

Lecture 26 — Proving loop correctness

- Program correctness
- Loops
 - Loop invariants
 - Termination conditions
 - Examples
- Hoare logic

Program correctness

- Program correctness is usually formalized using a kind of judgment called a *Hoare triple* (after C.A.R. Hoare):

$$P\{S\}Q$$

where P and Q are assertions involving the variables in the program, and S is the program (“ S ” for “statement”).

- This means: If P is true about the program variables, then if S is executed, then Q will be true when it finishes.
- Examples:
 - $x > 0 \{ x = x - 1 \} x \geq 0$
 - $x = x_0 \wedge y = y_0 \wedge x > 0 \wedge y > x \{ y = y - x \} \gcd(x, y) = \gcd(x_0, y_0)$

Proving loops: Partial correctness

- Suppose we want to prove a Hoare triple of the form:

$$P \{ \text{while } (b) \{ S \} \} Q$$

- A *loop invariant* for this loop is a condition I on the program variables (like P and Q) that is always true at the beginning and end of every iteration of S .
- To prove the above Hoare triple:
 - Prove I is an invariant: $b \wedge I \{ S \} I$
 - Prove I is true at the start: $P \wedge b \supset I$
 - Prove Q is true after the loop: $\neg b \wedge I \supset Q$

Proving loops: Termination

- The Hoare triple only proves *partial correctness*: Q holds *if the loop terminates*.
- To prove that a loop terminates, define a function T : *program variables* $\rightarrow$ integers. Then prove:
 1. For all values of the program variables $x, y, \dots$, $T(x, y, \dots) \geq 0$.
 2. If $x_0, y_0, \dots$ are the values of the program variables at the start of S and $x, y, \dots$ are their values after executing S once, then $T(x, y, \dots) < T(x_0, y_0, \dots)$
- Regardless of what T is, if these two conditions hold, the loop must terminate eventually.

Loop proving example 1

$$\begin{array}{l} x = n \wedge y = 1 \{ \\ \quad \text{while } (x \neq 0) \{ y = y * x; \quad x = x - 1; \} \\ \} y = n! \end{array}$$

- *Invariant I:*
- *I is an invariant:*
- *I holds at the start:*
- *Q holds at the end:*
- $T(x, y, n) =$
- $T(x, y, n) > 0$:
- $T(x, y, n) < T(x_0, y_0, n)$:

Loop proving example 2

$$a = lis \wedge b = 0 \{$$
$$\quad \text{while } (a \neq []) \{ b = b + \text{hd}(a); a = \text{tl}(a); \}$$
$$\} b = \Sigma lis$$

- *Invariant I:*
- *I is an invariant:*
- *I holds at the start:*
- *Q holds at the end:*
- $T(a, b, lis) =$
- $T(a, b, lis) > 0$:
- $T(a, b, lis) < T(a_0, b_0, lis)$:

Loop proving example 3

$$a > 0 \wedge b > 0 \wedge a = x \wedge b = y$$
$$\{ \text{ while } (a \neq b) \text{ if } (a > b) \ a = a - b;$$
$$\qquad \text{ else } b = b - a; \} \ a = \gcd(a_0, b_0)$$

- *Invariant I:*
- *I is an invariant:*
- *I holds at the start:*
- *Q holds at the end:*
- $T(a, b, x, y) =$
- $T(a, b, x, y) > 0$:
- $T(a, b, x, y) < T(a_0, b_0, x, y)$:

Loop proving example 4

$$\begin{aligned} &x = 0 \wedge y = 0 \{ \\ &\quad \text{while } (y < n) \{ y = y + 1; x := x + y; \} \\ &\} x = 1 + \dots + n \end{aligned}$$

- *Invariant I:*
- *I is an invariant:*
- *I holds at the start:*
- *Q holds at the end:*
- $T(x, y, n) =$
- $T(x, y, n) > 0$:
- $T(x, y, n) < T(x_0, y_0, n)$:

Loop proving example 5

$$x = 0 \wedge y = 1 \wedge z = 1 \wedge n \geq 1 \{$$
$$\quad \text{while } (z \neq n) \{ y = x + y; x = y - x; z = z + 1; \}$$
$$\} y = \text{fib}(n)$$

- *Invariant I:*
- *I is an invariant:*
- *I holds at the start:*
- *Q holds at the end:*
- $T(x, y, z, n) =$
- $T(x, y, z, n) > 0$:
- $T(x, y, z, n) < T(x_0, y_0, z_0, n)$:

Loop proving example 6

```
 $x = lst \wedge y = 0 \{$   
    while (x != []) { x = tl x; y = y + 1; }  
 $\} y = length(lst)$ 
```

- ***Invariant I:***
- ***I is an invariant:***
- ***I holds at the start:***
- ***Q holds at the end:***
- **$T(x, y, lst) =$**
- **$T(x, y, lst) > 0$:**
- **$T(x, y, lst) < T(x_0, y_0, lst)$:**

Loop proving example 7

$$\begin{aligned} &x = lst \wedge y = [] \{ \\ &\quad \text{while } (x \neq []) \{ y = \text{hd } x :: y; x = \text{tl } x; \} \\ &\} y = \text{reverse}(lst) \end{aligned}$$

- *Invariant I:*
- *I is an invariant:*
- *I holds at the start:*
- *Q holds at the end:*
- $T(x, y, lst) =$
- $T(x, y, lst) > 0$:
- $T(x, y, lst) < T(x_0, y_0, lst)$:

Hoare logic

- C.A.R. Hoare presented a logic — axioms and rules of inference, similar to SOS rules — for proving Hoare triples.

(Assignment) $P[e/x] \{ x = e \} P$ (While) $P \{ \text{while } (b) S \} Q$
 $I \wedge b \{ S \} I$
(if $P \wedge b \supset I$ and $P \wedge \neg b \supset Q$)

(Sequence) $P \{ S_1; S_2 \} Q$ (If) $P \{ \text{if } (b) S_1 \text{ else } S_2 \} Q$
 $P \{ S_1 \} R$
 $R \{ S_2 \} Q$
 $P \wedge b \{ S_1 \} Q$
 $P \wedge \neg b \{ S_2 \} Q$

(Consequence) $P \{ S \} Q$
 $P' \{ S \} Q'$
(if $P \supset P'$ and $Q' \supset Q$)

Example of a proof in Hoare's logic

(If) $\text{true} \{ \text{if } (x < 0) \ y = -x; \text{ else } y = x; \} \ y = |x|$
(Consequence) $x < 0 \{ y = -x \} y = |x| \quad (x < 0 \supset -x = |x|)$
(Assignment) $-x = |x| \{ y = -x \} y = |x|$
(Consequence) $x \not< 0 \{ y = x \} y = |x| \quad (x \not< 0 \supset x = |x|)$
(Assignment) $x = |x| \{ y = x \} y = |x|$