

# 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  $\mathcal{Set}$ ), and take this one as the default category. Let  $F$  be an endofunctor (so its type is  $\mathcal{A} \rightarrow \mathcal{A}$ ).

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

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

- The (pre)category  $CoAlg(F)$  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  $Alg(F)$  gives us the following about finality in  $CoAlg(F)$ .

**2 Finality.** An  $F$ -co-algebra  $\alpha$  is final (a final object in  $CoAlg(F)$ ) 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 $\varphi \rightarrow$ carrier of $\alpha$	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  $\mathcal{A} = \mathcal{Set}$  (though most formulas are valid in any category). First observe that by dualisation it also follows that a final  $F$ -co-algebra  $\alpha: t \rightarrow Ft$  is an isomorphism, so that it has an inverse, which we write as  $\alpha^\cup: Ft \rightarrow t$ . (In programming language terminology *types* are *sets*, and for this reason we use the letter  $t$  for the carrier of  $\alpha$ .)

We may call  $\alpha$  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  $\alpha$  means that for each  $\varphi: a \rightarrow Fa$  there exists precisely one  $f$  satisfying:

$$\begin{aligned} \varphi ; Ff &= f ; \alpha, && \text{or, equivalently} \\ f ; \alpha &= \varphi ; Ff && \text{or, equivalently} \\ f &= \varphi ; Ff ; \alpha^\cup && : a \rightarrow t . \end{aligned}$$

The second equation can be read as a definition of  $f$  by *induction on the  $\alpha$ -destruction of the result* of  $f$ . The third equation says how to compute  $f$ : first apply  $\varphi$ , 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  $\alpha$  be a final  $F$ -co-algebra, say with carrier  $t$ ” thus:

**datatype  $t$  has destructors  $\alpha: t \rightarrow Ft$  .**

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

**datatype  $t$  has destructors  $\beta: t \rightarrow Gt, \gamma: t \rightarrow Ht$  .**

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

**datatype  $t$  with *generate* has destructors  $\alpha: t \rightarrow Ft$  .**

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

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

$$\begin{aligned} \llbracket \varphi \rrbracket ; Tf &= \llbracket \varphi ; F(f, id) \rrbracket && \text{ana-map-Fusion} \\ \varphi ; F(f, f) = f ; \psi &\quad \Rightarrow \quad \llbracket \varphi \rrbracket ; Tf = \llbracket \psi \rrbracket && \text{ana-TRAFO} \end{aligned}$$

In the latter law the typing reads:

$$\varphi: a \rightarrow F(a, a) \quad f: a \rightarrow b \quad \psi: b \rightarrow F(b, b)$$

so

$$\llbracket \varphi \rrbracket: a \rightarrow Ta \quad \llbracket \psi \rrbracket: b \rightarrow Tb .$$

The programming language syntax now reads:

**datatype  $Ta$  with  $[-]$  has destructors  $\alpha: Ta \rightarrow FTa$  .**

## Examples

Throughout this section we speak as if  $\mathcal{A} = \mathcal{Set}$ . 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  $f: a \rightarrow 1$ . So, trivially, also for arbitrary  $\varphi: a \rightarrow a$  there exists precisely one  $f$  satisfying:

$$\varphi ; f = f ; id_1 \quad : \quad a \rightarrow 1 \quad .$$

Now observe that  $id_1$  and  $\varphi$  are  $Id$ -co-algebras:

$$\begin{aligned} id_1 & : \quad 1 \rightarrow Id\ 1 \\ \varphi & : \quad a \rightarrow Id\ a \quad . \end{aligned}$$

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

In the programming language syntax we may declare:

**datatype** *unit* **has destructors** *iden*: *unit*  $\rightarrow$  *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  $Stream\ a$  consists of all infinite sequences  $[a_0, a_1, \dots]$  with elements from  $a$ , and the head and tail function are defined thus:

$$\begin{aligned} hd & = \quad [a_0, a_1, \dots] \mapsto a_0 & : \quad Stream\ a \rightarrow a \\ tl & = \quad [a_0, a_1, \dots] \mapsto [a_1, \dots] & : \quad Stream\ a \rightarrow Stream\ a \quad . \end{aligned}$$

So, we have

$$\begin{aligned} hd \triangle tl & : \quad Stream\ a \rightarrow a \times Stream\ a \\ & = \quad Stream\ a \rightarrow (\underline{a} \times Id)\ Stream\ a \quad , \end{aligned}$$

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

Next, observe that for arbitrary  $e: b \rightarrow a$  and  $g: b \rightarrow b$  the two equations below fully determine a function  $f: b \rightarrow Stream\ a$ :

$$\begin{aligned} f ; hd & = \quad e & : \quad b \rightarrow a \\ f ; tl & = \quad g ; f & : \quad b \rightarrow Stream\ a \quad . \end{aligned}$$

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

$$\begin{aligned} f ; tl^n ; hd &= g^n ; e \quad \text{or, equivalently —in } \mathcal{Set} \text{—} \\ f(x) &= [e(x), e(g(x)), e(g(g(x))), \dots, e(g^n(x)), \dots] \quad . \end{aligned}$$

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

$$f ; hd \triangle tl = e \triangle g ; id_a \times f \quad .$$

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

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

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

$$\llbracket e \triangle g \rrbracket (x) = [e(x), e(g(x)), e(g(g(x))), \dots, e(g^n(x)), \dots] \quad .$$

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

**datatype** *Stream* *a* **with** *generate* **has destructors**

$$\begin{aligned} hd: \textit{Stream } a &\rightarrow a \\ tl: \textit{Stream } a &\rightarrow \textit{Stream } a \quad . \end{aligned}$$

**7 Iterate.** We continue the preceding discussion of streams.

Let  $f: a \rightarrow a$  be arbitrary, and consider the function  $f$ -iterate, denoted  $f^\omega$ :

$$f^\omega(x) = [x, fx, f^2x, f^3x, \dots] \quad : \quad a \rightarrow \textit{Stream } a \quad .$$

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

$$f^\omega = \llbracket id \triangle f \rrbracket \quad .$$

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

$$\begin{aligned} nats &= zero ; succ^\omega \quad : \quad 1 \rightarrow \textit{Stream } nat \\ ones &= one ; id^\omega \quad : \quad 1 \rightarrow \textit{Stream } nat \quad . \end{aligned}$$

Further, we have the following law:

$$g ; f = f ; h \quad \Rightarrow \quad g^\omega ; \textit{Stream } f = f ; h^\omega \quad . \quad \text{iterate-TRAFO}$$

The proof is easy:

$$\begin{aligned} &g^\omega ; \textit{Stream } f = f ; h^\omega \\ \equiv &\text{definition iterate} \end{aligned}$$

$$\begin{aligned}
& \llbracket id \triangle g \rrbracket ; Stream f = f ; \llbracket id \triangle h \rrbracket \\
\Leftarrow & \quad \text{ana-TRAFO} \\
& id \triangle g ; f \times f = f ; id \triangle h \\
\equiv & \quad \text{product} \\
& g ; f = f ; h \quad .
\end{aligned}$$

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

$$id^\omega ; Stream one = one ; id^\omega \quad (= ones) \quad .$$

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

$$\begin{aligned}
& f^\omega = \llbracket id \triangle f \rrbracket \\
\equiv & \quad \text{ana-CHARN (with functor } \underline{a} \times Id \text{)} \\
& f^\omega ; hd \triangle tl = id \triangle f ; id_a \times f^\omega \\
\equiv & \quad \text{product} \\
& f^\omega ; hd \triangle tl = id \triangle (f ; f^\omega) \\
\equiv & \quad \text{iterate-TRAFO (the condition } f ; f = f ; f \text{ is clearly true)} \\
& f^\omega ; hd \triangle tl = id \triangle (f^\omega ; f) \quad .
\end{aligned}$$

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:

$$\_^\omega : Id \rightarrow Stream \quad \text{in category } Alg(Id) \quad .$$

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

$$\begin{aligned}
& \_^\omega : Id \rightarrow Stream \quad \text{in } Alg(Id) \\
\equiv & \quad \text{definition naturality} \\
& f : g \rightarrow_{Id} h \quad \Rightarrow \quad Id f ; h^\omega = g^\omega ; Stream f \\
\equiv & \quad \text{definition of } \rightarrow_{Id} \text{ and } Id \text{, commutativity of equality} \\
& g ; f = f ; h \quad \Rightarrow \quad g^\omega ; Stream f = h^\omega ; f \quad .
\end{aligned}$$

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

$$[a, \dots], [b, \dots] \xrightarrow{zip} [(a, b), \dots] \xrightarrow{Stream f} [f(a, b), \dots] \quad .$$

Express  $zip$  and  $zipwith-f = zip ; Stream f$  both as a single anamorphism.

**9 Exercise.** Express the categorical product as a final co-algebra. (Hint: prove that  $exl \triangle exr$  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  $hd \triangle tl$  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  $nil \nabla cons$  is an initial  $F$ -algebra, where:

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

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.