CS 421 Lecture 23: Hoare logic

- Lecture outline
  - Proving properties of imperative programs
  - Hoare logic
  - Judgments, a.k.a. "Hoare formulas"
  - Axioms
  - Rules of inference

Review of last week

- Proof systems
  - Formal frameworks for writing proofs
  - Judgments, axioms, rules of inference
- Type systems
  - Used for type checking, type inference
  - Judgments of the form: $\Gamma \vdash e : \tau$
- Operational semantics
  - Used for proofs of correctness
  - Judgments of the form: $\sigma,\eta \vdash e \Downarrow v,\sigma'$

Review: operational semantics

- Operational semantics of functional languages
  - Based on expression evaluation
  - Proofs follow the structure of the expression

- Variants
  - OS_{subst}
  - OS_{clo}
  - OS_{state}

Example

```
let f = fun x -> 3 in (f 1, f true)
```

Today: proofs for imperative programs

- Hoare logic (or Hoare rules or Hoare formulas)
  - Prove correctness of imperative programs

- Specifies pre- and post-conditions for statement execution
  - Axiomatic semantics
  - Contract principle

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.
  - P {A} Q

- Examples:

x>0 { while (x>0)

{y := y*x; x := x-1;} } y = y * x!

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

More examples

```
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; }
```

$$\} \quad b_i = \sum_{k=0}^{n-i-1} a_k$$

(where b_i = hd (tl^i b) and similarly for a_k)

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 for Hoare logic

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

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

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

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

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

Rule of assignment

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

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

y=2 { x := y } x=2

Rule of assignment: examples

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

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

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

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

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

true { $x:=2$ } $x=2$

Rule of consequence

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

Rule of consequence: example

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

$$\frac{x = n \Rightarrow x+1 = n+1 \quad A \quad x = n+1 \Rightarrow x = n+1}{x = n \quad \{x := x+1\} \quad x = n+1}$$

A = $$\frac{}{x+1 = n+1 \quad \{x = x+1\} \quad x = n+1}$$

Which inferences are correct?

$$\frac{x > 0 \ \& \ x < 5 \quad \{x := x * x\} \quad x < 25}{x = 3 \quad \{x := x * x\} \quad x < 25}$$

$$\frac{x = 3 \quad \{x := x * x\} \quad x < 25}{x > 0 \ \& \ x < 5 \quad \{x := x * x\} \quad x < 25}$$

$$\frac{x * x < 25 \quad \{x := x * x\} \quad x < 25}{x > 0 \ \& \ x < 5 \quad \{x := x * x\} \quad x < 25}$$

Sequence rule

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

Sequence rule: example

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

$$\frac{\dfrac{}{\begin{matrix}x = x_0 \\ y = y_0\end{matrix} \& \quad t := x \quad \begin{matrix}t = x_0 \\ \& \\ x = x_0 \\ \& \\ y = y_0\end{matrix}} \qquad \dfrac{A \quad B}{\begin{matrix}t = x_0 \\ \& \\ x = x_0 \\ \& \\ y = y_0\end{matrix} \quad \{x = y; y = t\} \quad \begin{matrix}x = y_0 \\ y = x_0\end{matrix}}}{\begin{matrix}x = x_0 \\ y = y_0\end{matrix} \& \quad \{t := x; x := y; y := t\} \quad \begin{matrix}x = y_0 \\ y = x_0\end{matrix} \&}$$

Sequence rule: example

A =

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

B =

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

If rule

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

If rule: example

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

$$\dfrac{\dfrac{\Rightarrow \quad \dfrac{}{\ldots \quad \{x := -x\} \quad \ldots} \quad \Rightarrow}{\begin{matrix} x = 0 \\ \& \\ x < 0 \end{matrix} \quad \{x := -x\} \quad x = |x_0|} \qquad \dfrac{\Rightarrow \quad \dfrac{}{\ldots \quad \{x := x\} \quad \ldots} \quad \Rightarrow}{\begin{matrix} x = 0 \\ \& \\ \neg x < 0 \end{matrix} \quad \{x := x\} \quad x = |x_0|}}{x = x_0 \quad \begin{matrix} \{\text{if } x < 0 \\ \text{then } x := -x \\ \text{else } x := x\,\} \end{matrix} \quad x = |x_0|}$$

while rule

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

while rule: example

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

```
s = 0; i = 0
          { while (i < n) {
                   s := s+i;
                   i := i+1;
          }
```

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

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 & ¬b".
  - A formula of that form needs to be found (after which the rule of consequence can be used, proving P & ¬b ⇒ 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 $b := b - a$;

} $a = \gcd(a_0, b_0)$