

## (Dynamic) Semantics of Programs

- Operational – what the program does
- Denotational – the function computed by the program.
- Axiomatic – what is true after execution

We want to prove theorems of the form:

$$\{P\} S \{Q\} \quad \text{where}$$

$\{P\}$  and  $\{Q\}$  are sets of assertions about the program variables.

$S$  is a (possibly compound) statement

$S$  could in fact be a whole program

The theorem  $\{P\} S \{Q\}$  says that

If  $P$  (the *precondition*) is true just prior to executing  $S$ ,  
then  $Q$  (the *postcondition*) is true just after executing  $S$ .

Normally, the desired result of statement  $S$  is expressed by the postcondition  $Q$ . So given  $Q$ , we need to deduce a precondition  $P$  that, when true, will guarantee that  $Q$  holds after running  $S$ . In particular, we want the *weakest precondition*.

## The assignment statement

It turns out that given postcondition  $Q$  for an assignment statement  
 $\text{ident} \leftarrow \text{exp}$

the weakest precondition  $P$  that gives  $\{P\} S \{Q\}$   
can be determined immediately by the axiom:

$$\{ Q[ \text{exp} / \text{ident} ] \} \text{ident} \leftarrow \text{exp} \{ Q \}$$

In other words,  $P$  is the assertion we get by substituting the expres-  
sion on the right of the assignment for the identifier on the left,  
throughout  $Q$ .

Example: Consider the statement  $n \leftarrow (2*x)+1$

Suppose we want  $n > 1$  as a postcondition for this.

What is the weakest precondition that guarantees it?

By the above axiom, it is:  $(n > 1)[(2*x)+1 / n]$

which is  $((2*x)+1 > 1)$

$$= 2*x > 0 = x > 0$$

So we have proved

$$\{ x > 0 \} n \leftarrow (2*x)+1 \{ n > 1 \}$$

## The assignment statement: reasoning forward

Given assignment statement       $\text{ident} \leftarrow \text{exp}$

if we know a precondition  $P$  holds *prior* to executing the statement, can we figure out what postcondition  $Q$  will hold *after* executing the statement? Answer: SOMETIMES

Recall the assignment axiom:

$$\{ Q[\text{exp} / \text{ident}] \} \text{ ident} \leftarrow \text{exp} \{ Q \}$$

and consider again the statement  $n \leftarrow (2*x)+1$

If we know  $\{ x > 0 \}$  holds prior to the assignment, can we try to apply the axiom in reverse?

Look for " $(2*x)+1$ " in  $\{ x > 0 \}$  and replace them by  $n$ .

But  $(2*x)+1$  doesn't occur in  $\{ x > 0 \}$  – do some algebra.

$$\{ x > 0 \} = \{ 2*x > 0 \} = \{ (2*x)+1 > 1 \}$$

Now the replacement produces  $\{ n > 1 \}$

So we suspect that

$$\{ x > 0 \} \text{ } n \leftarrow (2*x)+1 \text{ } \{ n > 1 \}$$

**THIS DOESN'T ALWAYS WORK**

To be sure we have to check by using the axiom in its proper form: Start with  $\{ n > 1 \}$  and derive the precondition  $\{ x > 0 \}$

## Rule of Consequence

Consider the little theorem just proved:

$$\{ x > 0 \} \ n \leftarrow (2*x)+1 \ \{ n > 1 \}$$

Suppose, prior to executing  $n \leftarrow (2*x)+1$ ,  
we knew that  $x > 9$ ?

Well,  $x > 9 \implies x > 0$ ,

so the weakest precondition  $x > 0$  also holds.

Furthermore, suppose after executing  $n \leftarrow (2*x)+1$ ,  
we needed only that  $n \geq 0$ ?

Well,  $n > 1 \implies n \geq 0$ ,

and since the postcondition  $n > 1$  holds,

the weaker postcondition  $n \geq 0$  also holds.

As a result, if  $Q$  and  $P$  are, respectively, the postcondition and weakest precondition for statement  $S$ , and if both  $P' \implies P$  and  $Q \implies Q'$ , then we have  $\{P'\} S \{Q'\}$ .

Written as a rule of inference:

$$\frac{\{P\} S \{Q\}, P' \implies P, Q \implies Q'}{\{P'\} S \{Q'\}}$$

## Sequences of Statements

How do we combine preconditions and postcondition for single statements into preconditions and postconditions for multiple statements and/or whole programs?

Consider a two statement sequence, S1; S2.

Given a postcondition for S2, use the weakest precondition of S2 as the postcondition for S1.

Expressed as a rule of inference:

$$\frac{\{P\} S1 \{Q\}, \{Q\} S2 \{R\}}{\{P\} S1; S2 \{R\}} \quad \text{Composition Rule}$$

Consider the sequence:  $n \leftarrow 2*n; i \leftarrow i+1$

and the postcondition  $n = 2^i$

What is the weakest precondition for the sequence?

$\{n = 2^i[i+1 / i]\} i \leftarrow i+1 \{n = 2^i\}$  (assignment axiom)

$\{n = 2^{i+1}\} i \leftarrow i+1 \{n = 2^i\}$

$\{n = 2^{i+1}[2*n / n]\} n \leftarrow 2*n \{n = 2^{i+1}\}$  (assignment axiom)

$\{2*n = 2^{i+1}\} n \leftarrow 2*n \{n = 2^{i+1}\}$

$\{2*n = 2^{i+1}\} n \leftarrow 2*n; i \leftarrow i+1 \{n = 2^i\}$  (composition rule)

$\{n = 2^i\} n \leftarrow 2*n; i \leftarrow i+1 \{n = 2^i\}$

Given the conditional: **if B then S**

we would like to discover an

assertions P and Q such that

if P holds prior to the conditional,

then Q holds after the conditional, i.e.,

$$\{P\} \text{ if B then S } \{Q\}$$

But S will execute exactly when B holds,

so we need to establish

$$\{P \wedge B\} S \{Q\}$$

But if  $\neg B$  holds, S **does not** execute,

so we also need to establish

$$\{P \wedge \neg B\} \Rightarrow \{Q\}$$

The formal rule of inference for conditionals accounts for all these things:

$$\frac{\{P \wedge B\} S \{Q\}, P \wedge \neg B \Rightarrow Q}{\{P\} \text{ if } B \text{ then } S \{Q\}}$$

The extension to "if then else" is obvious:

$$\frac{\{P \wedge B\} S1 \{Q\}, \{P \wedge \neg B\} S2 \{Q\}}{\{P\} \text{ if } B \text{ then } S1 \text{ else } S2 \{Q\}}$$

Given the loop: **while B do S**

we would like to discover a

*loop invariant*  $P$  such that

$$\{P\} S \{P\}$$

But  $S$  will never execute unless  $B$  holds.

So if  $B$  and  $P$  are related, we need

$$\{P \wedge B\} S \{P\}$$

So if  $\{P \wedge B\}$  holds prior to the loop,  
then  $\{P\}$  will hold after the loop.

But we know more than just  $\{P\}$  after the loop:

Since the loop terminated,  $B$  must be false.

So after the loop,  $\{P \wedge \neg B\}$  holds.

The formal rule of inference for while loops accounts for all these things:

$$\frac{\{P \wedge B\} S \{P\}}{\{P\} \text{ while } B \text{ do } S \{P \wedge \neg B\}}$$

Recall that  $\sum_{i=1}^n i = \frac{n(n+1)}{2}$

input(n)

s ← 0

i ← 1

while i ≠ n+1                    (i≠n+1 ≡ B)

    s ← s+i

    i ← i+1

end while

$$\left\{ \underbrace{s = (i(i-1))/2}_P \wedge \underbrace{i=n+1}_{\neg B} \right\} \Rightarrow s = (n(n+1))/2$$

**Recall that**  $\sum_{i=1}^n i = \frac{n(n+1)}{2}$

input(n)

$\{s = 0\}[0/s] = \{0 = 0\} = \mathbf{true}$

$s \leftarrow 0$

$\{P[1/i]\} = \{s = (i(i-1))/2\}[1/i] = \{s = (1(1-1))/2\} = \{s=0\}$

$i \leftarrow 1$

$\{s = (i(i-1))/2\} \equiv P$

while  $i \neq n+1$                       ( $i \neq n+1 \equiv B$ )

$s \leftarrow s+i$

$i \leftarrow i+1$

end while

$$\left\{ \underbrace{s = (i(i-1))/2}_P \wedge \underbrace{i=n+1}_{\neg B} \right\} \Rightarrow s = (n(n+1))/2$$

$$\{s = 0\}[0/s] = \{0 = 0\} = \mathbf{true}$$

$$\{P[1/i]\} = \{s = (i(i-1))/2\}[1/i] = \{s = (1(1-1))/2\} = \{s=0\}$$

$$\{s = (i(i-1))/2\} \equiv P$$

**Recall that**  $\sum_{i=1}^n i = \frac{n(n+1)}{2}$

input(n)

$$\{s = 0\}[0/s] = \{0 = 0\} = \mathbf{true}$$

$s \leftarrow 0$

$$\{P[1/i]\} = \{s = (i(i-1))/2\}[1/i] = \{s = (1(1-1))/2\} = \{s=0\}$$

$i \leftarrow 1$

$$\{s = (i(i-1))/2\} \equiv P$$

while  $i \neq n+1$  ( $i \neq n+1 \equiv B$ )

$$\{P'[s+i/s]\} = \{s+i=(i(i+1))/2\} = \{s+i=(i^2+i)/2\} = \{s=(i(i-1))/2\} \equiv P$$

$s \leftarrow s+i$

$$\{P[i+1/i]\} = \{s = ((i+1)((i+1)-1))/2\} = \{s=(i(i+1))/2\} \equiv P'$$

$i \leftarrow i+1$

$$\{s = (i(i-1))/2\} \equiv P$$

end while

$$\left\{ \underbrace{s = (i(i-1))/2}_P \wedge \underbrace{i=n+1}_{\neg B} \right\} \Rightarrow s = (n(n+1))/2$$

$$\{P'[s+i/s]\} = \{s+i=(i(i+1))/2\} = \{s+i=(i^2+i)/2\} = \{s=(i(i-1))/2\} \equiv P$$

$$\{P[i+1/i]\} = \{s = ((i+1)((i+1)-1))/2\} = \{s=(i(i+1))/2\} \equiv P'$$

$$\{s = (i(i-1))/2\} \equiv P$$