A sequel to EWD592.

Recently I lectured on the structure of proofs for termination in Munich. Lack of time had prevented me from giving the argument why a variant function that decreases monotonically, must exist, and the next morning miss dr. Mila Majster, who had been in my audience, presented to me what she felt could be a counterexample. Her example was essentially:

\[
\begin{align*}
d &\text{ odd}(x) \text{ and } x \geq 3 \rightarrow x := x + 1 \\
\text{ even}(x) \text{ and } x \geq 2 \rightarrow x := x / 2
\end{align*}
\]

(The condition \( x \geq 2 \) has been added to prevent the value sequence 0, 0, 0, ...; the condition \( x \geq 3 \) has been added to prevent 1, 2, 1, 2, ... from occurring.)

My first guess at a single \( t \) was wrong. I have posed the problem to a number of my Dutch colleagues, and all first proposals were wrong. After my failure I decided to apply the technique of EWD592 writing \( t = t_1 + t_2 \), such that \( t_1 \) is decreased by the first alternative and left invariant by the second, and for \( t_2 \) the other way round. My construction was ingenious -- so ingenious that I showed it proudly to a number of people! -- but in St.Pierre de Chartreuse .... I discovered that it was wrong!

A next time I proceeded more carefully. Because I had the feeling that the binary representation of \( x \) could provide a good handle and the second alternative is in terms of the bits the simplest, I first looked for \( t_2 \).

Well the quantity that is obviously decreased by 1 by \( x := x / 2 \) is the number of trailing zero's or, alternatively, the number of significant digits. As first approximation I took the latter; because it depends on the position of the most-significant 1, it has the advantage of being undefined for \( x = 0 \), thereby justifying the constraint \( x \geq 2 \). The problem is that the first alternative increases it by 1 if \( x := x + 1 \) forms a power of 2. Because being a power of 2 is invariant under \( x := x / 2 \), a good proposal is \( t_2 = \text{the number of significant digits of } x \text{, decreased by 1 iff } x \text{ is a power of 2} \).

Searching for a \( t_1 \) took me more time. Because \( t_1 \) must be invariant under \( x := x / 2 \), it should be a function of the digit pattern as it extends itself from the most- to the least-significant 1 in \( x \). Because adding 1
will turn a number of trailing 1's into 0's and then a 0 into a 1, my first
guess was something like the number of "internal zero" (i.e. between the most-
and the least-significant 1's). I then realized, that the number of internal
zero's fails to be decreased by 1 if by \( x := x + 1 \) a power of 2 is formed.
Hence I concocted

\[ t_1 = \text{the number of internal 0's of } x, \text{ increased by } 1 \text{ iff } x \text{ is not a power of 2}. \]

This \( t_1 \) has the advantage of excluding \( x = 1 \), hence the condition \( x \geq 3 \).
Adding \( t_1 + t_2 \) I derived:

\[ t = 1 + \text{the number of significant digits} + \text{the number of internal 0's}, \]

\[ \text{decreased by 2 iff } x \text{ is a power of 2}. \]

\[
\begin{array}{ccc}
\text{x (in binary)} & \text{t(x)} \\
10 & 1 + 2 + 0 - 2 = 1 \\
11 & 1 + 2 + 0 = 3 \\
100 & 1 + 3 + 0 - 2 = 2 \\
101 & 1 + 3 + 1 = 5 \\
110 & 1 + 3 + 0 = 4 \\
111 & 1 + 3 + 0 = 4 \\
1000 & 1 + 4 + 0 - 2 = 3 \\
1001 & 1 + 4 + 2 = 7 \text{ etc.} \\
\end{array}
\]

The best thing that can be said for my \( t \) is that \( t_1 \) is exactly the
number of times the first alternative will be chosen, and \( t_2 \) is exactly the
number of times the second alternative will be chosen: it gives you all the
timing information. But if we are only interested in termination, the amount
of work we have spent seems excessive, and there is clearly room for techniques
of proving the existence of a variant function without actually constructing
one.

* * *

In St.Pierre de Chartreux M.Sintzoff showed me another way of proving
the termination of repetitive constructs. As yet I have no feeling for the
significance of his method, but it intrigued me enough to try to reconstruct
what he had shown me, and I think that in any case the method deserves to be
recorded. In order to prove for a given repetitive construct

\[
\text{DO: } \quad \text{do } B_1 \rightarrow S_1 \ ... \ ... \ B_j \rightarrow S_j \ ... \ ... \ B_n \rightarrow S_n \text{ ad }
\]

we have for a certain \( P \) and \( R \)

\[ P = \text{wp(DO, R)} \]
we don't prove that directly, but construct a sequence of repetitive constructs

$$\text{DO}_i: \quad \text{do } B_1 \rightarrow S_1 \quad \ldots \quad B_j \rightarrow S_j \ldots \quad B_n \rightarrow S_n \quad \text{od}$$

for $i = 0, 1, 2, \ldots$ such that $P_i = \text{wp}(\text{DO}_i, R)$

If for some value of $i$ --or in the limit for $i$ to infinity-- we have $P = P_i$ and $\text{DO} = \text{DO}_i$, then (2) has been proved.

With the abbreviations for $BB_i$ and $P_i$ :

$$BB_i = (\bigwedge_{j: 1 \leq j \leq n} \text{Bj}_i) \quad \text{and} \quad P_i = \text{BB}_i \text{ or } R$$

the method can be described as follows. Choose for $i = 0 : BB_0 = F$ --i.e. all guards false--; as a consequence we have $P_0 = R$. To perform the step from $i$ to $i+1$ we choose for the $B_{j+1}$ solutions of the equations ($1 \leq j \leq n$):

$$\left( B_j \rightarrow B_{j+1} \right) \quad \text{and} \quad \left( B_{j+1} \rightarrow (B_j \text{ or } (\text{wp}(S_j, P_j) \text{ and non } P_j)) \right)$$

If for a given value of $i$ the only solution of (6) is $B_{j+1} = B_j$ for all $j$, then the game clearly stops; otherwise we choose for at least one value of $j$ $B_{j+1} \neq B_j$. The result is that if $P_i$ holds initially, $\text{DO}_k$ for $k \geq i$ is certain to establish $R$ in at most $i$ steps. Formula (6) can be discovered by working backwards from the final state that should satisfy $R$; the last term "and non $P_i$" excludes that the weakening of $B_j$ to $B_{j+1}$ introduces the possibility of nontermination.

The charm of Sintzoff's method is that it is so constructive. With $S1: \ x := x + 1 \quad , \quad S2: \ x := x - 1 \quad , \quad S3: \ x := x - 2$ we derived with $R: \ x = 0$ and $P: \ x \geq 0$ without problems the following programs

$$\text{do } \text{false } \rightarrow x := x + 1 \quad \| \ x = 1 \rightarrow x := x - 1 \quad \| \ x \geq 2 \rightarrow x := x - 2 \quad \text{od}$$
$$\text{do } \text{false } \rightarrow x := x + 1 \quad \| \ \text{podd}(x) \rightarrow x := x - 1 \quad \| \ x \geq 2 \rightarrow x := x - 2 \quad \text{od}$$
$$\text{do } \text{podd}(x) \rightarrow x := x + 1 \quad \| \ \text{podd}(x) \rightarrow x := x - 1 \quad \| \ \text{peven}(x) \rightarrow x := x - 2 \quad \text{od}$$

The first program is deterministic, the second program is not, but the number of steps needed is a unique function of the initial value of $x$; in the last program, the number of steps is not (always) uniquely determined by the initial value of $x$. I spent a few hours with Sintzoff's method, and I think that they were well spent.

Plataanstraat 5
NUENEN - 4565
The Netherlands

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow