An intriguing example.

In the following all variables and all elements of the infinite arrays $f[0..]$ and $g[0..]$ are of type natural number.

Array f is ascending, i.e.

$$(\underline{A}\, x : x \geq 0 : f[x] \leq f[x+1]) \qquad (0)$$

and unbounded, i.e.

$$(\underline{A}\, y : y \geq 0 : (\underline{E}\, x : x \geq 0 : f[x] > y)) \qquad (1)$$

As a result of (1)

prog 0 : $\quad$ **do** $f[x] \leq y \rightarrow x := x+1$ **od**

terminates. Also — obviously —

prog 1 : $\quad$ **do** $f[x] > y \rightarrow g[y] := x ;\ y := y+1$ **od**

terminates. The "combined" program

```
x, y := 0, 0;
do f[x] ≤ y → x := x+1
[] f[x] > y → g[y] := x ; y := y+1
od
```

obviously fails to terminate. Hence, x and y are both unbounded: more and more of f will be taken into account, and more and more of g will be defined.

From (0) we derive

$$(\underline{N}\, i: i \geq 0: f[i] \leq f[x]) \geq x+1 \qquad (2)$$

The weakest precondition that $x := x+1$ establishes

$$(\underline{N}\, i: i \geq 0: f[i] \leq y) \geq x \qquad (3)$$

is, according to the axiom of assignment,

$$(\underline{N}\, i: i \geq 0: f[i] \leq y) \geq x+1 \quad ,$$

which, on account of (2), is implied by $f[x] \leq y$; hence, the first alternative leaves (3), which is established by $x, y := 0, 0$, invariant. So does the second alternative (obviously).

From $f[x] > y$ we derive, on account of (0)

$$(\underline{N}\, i: i \geq 0: f[i] \leq y) \leq x ,$$

which, in conjunction with (3) allows us to conclude that, then, $(\underline{N}\, i: i \geq 0: f[i] \leq y) = x$. Hence, we have the second invariant

$$(\underline{A}\, j: 0 \leq j < y: g[j] = (\underline{N}\, i: i \geq 0: f[i] \leq j)) \qquad (4)$$

and this is exactly the property I wanted to prove about my program

* * *

The example is - see EWD753 - inspired by the theorem of Lambek and Moser, a theorem Wim Feijen found when looking for functions to be programmed in SASL. As a matter of fact, my "combined" program was not the first program I wrote to solve this problem: it is a direct translation of the following SASL definitions I wrote first: (my syntax)

$$\begin{array}{l}\underline{\text{def}}\ k\ x\ y\ (p{:}q) = \\ \quad \underline{\text{if}}\ p \le y \rightarrow k\ (x+1)\ y\ q \\ \quad [\!]\ p > y \rightarrow x : k\ x\ (y+1)\ (p{:}q) \\ \quad \underline{\text{fi}} \\ \underline{\text{def}}\ g = k\ 0\ 0\ f\end{array} \qquad (5)$$

But even the proof of the fact that g is ascending - which in the iterative program follows trivially from the equally obvious invariant

$$y = 0 \ \underline{\text{cor}}\ g[y-1] \le x$$

— was very painful when I tried a proof technique à la EWD749 which does justice to the "functional" nature of applicative languages: (5) is expressed in terms of tails, my proof is in terms of finite prefixes. I think I should ask an expert. (See EWD759.)

Plataanstraat 5
5671 AL NUENEN
The Netherlands

5 November 1980
prof. dr. Edsger W. Dijkstra
Burroughs Research Fellow