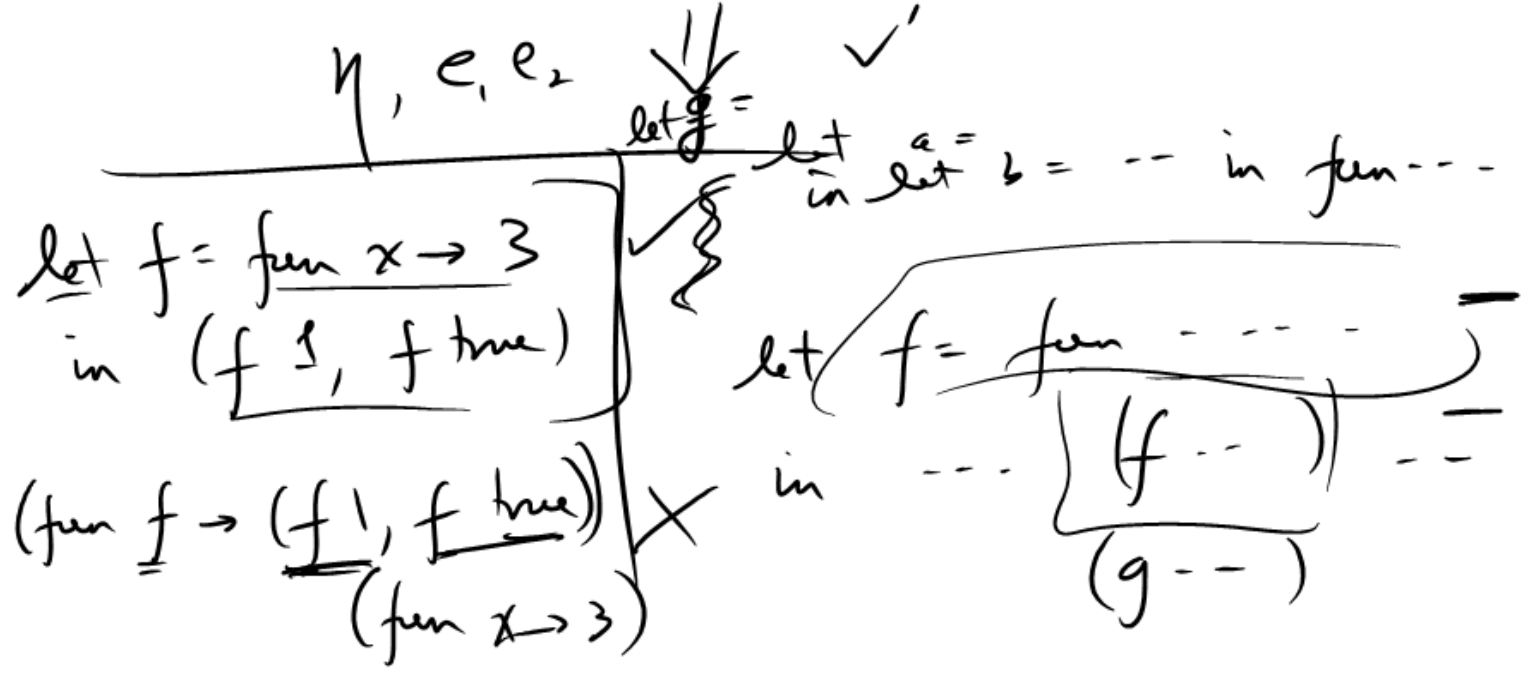


Class 24 – 4/22

Proving properties of imperative programs – “Hoare logic”

- Judgments, a.k.a. “Hoare formulas”
- Axioms
- Rules of inference

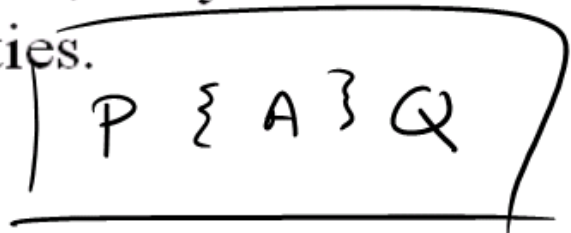
$$\eta, e_1 \Downarrow \langle \text{fun } x \rightarrow e, \eta' \rangle \quad \eta, e_2 \Downarrow v \quad \eta'[x \rightarrow v], e \Downarrow v'$$



Correctness of imperative programs

- “Hoare formula” says that if the variables in a program satisfy some properties, then after executing a given program, they satisfy some different properties.

- Examples:



$x > 0 \{ \text{while} (x > 0)$
 $\{ y := y * x; x := x - 1; \} \} y = y * x!$

$x = x_0 \ \& \ y = y_0 \{ t := x; x := y; y := t \}$
 $x = y_0 \ \& \ y = x_0$

true { if ($x < 0$) $x := -x$; } $x = |x|$

true { $n := \text{length}(a)$; $b := [\text{hd } a]$;
 $a := \text{tl } a$;
 while ($a \neq []$) {
 $b := (\text{hd } a + \text{hd } b) :: b$;
 $a := \text{tl } a$; }
}

$b_i = \sum_{k=0}^{n-i-1} a_k$ (where $b_i = \text{hd } (\text{tl}^i b)$,
and similarly for a_k)

Inference rules of Hoare logic

Judgments: $P \{S\} Q$ ✓

P, Q assertions about variables in the program

S a statement in this language:

Stmt \rightarrow Var := Expr | Stmt; Stmt
| if (Expr) then Stmt else Stmt
| while (Expr) Stmt

Inference rules of Hoare logic

$$\frac{P[e/x] \{x := e\} P}{\text{(Axiom of assignment)}} \quad \frac{P \ \& \ b \ \{S\} \ P}{P \ \{\text{while } (b) \ S\} \ P \ \& \ \neg b}$$
$$\frac{P \ \{S_1\} \ Q \quad Q \ \{S_2\} \ R}{P \ \{S_1; S_2\} \ R}$$

$$\frac{P \Rightarrow P' \quad P' \ \{S\} \ Q' \quad Q' \Rightarrow Q}{P \ \{S\} \ Q}$$

$$\frac{P \ \& \ b \ \{S_1\} \ Q \quad P \ \& \ \neg b \ \{S_2\} \ Q}{P \ \{\text{if } (b) \ \text{then } S_1 \ \text{else } S_2\} \ Q}$$

Rule of assignment:

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

$$x+1=2 \quad \{x := x+1\} \quad x = 2$$

$$y=2 \quad \{x := y\} \quad x = 2$$

Examples

$$y=2 \{ x:=y \} x=2$$

$$y=2 \{ x:=2 \} y=x$$

$$x+1=n+1 \{ x:=x+1 \} x=n+1$$

$$x+1=n \{ x:=x+1 \} x=n$$

2=2 $\{ x:=2 \} x=2$?

$$x=n \{ x:=x+1 \} x=n+1$$

Rule of consequence

$$\frac{P \Rightarrow P' \quad P' \{S\} Q' \quad Q' \Rightarrow Q}{P \{S\} Q}$$

(Assignment)

$$\underbrace{x=n}_{P} \Rightarrow \underbrace{x+1=n+1}_{P'} \quad \underbrace{x+1=n+1}_{P'} \{x:=x+1\} \underbrace{x=n+1}_{Q'} \Rightarrow \underbrace{x=n+1}_{Q}$$

$$\underbrace{x=n}_{P} \quad \{x:=x+1\}_S \quad \underbrace{x=n+1}_{Q}$$

Sequence rule

$$\frac{P \{S_1\} Q \quad Q \{S_2\} R}{P \{S_1; S_2\} R} \quad ?$$

$$P \{S_1; S_2\} R \quad ?$$

$$\frac{\begin{array}{l} t=x_0 \\ \& x=x_0 \\ \& y=y_0 \end{array} \{x:=y\} \begin{array}{l} t=x_0 \\ \& x=y_0 \\ \& y=y_0 \end{array} \quad \begin{array}{l} t=x_0 \\ \& x=y_0 \\ \& y=y_0 \end{array} \{y:=t\} \begin{array}{l} x=y_0 \\ \& y=x_0 \end{array}}{\quad}$$

$$\frac{\begin{array}{l} x=x_0 \\ y=y_0 \end{array} \{t:=x\} \begin{array}{l} t=x_0 \\ x=x_0 \\ y=y_0 \end{array} \quad \begin{array}{l} t=x_0 \\ x=x_0 \\ y=y_0 \end{array} \{x=y; y:=t\} \begin{array}{l} x=y_0 \\ y=x_0 \end{array}}{\quad}$$

$$x=x_0 \& y=y_0 \{t:=x; x:=y; y:=t\} x=y_0 \& y=x_0$$

If rule

$$\frac{P \wedge b \{S_1\} Q \quad P \wedge \neg b \{S_2\} Q}{P \{\text{if } (b) \text{ then } S_1 \text{ else } S_2\} Q}$$

$$\Rightarrow \frac{\frac{\text{Assign}}{\{x := -x\}}}{x = x_0 \wedge x < 0 \quad \{x := -x\} \quad x = |x_0|}$$

$$\frac{\frac{\text{Assign}}{\{x := x\}}}{x = x_0 \wedge x \geq 0 \quad \{x := x\} \quad x = |x_0|}$$

$$x = x_0 \quad \left\{ \begin{array}{l} \text{if } x < 0 \\ \text{then } x := -x \\ \text{else } x := x \end{array} \right\} \quad x = |x_0|$$

While rule

$$\frac{P \ \& \ b \ \{S\} \ P}{P \ \{\text{while } (b) \ S\} \ P \ \& \ \neg b}$$

$$\boxed{s=0}$$

while (i < n) {

s := s + i;

i := i + 1

}

}

Invariant:

$$s = \sum_{j=0}^{i-1} j$$

$$\Rightarrow s = \sum_{j=0}^{i-1} j \quad \& \quad i = n$$

Comments on Hoare logic

- Proofs in Hoare logic are *almost* syntax-directed, i.e. almost have the same shape as the program being proved. The only exceptions are the uses of the rule of consequence.
- Applying Hoare rules is largely mechanical – given A and Q , most of the proof (including P) can be generated automatically. Creativity is required mainly in determining the invariant in a while loop, because Q may not have the form “ $P \ \& \ \neg b$ ”, and so a formula of that form needs to be found (after which the rule of consequence can be used, proving $P \ \& \ \neg b \Rightarrow Q$).

Example: gcd algorithm

```
 $a > 0 \ \& \ b > 0 \ \& \ a = a_0 \ \& \ b = b_0 \{$   
  while ( $a \neq b$ )  
    if ( $a > b$ ) then  $a := a - b$ ;  
    else  $b := b - a$ ;  
}  $a = \text{gcd}(a_0, b_0)$ 
```