

Example 1.14

Let us consider the following program fragment, designed to calculate the product of two natural numbers a and b using only addition, and multiplication and division by 2:

```
r := 0;
while m > 0 do
  begin if m is odd then r := r + n;
        m := m div 2;
        n := n * 2;
  end;
```

Here, as usual, $m \text{ div } 2$ represents the whole number part of $m/2$. The intention is that if a and b are the initial values of the variables m and n , respectively, then the computation should terminate with variable r holding the value ab . For instance, the computation with $a = 11$ and $b = 26$ is as follows:

m	n	r
11	26	0
5	52	26
2	104	78
1	208	78
0	416	286

and the output is $286 = 11 \times 26$.

This particular program has a very simple structure: after assigning the value 0 to variable r , it just executes the `while` loop until the variable m holds the value 0. We prove by induction on k that, after k executions of the `while` loop, the current values of m , n , and r satisfy $mn + r = ab$.

Basis Consider $k = 0$, that is, just before the `while` loop is first executed. The values of m , n , and r are a , b , and 0, respectively, so the result is immediate.

Induction step Assume the result for k , and let the values of m , n , and r before and after the $(k + 1)$ th execution of the `while` loop be a_1 , b_1 , and c_1 , and a_2 , b_2 , and c_2 , respectively. The induction hypothesis assures us that $a_1b_1 + c_1 = ab$, and we have to show that $a_2b_2 + c_2 = ab$. We consider the cases a_1 even and odd.

If a_1 is even, then $a_2 = a_1/2$, $b_2 = 2b_1$, and $c_2 = c_1$, so

$$a_2b_2 + c_2 = (a_1/2)(2b_1) + c_1 = a_1b_1 + c_1 = ab$$

If a_1 is odd, then $a_2 = (a_1 - 1)/2$, $b_2 = 2b_1$, and $c_2 = c_1 + b_1$, so

$$\begin{aligned} a_2b_2 + c_2 &= [(a_1 - 1)/2](2b_1) + c_1 + b_1 \\ &= a_1b_1 - b_1 + c_1 + b_1 = a_1b_1 + c_1 = ab \end{aligned}$$

Hence result by induction.

From this we can deduce **partial correctness**: if the program terminates, then it gives the correct answer. By examining the `while` clause, we see that it can terminate only when m attains the value 0. But then $r = mn + r = ab$, as required. The other ingredient of a correctness proof is called **termination**. We have to show that the program always terminates. In the above example this is very easy, since on each execution of the `while` loop the value of m decreases. This means that at the a th stage at latest, m must attain the value 0.