.

Some Details:

Computing $P_0 \equiv \neg b \wedge Q$

$$\begin{aligned} P_0 &\equiv \neg (i \neq n) \wedge (s = n^2) \\ &\equiv (i = n) \wedge (s = n^2) \end{aligned}$$

Computing $P_1 \equiv b \wedge wp(S, P_0)$

$$\begin{aligned} P_1 &\equiv (i \neq n) \wedge wp(\mathbf{i:=i+1; s:=s+2*i-1}, ((i = n) \wedge (s = n^2))) \\ &\equiv (i \neq n) \wedge wp(\mathbf{i:=i+1}, \\ &\quad wp(\mathbf{s:=s+2*i-1}, ((i = n) \wedge (s = n^2)))) \\ &\equiv (i \neq n) \wedge wp(\mathbf{i:=i+1}, ((i = n) \wedge (s + 2 * i - 1 = n^2))) \\ &\equiv (i \neq n) \wedge (i + 1 = n) \wedge (s + 2 * (i + 1) - 1 = n^2) \\ &\equiv (i + 1 = n) \wedge (s = i^2) \end{aligned}$$

More Details — P_{k+1} by induction on k

Computing $P_{k+1} \equiv b \wedge wp(S, P_k)$

Inductive hypothesis: $P_k \equiv ((i + k = n) \wedge (s = i^2))$

$$\begin{aligned} P_{k+1} &\equiv (i \neq n) \wedge wp(\mathbf{i:=i+1; s:=s+2*i-1}, \\ &\quad ((i + k = n) \wedge (s = i^2))) \\ &\equiv (i \neq n) \wedge wp(\mathbf{i:=i+1}, \\ &\quad wp(\mathbf{s:=s+2*i-1}, ((i + k = n) \wedge (s = i^2)))) \\ &\equiv (i \neq n) \wedge wp(\mathbf{i:=i+1}, ((i + k = n) \wedge (s + 2 * i - 1 = i^2))) \\ &\equiv (i \neq n) \wedge ((i + 1) + k = n) \wedge (s + 2 * (i + 1) - 1 = i^2) \\ &\equiv (i + (k + 1) = n) \wedge (s = i^2) \end{aligned}$$

More Details – computing wp

Computation of $wp(W, (s = n^2))$:

$$\begin{aligned}wp(W, (s = n^2)) &\equiv \exists k. ((k \geq 0) \wedge P_k) \\ &\equiv \exists k. ((k \geq 0) \wedge (i + k = n) \wedge (s = i^2)) \\ &\equiv (s = i^2) \wedge \exists k. ((k \geq 0) \wedge (i + k = n)) \\ &\equiv (s = i^2) \wedge i \leq n\end{aligned}$$

Computation of weakest precondition of program:

$$\begin{aligned}wp(i:=0; s:=0; W, (s = n^2)) &\equiv wp(i:=0, wp(s:=0; W, (s = n^2))) \\ &\equiv wp(i:=0, wp(s:=0, wp(W, (s = n^2)))) \\ &\equiv wp(i:=0, wp(s:=0, (s = i^2 \wedge i \leq n))) \\ &\equiv wp(i:=0, (0 = i^2 \wedge i \leq n)) \\ &\equiv (0 = 0^2 \wedge 0 \leq n) \\ &\equiv 0 \leq n\end{aligned}$$

A Note on Simplifying Existential Quantifiers

On the last slide we simplified $\exists k. (k \geq 0) \wedge (i + k = n)$ to $i \leq n$.

Why is this correct? Simply recall that both are equivalent to

$$(i + 0 = n) \vee (i + 1 = n) \vee (i + 2 = n) \vee \dots$$

This is, again, a very useful trick!

Interesting Special Cases

- $wp(S, Q) \equiv True$ says that S terminates given *any* initial state and produces a state satisfying Q .
- $wp(S, Q) \equiv False$ says that there are *no* initial states for which S terminates with a state satisfying Q .
- $wp(S, True)$ provides the weakest precondition for which S terminates. It is therefore a meaningful thing to calculate.
(This contrasts with Hoare Logic, where $\{P\}S\{True\}$ always holds).
- $wp(S, False) \equiv False$ always.

Our next example will use this last fact.

Example 4

Consider this program *Prog*:

```
while (n ≠ 0) do
  n:=n-1;
  i:=i*2
```

where *n*, *i* are typed integer.

Say we wish to find $wp(\textit{Prog}, i = 2)$.

$$P_0 \equiv (n = 0) \wedge (i = 2)$$

$$P_1 \equiv (n \neq 0) \wedge wp(\mathbf{n := n - 1; i := i * 2}, P_0)$$

$$\equiv (n = 1) \wedge (i = 1)$$

$$P_2 \equiv (n \neq 0) \wedge wp(\mathbf{n := n - 1; i := i * 2}, P_1)$$

$$\equiv \mathbf{False}$$

Example 4 Ctd.

We now plug $P_2 \equiv \text{False}$ into our next step.

$$\begin{aligned} P_3 &\equiv (n \neq 0) \wedge wp(\mathbf{n := n - 1; i := i * 2, \mathbf{False}}) \\ &\equiv \mathbf{False} \end{aligned}$$

We can see that $P_k \equiv \text{False}$ for all $k > 1$.

We can hence find our weakest precondition:

$$\begin{aligned} &\exists k. ((k \geq 0) \wedge P_k) \\ &\equiv ((n = 0) \wedge (i = 2)) \vee ((n = 1) \wedge (i = 1)) \vee \text{False} \vee \text{False} \vee \dots \\ &\equiv ((n = 0) \wedge (i = 2)) \vee ((n = 1) \wedge (i = 1)) \end{aligned}$$

The intuition is that **if a loop cannot terminate after k steps then it cannot terminate after any larger number of steps.**

Other Properties

- Hoare Logic and Weakest Preconditions are related as follows:
 - $\{wp(S, Q)\} S \{Q\}$
 - If $\{P\} S \{Q\}$, and S terminates given any starting state satisfying P , then $P \rightarrow wp(S, Q)$
- If $Q \rightarrow R$ then $wp(S, Q) \rightarrow wp(S, R)$.
- $wp(S, Q \wedge R) \equiv (wp(S, Q) \wedge wp(S, R))$

What about

- $wp(S, \neg Q)$ and $\neg wp(S, Q)$?
- $wp(S, Q \vee R)$ and $(wp(S, Q) \vee wp(S, R))$?

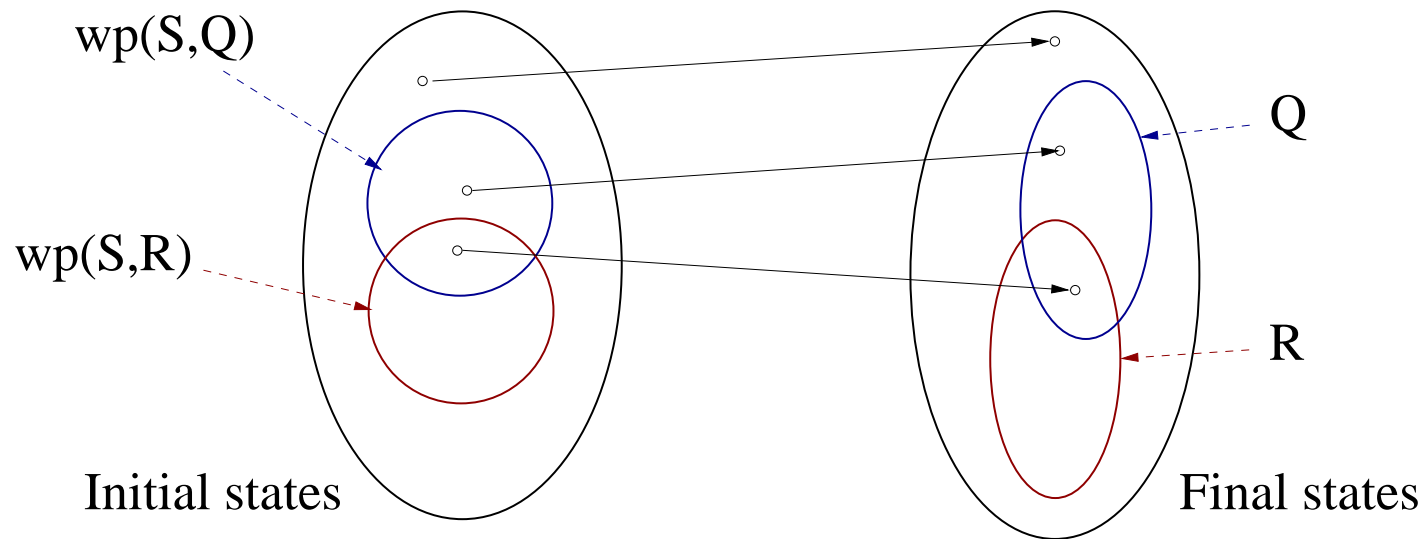
We'll come back to these questions.

Proof of Conjunction Theorem

- Want to prove: $wp(S, Q \wedge R) \equiv (wp(S, Q) \wedge wp(S, R))$
- Suppose initial state, σ , satisfies $wp(S, Q) \wedge wp(S, R)$.
 S will terminate starting in this state (σ) and then both Q and R will hold.
Thus σ satisfies $wp(S, Q \wedge R)$.
- Suppose now σ satisfies $wp(S, Q \wedge R)$.
Thus S terminates and, when it does, both Q and R hold.
Hence σ satisfies $wp(S, Q)$ and also satisfies $wp(S, R)$.
Hence σ satisfies $wp(S, Q) \wedge wp(S, R)$.

Conjunction Theorem - Diagrammatic View

The following diagram shows some of the cases; it is left to you to extend it to show all possible cases (including those on which S does not terminate).



How About Negation?

- A conjecture we might come up with is:

$$wp(S, \neg Q) \equiv \neg wp(S, Q)$$

This is invalid for any language with loops (because of nontermination):

Let S be the program `while 0 \neq 1 do x:=x` that loops forever, and let Q be the predicate *False*.

Now

$$wp(S, \neg Q) \equiv wp(S, True) \equiv False$$

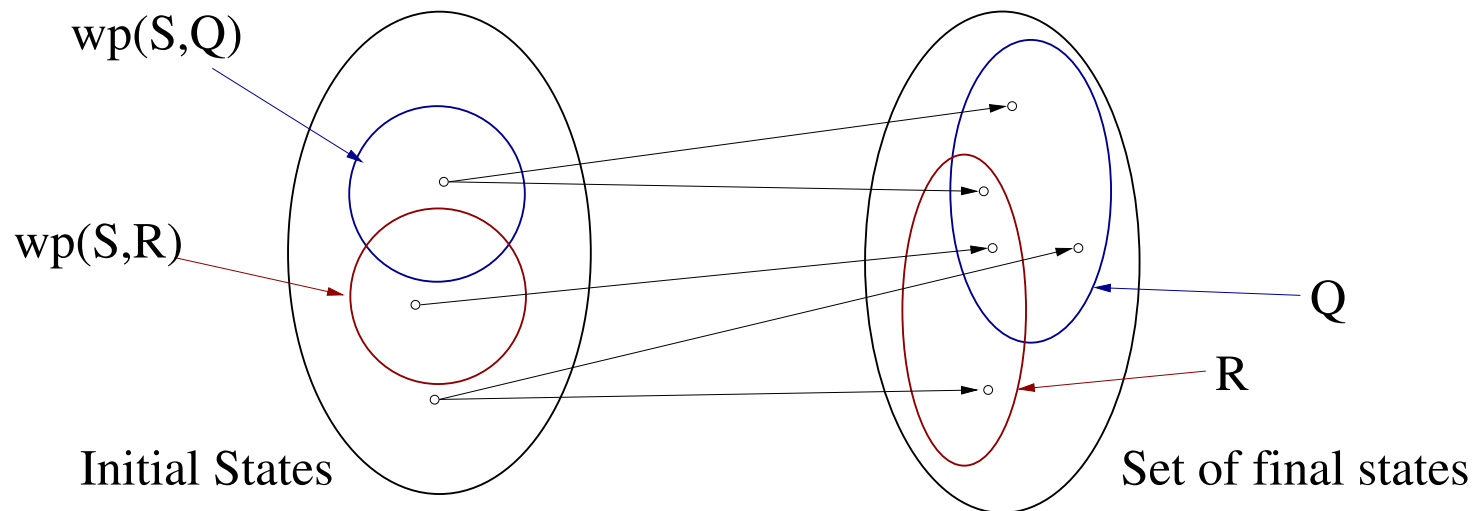
$$\neg wp(S, Q) \equiv \neg wp(S, False) \equiv \neg False \equiv True$$

How About Disjunction?

- The following is a likely looking proposition:

$$wp(S, Q \vee R) \equiv wp(S, Q) \vee wp(S, R)$$

- Indeed it *is* true for the imperative language we have seen.
- (but if we had **non-determinism** the left-to-right implication would hold but the other way would not, as the diagram below shows.)



Reflections

At the end of this section on [Formal Verification of Imperative Programs](#), we should ask why it is deemed worthwhile.

In particular, can we use these techniques to prove programs?

- Not for arbitrary programs - the language is too small and pure.
- However, the notation captures things we want to say.
- And the rules are valid in restricted scope
 - Recall the assignment axiom.
 - Conditional rules can help thinking.
 - Both `while` rules (Hoare and *wp*) aid intuition for arguing correctness.

This is foundational mathematics for programming; practical systems follow.