A somewhat open letter to D.A. Turner.

Dear David,

I read with considerable interest your "Program Proving and Applicative Languages", but when I tried to apply your techniques to the first slightly more ambitious example my assistant W.H.J. Feijen came up with, it turned out to be a very humbling experience; I clearly lacked the knack of dealing with SASL programs.

Consider the following short SASL definitions (my syntax! I trust you will forgive me that)

\[
\text{def } k \times y (p;q) = \\
\text{if } p \leq y \rightarrow k (x+1) y q \\
\text{if } p > y \rightarrow x : k x (y+1) (p;q) \\
\text{fi}; \\
\text{def } g = k 0 0 f
\]

Here \( f \) is supposed to be \underline{ascending}, i.e. to satisfy \((Po f)\) with

\[
Po (a;b;c) = a \leq b \land Po (b;c)
\]

and \underline{unbounded}, i.e. to satisfy \((\forall y: y \geq 0: Pt y f)\) with

\[
Pt y (p;q) = p > y \lor (Pt y q).
\]
When $p$ is ascending and unbounded, it is clearly an infinite list. (I try to use the more positive term "continued concatenation", for that, but that is another story.)

You would do me a great service by showing me a proof that when $p$ is ascending and unbounded, $g$ is ascending and unbounded.

The precise formal relation between $g$ and $p$ is that for all $y \geq 0$

$$\text{sub } y \ g = (\forall x : x \geq 0 : y \geq (\text{sub } x \ p)) \quad (0)$$

where $\text{sub}$ is defined by

$$\text{def } \text{sub } n \ (p;q) =$$

* if $n = 0 \rightarrow p$
* if $n > 0 \rightarrow \text{sub } (n-1) \ q$

and the $N$ should be read as: the number of distinct values $x$ in the range $x \geq 0$ such that $y \geq (\text{sub } x \ p)$. You would do me a great service by showing me as well, how you would prove (0). I am looking forward to your reply.

With my greetings and best wishes,

yours ever

5 November 1980

Edsger