Application: The correctness of iterative statements

Suppose that

- S is a statement in a programming language
- P and Q are boolean expressions involving program variables.

We write $\{P\}S\{Q\}$ to mean that

- If statement S starts in a state where P holds
- then it can only terminate in a state where Q holds.

(Such a triple is called a Hoare triple after C.A.R. Hoare. P is called the "precondition" and Q is called the "postcondition".)

For example the following are valid Hoare triples

- $\{i \le 100 \wedge j \le 100\}\ i := i + j\ \{i \le 200\}$
  (I am using the notation "$x := E$" for assignment of expression E to variable x. In C/C++ we would write "$x = E;$")
- $\{i = 4\}\ i := i + 1; i := i \times 2\ \{i = 10\}$

- $\{B^A = z \times y^x \wedge x > 0\}$
  $x := x - 1; z := z \times y$
  $\{B^A = z \times y^x\}$

We can also use variables that do not occur in the program state. So

$$\{i = K\}\ i := i + 1; i := i \times 2\ \{i = 2 \times K + 2\}$$

is a valid Hoare triple.[1]

Now consider the following triple with integer x, y, z, A, B

$$\{x = A \wedge x \geq 0 \wedge y = B\}$$
$$z := 1;$$
$$\textbf{while}\ \text{/*L*/}\ x > 0\ \textbf{do}\ (x := x - 1; z := z \times y)$$
$$\{z = B^A\}$$

We can see that the triple is valid as follows:

- Let I be "$B^A = z \times y^x \wedge x \geq 0$"
- Let $P(n)$ mean I holds the n^{th} time point L is reached.

---

[1] By the way, a Hoare triple $\{P\}x := E\{Q\}$ is valid iff

$$P \rightarrow Q[x := E] \text{ for all values of all variables}$$

And you can extend this to a sequence of assingments. E.g. $\{P\}x := E; y := F\{Q\}$ iff

$$P \rightarrow (Q[y := F])[x := E] \text{ for all values of all variables}$$

- We can show by simple induction that $P(n)$ is true for all $n \in \{1, 2, 3, ...\}$.
- Base step. $P(1)$ is true because when we first reach L it is right after the first assignment to z and so $x = A \wedge x \geq 0 \wedge y = B \wedge z = 1$.
- Inductive step. $P(k)$ is the ind. hyp. W.T.P $P(k+1)$
  * To show the inductive step, we first show the validity of
  $$\{I \wedge x > 0\}\; x := x - 1; z := z \times y\; \{I\}$$
  * The $(k+1)^{\text{th}}$ time point L is reached it is right after k^{th} iteration of the loop body.
  * By the ind. hyp. I holds at the start of the k^{th} iteration of the loop body; so does $x > 0$.
  * So by
  $$\{I \wedge x > 0\}\; x := x - 1; z := z \times y\; \{I\}$$
  I holds at the end of the k^{th} iteration of the loop body and hence the $(k+1)^{\text{th}}$ time point L is reached.
- If the loop is ever exited, it will be the case that I holds and also $x \leq 0$ holds.
- From $I \wedge x \leq 0$ we can deduce $x = 0$ and hence $z = B^A$.

We can replace the loop body by any statement S such that

$$\{I \wedge x > 0\}\ S\ \{I\}$$

For example we can replace the loop body by
$\mathbf{if}\ 2|x\ \mathbf{then}\ (x := x/2; y := y^2)\ \mathbf{else}\ (x := x-1; z := z \times y)$

Hoare's rule of iteration

We can generalize this technique to any loop $\mathbf{while}\ E\ \mathbf{do}\ S$ provided

- E does not affect the state.
- there is no way to exit the loop other than by E evaluating to false.

The general rule is

- If $\{I \wedge E\}\ S\ \{I\}$ is valid
- then $\{I\}\ \mathbf{while}\ E\ \mathbf{do}\ S\ \{I \wedge \neg E\}$ is valid

Application: The correctness of recursive subroutines

Consider the following subroutine in C++

```
int pow( int x, int y) {
        if( x==0 )
                return 1 ;
        else if( x % 2 != 0 )
                // x is odd
                return y * pow(x-1,y) ;
        else // x is even and not 0
                return pow(x/2,y*y) ;
}
```

Such a subroutine is called ‘recursive’ as it contains calls to itself.

What does this routine do?

Let $P(n)$ mean "for any y, pow(n, y) returns y^n".

Now we show for all $n \in \mathbb{N}$, $P(n)$ by *complete* induction.

Base Step:

- Any call 'pow(0, y)' returns 1, which is y^0.

Inductive Step:

- Let k be any natural greater than 0
- Assume as the ind. hyp. that for all naturals j less than k, $P(j)$.
- Let y be any integer
- Case k is odd
  - By the ind hyp 'pow$(k-1, y)$' returns y^{k-1}
  - The value returned by 'pow(k, y)' is $y \times pow(k-1, y)$

$$y \times pow(k-1, y) = y \times y^{k-1} = y^k$$

- Case k is even.
  - By the ind hyp 'pow$(k/2, y^2)$' returns $\left(y^2\right)^{k/2}$
  - The value returned by 'pow(k, y)' is $pow(k/2, y^2)$

$$pow(k/2, y^2) = \left(y^2\right)^{k/2} = y^k$$