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.

\[
(\forall x: x \geq 0: f[x] \leq f[x+1])
\]  

(0)

and unbounded, i.e.

\[
(\forall y: y \geq 0: (\exists x: x \geq 0: f[x] > y))
\]  

(1)

As a result of \( (1) \)

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

terminates. Also — obviously —

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

terminates. The "combined" program

\[
\begin{align*}
  x, y := 0, 0; \\
  \text{do } f[x] \leq y \rightarrow x := x + 1 \text{ od} \\
  \text{do } f[x] > y \rightarrow g[y] := x; y := y + 1 \text{ od}
\end{align*}
\]

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 (o) we derive
\[
(N \varepsilon: i \geq 0: f[i] \leq f[x]) \Rightarrow x + 1
\]  \( \text{(2)} \)

The weakest precondition that \( x := x + 1 \) establishes
\[
(N \varepsilon: i \geq 0: f[i] \leq y) \Rightarrow x
\]  \( \text{(3)} \)

is, according to the axiom of assignment,
\[
(N \varepsilon: i \geq 0: f[i] \leq y) \Rightarrow x + 1
\]

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 \( f[x] \geq y \) we derive, on account of (o)
\[
(N \varepsilon: i \geq 0: f[i] \leq y) \Rightarrow x
\]
which, in conjunction with (3) allows us to conclude that, then, \( (N \varepsilon: i \geq 0: f[i] \leq y) = x \). Hence, we have the second invariant
\[
(A \varepsilon: 0 \leq j < y: g[j] = (N \varepsilon: i \geq 0: f[i] \leq j))
\]  \( \text{(4)} \)

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)

\[
\begin{align*}
\text{def } & k \times y \; (p; q) = \\
& \begin{cases} 
  & \text{if } p \leq y \rightarrow k \; (x+1) \; y \; q \\
  & \text{if } p > y \rightarrow x : k \; x \; (y+1) \; (p; q)
\end{cases} \\
\text{def } & g = k \; 0 \; 0 \; f
\end{align*}
\]

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

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

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