Class 24 – 4/22

Proving properties of imperative programs – "Hoare logic"

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

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

let g = let a = in let b = -- in fun ---

let f = fun x → 3
in (f 1, f true) ✓

(fun f → (f 1, f true)) (fun x → 3) ✗

let f = fun ----
in --- (f --) --- (g --)

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.

$$\boxed{P \{ A \} Q}$$

- Examples:

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

  *x=x0 & y=y0* { t := x; x := y; y := t }
  *x=y0 & y=x0*

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

```
true { n := length(a);  b := [hd a];
        a := tl a;
        while (a != []) {
              b := (hd a + hd b) :: b;
              a := tl a; }
```

$$\} \; b_i = \sum_{k=0}^{n-i-1} a_k$$ (where $b_i$ = hd ($\mathrm{tl}^i$ 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 -> Var := Expr | Stmt;Stmt
        | if (Expr) then Stmt else Stmt
        | while (Expr) Stmt
```

# Inference rules of Hoare logic

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

(Axiom of assignment)

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

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

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

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

# Rule of assignment:

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

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

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

# 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

~~true~~ { x:=2 } x=2 ?

2 = 2

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

# Rule of consequence

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

(Assign)

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

# Sequence rule

$$\frac{P\ \{S_1\}\ Q \qquad Q\ \{S_2\}\ R}{P\ \{S_1;\ S_2\}\ R}$$

$$\frac{\frac{?}{t=x_0\ \&\ x=x_0\ \&\ y=y_0\ \{x:=y\}\ t=x_0\ \&\ x=y_0\ \&\ y=y_0} \qquad \frac{?}{t=x_0\ \&\ x=y_0\ \&\ y=y_0\ \{y:=t\}\ x=y_0\ \&\ y=x_0}}{}$$

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

# If rule

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

$$\dfrac{\dfrac{\Rightarrow \quad \dfrac{}{\_\_\ \{x := -x\}\ \_\_}\ \text{Assign} \quad \Rightarrow}{x = x_0\ \&\ x < 0\ \{x := -x\}\ x = |x_0|} \qquad \dfrac{\dfrac{}{\_\ \{x = x\}\ \_}\ \text{Assign}}{x = x_0\ \&\ x \not< 0\ \{x := x\}\ x = |x_0|}}{x = x_0\ \{\text{if } x < 0 \text{ then } x := -x \text{ else } x := x\}\ x = |x_0|}$$

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

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

# 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 ≠ b)
        if  (a > b) then a := a − b;
                    else b := b − a;
}
```

$a = gcd(a_0, b_0)$