Continuations continued (for the record)

The other day I found myself struggling with a program transformation that by now should be completely standard. Therefore, I decided to write it down, mainly for the sake of memorisation.

Suppose function F is defined recursively in terms of (known) functions b, f, g, and a binary operator $\oplus$, as follows:

$$
\begin{aligned}
F.x \;=\; & (\neg b.x \;\rightarrow\; f.x \\
& \;[]\; b.x \;\rightarrow\; x \oplus F.(g.x) \\
&) \;.
\end{aligned}
$$

The problem is to transform this definition into one that involves tail recursion only. If the operator $\oplus$ is associative an appeal to the Tail Recursion Theorem does the job.

For nonassociative $\oplus$ we observe:

$$
\begin{aligned}
& x \oplus F.(g.x) \\
= \;& \quad \{ \text{ syntactic convention w.r.t. } (x\oplus) \; \} \\
& (x\oplus).(F.(g.x)) \\
= \;& \quad \{ \text{ function composition } \} \\
& ((x\oplus) \circ F).(g.x) \;,
\end{aligned}
$$

and we observe that function composition is associative. Therefore, we introduce a function G as a generalisation of F; its specification is:

$$G.h \;=\; h \circ F \;.$$

Then, F can be defined — or: implemented — in terms of G by — with Id for the identity function — :

$$F \;=\; G.Id \;.$$

A recursive definition for G can be derived easily:

$$\begin{aligned} G.h.x \;=\; (&\neg b.x \;\rightarrow\; h.(f.x) \\ \;[]\; &b.x \;\rightarrow\; G.(h \circ (x\oplus)).(g.x) \\ )\;. & \end{aligned}$$

This is all very straightforward. What I was struggling with was the following transformation. The functions h involved in the implementation of F are continued compositions of functions of the kind $(x\oplus)$. These functions can be conveniently represented by lists, since both composition and catenation are associative. For this purpose we define an abstraction function φ, as a list homomorphism:

$$\begin{aligned} \varphi.[\,] &= Id \\ \varphi.[x] &= (x\oplus) \\ \varphi.(s +\!\!+ t) &= \varphi.s \circ \varphi.t \end{aligned}$$

We introduce a new function H with <u>specification</u>:

$$H.s \;=\; G.(\varphi.s) \quad , \text{ for list } s \;.$$

Hence : $F = H.[]$.

A definition for H is derivable from the definitions of G and φ, yielding :

$$\begin{aligned} H.s.x = (\ & \neg b.x \rightarrow \varphi.s.(f.x) \\ [\!] \ & b.x \rightarrow H.(s +\!\!+ [x]).(g.x) \\ ) \ . & \end{aligned}$$

Moreover, for expressions of the form $\varphi.s.y$ the following tail recursive definition is possible :

$$\begin{aligned} \varphi.[].y &= y \\ \varphi.(s +\!\!+ [x]).y &= \varphi.s.(x \oplus y) \end{aligned}$$

By "reversal of the list s" the operation $(+\!\!+ [x])$ becomes a cons operation $(x :)$. This is too simple a transformation too be recorded here.

In the above I have omitted all derivations because they are entirely of the kind nothing-else-you-can-do. That I struggled with this exercise was caused by laziness: I tried to write down the outcome by heart instead of recalculating it. Shame on me!

Eindhoven, 30 january 1992
Rob R. Hoogerwoord