Composition, λ-calculus, and some more

The other day I saw - courtesy Tony Hoare - for the associativity of functional composition a proof of the following structure.

For functions f, g, h we have to show

(0) $\quad f \circ (g \circ h) = (f \circ g) \circ h$

where $\circ$ is defined by: for functions p, q

(1) $\quad p \circ q = (\lambda z: p.(q.z))$;

from the λ-calculus we use

(2) $\quad (\lambda x: E).y =$ "E with y substituted for x"

We observe for any f, g, h

$f \circ (g \circ h)$

$=$ {(1) with $p, q := f, (g \circ h)$}

$(\lambda z: f.((g \circ h).z))$

$=$ {(1) with $p, q, z := g, h, x$}

$(\lambda z: f.((\lambda x: g.(h.x)).z))$

$=$ {(2) with $E, y := g.(h.x), z$}

$(\lambda z: f.(g.(h.z)))$

$=$ {(2) with $E, y := f.(g.x), h.z$}

$(\lambda z: (\lambda x: f.(g.x)).(h.z))$

$=$ {(1) with $p, q, z := f, g, x$}

$(\lambda z: (f \circ g).(h.z))$

$=$ {(1) with $p, q := (f \circ g), h$}

$(f \circ g) \circ h$

thus having proved (0). The use of the λ-calculus enabled us to carry out this proof in terms of equality of functions.

Let us now pursue what happens when we circumvent the λ-calculus and carry the argument out in equality of function applications. Our demonstrandum becomes

(4) $(f \circ (g \circ h)).z = ((f \circ g) \circ h).z$ for all z

and the definition of functional composition

(5) $(p \circ q).z = p.(q.z)$ for all z .

We observe for any z

$$
\begin{aligned}
& (f \circ (g \circ h)).z \\
= & \quad \{(5) \text{ with } p,q := f, (g \circ h)\} \\
& f.((g \circ h).z) \\
= & \quad \{(5) \text{ with } p,q := g, h\} \\
& f.(g.(h.z)) \\
= & \quad \{(5) \text{ with } p,q,z := f, g, h.z\} \\
& (f \circ g).(h.z) \\
= & \quad \{(5) \text{ with } p,q := (f \circ g), h\} \\
& ((f \circ g) \circ h).z
\end{aligned}
$$

thus having proved (5). As the instantiations of p,q show, these are in a sense different renderings of the "same" proof, but in another sense the proofs are very different!

In the proof of (0) —as the reader may verify—, the first two steps may be interchanged, and so may the last two; in the proof of (5) —as the reader may verify— the order of the steps is fixed. (The latter proof is, in fact, a genuine instance of "There is only one thing you can do.".) Since there is no virtue in spurious freedom this is a reason for preferring the latter.

More important is that the two middle steps of the former proof have completely disappeared in the latter one. We can view them as an artefact of the notation; from a technical point we can observe that, whereas (1) has only two free variables, (5) has three: the additional one enables us to subsume the identification of the subexpression h.2 in its instantiation. The more free variables a rewrite rule has, the more powerful it seems! In spite of all we hear in favour of "lifting", the transition from (5) to (1) does not seem an improvement here.

Another striking difference is in the depth of parenthesis nesting. In the latter proof it is homogeneously 2, in the former it begins and ends at 1, but reaches 4 in between (which seems excessive). The observation is made because

the difference is striking, but I hesitate to attach a moral to it. Symbol counts for the results in the former proof yield 7, 16, 23, 16, 23, 16, 7, (sum 108), for the latter 11, 11, 11, 11, 11 (sum 55); sum and maximum differ both by a factor of 2.

With thanks to Tony Hoare for exposing me to this example of the λ-calculus in action.

Post Scriptum In a hand-out of Tony's on Category Theory I saw a familiar ugliness that we can now recognize as a manifestation of the same phenomenon. In an example he introduces the (would-be) set Russell in the usual way

$$(6) \qquad Russel = \{ x \mid x \notin x \}$$

and then needs -again in the usual way- an argument by case analysis of a whole paragraph to show that the acceptance of the above definition leads to a contradiction. Had he defined Russel, instead of by (6), by

$$(7) \qquad x \in Russel \equiv x \notin x \quad \text{for all } x$$

the instantiation $x := Russel$ would have sufficed to produce the desired contradiction:

$$Russel \in Russel \equiv Russel \notin Russel \ .$$

It is exactly the same phenomenon: in λ-

notation, (6) could be rendered by defining the membership function ϵRussel by

$$\epsilon\text{Russel} = (\lambda x: x \notin x) \quad ;$$

in (7), this function is applied to a dummy, ready for instantiation.

I knew that the set notation as used in (6) is a notational mistake. (The first x plays a double rôle: it identifies the dummy and represents the general element.) I also knew from experience that it is almost always more convenient to name the membership predicate than to name the set. And now we know the simple reason why.

Austin, 16 September 1989

prof. dr. Edsger W. Dykstra
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188
USA