Program Correctness/Verification

Program Correctness

- We want to be able to *prove* that a given program meets the intended specifications.
  - This can often be done manually, or even by *automated program verification* tools.
    - One example is PVS (People's Verification System).
- A program is *correct* if it produces the correct output for every possible input.
  - A program has *partial correctness* if it produces the correct output for every input for which the program eventually halts.

Initial & Final Assertions

- A program's I/O specification can be given using *initial* and *final* assertions.
  - The *initial assertion p* is the condition that the program's input (its initial state) is guaranteed (by its user) to satisfy.
  - The *final assertion q* is the condition that the output produced by the program (its final state) is required to satisfy.
- *Hoare triple* notation:
  - The notation $p\{S\}q$ means that, for all inputs I such that $p(I)$ is true, if program S (given input I) halts and produces output $O = S(I)$, then $q(O)$ is true.
    - That is, S is partially correct with respect to specification p,q.

A Trivial Example

- Let S be the program fragment

  "$y := 2; z := x+y$"

- Let p be the initial assertion "$x = 1$".
  - The variable x will hold 1 in all initial states.
- Let q be the final assertion "$z = 3$".
  - The variable z must hold 3 in all final states.
- Prove $p\{S\}q$.
  - **Proof:** If x=1 in the program's input state, then after running y:=2 and z:=x+y, z will be 1+2=3.

Hoare Triple Inference Rules

- Deduction rules for Hoare Triple statements.
- A simple example: *The composition rule:*

$$\frac{\begin{array}{c} p\{S_1\}q \\ q\{S_2\}r \end{array}}{\therefore\ p\{S_1;\ S_2\}r}$$

- **It says:** If program S_1 given condition p produces condition q, and S_2 given q produces r, then the program "S_1 followed by S_2", if given p, yields r.

Inference rule for **if** statements

$$\begin{array}{l} (p \wedge cond)\{S\}q \\ \underline{(p \wedge \neg cond) \rightarrow q} \\ \therefore p\{\textbf{if } cond \textbf{ then } S\}q \end{array}$$

- Example: Show that: **T** $\{$**if** $x>y$ **then** $y:=x\}$ $y \geq x$.
- **Proof:** If initially $x>y$, then the if body is executed, setting $y=x$, and so afterwards $y \geq x$ is true. Otherwise, $x \leq y$ and so $y \geq x$. In either case $y \geq x$ is true. So the rule applies, and so the fragment meets the specification.

if-then-else rule

$$\frac{\begin{array}{l}(p \wedge cond)\{S_1\}q \\ (p \wedge \neg cond)\{S_2\}q\end{array}}{\therefore\quad p\{\textbf{if } cond \textbf{ then } S_1 \textbf{ else } S_2\}q}$$

Example: Show that

$\mathbf{T}\ \{\textbf{if } x<0 \textbf{ then } abs:=-x \textbf{ else } abs:=x\}\ abs=|x|$

- If $x<0$ then after the **if** body, *abs* will be $|x|$. If $\neg(x<0)$, *i.e.*, $x\geq 0$, then after the **else** body, $abs=x$, which is $|x|$. So the rule applies.

Loop Invariants

- For a while loop "**while** *cond S*", we say that *p* is a loop invariant of this loop if $(p \wedge cond)\{S\}p$.
  - If *p* (and the continuation condition *cond*) is true before executing the body, then *p* remains true afterwards.
    - And so *p* stays true through *all* subsequent iterations.
- This leads to the inference rule:

$$\begin{array}{ll} & \underline{(p \wedge cond)\{S\}p} \\ \therefore & p\{\mathbf{while}\ cond\ S\}(\neg cond \wedge p) \end{array}$$

Loop Invariant Example

Prove that the following Hoare triple holds:

T $\{i := 1;\ fact := 1;$ **while** $i<n\ \{i{+}{+};\ fact *= i\}\}$ $(fact = n!)$

Proof. Note that $p :\equiv$ "$fact = i! \land i \le n$" is a loop invariant, and is true before the loop. Thus, after the loop we have $\neg cond \land p \Leftrightarrow \neg(i<n) \land fact = i! \land i \le n \Rightarrow i = n \land fact = i! \Rightarrow fact = n!$. ■

Big Example

procedure *multiply*(m,n: integers) $m,n \in \mathbf{Z}$

  if $n<0$ **then** $a:=-n$ **else** $a:=n$

$a=|n|$

  $k:=0$; $x:=0$ $x = mk \wedge k \leq a$

  while $k<a$ { Maintains loop invariant:

    $x \mathrel{+}= m$; k++ $x = mk \wedge k \leq a$

  } $x = mk \wedge k=a \;\therefore\; x = ma = m|n|$

$\therefore (n<0 \wedge x=-mn) \vee (n \geq 0 \wedge x=mn)$

  if $n<0$ **then** $prod := -x$ **else** $prod:=x$

$prod = mn$