SYDNEY TYPE THEORY MEETUP – 24/07/17

Polymorphic functions & naturality – Richard Garner

(richard.garner@mq.edu.au)

In Haskell, we can write functions of type $\forall a.\ [a] \to [a]$:

$$\left.\begin{array}{l}
\bullet\ \text{reverse} :: \text{forall } a.\ [a] \to [a] \\
\bullet\ \text{tail} \\
\bullet\ \lambda_ \to [\,] \\
\bullet\ \text{id} \\
\bullet\ \text{cycle left, cycle right} \\
\bullet\ h\ [\,] = [\,] \\
\quad h\ [x] = [x, x] \\
\quad h\ [x, y] = [y, x, y] \\
\quad h\ ls = ls \mathbin{++} \text{reverse } ls
\end{array}\right\} + \text{composites}$$

Non-examples

- $\text{sort} :: \forall a.\ \text{Ord } a \Rightarrow [a] \to [a]$
- $\text{nub} :: \forall a.\ \text{Eq } a \Rightarrow [a] \to [a]$

Problem: classify all functions of type $\forall a.\ [a] \to [a]$.

To solve: work with idealised version of Haskell without laziness, and with only strict, total functions.
Now use some category theory! Basic idea:

$$\boxed{\text{Polymorphism} = \text{naturality}}.$$

In this particular case, "naturality" means the following:

If $\theta :: \forall a.\ [a] \longrightarrow [a]$, and $f :: A \longrightarrow B$, then whenever $ls :: [A]$, we have

instantiation of θ at the type A

$$(\text{map } f)\ (\theta_A\ ls) = \theta_B\ ((\text{map } f)\ ls) \quad (*)$$

eg:

$$(\text{map } f)\ (\text{reverse } ls) = \text{reverse}\ ((\text{map } f)\ ls)$$

In categorical lingo:-

- We have a category <u>Hask</u> whose objects are (non-lazy) Haskell types, and whose morphisms $A \longrightarrow B$ are (strict, total) Haskell functions of type $A \to B$ (identified up to extensional equivalence).
- We have a functor

$$\begin{aligned} [\ ] :\ &\underline{\text{Hask}} \longrightarrow \underline{\text{Hask}} \\ &A \longmapsto [A] \\ f : A \longrightarrow B &\longmapsto \text{map } f : [A] \longrightarrow [B] \end{aligned}$$

(satisfying axioms).

In Haskell: Class Functor f where

fmap :: (a → b) → (f a) → f b

and $\left\{\begin{matrix}[\ ] \\ \text{Maybe} \\ \text{IO} \\ \{\text{any monad}\}\end{matrix}\right\}$ is an instance of this.

- A natural transformation $\theta : [\ ] \Rightarrow [\ ]$ comprises:-

- for each $A \in \underline{\text{Hask}}$, a morphism $\theta_A : [A] \longrightarrow [A]$ (components)
- such that for any $f : A \to B$ in $\underline{\text{Hask}}$, the following composites coincide:

$$\begin{tikzcd} {[A]} \arrow[r, "\theta_A"] \arrow[d, "\text{map } f"'] & {[A]} \arrow[d, "\text{map } f"] \\ {[B]} \arrow[r, "\theta_B"] & {[B]} \end{tikzcd} \qquad (*)$$

ie: $\theta_B \cdot (\text{map } f) = (\text{map } f) \cdot \theta_A$

eg: reverse . (map f) = (map f) . reverse.

Idea from category theory:

polymorphic $f^{\underline{n}}$s of type $\forall a. [a] \longrightarrow [a]$ <u>are the same as</u> natural transformations $[\,] \Rightarrow [\,]$.

So to classify the former, we may as well classify the latter.
And we can do this!

Suppose we're given $\theta : [\,] \Rightarrow [\,]$. We want to know how the component/instantiation $\theta_A : [A] \longrightarrow [A]$ acts on some given list $[x_0, .., x_{n-1}]$.

Consider a type

Data $\text{Enum}_n = 0 \mid 1 \mid \cdots \mid n-1$.

There's a function $f : \text{Enum}_n \longrightarrow A$ defined by

$$\begin{aligned} f\ 0 &= x_0 \\ &\vdots \\ f\ (n-1) &= x_{n-1} \end{aligned}$$

and now $[x_0, .., x_{n-1}] = (\text{map } f)\ [0, .., n-1]$.

So now we can consider

$$\begin{tikzcd}
{[\text{Enum}_n]} \arrow[r, "\theta_{\text{Enum}_n}"] \arrow[d, "\text{map } f"'] & {[\text{Enum}_n]} \arrow[d, "\text{map } f"] \\
{[A]} \arrow[r, "\theta_A"] & {[A]}
\end{tikzcd}$$

.

Tracing $[0, .., n-1]$ in $[\text{Enum}_n]$ around this square, we get:

$$\begin{tikzcd}
{[0, .., n-1]} \arrow[r, mapsto] \arrow[d, mapsto] & ls \arrow[dr, mapsto] & \\
{[x_0, .., x_{n-1}]} \arrow[r, mapsto] & \theta_A\ [x_0, .., x_{n-1}] \arrow[r, phantom, "="] & (\text{map } f)\ ls
\end{tikzcd}$$

Now, $ls = [i_0, .., i_k]$ where each i is in $0, .., n-1$

So $(\text{map } f)\ ls = [x_{i_0}, .., x_{i_k}]$.

In summary: if $\theta_{\text{Enum}_n}\ [0, .., n-1] = [i_0, .., i_k]$

then $\theta_A\ [x_0, .., x_{n-1}] = [x_{i_0}, .., x_{i_k}]$

(This argument is an instance of the Yoneda lemma in category theory)

What this means: to give $\theta : [\ ] \Rightarrow [\ ]$ is equally to give, for each natural number n, a list $l_n :: [\text{Enum}_n]$. We can encode this data as (certain) Haskell functions $\text{Int} \longrightarrow [\text{Int}]$.

... and we can do this in practice! (See example.hs)