

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.

$$(\underline{\forall} x: x \geq 0: f[x] \leq f[x+1]) \quad (0)$$

and unbounded, i.e.

$$(\underline{\forall} y: y \geq 0: (\underline{\exists} x: x \geq 0: f[x] > y)) \quad (1)$$

As a result of (1)

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

terminates. Also - obviously -

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

terminates. The "combined" program

```

x, y := 0, 0;
do  $f[x] \leq y \rightarrow x := x+1$ 
  ∩  $f[x] > y \rightarrow g[y] := x; y := y+1$ 
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

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

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

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

is, according to the axiom of assignment,

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

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

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

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)

$$\begin{aligned} \underline{\text{def}} \ k \ x \ y \ (p:q) &= & (5) \\ \quad \text{if } p \leq y &\rightarrow k \ (x+1) \ y \ q \\ \quad \text{if } p > y &\rightarrow x : k \ x \ (y+1) \ (p:q) \\ \underline{\text{f:}} \\ \underline{\text{def}} \ g &= k \ 0 \ 0 \ f \end{aligned}$$

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

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

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.)

Plataamstraat 5
5671 AL NUENEN
The Netherlands

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