Hamming's exercise in SASL.

In a slightly simplified version, the problem is to generate in increasing order "all" positive integers that have no other prime factors than 2 and 3. The solution à la SASL that was designed by J.L.A. van de Snepscheut was essentially the following:

```
def times n (a:b) = n*a : times n b ;
def merge (a0:b0) (a1:b1) =
      if a0 < a1 → a0 : merge b0 (a1:b1)
      [] a0 = a1 → a1 : merge b0 b1
      [] a0 > a1 → a1 : merge (a0:b0) b1
      fi ;
def x = 1 : merge (times 2 x) (times 3 x) ;
x                                                        (0)
```

This note records how —at the present state of my art— I prove the correctness of this program.

* * *

I need a notation for the n-th element of a continued concatenation:

```
sub n (a:b) = if n=0 → a
              [] n>0 → sub (n-1) b
              fi                     .
```

I need it in order to be able to express a simple theorem about the function dist which —remember the low binding power of the concatenation as indicated by the colon— is given by

$$\text{dist}\ f\ (a:b) = f\,a : \text{dist}\ f\ b \quad .$$

<u>Theorem 0</u>:

$$(\underline{A}\,n : n \geq 0 : \text{sub}\ n\ (\text{dist}\ f\ x) = f\ (\text{sub}\ n\ x))$$

<u>Proof of Theorem 0</u>. By mathematical induction (by what else?)

<u>Case</u> $n = 0$:

$$\begin{array}{l} \text{sub}\ 0\ (\text{dist}\ f\ (a:b)) = \\ \text{sub}\ 0\ (f\,a : \text{dist}\ f\ b) = \\ f\,a = \\ f\,(\text{sub}\ 0\ (a:b)) \end{array}$$

<u>Case</u> $n > 0$:

$$\begin{array}{l} \text{sub}\ n\ (\text{dist}\ f\,(a:b)) = \\ \text{sub}\ n\ (f\,a : \text{dist}\ f\ b) = \\ \text{sub}\ (n-1)\ (\text{dist}\ f\ b) = \\ f\,(\text{sub}\,(n-1)\ b) = \\ f\,(\text{sub}\ n\ (a:b)) \end{array}$$

(End of Proof of Theorem 0).

I think Theorem 0 so trivial that most people are willing to apply it without bothering to think about its proof. It should be formulated and proved once.

Let min s be the smallest element from a set s of natural numbers. For an infinite set s we define the continued concatenation sort s by

$$\text{sort } s = \min s \;:\; \text{sort } (s - \{\min s\}) \;.$$

In terms of the set formator $\underline{U}$ we have to prove that in (0)

$$x = \text{sort } (\underline{U}\; n2, n3 : n2 \geq 0, n3 \geq 0 : 2^{n2} \cdot 3^{n3}) \qquad (1)$$

To do so, we first prove about merge

<u>Theorem 1</u>. For two infinite sets s0 and s1 of natural numbers — the union of which we denote by s0+s1 — we have

$$\text{merge } (\text{sort } s0)(\text{sort } s1) = \text{sort } (s0 + s1)$$

<u>Proof of Theorem 1</u> The theorem to be proved is of the form

$$(\underline{A}\; s0, s1 :: \; p\; s0\; s1 = q\; s0\; s1)$$

where p and q are functions of two infinite sets of natural numbers; their values being continued concatenations we prove

$$(\underline{A}\; n : n \geq 0 : (\underline{A}\; s0, s1 :: \; \text{sub } n\; (p\; s0\; s1) = \text{sub } n\; (q\; s0\; s1)))$$

and this is proved using mathematical induction over n. The proof is straightforward and left to the reader. (End of Proof of Theorem 1.)

It is now not difficult to show that x , as given by (1) satisfies (0). Thanks to Theorem 0 and $p < q \Rightarrow 2 \cdot p < 2 \cdot q$ we find

$$\text{times } 2\ x = \text{sort } (\underline{U}\ n2, n3 : n2 \geq 1, n3 \geq 0 : 2^{n2} \cdot 3^{n3})$$

and, similarly,

$$\text{times } 3\ x = \text{sort } (\underline{U}\ n2, n3 : n2 \geq 0, n3 \geq 1 : 2^{n2} \cdot 3^{n3})$$

and then, thanks to Theorem 1

$$\text{merge (times 2 x) (times 3 x)} =$$
$$\text{sort } (\underline{U}\ n2, n3 : n2 \geq 0 \wedge n3 \geq 0 \wedge n2 + n3 \geq 1 : 2^{n2} \cdot 3^{n3})$$

from which immediately follows

$$1 : \text{merge (times 2 x) (times 3 x)} = x \quad .$$

* * *

In order that program (0) is an effective program I think that we have to prove more. It seems that we have to prove the uniqueness of solution (1).

With uniq defined by

$$\text{uniq } n\ x = (\underline{A}\ i : 0 \leq i < n : \text{sub } i\ x \neq \omega)$$

I think we should prove in succession

uniq n x ⇒ uniq n (times m x)

(uniq n y) ∧ (uniq n z) ⇒ uniq n (merge y z)

and, finally, from the last definition of (0)

uniq n x ⇒ uniq (n+1) x .

These proofs are very easy, but my current feeling is that we are not allowed to omit them.

(Continued after walking with the dogs.) My choice of identifier "uniq" betrays my misunderstanding. Suppose the last definition had been

x = times 2 x . (2)

This —as is easily verified— has one solution, viz.

x = zeroes

(with <u>def</u> zeroes = 0 : zeroes). The solution is unique, but I expect no implementation to find it. We fool ourselves when we regard a SASL program just as a set of recursive equations. They are recursive definitions, quite definitely with a direction! Perhaps it would have been more honest to write the definition

zeroes := 0 : zeroes .

Plataanstraat 5
5671 AL NUENEN
The Netherlands

15 June 1981

prof.dr. Edsger W. Dijkstra
Burroughs Research Fellow