Co-algebras — a short summary

MMF, 28 feb 1994

Co-algebras are dual to algebras, and finality is dual to initiality. We shall be very brief in the formal exposition, since it is just a matter of dualisation, but more elaborate in the informal explanation and examples.

1 Co-algebras. Let $\mathcal{A}$ be a category (for example $\mathcal{S}et$), and take this one as the default category. Let F be an endofunctor (so its type is $\mathcal{A} \to \mathcal{A}$).

- An F-co-algebra is: a morphism φ of type $a \to Fa$ for some a, called its *carrier*.
- An F-co-algebra homomorphism from φ to ψ is: a morphism f satisfying

$$\varphi \,;\, Ff \quad = \quad f \,;\, \psi \quad .$$

- The (pre)category $\mathcal{C}o\mathcal{A}lg(F)$ of F-co-algebras is defined thus: it is built upon category $\mathcal{A}$ (so it inherits the composition and identities from $\mathcal{A}$), its objects are F-co-algebras, its morphisms are F-co-algebra homomorphisms, its typing is defined as you would expect from the definition of 'homomorphism'.

Dualisation of initiality in $\mathcal{A}lg(F)$ gives us the following about finality in $\mathcal{C}o\mathcal{A}lg(F)$.

2 Finality. An F-co-algebra α is final (a final object in $\mathcal{C}o\mathcal{A}lg(F)$) if: there exists a mapping $[\![\_]\!]$ satisfying ana-Type and ana-Charn, and therefore also the other laws:

$\varphi\ F$-co-algebra	$\Rightarrow$	$[\![\varphi]\!]$: carrier of $\varphi \to$ carrier of α	ana-Type
$f = [\![\varphi]\!]$	$\equiv$	$\varphi \,;\, Ff = f \,;\, \alpha$	ana-Charn
$\varphi \,;\, F\,[\![\varphi]\!] = [\![\varphi]\!] \,;\, \alpha$			ana-Self
$id = [\![\alpha]\!]$			ana-Id
$\varphi \,;\, Ff = f \,;\, \alpha \ \wedge\ \varphi \,;\, Fg = g \,;\, \alpha$	$\Rightarrow$	$f = g$	ana-Uniq
$\varphi \,;\, Ff = f \,;\, \psi$	$\Rightarrow$	$[\![\varphi]\!] = f \,;\, [\![\psi]\!]$.	ana-Fusion

We call a morphism of the form $[\![\varphi]\!]$ an *anamorphism*.

3 Disussion. For an intuitive understanding of the above results, we give here an explanation in terms of category $\mathcal{A} = \mathcal{S}et$ (though most formulas are valid in any category). First observe that by dualisation it also follows that a final F-co-algebra α: $t \to Ft$ is an isomorphism, so that it has an inverse, which we write as $\alpha^{\cup}$: $Ft \to t$. (In programming language terminology *types* are *sets*, and for this reason we use the letter t for the carrier of α .)

We may call α a *destructor* since it decomposes an element of its carrier t into its constituents: a value in the set Ft, an F-structure over t. It is right to say 'decomposes' and 'constituents' because subsequent application of $\alpha^{\cup}$ yields the original value in t back again: $\alpha ; \alpha^{\cup} = id_t$.

Finality of α means that for each $\varphi\colon a \to Fa$ there exists precisely one f satisfying:

$$
\begin{array}{lcll}
\varphi ; Ff & = & f ; \alpha, & \text{or, equivalently} \\
f ; \alpha & = & \varphi ; Ff & \text{or, equivalently} \\
f & = & \varphi ; Ff ; \alpha^{\cup} & : \quad a \to t \quad .
\end{array}
$$

The second equation can be read as a definition of f *by induction on the α-destruction of the result* of f. The third equation says how to compute f: first apply φ, then do f recursively on the results, and finally apply $\alpha^{\cup}$, thus constructing a value in the set t.

In a programming language syntax we write the declaration "let α be a final F-co-algebra, say with carrier t" thus:

$$\textbf{datatype } t \textbf{ has destructors } \alpha\colon t \to Ft \quad .$$

In case F has the form $F = G \times H$, we know that $\alpha = \beta \vartriangle \gamma$ and we may write:

$$\textbf{datatype } t \textbf{ has destructors } \beta\colon t \to Gt,\ \gamma\colon t \to Ht \quad .$$

Moreover, the programmer may introduce another name for $[\![\_]\!]$, say *generate*, and introduce that by a with-clause thus:

$$\textbf{datatype } t \textbf{ with } \mathit{generate} \textbf{ has destructors } \alpha\colon t \to Ft \quad .$$

4 Parameterisation. Let F have an extra parameter, a say, so that α and t depend on a. Writing Ta instead of t_a, we have $\alpha_a\colon Ta \to F(a, Ta)$.

By dualisation of the theory for algebras, it follows that α is natural in a, that T is a functor, and that:

$$[\![\varphi]\!] ; Tf = [\![\varphi ; F(f, id)]\!] \qquad \text{ana-map-Fusion}$$

$$\varphi ; F(f, f) = f ; \psi \quad \Rightarrow \quad [\![\varphi]\!] ; Tf = [\![\psi]\!] \quad . \qquad \text{ana-}\textsc{Trafo}$$

In the latter law the typing reads:

$$\varphi\colon a \to F(a, a) \qquad f\colon a \to b \qquad \psi\colon b \to F(b, b)$$

so

$$[\![\varphi]\!]\colon a \to Ta \qquad\qquad [\![\psi]\!]\colon b \to Tb \quad .$$

The programming language syntax now reads:

$$\textbf{datatype } Ta \textbf{ with } [\![\_]\!] \textbf{ has destructors } \alpha\colon Ta \to FTa \quad .$$

Examples

Throughout this section we speak as if $\mathcal{A} = \mathcal{S}et$. But a lot of the formal claims hold in any category.

5 Final Id-co-algebra. Consider a one-point set, or more generally a final object 1 in $\mathcal{A}$. Then, for arbitrary a there exists precisely one $f\colon a \to 1$. So, trivially, also for arbitrary $\varphi\colon a \to a$ there exists precisely one f satisfying:

$$\varphi \mathbin{;} f \quad = \quad f \mathbin{;} id_1 \quad : \quad a \to 1 \quad .$$

Now observe that id_1 and φ are Id-co-algebras:

$$\begin{array}{lcl} id_1 & : & 1 \to Id\, 1 \\ \varphi & : & a \to Id\, a \quad . \end{array}$$

Thus we have shown that $id_1\colon 1 \to 1$ is a final Id-co-algebra.

In the programming language syntax we may declare:

datatype *unit* **has destructors** *iden*: *unit* → *unit* .

Then *unit* is (isomorphic to) 1, and *iden* is its identity.

6 Streams. Let $F(x, y) = x \times y$, so that $F(a, \_) = \underline{a} \times Id$. We shall show that the datatype of streams over a is a final $F(a, \_)$-co-algebra.

First, recall the datatype of streams over a: the set $\mathit{Stream}\, a$ consists of all infinite sequences $[a_0, a_1, \ldots]$ with elements from a, and the head and tail function are defined thus:

$$\begin{array}{lclcl} hd & = & [a_0, a_1, \ldots] \mapsto a_0 & : & \mathit{Stream}\, a \to a \\ tl & = & [a_0, a_1, \ldots] \mapsto [a_1, \ldots] & : & \mathit{Stream}\, a \to \mathit{Stream}\, a \quad . \end{array}$$

So, we have

$$\begin{array}{lcl} hd \vartriangle tl & : & \mathit{Stream}\, a \to a \times \mathit{Stream}\, a \\ & = & \mathit{Stream}\, a \to (\underline{a} \times Id)\, \mathit{Stream}\, a \quad , \end{array}$$

and $hd \vartriangle tl$ is a $(\underline{a} \times Id)$-co-algebra.

Next, observe that for arbitrary $e\colon b \to a$ and $g\colon b \to b$ the two equations below fully determine a function $f\colon b \to \mathit{Stream}\, a$:

$$\begin{array}{lclcl} f \mathbin{;} hd & = & e & : & b \to a \\ f \mathbin{;} tl & = & g \mathbin{;} f & : & b \to \mathit{Stream}\, a \quad . \end{array}$$

Indeed, by induction on n one can easily prove that the two equations imply:

$$\begin{array}{lcl} f \mathbin{;} tl^n \mathbin{;} hd &=& g^n \mathbin{;} e \qquad \text{or, equivalently —in } \mathcal{S}et\text{—} \\ f\,(x) &=& [e(x), e(g(x)), e(g(g(x)), \ldots, e(g^n(x)), \ldots] \quad . \end{array}$$

Using the product laws, the two equations above can be written as one equation:

$$f \mathbin{;} hd \vartriangle tl \quad = \quad e \vartriangle g \mathbin{;} id_a \times f \quad .$$

Putting $\alpha = hd \vartriangle tl$ and $\varphi = e \vartriangle g$, this equation reads:

$$f \mathbin{;} \alpha \quad = \quad \varphi \mathbin{;} (\underline{a} \times Id)\, f \quad .$$

So, since this determines f uniquely, $\alpha = hd \vartriangle tl$ is a final $(\underline{a} \times Id)$-co-algebra. Moreover, since $[\![\varphi]\!]$ is a notation for the f so defined, we have:

$$[\![e \vartriangle g]\!]\,(x) \quad = \quad [e(x), e(g(x)), e(g(g(x)), \ldots, e(g^n(x)), \ldots] \quad .$$

Since $hd \vartriangle tl$ is a final co-algebra, we may declare it by:

datatype *Stream a* **with** *generate* **has destructors**
 $hd\colon\ \mathit{Stream}\ a \to a$
 $tl\colon\ \mathit{Stream}\ a \to \mathit{Stream}\ a \quad .$

7 Iterate. We continue the preceding discussion of streams.
Let $f\colon\ a \to a$ be arbitrary, and consider the function f-iterate, denoted f^ω:

$$f^\omega\,(x) \quad = \quad [x, fx, f^2x, f^3x, \ldots] \quad : \quad a \to \mathit{Stream}\ a \quad .$$

From the formulas above, it is now immediate that we can express f^ω as an anamorphism:

$$f^\omega \quad = \quad [\![id \vartriangle f]\!] \quad .$$

As an example, the stream of natural numbers, and the stream of ones now read:

$$\begin{array}{lclcl} nats &=& zero \mathbin{;} succ^\omega &:& 1 \to \mathit{Stream}\ nat \\ ones &=& one \mathbin{;} id^\omega &:& 1 \to \mathit{Stream}\ nat \quad . \end{array}$$

Further, we have the following law:

$$g \mathbin{;} f = f \mathbin{;} h \quad \Rightarrow \quad g^\omega \mathbin{;} \mathit{Stream}\ f = f \mathbin{;} h^\omega \quad . \qquad \text{iterate-Trafo}$$

The proof is easy:

$$g^\omega \mathbin{;} \mathit{Stream}\ f = f \mathbin{;} h^\omega$$
$\equiv$ definition iterate

$$
\begin{array}{ll}
 & [\![id \vartriangle g]\!] \mathbin{;} \mathit{Stream}\, f = f \mathbin{;} [\![id \vartriangle h]\!] \\
\Leftarrow & \qquad \text{ana-}\textsc{Trafo} \\
 & id \vartriangle g \mathbin{;} f \times f = f \mathbin{;} id \vartriangle h \\
\equiv & \qquad \text{product} \\
 & g \mathbin{;} f = f \mathbin{;} h \quad .
\end{array}
$$

As an application of iterate-TRAFO we find another way to express the stream of ones:

$$
id^{\omega} \mathbin{;} \mathit{Stream}\; one \quad = \quad one \mathbin{;} id^{\omega} \quad (= \quad ones) \quad .
$$

As another application we derive an alternative recursive equation for f-iterate:

$$
\begin{array}{ll}
 & f^{\omega} = [\![id \vartriangle f]\!] \\
\equiv & \qquad \text{ana-}\textsc{Charn} \text{ (with functor } \underline{a} \times Id\,) \\
 & f^{\omega} \mathbin{;} hd \vartriangle tl = id \vartriangle f \mathbin{;} id_a \times f^{\omega} \\
\equiv & \qquad \text{product} \\
 & f^{\omega} \mathbin{;} hd \vartriangle tl = id \vartriangle (f \mathbin{;} f^{\omega}) \\
\equiv & \qquad \text{iterate-}\textsc{Trafo} \text{ (the condition } f \mathbin{;} f = f \mathbin{;} f \text{ is clearly true)} \\
 & f^{\omega} \mathbin{;} hd \vartriangle tl = id \vartriangle (f^{\omega} \mathbin{;} f) \quad .
\end{array}
$$

By the way, law iterate-TRAFO expresses nothing but the fact that the iterate operation $\_^{\omega}$ is, in a sense, natural in its parameter $\_$. Precisely, we claim that $\_^{\omega}$ is a natural transformation with type:

$$
\_^{\omega} \quad : \quad Id \stackrel{.}{\to} \mathit{Stream} \quad \text{in category } \mathcal{A}lg(Id) \quad .
$$

To prove this claim, we first observe that an Id-algebra is just a morphism of type $a \to a$ for some a, and so it makes sense to consider Stream as a functor on $\mathcal{A}lg(Id)$. Next, we unfold the naturality claim, and find that it is precisely law iterate-TRAFO:

$$
\begin{array}{ll}
 & \_^{\omega}\colon\ Id \stackrel{.}{\to} \mathit{Stream} \quad \text{in } \mathcal{A}lg(Id) \\
\equiv & \qquad \text{definition naturality} \\
 & f\colon\ g \to_{Id} h \qquad \Rightarrow \qquad Id\, f \mathbin{;} h^{\omega} = g^{\omega} \mathbin{;} \mathit{Stream}\, f \\
\equiv & \qquad \text{definition of } \to_{Id} \text{ and } Id\text{, commutativity of equality} \\
 & g \mathbin{;} f = f \mathbin{;} h \qquad \Rightarrow \qquad g^{\omega} \mathbin{;} \mathit{Stream}\, f = h^{\omega} \mathbin{;} f \quad .
\end{array}
$$

8 Exercise. The function zip maps a pair of streams into a stream of pairs, like a zipper, and the function $zipwith\text{-}f$ applies in addition function f to each pair in the result stream of zip:

$$
[a, \ldots], [b, \ldots] \stackrel{zip}{\mapsto} [(a, b), \ldots] \stackrel{\mathit{Stream}\, f}{\mapsto} [f(a, b), \ldots] \quad .
$$

Express zip and $zipwith\text{-}f = zip \mathbin{;} \mathit{Stream}\, f$ both as a single anamorphism.

9 Exercise. Express the categorical product as a final co-algebra. (Hint: prove that $exl \vartriangle exr$ is a final $\underline{a} \times \underline{b}$-co-algebra.)

10 Note. Recall that the set $Stream\, a$ consists of the *infinite* sequences over a, and that its destructor $hd \vartriangle tl$ is a final $(\underline{a} \times Id)$-co-algebra.

Also, recall that the set $Seq\, a$ consists of the *finite* sequences over a, and its constructor $nil \triangledown cons$ is an initial F-algebra, where:

$$F \quad = \quad \underline{1} + \underline{a} \times Id \quad .$$

The remarkable point is that the carrier of the final F-co-algebra (for that same F) is a set that consists of all *finite and infinite* sequences over a. We don't prove that claim here.