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)

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

Here f is supposed to be <u>ascending</u>, i.e. to satisfy $(P0\ f)$ with

$$P0\ (a{:}b{:}c) = a \le b \wedge P0\ (b{:}c)$$

and <u>unbounded</u>, i.e. to satisfy $(\underline{A}\, y{:}\, y \ge 0{:}\, P1\ y\ f)$ with

$$P1\ y\ (p{:}q) = p > y \vee (P1\ y\ q)\ .$$

When f 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 f is ascending and unbounded, g is ascending and unbounded.

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

$$\text{sub } y\ g = (\underline{N}\ x : x \geq 0 : y \geq (\text{sub } x\ f)) \qquad (0)$$

where sub is defined by

$\underline{\text{def}}$ sub n $(p{:}q)$ =
  $\underline{\text{if}}$ $n = 0 \rightarrow p$
  $[\!]$ $n > 0 \rightarrow$ sub $(n\text{-}1)$ q
  $\underline{\text{fi}}$

and the $(\underline{N} \ldots)$ should be read as: the number of distinct values x in the range $x \geq 0$ such that $y \geq (\text{sub } x\ f)$. 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

Edsger

5 November 1980