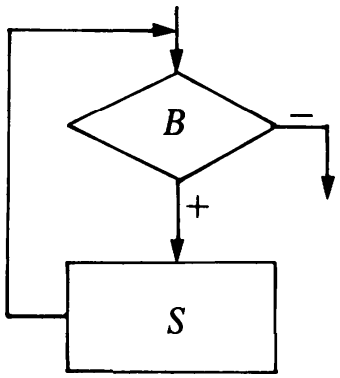
6 FINITENESS OF PROGRAMS

The loop structure is the most characteristic element in every computer program because it implies the *repetition* of an action, and automata are particularly well suited to repetitive tasks. The computer's ability to maintain accuracy and reliability after thousands of repetitions is particularly valuable. On the other hand, it is precisely this untiring and indiscriminate "obedience" to a program that requires an increased caution on the part of the programmer. Consequently, one important property required of every algorithm ~~program~~ is that it *terminates* after a *finite* number of repetitions. Unfortunately, processes that do not terminate are a fairly common and costly phenomenon in computer installations everywhere. These can be avoided, however, through an increased meticulousness during program design and verification. To explain the precautions necessary to ensure termination, we can use the fundamental loop structure shown in (6.1).



(6.1)

A minimal requirement for termination is that statement S must change the value of one or more variables in such a way that after a finite number of passes, condition B can no longer be satisfied. In general, the finiteness of a

repetition can be formally derived by postulating an integer function N, depending on certain variables of the program, and by showing that

(a) if B is satisfied, then $N > 0$ and
(b) each execution of S decreases the value of N.

The application of this rule is trivial in the case of program (5.2). Statement S is

$$r := r - y$$
$$q := q + 1$$

and condition B is

$$r \geqq y$$

where r, q, and y are natural numbers. We choose $N = r - y$. Since

(a) $r \geqq y$ implies $N \geqq 0$ and (Osservazione: nella definizione generica, 'N>0' deve essere 'n>=0')
(b) the execution of S decreases the value of r and therefore of N, finiteness is evidently guaranteed. Note, in particular, the necessity of the initial condition $y > 0$.

As a second example, we can use the application of the derivation rule to program (5.18). Its statement S consists of two alternatives,

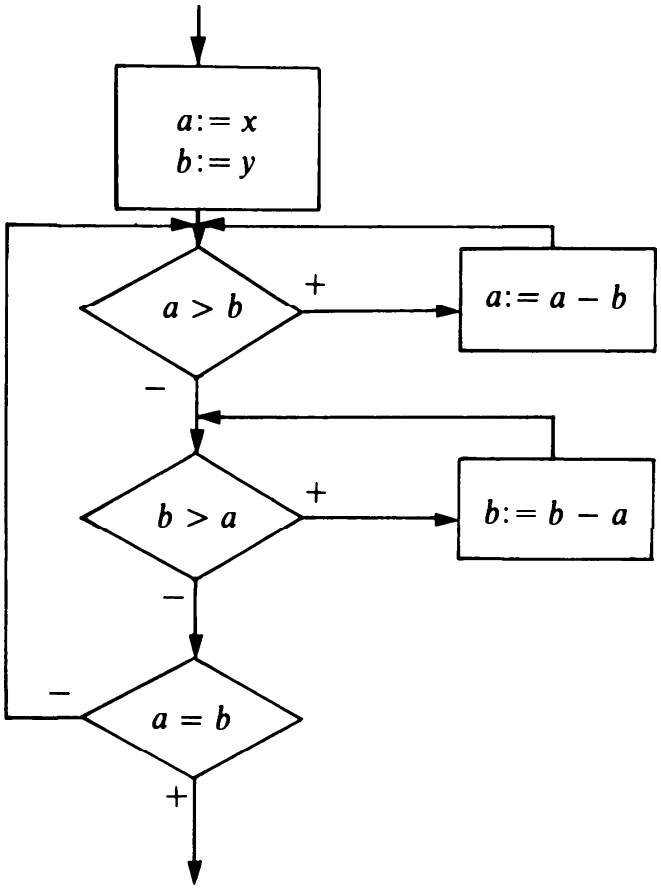
$$a := a - b \quad \text{if } a > b, \qquad \text{or} \qquad b := b - a \quad \text{if } b > a$$

and condition B is $a = b$. Again, a and b are natural numbers with initial values $a > 0$, $b > 0$, and $a \neq b$. A suitable choice for $N(a, b)$ turns out to be $N = \max(a, b)$. The effect of S upon N must be considered in two separate cases. If $a > b$, then b remains unchanged and a is decreased by b. Since $a > 0$, $b > 0$, and $a \neq b$ initially, the first two relations remain unchanged (invariant) and N decreases. If $b > a$, then a remains unchanged and $b = \max(a, b) = N$ decreased by a. Thus since $N = \max(a, b)$ decreases during each repetition and, on the other hand, $\min(a, b)$ remains positive, it follows that at some time, $\max(a, b) = \min(a, b)$, and therefore $a \neq b$ is no longer satisfied. This terminates the repetition.

EXERCISES

6.1 Determine the range of values of x and y that will guarantee the finiteness of the programs in Exercises 5.1 and 5.3.

6.2 For which range of values of x and y is the following program finite?



(6.2)

6.3 In which cases does program (6.2) compute the greatest common divisor of x and y? Determine the necessary and sufficient assertions.