Program variables and logical variables

The variables which we have seen so far in the programs that we verify are called program variables. They can also appear in the preconditions and postconditions of specifications. Sometimes, in order to formulate specifications, we need to use other variables which do not appear in programs.

1. Another version of the factorial program might have been `Fac2`:

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

Unlike the previous version, it 'consumes' the input x. Nevertheless, it correctly calculates the factorial of x and stores the value in y; and we would like to express that as a Hoare triple. However, it is not a good idea to write $\{x \geq 0\}$ `Fac2` $\{y = x!\}$ because, if the program terminates, then x will be 0 and y will be the factorial of the initial value of x.

We need a way of remembering the initial value of x, to cope with the fact that it is modified by the program. Logical variables achieve just that: in the specification $\{x = x_0 \wedge x \geq 0\}$ `Fac2` $\{y = x_0!\}$ the x_0 is a logical variable and we read it as being universally quantified in the precondition. Therefore, this specification reads: for all integers x_0, if x equals x_0, $x \geq 0$ and we run the program such that it terminates, then the resulting state will satisfy y equals $x_0!$. This works since x_0 cannot be modified by `Fac2` as x_0 does not occur in `Fac2`.

2. Consider the program `Sum`:

```
z = 0;
while (x > 0) {
    z = z + x;
    x = x - 1;
    }
```

This program adds up the first x integers and stores the result in z. Thus, $\{x = 3\}$ `Sum` $\{z = 6\}$, $\{x = 8\}$ `Sum` $\{z = 36\}$ etc. We know from Theorem 1.31 on page 41 that $1 + 2 + \cdots + x = x(x+1)/2$ for all $x \geq 0$, so

we would like to express, as a Hoare triple, that the value of z upon termination is $x_0(x_0+1)/2$ where x_0 is the initial value of x. Thus, we write $\{x = x_0 \wedge x \geq 0\}$ `Sum` $\{z = x_0(x_0+1)/2\}$.

Variables like x0 in these examples are called logical variables, because they occur only in the logical formulas that constitute the precondition and postcondition; they do not occur in the code

to be verified. The state of the system gives a value to each program variable, but not for the logical variables.