**Dot or juxtaposition?**

About 5 years ago I adopted the convention -- which is, I think, due to E.W. Dijkstra and W.H.J. Feijen -- to denote (function) application by an explicit symbol, namely the infix operator · ("dot"), instead of juxtaposition. So I write f·x instead of f x or f(x) .

In the SASL/KRC/Miranda language family application is denoted by (unparenthesized) juxtaposition; furthermore, the practitioners of the Bird/Meertens calculus consistently use juxtaposition for application. These simple facts always confused me a little: is the use of an explicit symbol really better or is it just exaggerated purism? Let me explain my confusion by giving an argument in favour of juxtaposition first. Next, I present an observation that, for me, settles the issue in favour of an explicit operator.

Juxtaposition saves symbols; thus, it contributes to the brevity and, possibly, clarity of our formulae. For example, in one of my larger functional programs 17 % of the symbols are dots. Of course, juxtaposition can be used with only one (context-independent) meaning. It seems, therefore, wise to use it for one of the more elementary operations. Indeed, application is a good candidate.

Application can be considered as a binary operation, mapping a function and an argument to a value. By using an explicit infix operator for it we recognise that application itself is a (two-argument) function that, in some respect, does not differ from other binary operators. We illustrate this as follows.

For binary operator $\oplus$ , say, we denote the two-argument function represented by it by $(\oplus)$ , and we denote the one-argument functions obtained by fixing one of $\oplus$'s arguments by $(a\oplus)$ and $(\oplus b)$ respectively. That is, we have:

$$(\oplus)\cdot x\cdot y = x \oplus y \ ,$$
$$(a\oplus)\cdot y = a \oplus y \ ,$$
$$(\oplus b)\cdot x = x \oplus b \ , \quad \text{for all } a, b, x, y \ .$$

This convention proves to be convenient because it enables us to consider the standard operators as values in their own right. For example, $(\oplus)$ can be

used as argument in an application of a function that "expects" a two-argument function for its parameter.

We now have: $(\cdot)$ is the function that applies functions to arguments, $(f\cdot)$ is not very useful because it is equivalent to $f$, and $(\cdot a)$ is the function that applies functions to the constant argument $a$. For example, if we use $s\cdot i$, for natural $i$, to denote the $i$-th element of list $s$ then $s\cdot 0$ is the head element of $s$; hence, the function that maps lists to their head elements is $(\cdot 0)$.

As another example, let functions $K$ and $C$ be defined by:

$$K\cdot x\cdot y = x \ ,$$
$$C\cdot x\cdot y = y \ .$$

Using $K$ and $C$ we can now define a pair-forming function and its inverses in two different ways -- where $\circ$ denotes function composition -- :

(0)
$$\begin{aligned} pair\cdot x\cdot y\cdot s &= s\cdot x\cdot y \ , \\ fst\cdot p &= p\cdot K \ , \\ snd\cdot p &= p\cdot C \ . \end{aligned}$$

(1)
$$\begin{aligned} pair\cdot x\cdot y &= (\cdot y)\circ(\cdot x) \ , \\ fst &= (\cdot K) \ , \\ snd &= (\cdot C) \ . \end{aligned}$$

In both cases we have $fst\cdot(pair\cdot x\cdot y) = x$ and $snd\cdot(pair\cdot x\cdot y) = y$. Although in this case I prefer definition (0) to definition (1), because the former matches our manipulative needs better than the latter does, the greater level of abstraction of definition (1) may sometimes be advantageous -- honesty forces me to admit that I am still looking for a more convincing example -- .

Eindhoven, 28 november 1989
Rob Hoogerwoord
department of mathematics and computing science
Eindhoven University of Technology