Separating my concerns: a sequel to rh199

An omission in rh199 gave rise to the following observations. For fixed A we consider the datatype $T{\cdot}B$ defined by:

$$T{\cdot}B \;=\; B \leftarrow A \;.$$

For function g in $C \leftarrow B$ we define a "lifted" version $g*$ ("g map") in $T{\cdot}C \leftarrow T{\cdot}B$ by:

$$g* \cdot f \;=\; g \circ f \quad , \quad \text{for all } f \text{ in } T{\cdot}B \,.$$

(aside: because apparently $g* = (g\circ)$ the introduction of the new symbol $*$ looks a bit overdone, but nevertheless there are good reasons for doing so.
)

We now study polymorphic functions P that have type $T{\cdot}B \leftarrow T{\cdot}B$, for all B. Such a function P is called (by category theorists) a natural transformation if for all B, C and g in $C \leftarrow B$ we have:

$$P \circ g* \;=\; g* \circ P \;.$$

If P has an inverse then an easy calculation shows that P is a natural transformation if and only if P^{-1} is a natural transformation. Backhouse calls a natural transformation that has an inverse a "witness to the isomorphism" between $T{\cdot}B$ and $T{\cdot}B$. A very trivial example of such a witness is the identity function but, as we shall show, many more such witnesses exist.

We now study a special case. For function p in $A \leftarrow A$ we define P by:

$$P.f \;=\; f \circ p \quad , \quad \text{for all } f \text{ in } T.B \;.$$

Thus defined P is a natural transformation:

$$
\begin{array}{ll}
 & (P \circ g*).f \\
= & \quad \{ \text{ definition of } \circ \ \} \\
 & P.(g* . f) \\
= & \quad \{ \text{ definitions of } P \text{ and } * \ \} \\
 & (g \circ f) \circ p \\
= & \quad \{ \text{ associativity of } \circ \ \} \\
 & g \circ (f \circ p) \\
= & \quad \{ \text{ definitions of } * \text{ and } P \ \} \\
 & g* . (P.f) \\
= & \quad \{ \text{ definition of } \circ \ \} \\
 & (g* \circ P).f \;.
\end{array}
$$

So, in this case it all boils down to associativity of function composition. (The other steps in this derivation only serve to eliminate and reintroduce names.) Observing that $g* = (g\circ)$ and that $P = (\circ p)$, P's being a natural transformation boils down to:

$$(\circ p) \circ (g \circ) \;=\; (g \circ) \circ (\circ p) \;,$$

which is just another formulation of the associativity of function composition.

Moreover, we have that P has an inverse if p has an inverse; actually, for q in $A \leftarrow A$ and Q in $T.B \leftarrow T.B$ such that, for all f, $Q.f = f \circ q$, we even have:

$$p^{-1} = q \quad \Rightarrow \quad P^{-1} = Q \ .$$

(Phrased differently: $(\circ\, p)^{-1} = (\circ\, p^{-1})$.)

<u>proof</u>:

$$\begin{array}{ll} & (Q \circ P).f \\ = & \quad \{ \text{ definitions of } \circ, P, Q \ \} \\ & (f \circ p) \circ q \\ = & \quad \{ \text{ associativity of } \circ \text{ (again!) } \} \\ & f \circ (p \circ q) \\ = & \quad \{ \ p^{-1} = q \ \} \\ & f \ , \end{array}$$

and, similarly, one derives $(P \circ Q).f = f$.

□

In words, we conclude that <u>every</u> invertible function in $A \leftarrow A$ —for finite A this amounts to "every permutation of A"— generates a witness to the isomorphism between $T.B$ and $T.B$.

* * *

The above has been obtained from an example concerning reversal of (finite) lists, by abstraction from irrelevant details. Reintroducing some of those details we choose, for all natural j :

$$A = [0,j] \quad \text{(i.e. the interval } \{i \mid 0 \le i \le j\} \text{).}$$

In the interval $[0,j]$ the function $(j-)$ is its own inverse. Now we specify rev in $T.B \leftarrow T.B$ by:

(0) $\quad rev.f = f \circ (j-)$, for all f in $T.B$.

(By the way, this is the most concise specification of rev I have ever seen, and it is new to me. Although (0) defines rev as a function, it is a specification for programs —like recursive definitions— implementing that function.)

Because this is just an instance of the above "theory" — it is too simple to be called that way— we are entitled to conclude that

- rev is a witness to the isomorphism between $B \leftarrow [0,j]$ and $B \leftarrow [0,j]$, with $rev \circ g* = g* \circ rev$, and that
- (as I pointed out in rh199) this fact follows from the above specification alone without construction of a "program", but also that
- rev is just one out of a huge collection of such witnesses, as every permutation instead of $(j-)$ does the job; so, requiring rev to be such a witness is far too weak to be useful as a specification (and any "derivation" from this requirement alone must involve either cheating or sheer luck).

The above "theory" is very elementary; in both proofs the key notion is the associativity of function composition. Of course, this smells of category theory, but understanding the above requires no knowledge of whatever theory.

In the above I have confined myself to viewing lists as functions and I (deliberately) ignored their structure, Doing justice to the latter is, however, easy. With $\mathcal{L}_j \cdot B$ for the type of lists of length j and with elements of type B, we postulate (not unreasonably), for all j:

$$g* \text{ in } \mathcal{L}_j \cdot C \leftarrow \mathcal{L}_j \cdot B \quad , \text{ for all } g \text{ in } C \leftarrow B .$$

(That is: $*$ preserves list structure; this additional requirement, which is not imposed upon $\circ$, justifies the introduction of $*$ besides $\circ$.)
We now strengthen the specification of rev with:

(1) $\qquad$ rev in $\mathcal{L}_j \cdot B \leftarrow \mathcal{L}_j \cdot B$.

(Universally quantified over B and j, the conjunction of (0) and (1) completely specifies list reversal.)
The typing rule for function composition now immediately yields:

$\qquad$ rev $\circ$ $g*$ and $g* \circ$ rev both have type
$\qquad$ $\mathcal{L}_j \cdot C \leftarrow \mathcal{L}_j \cdot B$, as required.

The above story is completely independent of how the "functional view" of lists is related to the "structural view", that is, independent of how lists of a certain structure are interpreted as functions. As a result, the above is equally well applicable to "cons-lists", "snoc-lists", "join-lists" or what have you. From the point of view of structure, the difference between "cons-lists" and "snoc-lists" is only a notational artefact; only by means of their interpretation as functions can the two kinds of lists be distinguished in a meaningful way.

This exercise raises the question, at least for me, what is the right amount of emphasis to be placed upon datastructures compared to the full algebraic structures represented by these datastructures. The most telling example, in this respect, is the algebraic poorness of $\mu(X \mapsto \mathbb{1}+X)$ compared to the richness of the natural numbers with addition, subtraction, multiplication, et cetera. As I have currently no idea about the answer to this question, its discussion has to be postponed for a while. (But, he who doubts does not sin.)

__acknowledgements__: to Roland Backhouse who triggered all this simply by his asking how I would derive a definition for rev, and to the ETAC for suggesting some improvements of the presentation.

□

Eindhoven, 9 september 1993
Rob R. Hoogerwoord