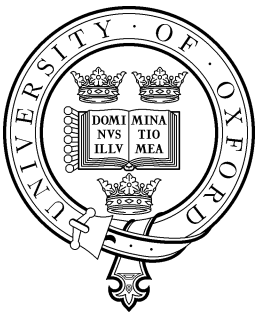
Computing Science Group

Proving The Unique Fixed-Point Principle Correct
An Adventure with Category Theory

Ralf T. W. Hinze, Daniel W. H. James

CS-RR-11-03



Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD

PROVING THE UNIQUE FIXED-POINT PRINCIPLE CORRECT

AN ADVENTURE WITH CATEGORY THEORY

RALF T. W. HINZE AND DANIEL W. H. JAMES

Abstract. Say you want to prove something about an infinite data-structure, such as a stream or an infinite tree, but you would rather not subject yourself to coinduction. The unique fixed-point principle is an easy-to-use, calculational alternative. The proof technique rests on the fact that certain recursion equations have unique solutions; if two elements of a coinductive type satisfy the same equation of this kind, then they are equal. In this paper we precisely characterize the conditions that guarantee a unique solution. Significantly, we do so not with a syntactic criterion, but with a semantic one that stems from the categorical notion of naturality. Our development is based on distributive laws and bialgebras, and draws heavily on Turi and Plotkin's pioneering work on mathematical operational semantics. Along the way, we break down the design space in two dimensions, leading to a total of nine points. Each gives rise to varying degrees of expressiveness, and we will discuss three in depth. Furthermore, our development is generic in the syntax of equations and in the behaviour they encode—we are not caged in the world of streams.

1. Introduction

"Whence cometh this?" Our aim is to provide an elegant proof of correctness for an elegant proof principle. Elegance comes, in large part, through simplicity, and specifically we value the simplicity afforded by the notion of naturality and initial/final algebra/coalgebra semantics. The key component for correctness of the unique fixed-point principle is a sound characterization of what gives a recursion equation a unique solution.

"Why does uniqueness matter?" Uniqueness has two complementary perspectives: programs and proofs. When read as a program, the unique solution implies that it is well-defined. When the unicity is utilized in a proof, we are able to show that two given solutions are equal—the unique fixed-point principle (UFP).

"Why not a syntactic criterion?" In prior work [Hinze, 2010] a simple syntactic criterion, specific to stream equations, was proffered, but this is unsatisfactory. A criterion must exclude all bad things and accept as many good things as possible. As criteria are complicated to accept more good things, so is the understanding and trust. Their syntactic nature makes them intrinsically fragile. Just as it is often easy to satisfy a criterion by a program transformation, it is just as easy to lose the satisfaction.

"Category complex" Our theoretical underpinnings, distributive laws and bialgebras, draw on Turi and Plotkin's mathematical operational semantics [Turi and Plotkin, 1997], and just as in theirs, the following pages contain plenty of category theory. However, our concern throughout is making the theory accessible. We do so by grounding it in an application, and targeting the categorical parlance to readers who are familiar with the Algebra of Programming [Bird and De Moor, 1997]. This is the holy trinity of categories, functors and natural transformations,

2010 *Mathematics Subject Classification.* Primary 68Q55; Secondary 68N18 68Q60.
Key words and phrases. unique fixed-points, bialgebras, distributive laws.

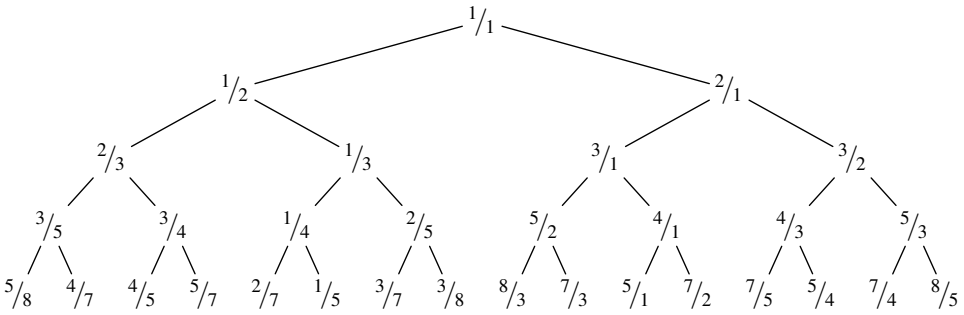


FIGURE 1. The Bird tree

along with algebras and coalgebras, which we will summarize. For the categorically enlightened, we will note the more direct reasoning.

"Interpretation of categorical structures" We will introduce a selection of categorical structures and discuss their interpretation in this domain. It is well known that free monads can been seen as terms with variables and cofree comonads as labelled trees, but we will see how these and two other structures influence the expressivity of recursion equations, by the 'language features' they induce. These features are positioned in two dimensions: one coupled to the expressiveness of terms on the right-hand side of equations, and the other to the expressiveness of patterns on the left-hand side. We will focus on the former initially and then introduce the latter as the dual situation. Notably, we will consider what Niqui and Rutten [2010] calls *sampling* functions, and what we would more loosely describe as stream operators that consume more than they produce.

"Beauty and Elegance" We hope to showcase the beauty of the unique fixed-point principle. Its elegance is due, in no small part, to its calculational style, a style that proofs will follow throughout.

2. The Unique Fixed-Point Principle

2.1. **Infinite Trees.** In Figure 1 we can see the first four levels of the *Bird Tree* [Hinze, 2009], an infinite tree in which you can find every positive rational number exactly once. It has several remarkable properties that come from its nature as a *fractal* object—its subtrees are similar to the whole tree. The tree can be transformed into its left subtree by incrementing and then taking the reciprocal of every element; the right subtree is obtained by swapping these operations. This description can be nicely captured by a *corecursive* definition (we will use Haskell as a meta-language for the category of sets and total functions):

$$bird = Node\ 1\ (1\ /\ (bird + 1))\ ((1\ /\ bird) + 1)\ .$$

The picture suggests that taking the reciprocal of each element is the same as mirroring the tree, $mirror\ bird = 1\ /\ bird$, and this is indeed the case. We shall see that we can prove this effortlessly using the unique fixed-point principle.

Before we get to the proof, we must introduce some definitions.

$$\mathbf{data}\ \mathsf{Tree}\ x = Node\ \{root :: x, left :: \mathsf{Tree}\ x, right :: \mathsf{Tree}\ x\}$$

The type $\mathsf{Tree}\ x$ is a so-called *coinductive datatype*. Its definition is similar to the textbook definition of binary trees, except that there are no leaves, so we cannot build a finite tree. And without leaves, *mirror* is a one-liner:

$$mirror\ (Node\ x\ l\ r) = Node\ x\ (mirror\ r)\ (mirror\ l)\ .$$

The definition of *bird* uses + and / lifted to trees. We obtain these liftings for free as Tree is a so-called *idiom* [McBride and Paterson, 2008]:

$$
\begin{array}{l}
\textbf{class } \mathit{Idiom}\ f \textbf{ where} \\
\quad \mathit{pure} :: x \to f\ x \\
\quad (\diamond) \quad :: f\ (x \to y) \to (f\ x \to f\ y) \\
\textbf{instance } \mathit{Idiom}\ \mathsf{Tree} \textbf{ where} \\
\quad \mathit{pure}\ x = t \textbf{ where } t = \mathit{Node}\ x\ t\ t \\
\quad t \diamond u \quad = \mathit{Node}\ (\mathit{root}\ t\ \$\ \mathit{root}\ u)\ (\mathit{left}\ t \diamond \mathit{left}\ u)\ (\mathit{right}\ t \diamond \mathit{right}\ u)\ .
\end{array}
$$

The call *pure* x constructs an infinite tree of xs; idiomatic application $\diamond$ takes a tree of functions and a tree of arguments to a tree of results. Using *pure* and $\diamond$, we can lift arithmetic operations *generically* to idioms.

$$
\begin{array}{l}
\textbf{instance } (\mathit{Idiom}\ f, \mathit{Num}\ x) \Rightarrow \mathit{Num}\ (f\ x) \textbf{ where} \\
\quad \begin{array}{ll}
\mathit{fromInteger}\ n & = \mathit{pure}\ (\mathit{fromInteger}\ n) \\
\mathit{negate}\ u & = \mathit{pure}\ \mathit{negate} \diamond u \\
u + v & = \mathit{pure}\ (+) \diamond u \diamond v \\
u \times v & = \mathit{pure}\ (\times) \diamond u \diamond v \\
u - v & = \mathit{pure}\ (-) \diamond u \diamond v
\end{array}
\end{array}
$$

Since the operations are defined pointwise, the familiar arithmetic laws also hold for trees. Mirroring a tree preserves the idiomatic structure, the function *mirror* is an *idiom homomorphism*: $\mathit{mirror}\ (\mathit{pure}\ x) = \mathit{pure}\ x$ and $\mathit{mirror}\ (x \diamond y) = \mathit{mirror}\ x \diamond \mathit{mirror}\ y$. This implies, for instance, that $\mathit{mirror}\ (u + v) = \mathit{mirror}\ u + \mathit{mirror}\ v$.

Let us return to the promised proof and the unique fixed-point principle. Consider the recursion equation $x = \mathit{Node}\ y\ l\ r$, where l and r possibly contain the variable x, but *not* the expressions *root* x, *left* x or *right* x. Equations in this syntactic form possess a *unique solution*. Uniqueness can be exploited to prove that two infinite trees are equal: if they satisfy the same equation, then they are.

$$
\begin{array}{ll}
 & \mathit{mirror}\ \mathit{bird} \\
= & \quad \{\ \text{definitions of } \mathit{mirror} \text{ and } \mathit{bird}\ \} \\
 & \mathit{Node}\ 1\ (\mathit{mirror}\ ((1\ /\ \mathit{bird}) + 1))\ (\mathit{mirror}\ (1\ /\ (\mathit{bird} + 1))) \\
= & \quad \{\ \mathit{mirror} \text{ is an idiom homomorphism}\ \} \\
 & \mathit{Node}\ 1\ ((1\ /\ \mathit{mirror}\ \mathit{bird}) + 1)\ (1\ /\ (\mathit{mirror}\ \mathit{bird} + 1)) \\
\propto & \quad \{\ x = \mathit{Node}\ 1\ ((1\ /\ x) + 1)\ (1\ /\ (x + 1)) \text{ has a unique solution}\ \} \\
 & \mathit{Node}\ 1\ ((1\ /\ (1\ /\ \mathit{bird})) + 1)\ (1\ /\ ((1\ /\ \mathit{bird}) + 1)) \\
= & \quad \{\ \text{arithmetic}\ \} \\
 & 1\ /\ \mathit{Node}\ 1\ (1\ /\ (\mathit{bird} + 1))\ ((1\ /\ \mathit{bird}) + 1) \\
= & \quad \{\ \text{definition of } \mathit{bird}\ \} \\
 & 1\ /\ \mathit{bird}
\end{array}
$$

The link $\propto$ indicates that the proof rests on the *unique fixed-point principle*; the recursion equation is given within the curly braces. The upper part shows that *mirror bird* satisfies the equation $x = \mathit{Node}\ 1\ ((1\ /\ x) + 1)\ (1\ /\ (x + 1))$; the lower part establishes that $1\ /\ \mathit{bird}$ satisfies the same equation. The symbol $\propto$ links the two parts, effectively proving the equality of both expressions.

We mentioned that the Bird Tree contains every positive rational exactly once. A proof that exclusively builds on the unique fixed-point principle can be found in Hinze [2009].

2.2. **Streams.** Let us consider a second coinductive type, one that will accompany us for the rest of the paper: the type of streams, infinite sequences of elements.

$$\begin{aligned}&\textbf{data}\ \mathsf{Stream}\ x = \mathit{Cons}\ \{\mathit{head} :: x, \mathit{tail} :: \mathsf{Stream}\ x\}\\&(\prec)\quad :: x \to \mathsf{Stream}\ x \to \mathsf{Stream}\ x\\&x \prec s = \mathit{Cons}\ x\ s\end{aligned}$$

Like the type of infinite trees, Stream is an idiom.

$$\begin{aligned}&\textbf{instance}\ \mathit{Idiom}\ \mathsf{Stream}\ \textbf{where}\\&\quad \mathit{pure}\ x = s\ \textbf{where}\ s = x \prec s\\&\quad s \diamond t\quad = (\mathit{head}\ s\ \$\ \mathit{head}\ t) \prec (\mathit{tail}\ s \diamond \mathit{tail}\ t)\end{aligned}$$

Using this vocabulary, we can define, for instance, the stream of Fibonacci numbers.

$$\begin{aligned}\mathit{fib}\ &= 0 \prec \mathit{fib}'\\\mathit{fib}' &= 1 \prec \mathit{fib} + \mathit{fib}'\end{aligned}$$

The Fibonacci numbers satisfy a myriad of properties. For example, if we form the stream of their partial sums and increment the result, we obtain fib'. Again, we shall see that the UFP allows for a concise proof. But first, we have to capture summation as a stream operator.

$$\Sigma\ s = 0 \prec s + \Sigma\ s$$

Turning to the proof of $\Sigma\ \mathit{fib} + 1 = \mathit{fib}'$, we can either show that $\Sigma\ \mathit{fib} + 1$ satisfies the defining equation of fib', or that $\mathit{fib}' - 1$ satisfies the recursion equation of $\Sigma\ \mathit{fib}$. Both approaches work, here is the calculation for the former.

$$\begin{aligned}&\Sigma\ \mathit{fib} + 1\\=\quad &\{\ \text{definition of } \Sigma\ \}\\&(0 \prec \mathit{fib} + \Sigma\ \mathit{fib}) + 1\\=\quad &\{\ \text{arithmetic}\ \}\\&1 \prec \mathit{fib} + \Sigma\ \mathit{fib} + 1\end{aligned}$$

A related property is the following: if we sum the Fibonacci numbers at odd positions, we obtain the Fibonacci numbers at even positions. The so-called *sampling functions* [Niqui and Rutten, 2010] *even* and *odd* enjoy simple corecursive definitions.

$$\begin{aligned}\mathit{even}\ s &= \mathit{head}\ s \prec \mathit{odd}\quad (\mathit{tail}\ s)\\\mathit{odd}\quad s &= \qquad\qquad\quad \mathit{even}\ (\mathit{tail}\ s)\end{aligned}$$

Turning to the proof of $\Sigma\ (\mathit{odd}\ \mathit{fib}) = \mathit{even}\ \mathit{fib}$, we reason:

$$\begin{aligned}&\Sigma\ (\mathit{odd}\ \mathit{fib})\\=\quad &\{\ \text{definition of } \Sigma\ \}\\&0 \prec \mathit{odd}\ \mathit{fib} + \Sigma\ (\mathit{odd}\ \mathit{fib})\\=\quad &\{\ \text{definition of } \mathit{odd}\ \}\\&0 \prec \mathit{even}\ \mathit{fib}' + \Sigma\ (\mathit{odd}\ \mathit{fib})\\\propto\quad &\{\ x = 0 \prec \mathit{even}\ \mathit{fib}' + x\ \}\\&0 \prec \mathit{even}\ \mathit{fib}' + \mathit{even}\ \mathit{fib}\\=\quad &\{\ \mathit{even} \text{ is an idiom homomorphism and arithmetic}\ \}\\&0 \prec \mathit{even}\ (\mathit{fib} + \mathit{fib}')\\=\quad &\{\ \text{definitions of } \mathit{fib}' \text{ and } \mathit{odd}\ \}\\&0 \prec \mathit{odd}\ \mathit{fib}'\end{aligned}$$

$$= \quad \{ \text{ definitions of } \mathit{fib} \text{ and } \mathit{even} \ \}$$
$$\mathit{even}\ \mathit{fib}$$

This completes our short survey. The UFP is not only easy-to-use, but also surprisingly powerful: in prior work [Hinze, 2010] we have shown how to redevelop the theory of recurrences, finite calculus and generating functions using streams and stream operators, building solely on the cornerstone of unique solutions.

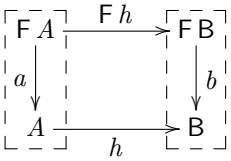
What remains to be done? We have been somewhat vague about the syntactic conditions that guarantee uniqueness. We shall see that systems of recursion equations can be classified along two dimensions, leading to a total of nine different point. The system for *fib* falls into one ("consume one stream element, produce one"), the system for *even* into another ("consume many, but don't nest calls"). When defining streams we cannot mix styles. For instance, the equation $x = 0 \prec \mathit{even}\ x$ has infinitely many solutions. We shall see that we can capture the conditions that guarantee unicity semantically, using the categorical concept of naturality.

Furthermore, we abstract away from the type of infinite trees and streams. The development is generic both in the syntax and in the behaviour—which operations are defined and over which coinductive type. An appropriate setting is provided by the categorical notion of algebras and coalgebras which we introduce next. The resulting proofs are not only more general, they are also shorter than specific instances that have appeared in the literature [Rutten, 2003, Silva and Rutten, 2010].

3. Warm-up

3.1. **Initial Algebras and Final Coalgebras.** We hope the reader has encountered the material of this section before, but we will reiterate it here as it serves as a simple demonstration of the power of *duality*. We will invoke the power to construct 'the opposite thing' time and time again.

Let $\mathsf{F} : \mathscr{C} \to \mathscr{C}$ be an endofunctor. An F-algebra is a pair $\langle A, a\rangle$ consisting of an object $A : \mathscr{C}$ and an arrow $a : \mathsf{F}\,A \to A : \mathscr{C}$. We say that A is the carrier and a is the action of the algebra; however, we often refer to the algebra simply by a as it determines the carrier. An F-homomorphism between algebras $\langle A, a\rangle$ and $\langle \mathsf{B}, b\rangle$ is an arrow $h : A \to \mathsf{B} : \mathscr{C}$ such that $h \cdot a = b \cdot \mathsf{F}\,h$.



A characteristic of functors is that they preserve identity and composition; this entails that F-homomorphisms compose and have an identity. Consequently, F-algebras and F-homomorphisms form a category, called F-**Alg**$(\mathscr{C})$. If this category has an initial object, it is called the *initial* F-*algebra* $\langle \mu\mathsf{F}, \mathit{in}\rangle$. Initiality implies that it has a unique F-homomorphism to any F-algebra $\langle A, a\rangle$, which is written $(\!|a|\!)$ and called *fold*. It satisfies the *uniqueness property*

$$h = (\!|a|\!) \quad \Longleftrightarrow \quad h \cdot \mathit{in} = a \cdot \mathsf{F}\,h \ . \tag{3.1}$$

We will now seize the opportunity to dualize these constructions to the opposite things: F-*coalgebras* and *unfolds*. An F-coalgebra is a pair $\langle C, c\rangle$ of an object $C : \mathscr{C}$ and an arrow $c : C \to \mathsf{F}\,C : \mathscr{C}$. An F-homomorphism between coalgebras $\langle C, c\rangle$ and $\langle D, d\rangle$ is an arrow $h : C \to D : \mathscr{C}$ such that $\mathsf{F}\,h \cdot c = d \cdot h$. In the same way, we can form a category F-**Coalg**$(\mathscr{C})$. If this category has a final object, it is called the *final* F-*coalgebra* $\langle \nu\mathsf{F}, \mathit{out}\rangle$. Being the final object, it has a unique F-homomorphism to

it from any F-coalgebra $\langle C, c\rangle$, which is written $[\![c]\!]$ and called *unfold*. It satisfies the following *uniqueness property*.

$$h = [\![c]\!] \iff \mathsf{F}\, h \cdot c = out \cdot h \tag{3.2}$$

In case you were wondering, *final algebras* and *initial coalgebras* are unexciting, although we will find a use for them. The final algebra is $\langle 1, !_{\mathsf{F}\,1}\rangle$, and the initial coalgebra is $\langle 0, \mathsf{i}_{\mathsf{F}\,0}\rangle$.

$$\begin{array}{ccc} \mathsf{F}\,A & \xrightarrow{\mathsf{F}\,!_A} & \mathsf{F}\,1 \\ {\scriptstyle a}\downarrow & & \downarrow{\scriptstyle !_{\mathsf{F}\,1}} \\ A & \xrightarrow[!_A]{} & 1 \end{array} \qquad \begin{array}{ccc} 0 & \xrightarrow{\mathsf{i}_C} & C \\ {\scriptstyle \mathsf{i}_{\mathsf{F}\,0}}\downarrow & & \downarrow{\scriptstyle c} \\ \mathsf{F}\,0 & \xrightarrow[\mathsf{F}\,\mathsf{i}_C]{} & \mathsf{F}\,C \end{array}$$

The diagrams commute as there is only one arrow from $\mathsf{F}\,A$ to 1 and only one arrow from 0 to $\mathsf{F}\,C$.

3.2. **Initial into Final.** The carriers of the initial algebra and final coalgebra are usually distinct objects (cf. algebraically compact categories [Freyd, 1992]), however, the former can be embedded into the latter.

$$\mu\mathsf{F} \to \nu\mathsf{F} \tag{3.3}$$

There are two options for defining this arrow. The embedding goes from $\mu\mathsf{F}$ so we could express it as a fold; this dictates that we need an algebra with type $\mathsf{F}\,(\nu\mathsf{F}) \to \nu\mathsf{F}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{out : \nu\mathsf{F} \to \mathsf{F}\,(\nu\mathsf{F})}(final)}{\mathsf{F}\,out : \mathsf{F}\,(\nu\mathsf{F}) \to \mathsf{F}\,(\mathsf{F}\,(\nu\mathsf{F}))}(functor)}{[\![\mathsf{F}\,out]\!] : \mathsf{F}\,(\nu\mathsf{F}) \to \nu\mathsf{F}}(unfold)}{(\!|[\![\mathsf{F}\,out]\!]|\!) : \mu\mathsf{F} \to \nu\mathsf{F}}(fold)$$

The embedding also goes to $\nu\mathsf{F}$ so we could express it as an unfold; this dictates that we need a coalgebra with type $\mu\mathsf{F} \to \mathsf{F}\,(\mu\mathsf{F})$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{in : \mathsf{F}\,(\mu\mathsf{F}) \to \mu\mathsf{F}}(initial)}{\mathsf{F}\,in : \mathsf{F}\,(\mathsf{F}\,(\mu\mathsf{F})) \to \mathsf{F}\,(\mu\mathsf{F})}(functor)}{(\!|\mathsf{F}\,in|\!) : \mu\mathsf{F} \to \mathsf{F}\,(\mu\mathsf{F})}(fold)}{[\![(\!|\mathsf{F}\,in|\!)]\!] : \mu\mathsf{F} \to \nu\mathsf{F}}(unfold)$$

A key observation is that in and $(\!|\mathsf{F}\,in|\!)$ form an isomorphism.

$$\begin{array}{ll} & in \cdot (\!|\mathsf{F}\,in|\!) = id_{\mu\mathsf{F}} \\ \iff & \{\ \text{fold reflection (C.1)}\ \} \\ & in \cdot (\!|\mathsf{F}\,in|\!) = (\!|in|\!) \\ \Longleftarrow & \{\ \text{fold fusion (C.3)}\ \} \\ & in \cdot \mathsf{F}\,in = in \cdot \mathsf{F}\,in \end{array} \qquad \begin{array}{ll} & (\!|\mathsf{F}\,in|\!) \cdot in = id_{\mathsf{F}\,(\mu\mathsf{F})} \\ \iff & \{\ \text{fold computation (C.2)}\ \} \\ & \mathsf{F}\,in \cdot \mathsf{F}\,(\!|\mathsf{F}\,in|\!) = id \\ \iff & \{\ \mathsf{F}\ \text{functor and other proof}\ \} \\ & \mathsf{F}\,id = id \end{array}$$

And dually, out and $[\![\mathsf{F}\,out]\!]$ also form an isomorphism.

$$\begin{array}{ll} & [\![\mathsf{F}\,out]\!] \cdot out = id_{\nu\mathsf{F}} \\ \iff & \{\ \text{unfold reflection (D.1)}\ \} \\ & [\![\mathsf{F}\,out]\!] \cdot out = [\![out]\!] \\ \Longleftarrow & \{\ \text{unfold fusion (D.3)}\ \} \\ & \mathsf{F}\,out \cdot out = \mathsf{F}\,out \cdot out \end{array} \qquad \begin{array}{ll} & out \cdot [\![\mathsf{F}\,out]\!] = id_{\mathsf{F}\,(\nu\mathsf{F})} \\ \iff & \{\ \text{unfold computation (D.2)}\ \} \\ & \mathsf{F}\,[\![\mathsf{F}\,out]\!] \cdot \mathsf{F}\,out = id \\ \iff & \{\ \mathsf{F}\ \text{functor and other proof}\ \} \\ & \mathsf{F}\,id = id \end{array}$$

Using these isomorphisms, along with the uniqueness of fold and unfold, we can give a calculation proof that the embedding is unique.

$$
\begin{aligned}
& [\!(\mathsf{F}\ in)\!] = (\!|[\mathsf{F}\ out]|\!) \\
\iff & \quad \{\ \text{uniqueness of fold (3.1)}\ \} \\
& [\!(\mathsf{F}\ in)\!] \cdot in = [\mathsf{F}\ out] \cdot \mathsf{F}\ [\!(\mathsf{F}\ in)\!] \\
\iff & \quad \{\ out\ \text{isomorphism}\ \} \\
& out \cdot [\!(\mathsf{F}\ in)\!] \cdot in = \mathsf{F}\ [\!(\mathsf{F}\ in)\!] \\
\iff & \quad \{\ \text{unfold computation law (D.2)}\ \} \\
& \mathsf{F}\ [\!(\mathsf{F}\ in)\!] \cdot (\mathsf{F}\ in) \cdot in = \mathsf{F}\ [\!(\mathsf{F}\ in)\!] \\
\iff & \quad \{\ in\ \text{isomorphism}\ \} \\
& \mathsf{F}\ [\!(\mathsf{F}\ in)\!] = \mathsf{F}\ [\!(\mathsf{F}\ in)\!]
\end{aligned}
$$

There is, of course, a dual proof, which rests on the uniqueness of unfold. The two forms of the unique arrow can be seen in the following diagram:

$$
\begin{array}{ccc}
\mathsf{F}\,(\mu\mathsf{F}) & \dashrightarrow & \mathsf{F}\,(\nu\mathsf{F}) \\
{\scriptstyle in}\downarrow & (\!|[\mathsf{F}\ out]|\!) & \downarrow{\scriptstyle [\mathsf{F}\ out]} \\
\mu\mathsf{F} & \dashrightarrow & \nu\mathsf{F}
\end{array}
\qquad
\begin{array}{ccc}
\mu\mathsf{F} & \dashrightarrow & \nu\mathsf{F} \\
{\scriptstyle (\!|\mathsf{F}\ in|\!)}\downarrow & [\!(\mathsf{F}\ in)\!] & \downarrow{\scriptstyle out} \\
\mathsf{F}\,(\mu\mathsf{F}) & \dashrightarrow & \mathsf{F}\,(\nu\mathsf{F})
\end{array}
$$

3.3. **Natural Transformations of Algebras and Coalgebras.** Let $\mathsf{F}, \mathsf{G} : \mathscr{C} \to \mathscr{C}$ be endofunctors, and $\alpha : \mathsf{F} \leftarrow \mathsf{G}$ a natural transformation. We can turn α into a functor α-**Alg** : F-**Alg** $\to$ G-**Alg** between the categories of F- and G-algebras.

$$
\begin{aligned}
\alpha\text{-}\mathbf{Alg}\ \langle X, a : \mathsf{F}\ X \to X\rangle &= \langle X, a \cdot \alpha\ X : \mathsf{G}\ X \to X\rangle \\
\alpha\text{-}\mathbf{Alg}\ h &= h
\end{aligned}
$$

We must show that homomorphisms are preserved by α-**Alg**. For algebras $a : \mathsf{F}\ X \to X$, $b : \mathsf{F}\ Y \to Y$, and $h : X \to Y$,

$$
h \cdot \alpha\text{-}\mathbf{Alg}\ a = \alpha\text{-}\mathbf{Alg}\ b \cdot \mathsf{G}\ h \quad \Longleftarrow \quad h \cdot a = b \cdot \mathsf{F}\ h \ .
$$

Proof.

$$
\begin{aligned}
& h \cdot \alpha\text{-}\mathbf{Alg}\ a \\
= & \quad \{\ \text{definition of}\ \alpha\text{-}\mathbf{Alg}\ \} \\
& h \cdot a \cdot \alpha\ X \\
= & \quad \{\ \text{assumption}\ h \cdot a = b \cdot \mathsf{F}\ h\ \} \\
& b \cdot \mathsf{F}\ h \cdot \alpha\ X \\
= & \quad \{\ \alpha : \mathsf{F} \leftarrow \mathsf{G}\ \text{is natural}\ \} \\
& b \cdot \alpha\ Y \cdot \mathsf{G}\ h \\
= & \quad \{\ \text{definition of}\ \alpha\text{-}\mathbf{Alg}\ \} \\
& \alpha\text{-}\mathbf{Alg}\ b \cdot \mathsf{G}\ h \quad \square
\end{aligned}
$$

We will see various instantiations of α-**Alg** later on, where its functor properties will come in handy.

Dually, let $\alpha : \mathsf{F} \dot{\to} \mathsf{G}$, then α-**Coalg** : F-**Coalg** $\to$ G-**Coalg**.

$$\begin{aligned}\alpha\text{-}\mathbf{Coalg}\,\langle X, c : X \to \mathsf{F}\,X\rangle &= \langle X, \alpha\,X \cdot c : X \to \mathsf{G}\,X\rangle\\ \alpha\text{-}\mathbf{Coalg}\,h &= h\end{aligned}$$

4. Meet Iniga and Finn

Once upon a time a teacher had a pair of bright and capable students, who, for better or worse, were hooked on category theory. The first, Iniga, was a go-getting student with plenty of initiative. Interestingly, this was in stark contrast to Finn, a reserved character who perceived the world with a sense of finality.

The teacher posed them the problem of showing that stream equations possess unique solutions. Owing to their polar opposite outlooks, Iniga and Finn took divergent approaches to tackling the problem, but as we will discover, their approaches turned out to be two sides of the same coin.

The teacher started with a minimalistic example, asking them to consider the following stream equations.

$$\begin{aligned}\mathit{one} &= 1 \prec \mathit{one}\\ \mathit{plus}\ (\mathit{Cons}\ m\ s, \mathit{Cons}\ n\ t) &= m + n \prec \mathit{plus}\ (s, t)\end{aligned}$$

Streams of natural numbers are the resultant behaviour of these equations, so the teacher provided the functor $\mathsf{B}\,X = \mathbb{N} \times X$ as the *behaviour functor*. We can give this a Haskell rendering:

$$\mathbf{data}\ \mathsf{B}\ x = \mathit{Cons}\ (\mathbb{N}, x)\ .$$

For simplicity, the teacher fixed the element type of streams. An element of $\nu\mathsf{B}$, the carrier of the final coalgebra of the behaviour functor, is a stream of natural numbers: $\nu\mathsf{B} \cong \mathsf{Stream}\,\mathbb{N}$.

The stream constant *one* and the stream operator *plus* in the example stream equations are also modelled categorically with the functor $\mathsf{S}\,X = 1 + X \times X$ as the *syntax functor*.

$$\mathbf{data}\ \mathsf{S}\ x = \mathit{One} \mid \mathit{Plus}\ (x, x)$$

An element of $\mu\mathsf{S}$, the initial algebra carrier of the syntax functor, is a finite, closed term, built from the syntax constructors of S.

Iniga (taking the initiative): Ok, given these definitions we can model the stream equations by a simple function.

$$\begin{aligned}\lambda\ (\mathit{One}) &= \mathit{Cons}\ (1, \mathit{One})\\ \lambda\ (\mathit{Plus}\ (\mathit{Cons}\ (m, s), \mathit{Cons}\ (n, t))) &= \mathit{Cons}\ (m + n, \mathit{Plus}\ (s, t))\end{aligned}$$

Teacher: Observe that λ is really a natural transformation of type $\mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}$. This is crucial: the syntactic requirements on stream equations to ensure uniqueness of solutions are captured by the naturality requirement on λ. Its type can be seen as a promise that only the head of the incoming stream will be inspected and that an element of the outgoing stream will be constructed. Can you see how the slogan "consume one, produce one" translates?

Iniga: Yes! An interpretation of the syntax is then given by an S-algebra $a : \mathsf{S}\,(\nu\mathsf{B}) \to \nu\mathsf{B}$ whose carrier is the final B-coalgebra $\langle \nu\mathsf{B}, \mathit{out}\rangle$. The algebra a takes a level of syntax over a stream and turns it into a stream.

Teacher: How do we model that a respects the stream equations captured by λ? Your algebra a has to satisfy the following law:

$$\mathit{out} \cdot a\ =\ \mathsf{B}\,a \cdot \lambda\,(\nu\mathsf{B}) \cdot \mathsf{S}\,\mathit{out}\ :\ \mathsf{S}\,(\nu\mathsf{B}) \to \mathsf{B}\,(\nu\mathsf{B})\ . \tag{4.1}$$

The law states that unrolling the result of a is the same as unrolling the arguments of the syntax, $\mathsf{S}\,\mathit{out}$, applying the stream equations $\lambda\,(\nu\mathsf{B})$, and then interpreting the tail, $\mathsf{B}\,a$.

Iniga: Great, for our example I will rearrange the law to observe the Haskell convention of definition by pattern matching, $a \cdot \mathsf{S}\, out^\circ = out^\circ \cdot \mathsf{B}\, a \cdot \lambda\,(\nu\mathsf{B})$. If I instantiate this law to our running example, I obtain a definition of the algebra a:

$$\begin{aligned} & a\ One \\ & \quad = Out^\circ\ (Cons\ (1, a\ One)) \\ & a\ (Plus\ (Out^\circ\ (Cons\ (m, s)), Out^\circ\ (Cons\ (n, t)))) \\ & \quad = Out^\circ\ (Cons\ (m + n, a\ (Plus\ (s, t)))) \ . \end{aligned}$$

With a, I can now define the semantic counterparts of *One* and *Plus*, the stream constant $\underline{one}$ and the stream operator $\underline{plus}$, underlined to emphasize that they are semantic entities:

$$\begin{aligned} \underline{one} &= a\ One \\ \underline{plus}\ (s, t) &= a\ (Plus\ (s, t)) \ . \end{aligned}$$

Teacher (interrupting): You are actually building upon the isomorphism $\mathsf{S}\, X \to X \cong (1 \to X) \times (X \times X \to X)$ here: the pair of functions, $\underline{one}$ and $\underline{plus}$, is just another way of writing the algebra a.

Iniga: Using $Cons\ a\ s$ and $a \prec s$ as shorthands for $Out^\circ\ (Cons\ (a, s))$, the definition of a is the same as,

$$\begin{aligned} \underline{one} &= 1 \prec \underline{one} \\ \underline{plus}\ (Cons\ m\ s, Cons\ n\ t) &= m + n \prec \underline{plus}\ (s, t) \ , \end{aligned}$$

that is $\underline{one}$ and $\underline{plus}$ satisfy the original stream equations. Again, the notation makes clear that we have to read the stream operators semantically—$\underline{one}$ and $\underline{plus}$ are the entities defined by the system.

Teacher: We can wrap this up by showing that the law (4.1) *uniquely* determines a:

$$\begin{aligned} & out \cdot a = \mathsf{B}\, a \cdot \lambda\,(\nu\mathsf{B}) \cdot \mathsf{S}\, out \\ \iff & \quad \{ \text{ uniqueness of unfold (3.2) } \} \\ & a = [\![\lambda\,(\nu\mathsf{B}) \cdot \mathsf{S}\, out]\!] \end{aligned}$$

So $[\![\lambda\,(\nu\mathsf{B}) \cdot \mathsf{S}\, out]\!]$ is the unique solution of the stream equations. Furthermore, the fold $(\!|a|\!) : \mu\mathsf{S} \to \nu\mathsf{B}$ takes syntax to behaviour by evaluating a term. Finn, what are your thoughts?

Finn: To start with, I would write the stream equations differently. I find them too Haskell-like, and I prefer what Jan Rutten calls "behavioural differential equations" [Rutten, 2003].

$$\begin{aligned} & head\ one = 1 \\ & tail\ \ one = one \\ & head\ (plus\ (s, t)) = head\ s + head\ t \\ & tail\ \ (plus\ (s, t)) = plus\ (tail\ s, tail\ t) \end{aligned}$$

A semantics is given by a B-coalgebra $c : \mu\mathsf{S} \to \mathsf{B}\,(\mu\mathsf{S})$ whose carrier is the initial S-algebra $\langle \mu\mathsf{S}, in \rangle$. The coalgebra c takes a term and produces the first number of the defined stream, and a term to generate the rest of the stream.

Teacher: Just as for Iniga, your coalgebra c has to satisfy the following law:

$$c \cdot in = \mathsf{B}\, in \cdot \lambda\,(\mu\mathsf{S}) \cdot \mathsf{S}\, c : \mathsf{S}\,(\mu\mathsf{S}) \to \mathsf{B}\,(\mu\mathsf{S}) \ . \tag{4.2}$$

The law states that building a term and applying c is the same as giving a semantics to the subterms, $\mathsf{S}\, c$, applying the stream equations $\lambda\,(\mu\mathsf{S})$, and building a term in the tail of a stream, $\mathsf{B}\, in$.

Finn: I will follow Iniga's lead and specialize the law to our example, obtaining a definition of c:

$$\begin{aligned} & c\ (In\ One) \\ & \quad = Cons\ (1, In\ One) \\ & c\ (In\ (Plus\ (s,t))) \\ & \quad = Cons\ (head\ (c\ s) + head\ (c\ t), In\ (Plus\ (tail\ (c\ s), tail\ (c\ t))))\ , \end{aligned}$$

where $head\,(Cons\,(a,s)) = a$ and $tail\,(Cons\,(a,s)) = s$. Given a *stream program*, my c gives the head of the stream and a stream program for the tail of the stream. I can now define the semantic counterparts of *head* and *tail*:

$$\begin{aligned} \underline{head}\ s &= head\ (c\ s) \\ \underline{tail}\ \ s &= tail\ \ (c\ s)\ . \end{aligned}$$

Teacher (interrupting): You are building upon the isomorphism $X \to \mathsf{B}\,X \cong (X \to \mathbb{N}) \times (X \to X)$ here: $\underline{head}$ and $\underline{tail}$ is just another way of writing c.

Finn: Using *one* as a shorthand for *In One* and *plus* (s,t) for $In\,(Plus\,(s,t))$, the definition of c is the same as,

$$\begin{aligned} & \underline{head}\ one = 1 \\ & \underline{tail}\ \ one = one \\ & \underline{head}\ (plus\ (s,t)) = \underline{head}\ s + \underline{head}\ t \\ & \underline{tail}\ \ (plus\ (s,t)) = plus\ (\underline{tail}\ s, \underline{tail}\ t)\ , \end{aligned}$$

that is, $\underline{head}$ and $\underline{tail}$ satisfy the original stream equations. The notation emphasizes that we have to read the stream selectors semantically—$\underline{head}$ and $\underline{tail}$ are the entities defined by the system.

Teacher: Again, we can show that the law (4.2) determines c:

$$\begin{aligned} & c \cdot in = \mathsf{B}\,in \cdot \lambda\,(\mu\mathsf{S}) \cdot \mathsf{S}\,c \\ \iff & \quad \{\ \text{uniqueness of fold (3.1)}\ \} \\ & c = (\!|\mathsf{B}\,in \cdot \lambda\,(\mu\mathsf{S})|\!) \end{aligned}$$

So $(\!|\mathsf{B}\,in \cdot \lambda\,(\mu\mathsf{S})|\!)$ is the *unique* solution of your stream equations. And the unfold $[\![c]\!] : \mu\mathsf{S} \to \nu\mathsf{B}$ takes syntax to behaviour by unrolling a complete stream.

Iniga and Finn, you should reconcile your two viewpoints. Your semantic functions are of type $\mu\mathsf{S} \to \nu\mathsf{B}$, so is the fold of Iniga's algebra the same as the unfold of Finn's coalgebra: $(\!|a|\!) = [\![c]\!]$? Did you notice that we made use of the naturality of λ: Iniga used λ at type $\nu\mathsf{B}$, while Finn required the $\mu\mathsf{S}$ instance? Of course, we have only discussed a minimalistic example here, and we are not immediately able to model stream equations such as the ones that define the Fibonacci stream.

It's also important to note that your approaches are not perfectly symmetric: Iniga can define $\underline{one} : \nu\mathsf{B}$ and $\underline{plus} : \nu\mathsf{B} \times \nu\mathsf{B} \to \nu\mathsf{B}$, Finn can define $\underline{head} : \mu\mathsf{S} \to \mathbb{N}$ and $\underline{tail} : \mu\mathsf{S} \to \mu\mathsf{S}$, but not the other way round.

Finn: Hmm, to define $\underline{plus}$, which takes two streams as an argument, I would need a means to embed streams into terms.

$$\underline{plus}\ (s,t) = evaluate\ (Plus\ (?s, ?t))$$

I have used '?' as a placeholder for this shortcoming.

Teacher: We shall see that the *free monad* is the right structure.

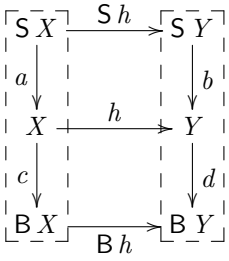
Iniga: Also, what about device to label streams with terms?

Teacher: We shall see that the cofree comonad is the right structure.

Epilogue. Now that we have met Iniga and Finn and got a taste for the problem that their teacher posed to them, we will move on to introduce the infrastructure that is needed for the reconciliation.

5. Bialgebras

Let $\mathsf{S}, \mathsf{B} : \mathscr{C} \to \mathscr{C}$ be functors. A bialgebra is a triple $\langle X, a, c\rangle$ consisting of an object $X : \mathscr{C}$, an arrow $a : \mathsf{S}\,X \to X : \mathscr{C}$, and an arrow $c : X \to \mathsf{B}\,X : \mathscr{C}$. It is an S-algebra and a B-coalgebra with a common carrier. Let $\langle X, a, c\rangle$ and $\langle Y, b, d\rangle$ be bialgebras and $h : X \to Y : \mathscr{C}$ an arrow. Then h is a bialgebra homomorphism if it is both an S-algebra homomorphism and a B-coalgebra homomorphism.



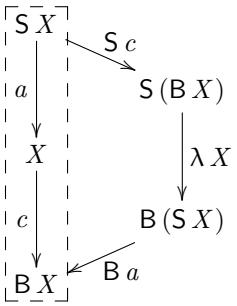
Identity is a bialgebra homomorphism and homomorphisms compose. Consequently, bialgebras and their homomorphisms form a category, called $\mathbf{Bialg}(\mathscr{C})$.

We are concerned with λ-bialgebras, which are bialgebras equipped with a so-called *distributive law* $\lambda : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}$. This extra structure imposes a coherence condition on bialgebras.

$$c \cdot a = \mathsf{B}\,a \cdot \lambda\,X \cdot \mathsf{S}\,c \tag{5.1}$$

The condition is also called the *pentagonal law*.

(5.2)



The category of bialgebras that satisfy the pentagonal law (5.2) is denoted λ-$\mathbf{Bialg}(\mathscr{C})$. It is a full subcategory of $\mathbf{Bialg}(\mathscr{C})$.

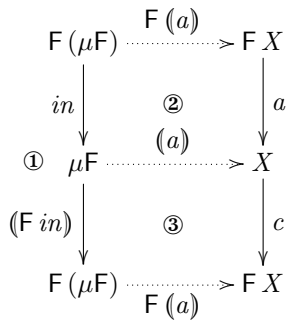
6. Reheat

Let us revisit our warm-up example of the embedding $\mu\mathsf{F} \to \nu\mathsf{F}$, but now we will use λ-bialgebras. While this usage will be a degenerative case of the theory, it will serve as an introduction to a recurring pattern that will soon emerge.

Instead of two functors, we have just one, F. Therefore, our natural transformation must have the type $\lambda : \mathsf{F}\circ\mathsf{F} \dot{\to} \mathsf{F}\circ\mathsf{F}$. The identity $id_{\mathsf{F}\circ\mathsf{F}}$ will do nicely.

The algebra $\langle \mu\mathsf{F}, in\rangle$ is the initial F-algebra—the initial object in F-$\mathbf{Alg}$. We will now show that from this we can form the initial object in $id_{\mathsf{F}\circ\mathsf{F}}$-$\mathbf{Bialg}$. We showed earlier that in and $(\!|\mathsf{F}\ in|\!)$ form an isomorphism. Therefore, we will choose $\langle \mu\mathsf{F}, in, (\!|\mathsf{F}\ in|\!)\rangle$ as the initial bialgebra. This situation is illustrated in the following

diagram.

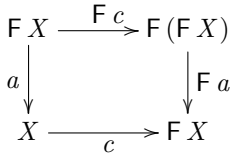


A number of proof obligations now arise.

First, we must show that $\langle \mu\mathsf{F}, in, (\!|\mathsf{F}\ in|\!)\rangle$ is a λ-bialgebra (①)—it has the right types, but is must also satisfy (5.1). As λ is the identity, the coherence condition simplifies to $(\!|\mathsf{F}\ in|\!) \cdot in = \mathsf{F}\ in \cdot \mathsf{F}\ (\!|\mathsf{F}\ in|\!)$. This is exactly the computation law of fold (C.2).

Second, the fact that $(\!|a|\!)$ is an algebra homomorphism is by construction—the top half of the diagram commutes (②). Moreover, the uniqueness of this arrow also comes for free.

All that remains is to prove that $(\!|a|\!)$ is also a coalgebra homomorphism—that the bottom half of the diagram commutes (③): $c \cdot (\!|a|\!) = \mathsf{F}\ (\!|a|\!) \cdot (\!|\mathsf{F}\ in|\!)$. Before we proceed to the proof, let us revisit the pentagonal law. As λ is the identity, we can simplify and rearrange the diagram to show that c is a not only a coalgebra, but also an algebra homomorphism: $c : a \to \mathsf{F}\ a : \mathsf{F}\text{-}\mathbf{Alg}$.



As a result, we can directly invoke the fusion law of fold (C.3) to state that $c \cdot (\!|a|\!) = (\!|\mathsf{F}\ a|\!)$. Now we can proceed.

$$
\begin{array}{ll}
 & c \cdot (\!|a|\!) = \mathsf{F}\ (\!|a|\!) \cdot (\!|\mathsf{F}\ in|\!) \\
\iff & \{\ \text{fold fusion (C.3) with } c : a \to \mathsf{F}\ a : \mathsf{F}\text{-}\mathbf{Alg}\ \} \\
 & (\!|\mathsf{F}\ a|\!) = \mathsf{F}\ (\!|a|\!) \cdot (\!|\mathsf{F}\ in|\!) \\
\impliedby & \{\ \text{fold fusion (C.3)}\ \} \\
 & \mathsf{F}\ (\!|a|\!) : \mathsf{F}\ in \to \mathsf{F}\ a : \mathsf{F}\text{-}\mathbf{Alg} \\
\iff & \{\ \mathsf{F}\ \text{functor}\ \} \\
 & (\!|a|\!) : in \to a : \mathsf{F}\text{-}\mathbf{Alg}
\end{array}
$$

Having discharged the necessary proof obligations, we can now state that $\langle \mu\mathsf{F}, in, (\!|\mathsf{F}\ in|\!)\rangle$ is indeed the initial λ-bialgebra. Furthermore, $(\!|a|\!)$ is the unique homomorphism from the initial λ-bialgebra to any λ-bialgebra $\langle X, a, c\rangle$.

We can dualize these results to state that: $\langle \nu\mathsf{F}, [\![\mathsf{F}\ out]\!], out\rangle$ is the final λ-bialgebra; and $[\![c]\!]$ is the unique homomorphism to any λ-bialgebra $\langle X, a, c\rangle$.

$$\begin{array}{ccc}
\mathsf{F}\,X & \xrightarrow{\mathsf{F}\,[\![c]\!]} & \mathsf{F}\,(\nu\mathsf{F}) \\
{\scriptstyle a}\downarrow & & \downarrow{\scriptstyle [\![\mathsf{F}\ out]\!]} \\
X & \xrightarrow[[\![c]\!]]{} & \nu\mathsf{F} \\
{\scriptstyle c}\downarrow & & \downarrow{\scriptstyle out} \\
\mathsf{F}\,X & \xrightarrow[\mathsf{F}\,[\![c]\!]]{} & \mathsf{F}\,(\nu\mathsf{F})
\end{array}$$

Let us return to the embedding of $\mu\mathsf{F}$ in $\nu\mathsf{F}$. The following diagram depicts the homomorphism between initial and final λ-bialgebras. The embedding arrow is unique, and we can give two justifications for it being so: namely that it is the unique homomorphism both from the initial λ-bialgebra and to the final λ-bialgebra. For the same two justifications, we can give two definitions of this arrow, and by uniqueness they are equal.

$$\begin{array}{ccc}
\mathsf{F}\,(\mu\mathsf{F}) & \xrightarrow{\qquad} & \mathsf{F}\,(\nu\mathsf{F}) \\
{\scriptstyle in}\downarrow & & \downarrow{\scriptstyle [\![\mathsf{F}\ out]\!]} \\
\mu\mathsf{F} & \xrightarrow[[\![(\mathsf{F}\ in)]\!]]{(\!|[\![\mathsf{F}\ out]\!]|\!)} & \nu\mathsf{F} \\
{\scriptstyle (\!|\mathsf{F}\ in|\!)}\downarrow & & \downarrow{\scriptstyle out} \\
\mathsf{F}\,(\mu\mathsf{F}) & \xrightarrow{\qquad} & \mathsf{F}\,(\nu\mathsf{F})
\end{array}$$

Summary. We have implicitly set out a template here that will be used again and again. In this case the two functors of the bialgebras were the same and the λ was trivial, but we will gradually complicate matters by using distinct functors and then adding more structure. This will be in the form of natural transformations and the consequent coherence conditions.

7. Endofunctors over Endofunctors

We will now use λ-bialgebras to explicate Iniga and Finn's conversation with their teacher, and begin to reconcile their solutions.

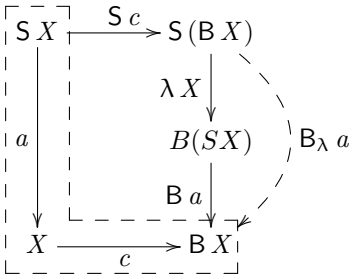
Let $\mathsf{S}, \mathsf{B} : \mathscr{C} \to \mathscr{C}$ be functors, and $\lambda : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}$ be a natural transformation. We will read these to imply syntax and behaviour functors, and a distributive law modelling a set of equations. Using λ-bialgebras, we will characterize the semantic function from syntax to behaviour—the arrow from the least fixed-point of S to the greatest fixed-point of B.

Before, we extended the initial object in F-**Alg** into the initial object in $id_{\mathsf{F}\circ\mathsf{F}}$-**Bialg**, with the coalgebra $(\!|\mathsf{F}\ in|\!)$. Our new scenario is no longer so simple. If the carrier of the initial λ-bialgebra has been fixed as $\mu\mathsf{S}$, then the coalgebra will have type $\mu\mathsf{S} \to \mathsf{B}\,(\mu\mathsf{S})$. This is exactly Finn's coalgebra, and his teacher has derived it: $(\!|\mathsf{B}\ in \cdot \lambda\,(\mu\mathsf{S})|\!)$. As one might guess, the laws the teacher provided came from λ-bialgebras. Let us take a step back to re-examine λ and the pentagonal law.

7.1. **Lifting Endofunctors to Algebras.**

The pentagonal law confers a useful property both on the algebra and the coalgebra component of a λ-bialgebra. Let

us illustrate this first for the coalgebra component by redrawing diagram (5.2).



Here we can see that c is not only a B-coalgebra, but also an S-algebra homomorphism from $\langle X, a\rangle$ to $\langle \mathsf{B}\,X, \mathsf{B}\,a \cdot \lambda\,X\rangle$.

We can characterize this situation as lifting the endofunctor $\mathsf{B} : \mathscr{C} \to \mathscr{C}$ to a functor on S-algebras; we will give it the name $\mathsf{B}_\lambda : \mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C}) \to \mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C})$, and define it as,

$$\mathsf{B}_\lambda\,\langle X, a : \mathsf{S}\,X \to X\rangle = \langle \mathsf{B}\,X, \mathsf{B}\,a \cdot \lambda\,X : \mathsf{S}\,(\mathsf{B}\,X) \to \mathsf{B}\,X\rangle \ , \tag{7.1}$$

$$\mathsf{B}_\lambda\,h = \mathsf{B}\,h \ . \tag{7.2}$$

We shall usually omit the object part of the algebra and apply B_λ only to the arrow part. We need to show that B_λ is functorial. As B is a functor and $\mathsf{B}_\lambda\,h = \mathsf{B}\,h$, it preserves composition and the identity. Additionally we must show that it preserves S-algebra homomorphisms.

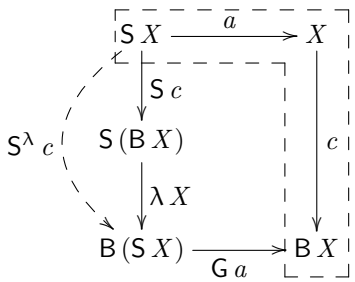
$$\mathsf{B}\,h : \mathsf{B}_\lambda\,a \to \mathsf{B}_\lambda\,b : \mathsf{S}\text{-}\mathbf{Alg} \iff h : a \to b : \mathsf{S}\text{-}\mathbf{Alg} \tag{7.3}$$

Proof.

$$\begin{aligned}
& \mathsf{B}\,h \cdot \mathsf{B}_\lambda\,a \\
= & \quad \{ \text{ definition of } \mathsf{B}_\lambda \ \} \\
& \mathsf{B}\,h \cdot \mathsf{B}\,a \cdot \lambda\,X \\
= & \quad \{ \ \mathsf{B} \text{ functor and assumption } h : a \to b : \mathsf{S}\text{-}\mathbf{Alg} \ \} \\
& \mathsf{B}\,b \cdot \mathsf{B}\,(\mathsf{S}\,h) \cdot \lambda\,X \\
= & \quad \{ \ \lambda : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S} \text{ is natural } \} \\
& \mathsf{B}\,b \cdot \lambda\,X \cdot \mathsf{S}\,(\mathsf{B}\,h) \\
= & \quad \{ \text{ definition of } \mathsf{B}_\lambda \ \} \\
& \mathsf{B}_\lambda\,b \cdot \mathsf{S}\,(\mathsf{B}\,h) \quad \square
\end{aligned}$$

Therefore, we can give c, viewed as an algebra homomorphism, the more succinct type $c : a \to \mathsf{B}_\lambda\,a : \mathsf{S}\text{-}\mathbf{Alg}$.

Dually, a is both an S-algebra and a B-coalgebra homomorphism,



with the type $a : \mathsf{S}^\lambda\ c \to c : \mathsf{B}\text{-}\mathbf{Coalg}$, where the lifted functor $\mathsf{S}^\lambda : \mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C}) \to \mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C})$ is defined as,

$$\mathsf{S}^\lambda\ \langle X, c : X \to \mathsf{B}\ X\rangle = \langle \mathsf{S}\ X, \lambda\ X \cdot \mathsf{S}\ c : \mathsf{S}\ X \to \mathsf{B}\ (\mathsf{S}\ X)\rangle \ , \tag{7.4}$$

$$\mathsf{S}^\lambda\ h = \mathsf{S}\ h \ . \tag{7.5}$$

Again, S^λ preserves composition and the identity as $\mathsf{S}^\lambda\ h = \mathsf{S}\ h$. It remains to show that S^λ preserves coalgebra homomorphisms.

$$\mathsf{S}\ h : \mathsf{S}^\lambda\ c \to \mathsf{S}^\lambda\ d : \mathsf{B}\text{-}\mathbf{Coalg} \iff h : c \to d : \mathsf{B}\text{-}\mathbf{Coalg} \tag{7.6}$$

The proof is simply the dual of that for (7.3).

Summary. The functors B_λ and S^λ are liftings. We can see this in following diagrams, where U is the forgetful functor to the underlying category.

$$\begin{array}{ccc} \mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C}) & \xrightarrow{\mathsf{B}_\lambda} & \mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C}) \\ {\scriptstyle\mathsf{U}}\downarrow & & \downarrow{\scriptstyle\mathsf{U}} \\ \mathscr{C} & \xrightarrow[\mathsf{B}]{} & \mathscr{C} \end{array} \qquad \begin{array}{ccc} \mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C}) & \xrightarrow{\mathsf{S}^\lambda} & \mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C}) \\ {\scriptstyle\mathsf{U}}\downarrow & & \downarrow{\scriptstyle\mathsf{U}} \\ \mathscr{C} & \xrightarrow[\mathsf{S}]{} & \mathscr{C} \end{array}$$

7.2. **Initial Object and Final Object.** Our initial λ-bialgebra will be $\langle \mu\mathsf{S}, in, (\!|\mathsf{B}_\lambda\ in|\!)\rangle$, as depicted below.

$$\begin{array}{ccc} \mathsf{S}\ (\mu\mathsf{S}) & \overset{\mathsf{S}\ (\!|a|\!)}{\dashrightarrow} & \mathsf{S}\ X \\ {\scriptstyle in}\downarrow & \textcircled{2} & \downarrow{\scriptstyle a} \\ \textcircled{1}\ \ \mu\mathsf{S} & \overset{(\!|a|\!)}{\dashrightarrow} & X \\ {\scriptstyle(\!|\mathsf{B}_\lambda\ in|\!)}\downarrow & \textcircled{3} & \downarrow{\scriptstyle c} \\ \mathsf{B}\ (\mu\mathsf{S}) & \underset{\mathsf{B}\ (\!|a|\!)}{\dashrightarrow} & \mathsf{B}\ X \end{array}$$

We have three proof obligations. First we must show that the triple $\langle \mu\mathsf{S}, in, (\!|\mathsf{B}_\lambda\ in|\!)\rangle$ is indeed a λ-bialgebra (①)—it has the right types, but it must also satisfy (5.1).

$$\begin{array}{ll} & (\!|\mathsf{B}_\lambda\ in|\!) \cdot in \\ = & \quad \{\ \text{fold computation (C.2)}\ \} \\ & \mathsf{B}_\lambda\ in \cdot \mathsf{S}\ (\!|\mathsf{B}_\lambda\ in|\!) \\ = & \quad \{\ \text{definition of } \mathsf{B}_\lambda\ (7.1)\ \} \\ & \mathsf{B}\ in \cdot \lambda\ (\mu\mathsf{S}) \cdot \mathsf{S}\ (\!|\mathsf{B}_\lambda\ in|\!) \end{array}$$

The second obligation, that $(\!|a|\!)$ is an S-algebra homomorphism is by construction—the top half of the diagram commutes (②). Moreover, the uniqueness of this arrow

comes for free. Finally, we must show that $(\!|a|\!)$ is also a B-coalgebra homomorphism—that the bottom half of the diagram commutes (③).

$$
\begin{array}{ll}
 & c \cdot (\!|a|\!) = \mathsf{B}\,(\!|a|\!) \cdot (\!|\mathsf{B}_\lambda\ in|\!) \\
\Longleftrightarrow & \quad \{ \text{ fold fusion (C.3) with } c : a \to \mathsf{B}_\lambda\ a : \mathsf{S}\text{-}\mathbf{Alg} \ \} \\
 & (\!|\mathsf{B}_\lambda\ a|\!) = \mathsf{B}\,(\!|a|\!) \cdot (\!|\mathsf{B}_\lambda\ in|\!) \\
\Longleftarrow & \quad \{ \text{ fold fusion (C.3) } \} \\
 & \mathsf{B}\,(\!|a|\!) : \mathsf{B}_\lambda\ in \to \mathsf{B}_\lambda\ a : \mathsf{S}\text{-}\mathbf{Alg} \\
\Longleftarrow & \quad \{ \ \mathsf{B}_\lambda \text{ functor (7.3) } \} \\
 & (\!|a|\!) : in \to a : \mathsf{S}\text{-}\mathbf{Alg}
\end{array}
$$

We can dualize the results above. We have just used Finn's coalgebra to construct the initial λ-bialgebra, so naturally we will use Iniga's algebra to construct the final λ-bialgebra. Indeed, $\langle \nu\mathsf{B}, [\![\mathsf{S}^\lambda\ out]\!], out\rangle$ is the final λ-bialgebra; and $[\![c]\!]$ is the unique homomorphism from any λ-bialgebra $\langle X, a, c\rangle$ to the final λ-bialgebra. The duality extends to the satisfaction of the proof obligations.

We have defined the initial and final λ-bialgebras, and we are now in a position to state the homomorphism between them—the semantic function from syntax to behaviour $\mu\mathsf{S} \to \nu\mathsf{B}$.

$$
\begin{array}{ccc}
\mathsf{S}\,(\mu\mathsf{S}) & \dashrightarrow & \mathsf{S}\,(\nu\mathsf{B}) \\
{\scriptstyle in}\downarrow & & \downarrow{\scriptstyle [\![\mathsf{S}^\lambda\ out]\!]} \\
\mu\mathsf{S} & \overset{(\!|[\![\mathsf{S}^\lambda\ out]\!]|\!)}{\dashrightarrow} & \nu\mathsf{B} \\
{\scriptstyle (\!|\mathsf{B}_\lambda\ in|\!)}\downarrow & & \downarrow{\scriptstyle out} \\
\mathsf{B}\,(\mu\mathsf{S}) & \dashrightarrow & \mathsf{B}\,(\nu\mathsf{B})
\end{array}
$$

The similarities to Section 6 should be plainly clear to the reader. We have two constructions for the λ-bialgebra homomorphism of type $\mu\mathsf{S}{\to}\nu\mathsf{B}$, and by uniqueness properties attributed to initial and final objects, these constructions are equal. And just like that, we have the basic resolution of Iniga and Finn's seemingly opposing points of view.

In a manner of speaking, Iniga and Finn's personalities would appear to be entwined. Iniga thought in terms of initial algebras and folds, but ended up constructing the final λ-bialgebra, and vice versa for Finn. This is not a coincidence as the category of bialgebras is isomorphic to a category of algebras over coalgebras.

$$
\begin{array}{ll}
 & \langle X, a, c\rangle : \lambda\text{-}\mathbf{Bialg}(\mathscr{C}) \\
\Longleftrightarrow & \quad \{ \text{ definition of λ-bialgebra } \} \\
 & c \cdot a = \mathsf{B}\ a \cdot \lambda\, X \cdot \mathsf{S}\ c \\
\Longleftrightarrow & \quad \{ \text{ definition of } \mathsf{S}^\lambda \text{ (7.4) } \} \\
 & c \cdot a = \mathsf{B}\ a \cdot \mathsf{S}^\lambda\ c \\
\Longleftrightarrow & \quad \{ \text{ definition of } \mathsf{B}\text{-coalgebra homomorphism } \} \\
 & a : \mathsf{S}^\lambda\ \langle X, c\rangle \to \langle X, c\rangle : \mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C}) \\
\Longleftrightarrow & \quad \{ \text{ definition of an } \mathsf{S}^\lambda\text{-algebra } \} \\
 & \langle\langle X, c\rangle, a\rangle : \mathsf{S}^\lambda\text{-}\mathbf{Alg}(\mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C}))
\end{array}
$$

The proof shows that the objects are in one-to-one correspondence. A similar calculation establishes a bijection between arrows.

$$
\begin{aligned}
& h : \langle X_1, a_1, c_1\rangle \to \langle X_2, a_2, c_2\rangle \\
\iff & \quad \{ \text{ definition of } \lambda\text{-bialgebra homomorphism } \} \\
& h \cdot a_1 = a_2 \cdot \mathsf{S}\, h \ \ \wedge \ \ \mathsf{B}\, h \cdot c_1 = c_2 \cdot h \\
\iff & \quad \{ \text{ definition of } \mathsf{B}\text{-coalgebra homomorphism } \} \\
& h \cdot a_1 = a_2 \cdot \mathsf{S}\, h \ \ \wedge \ \ h : \langle X_1, c_1\rangle \to \langle X_2, c_2\rangle : \mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C}) \\
\iff & \quad \{ \text{ definition of } \mathsf{S}^\lambda\text{-algebra homomorphism } \} \\
& h : \langle\langle X_1, c_1\rangle, a_1\rangle \to \langle\langle X_2, c_2\rangle, a_2\rangle : \mathsf{S}^\lambda\text{-}\mathbf{Alg}(\mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C}))
\end{aligned}
$$

As a summary of our construction above, for the categorically enlightened, the final λ-bialgebra is determined by the final S^λ-algebra. Recall that the *final* S-algebra is $\langle 1, !_{\mathsf{S}\,1}\rangle$. Consequently, the final S^λ-algebra is $\langle 1, !_{\mathsf{S}^\lambda\,1}\rangle = \langle\langle \nu\mathsf{B}, \mathit{out}\rangle, [\![\mathsf{S}^\lambda\ \mathit{out}]\!]\rangle$ as $\langle \nu\mathsf{B}, \mathit{out}\rangle$ is the final object in $\mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C})$.

Dually, the category of bialgebras is isomorphic to a category of coalgebras over algebras. We can see this again for objects.

$$
\begin{aligned}
& \langle X, a, c\rangle : \lambda\text{-}\mathbf{Bialg}(\mathscr{C}) \\
\iff & \quad \{ \text{ definition of } \lambda\text{-bialgebra } \} \\
& c \cdot a = \mathsf{B}\, a \cdot \lambda\, X \cdot \mathsf{S}\, c \\
\iff & \quad \{ \text{ definition of } \mathsf{B}_\lambda \ \} \\
& c \cdot a = \mathsf{B}_\lambda\, a \cdot \mathsf{S}\, c \\
\iff & \quad \{ \text{ definition of algebra homomorphism } \} \\
& c : a \to \mathsf{B}_\lambda\, a : \mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C}) \\
\iff & \quad \{ \text{ definition of coalgebra } \} \\
& \langle\langle X, a\rangle, c\rangle : \mathsf{B}_\lambda\text{-}\mathbf{Coalg}(\mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C}))
\end{aligned}
$$

A similar calculation again establishes a bijection between arrows.

$$
\begin{aligned}
& h : \langle X_1, a_1, c_1\rangle \to \langle X_2, a_2, c_2\rangle \\
\iff & \quad \{ \text{ definition of } \lambda\text{-bialgebra homomorphism } \} \\
& h \cdot a_1 = a_2 \cdot \mathsf{S}\, h \ \ \wedge \ \ \mathsf{B}\, h \cdot c_1 = c_2 \cdot h \\
\iff & \quad \{ \text{ definition of } \mathsf{S}\text{-algebra homomorphism } \} \\
& h : \langle X_1, a_1\rangle \to \langle X_2, a_2\rangle \ \ \wedge \ \ \mathsf{B}\, h \cdot c_1 = c_2 \cdot h \\
\iff & \quad \{ \text{ definition of } \mathsf{B}_\lambda\text{-coalgebra homomorphism } \} \\
& h : \langle\langle X_1, a_1\rangle, c_1\rangle \to \langle\langle X_2, a_2\rangle, c_2\rangle : \mathsf{B}_\lambda\text{-}\mathbf{Coalg}(\mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C}))
\end{aligned}
$$

As one might expect, the initial λ-bialgebra is determined by the initial B_λ-coalgebra. Therefore, we have a double isomorphism with respect to the category of bialgebras.

$$
\lambda\text{-}\mathbf{Bialg}(\mathscr{C}) \cong \begin{cases} \mathsf{S}^\lambda\text{-}\mathbf{Alg}(\mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C})) \\ \mathsf{B}_\lambda\text{-}\mathbf{Coalg}(\mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C})) \end{cases} \tag{7.7}
$$

The double isomorphism says that there are actually two ways to determine initial and final objects in $\lambda\text{-}\mathbf{Bialg}(\mathscr{C})$. The reader is encouraged to work out the details.

8. Pointed Functors over Endofunctors

Remember that Finn remarked that in order to define *plus*, the semantic counterpart to the syntax *Plus*, he needed to embed streams into terms: he required some facility for *variables*. Iniga and Finn were using distributive laws of type $\mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}$

and these are not sufficiently expressive to model the recursion equations such as *bird* and *fib* as their right-hand sides consist of more than one layer of syntax. In general, to address the lack of variables and layers of syntax, we need terms. As Iniga and Finn's teacher hinted at, the *free monad* is the right structure to provide this. However, rather than making a beeline for free monads, we will visit *pointed functors* as a stepping stone. This is an adventure with category theory after all, and the fun is in the journey.

Example 8.1. Suspend your disbelief and suppose that you need the identity operator on streams, defined by the equation,

$$id\ (Cons\ m\ s) = m \prec s\ .$$

A system containing this equation cannot be turned into a distributive law $\lambda : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}$ as the stream s is not an element of the syntax functor S. To solve this, we can allow for variables or constructors of S.

$$\begin{aligned}&\mathbf{data}\ \mathsf{P}\ x = Var\ x \mid Con\ (\mathsf{S}\ x)\\ &\mathbf{data}\ \mathsf{S}\ x = Id\ x \mid One \mid \ldots\end{aligned}$$

A system of recursion equations is now captured by a natural transformation ρ of type $\mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{P}$.

$$\begin{aligned}\rho\ (Id\ (Cons\ (m, s))) &= Cons\ (m, Var\ s)\\ \rho\ (One) &= Cons\ (1,\ Con\ One) \quad \ldots\end{aligned}$$

Note that we have only replaced S on the right-hand side, where there is a need. We shall later restore symmetry and show how to turn ρ into a distributive law (Section 8.3). Furthermore, this is a very limited introduction of variables: one can either have a variable, or a constructor, but no variables as arguments. □

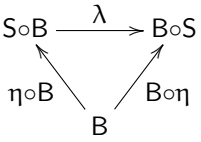
The Haskell type P is the so-called free pointed functor of S. We will discuss pointed functors in general and then return to the free construction in Section 8.1.

Definition 8.2. We say that S is *pointed* if it is equipped with a natural transformation $\eta : \mathsf{Id} \dot{\to} \mathsf{S}$.

We are going to build on the picture we laid out in the previous section by replacing the plain endofunctor with a pointed functor. The extra structure that we have introduced with η has two implications: first with regards to the distributive law λ and second with regards to constructing algebras of pointed functors.

Condition 8.3. A distributive law $\lambda : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}$ for a pointed functor S has an additional coherence condition to satisfy:

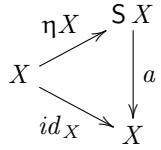
$$\lambda \cdot \eta\circ\mathsf{B} = \mathsf{B}\circ\eta\ , \tag{8.1}$$



The condition says that there are two ways to construct an arrow from behaviour to behaviour over syntax, and that these must be equal.

Condition 8.4. If we construct an algebra $\langle X, a : \mathsf{S}\ X \to X\rangle$ of a pointed functor S, then it must respect η:

$$a \cdot \eta X = id_X\ , \tag{8.2}$$



For full specificity we will say that (S,η)-**Alg**$(\mathscr{C})$ is the category of S-algebras that respect η. This is a full subcategory of S-**Alg**$(\mathscr{C})$. Henceforth, we will be working with λ-bialgebras based on (S,η)-algebras and B-coalgebras.

The double isomorphism (7.7) succinctly tells the story of initial and final objects in λ-**Bialg**. In a sense, Conditions 8.3 and 8.4 ensure that we can establish an analogous isomorphism for pointed functors. The following two properties prepare the ground.

Property 8.5. Let $c : X \to \mathsf{B}\,X$ be a B-coalgebra, then

$$\eta X : c \to \mathsf{S}^\lambda\, c : \mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C}) \ , \tag{8.3}$$

is the lifting of η to a B-coalgebra homomorphism.

Proof.

$$\begin{aligned}
& \mathsf{S}^\lambda\, c \cdot \eta X \\
= \quad & \{ \text{ definition of } \mathsf{S}^\lambda \ (7.4) \ \} \\
& \lambda\, X \cdot \mathsf{S}\, c \cdot \eta X \\
= \quad & \{ \ \eta : \mathsf{Id} \dot{\to} \mathsf{S} \text{ is natural } \} \\
& \lambda\, X \cdot \eta(\mathsf{B}\, X) \cdot c \\
= \quad & \{ \text{ coherence of } \lambda \text{ with } \eta \ (8.1) \ \} \\
& \mathsf{B}\,(\eta X) \cdot c \quad \square
\end{aligned}$$

In other words, the lifted functor S^λ is pointed as well and we can form $(\mathsf{S}^\lambda,\eta)$-**Alg**$(\mathsf{B}$-**Coalg**$(\mathscr{C}))$.

Property 8.6. The functor B_λ preserves respect for η.

$$\mathsf{B}_\lambda\, a \cdot \eta(\mathsf{B}\, X) = id_{\mathsf{B}\,X} \quad \Longleftarrow \quad a \cdot \eta X = id_X \tag{8.4}$$

Proof.

$$\begin{aligned}
& \mathsf{B}_\lambda\, a \cdot \eta(\mathsf{B}\, X) \\
= \quad & \{ \text{ definition of } \mathsf{B}_\lambda \ (7.1) \ \} \\
& \mathsf{B}\, a \cdot \lambda\, X \cdot \eta(\mathsf{B}\, X) \\
= \quad & \{ \text{ coherence of } \lambda \text{ with } \eta \ (8.1) \ \} \\
& \mathsf{B}\, a \cdot \mathsf{B}\,(\eta X) \\
= \quad & \{ \ \mathsf{B} \text{ functor and assumption } a \cdot \eta X = id_X \ \} \\
& id_{\mathsf{B}\,X} \quad \square
\end{aligned}$$

In other words, B_λ is an endofunctor on (S,η)-**Alg**$(\mathscr{C})$ and we can form B_λ-**Coalg**$((\mathsf{S},\eta)$-**Alg**$(\mathscr{C}))$.

Summary. Properties 8.5 and 8.6 imply that the double isomorphism (7.7) carries over to the new setting.

$$\lambda\text{-}\mathbf{Bialg}(\mathscr{C}) \cong \begin{cases} (\mathsf{S}^\lambda,\eta)\text{-}\mathbf{Alg}(\mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C})) \\ \mathsf{B}_\lambda\text{-}\mathbf{Coalg}((\mathsf{S},\eta)\text{-}\mathbf{Alg}(\mathscr{C})) \end{cases} \tag{8.5}$$

8.1. **Free Pointed Functor.** Let $\mathsf{S} : \mathscr{C} \to \mathscr{C}$ be an endofunctor. There is a canonical pointed functor, with pleasant properties, that we can construct from S. This is the *free* pointed functor of S, the categorical version of the Haskell type P we saw in Example 8.1,

$$\mathsf{P}\, X = X + \mathsf{S}\, X \ . \tag{8.6}$$

The natural transformation $\eta : \mathsf{Id} \dot{\to} \mathsf{P}$ that equips the free pointed functor is simply $\eta = \mathit{inl}$. Our λ-bialgebras now have P-algebras, but what about all the S-algebras that we have used previously? All is not lost, in fact far from it.

Theorem 8.1. *The category of algebras for the free pointed functor is isomorphic to the category of* S*-algebras:*

$$(\mathsf{P}, \eta)\text{-}\mathbf{Alg}(\mathscr{C}) \cong \mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C}) \ .$$

The following definitions are the witnesses to this isomorphism.

$$\lceil \langle X, a : \mathsf{S}\, X \to X \rangle \rceil = \langle X, \mathit{id}_X \triangledown a : \mathsf{P}\, X \to X \rangle \qquad \lceil h \rceil = h \tag{8.7}$$

$$\lfloor \langle X, b : \mathsf{P}\, X \to X \rangle \rfloor = \langle X, b \cdot \mathit{inr} : \mathsf{S}\, X \to X \rangle \qquad \lfloor h \rfloor = h \tag{8.8}$$

In particular, $\lceil - \rceil$ *preserves and reflects homomorphisms.*

$$h : \lceil a \rceil \to \lceil b \rceil : (\mathsf{P}, \eta)\text{-}\mathbf{Alg}(\mathscr{C}) \iff h : a \to b : \mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C}) \tag{8.9}$$

Proof.

(i) Given an S-algebra a, we can cast it up to a P-algebra $\lceil a \rceil$. Likewise, we can cast a P-algebra b down to an S-algebra $\lfloor b \rfloor$. The following proves both directions of the isomorphism.

$$\begin{aligned}
&\lfloor \lceil a \rceil \rfloor \\
&= \quad \{ \text{ definition of } \lceil - \rceil \ \} \\
&\lfloor \mathit{id}_X \triangledown a \rfloor \\
&= \quad \{ \text{ definition of } \lfloor - \rfloor \ \} \\
&(\mathit{id}_X \triangledown a) \cdot \mathit{inr} \\
&= \quad \{ \text{ join computation (B.3b) } \} \\
&a
\end{aligned}
\qquad\qquad
\begin{aligned}
&\lceil \lfloor b \rfloor \rceil \\
&= \quad \{ \text{ definitions of } \lfloor - \rfloor \text{ and } \lceil - \rceil \ \} \\
&\mathit{id}_X \triangledown b \cdot \mathit{inr} \\
&= \quad \{ \ b \text{ respects } \eta \text{ (8.2) } \} \\
&b \cdot \mathit{inl} \triangledown b \cdot \mathit{inr} \\
&= \quad \{ \text{ join fusion (B.4) } \} \\
&b \cdot (\mathit{inl} \triangledown \mathit{inr}) \\
&= \quad \{ \text{ join reflection (B.2) } \} \\
&b
\end{aligned}$$

(ii) $\lfloor - \rfloor$ is functorial—it maps P-homomorphisms to S-homomorphisms—as $\mathit{inr} : \mathsf{S} \dot{\to} \mathsf{P}$ is natural and $\lfloor - \rfloor = \mathit{inr}\text{-}\mathbf{Alg}$ (cf. Section 3.3).

(iii) $\lceil - \rceil$ maps S-homomorphisms to P-homomorphisms.

$$\begin{aligned}
&h \cdot \lceil a \rceil \\
&= \quad \{ \text{ definition of } \lceil - \rceil \text{ (8.7) } \} \\
&h \cdot (\mathit{id} \triangledown a) \\
&= \quad \{ \text{ join fusion (B.4) } \} \\
&h \cdot \mathit{id} \triangledown h \cdot a \\
&= \quad \{ \text{ identity and assumption } h : a \to b : \mathsf{S}\text{-}\mathbf{Alg} \ \} \\
&\mathit{id} \cdot h \triangledown b \cdot \mathsf{S}\, h \\
&= \quad \{ \text{ join functor fusion (B.5) } \} \\
&(\mathit{id} \triangledown b) \cdot (h + \mathsf{S}\, h) \\
&= \quad \{ \text{ definitions of } \lceil - \rceil \text{ (8.7) and } \mathsf{P} \text{ (8.6) } \} \\
&\lceil b \rceil \cdot \mathsf{P}\, h
\end{aligned}$$

(iv) Finally, $\lceil \langle X, a \rangle \rceil$ has to be an algebra for the pointed functor—it must respect η (8.2).

$$\begin{aligned} & \lceil a \rceil \cdot \eta X \\ = \quad & \{ \text{ definitions of } \lceil - \rceil \text{ (8.7) and } \eta \ \} \\ & (id_X \triangledown a) \cdot inl \\ = \quad & \{ \text{ join computation (B.3a) } \} \\ & id_X \quad \square \end{aligned}$$

Summary. We can think of $\lfloor - \rfloor$ and $\lceil - \rceil$ as *casting* operators between (P, η)-algebras and S-algebras—they are functors that form an isomorphism of categories.

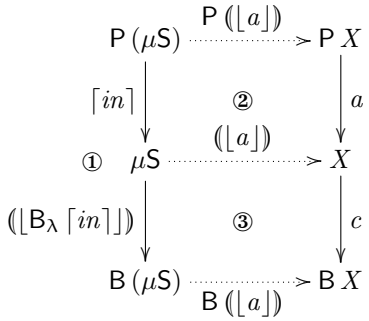
$$\begin{array}{c} (\mathsf{P}, \eta)\text{-}\mathbf{Alg} \\ \lceil - \rceil \uparrow \cong \downarrow \lfloor - \rfloor \\ \mathsf{S}\text{-}\mathbf{Alg} \end{array}$$

8.2. **Initial Object and Final Object.** The double isomorphism (8.5) immediately suggests how to define initial and final objects in the new setting. Nonetheless, we will slow down a bit and go through the construction step by step.

In Section 7 we explored λ-bialgebras over S and B, the functors representing syntax and behaviour, respectively. Despite the fact that we are now using the free pointed functor of S, the carrier of the initial λ-bialgebra will remain the same, as we are not changing our objects of syntax. Instead, we are generalizing the evaluation of our syntax. The initial λ-bialgebra will be $\langle \mu\mathsf{S}, a : \mathsf{P}(\mu\mathsf{S}) \to \mu\mathsf{S}, c : \mu\mathsf{S} \to \mathsf{B}(\mu\mathsf{S}) \rangle$, for some a and c that we will now determine.

Previously the algebra component of the initial λ-bialgebra was simply $in : \mathsf{S}(\mu\mathsf{S}) \to \mu\mathsf{S}$. This can no longer be the case; we need an algebra $a : \mathsf{P}(\mu\mathsf{S}) \to \mu\mathsf{S}$. However, now that we can freely cast between S and (P, η)-algebras, we can use $\lceil in \rceil : \mathsf{P}(\mu\mathsf{S}) \to \mu\mathsf{S}$.

The previous coalgebra component was $(\!|\mathsf{B}_\lambda\ in|\!)$, and this also no longer has the right type, as our λ has changed. Now B_λ lifts the functor B to a functor on (P, η)-algebras, not S-algebras; $(\!|-|\!)$ expects an S-algebra, and in is an S-algebra. We can satisfy these expectations with selective usage of casting: we can cast in up to a (P, η)-algebra so that we can apply B_λ, and furthermore, we can cast the image of $\mathsf{B}_\lambda \lceil in \rceil$ down so that it is an S-algebra that we can fold. We see all this information in the following diagram.



The claim is that $\langle \mu\mathsf{S}, \lceil in \rceil, (\!|\lfloor \mathsf{B}_\lambda \lceil in \rceil \rfloor|\!) \rangle$ is the initial λ-bialgebra and $(\!|\lfloor a \rfloor|\!)$ is the unique homomorphism to any λ-bialgebra $\langle X, a, c \rangle$. There are the three usual proof obligations we must satisfy. For reasons that will become clear, we will start by showing that $(\!|\lfloor a \rfloor|\!)$ is (P, η)-algebra homomorphism (②).

$$(\!|\lfloor a \rfloor|\!) : \lceil in \rceil \to a : (\mathsf{P}, \eta)\text{-}\mathbf{Alg} \tag{8.10}$$

This is a direct consequence of Theorem 8.1.

$$\begin{aligned}
& (\!(\lfloor a \rfloor)\!) : \lceil in \rceil \to a : (\mathsf{P},\eta)\text{-}\mathbf{Alg} \\
\iff & \quad \{ \text{ isomorphism } (\mathsf{P},\eta)\text{-}\mathbf{Alg} \cong \mathsf{S}\text{-}\mathbf{Alg} \ (8.9) \ \} \\
& (\!(\lfloor a \rfloor)\!) : in \to \lfloor a \rfloor : \mathsf{S}\text{-}\mathbf{Alg}
\end{aligned}$$

Next we will show that $\langle \mu\mathsf{S}, \lceil in \rceil, (\!(\lfloor \mathsf{B}_\lambda \lceil in \rceil \rfloor)\!) \rangle$ is indeed a λ-bialgebra, in that it satisfies the pentagonal law (5.1) (①).

$$\begin{aligned}
& (\!(\lfloor \mathsf{B}_\lambda \lceil in \rceil \rfloor)\!) \cdot \lceil in \rceil \\
= & \quad \{ \ (\!(\lfloor \mathsf{B}_\lambda \lceil in \rceil \rfloor)\!) : \lceil in \rceil \to \mathsf{B}_\lambda \lceil in \rceil : (\mathsf{P},\eta)\text{-}\mathbf{Alg} \ (8.10) \ \} \\
& \mathsf{B}_\lambda \lceil in \rceil \cdot \mathsf{P} \, (\!(\lfloor \mathsf{B}_\lambda \lceil in \rceil \rfloor)\!) \\
= & \quad \{ \text{ definition of } \mathsf{B}_\lambda \ (7.1) \ \} \\
& \mathsf{B} \lceil in \rceil \cdot \lambda \, (\mu\mathsf{S}) \cdot \mathsf{P} \, (\!(\lfloor \mathsf{B}_\lambda \lceil in \rceil \rfloor)\!)
\end{aligned}$$

Furthermore, (8.2) is satisfied since $\lceil - \rceil$ creates such an algebra.

Finally, we will show that $(\!(\lfloor a \rfloor)\!)$ is a B-coalgebra homomorphism (③). We know from the pentagonal law that c is a (P,η)-algebra homomorphism, $c : a \to \mathsf{B}_\lambda \, a : (\mathsf{P},\eta)\text{-}\mathbf{Alg}$. By (8.9) c is also an S-algebra homomorphism, $c : \lfloor a \rfloor \to \lfloor \mathsf{B}_\lambda \, a \rfloor : \mathsf{S}\text{-}\mathbf{Alg}$, and as a direct consequence of fold fusion (C.3), $c \cdot (\!(\lfloor a \rfloor)\!) = (\!(\lfloor \mathsf{B}_\lambda \, a \rfloor)\!)$.

$$\begin{aligned}
& c \cdot (\!(\lfloor a \rfloor)\!) = \mathsf{B} \, (\!(\lfloor a \rfloor)\!) \cdot (\!(\lfloor \mathsf{B}_\lambda \lceil in \rceil \rfloor)\!) \\
\iff & \quad \{ \text{ fold fusion } (C.3) \text{ with } c : \lfloor a \rfloor \to \lfloor \mathsf{B}_\lambda \, a \rfloor : \mathsf{S}\text{-}\mathbf{Alg} \ \} \\
& (\!(\lfloor \mathsf{B}_\lambda \, a \rfloor)\!) = \mathsf{B} \, (\!(\lfloor a \rfloor)\!) \cdot (\!(\lfloor \mathsf{B}_\lambda \lceil in \rceil \rfloor)\!) \\
\impliedby & \quad \{ \text{ fold fusion } (C.3) \ \} \\
& \mathsf{B} \, (\!(\lfloor a \rfloor)\!) : \lfloor \mathsf{B}_\lambda \lceil in \rceil \rfloor \to \lfloor \mathsf{B}_\lambda \, a \rfloor : \mathsf{S}\text{-}\mathbf{Alg} \\
\iff & \quad \{ \text{ isomorphism } (\mathsf{P},\eta)\text{-}\mathbf{Alg} \cong \mathsf{S}\text{-}\mathbf{Alg} \ (8.9) \ \} \\
& \mathsf{B} \, (\!(\lfloor a \rfloor)\!) : \mathsf{B}_\lambda \lceil in \rceil \to \mathsf{B}_\lambda \, a : (\mathsf{P},\eta)\text{-}\mathbf{Alg} \\
\impliedby & \quad \{ \ \mathsf{B}_\lambda \text{ functor } (7.3) \ \} \\
& (\!(\lfloor a \rfloor)\!) : \lceil in \rceil \to a : (\mathsf{P},\eta)\text{-}\mathbf{Alg}
\end{aligned}$$

We have already shown that the last statement holds (8.10).

As before, the final λ-bialgebra is $\langle \nu\mathsf{B}, [\![\mathsf{P}^\lambda \, out]\!], out \rangle$. The unique λ-bialgebra homomorphism to the final λ-bialgebra from any λ-bialgebra $\langle X, a, c \rangle$ is $[\![c]\!]$.

$$\begin{array}{ccc}
\mathsf{P}\, X & \xrightarrow{\mathsf{P}\,[\![c]\!]} & \mathsf{P}\,(\nu\mathsf{B}) \\
{\scriptstyle a}\downarrow & & \downarrow{\scriptstyle [\![\mathsf{P}^\lambda \, out]\!]} \\
X & \xrightarrow[[\![c]\!]]{} & \nu\mathsf{B} \\
{\scriptstyle c}\downarrow & & \downarrow{\scriptstyle out} \\
\mathsf{B}\, X & \xrightarrow[\mathsf{B}\,[\![c]\!]]{} & \mathsf{B}\,(\nu\mathsf{B})
\end{array}$$

There is one final proof obligation: we have to show that $[\![\mathsf{P}^\lambda \, out]\!]$ respects η (8.2).

$$\begin{aligned}
& [\![\mathsf{P}^\lambda \, out]\!] \cdot \eta(\nu\mathsf{B}) = id_{\nu\mathsf{B}} \\
\iff & \quad \{ \text{ unfold reflection } (D.1) \ \} \\
& [\![\mathsf{P}^\lambda \, out]\!] \cdot \eta(\nu\mathsf{B}) = [\![out]\!] \\
\impliedby & \quad \{ \text{ unfold fusion } (D.3) \ \} \\
& \eta(\nu\mathsf{B}) : out \to \mathsf{P}^\lambda \, out
\end{aligned}$$

The last statement holds as P^λ is pointed (8.3).

Putting things together, we can give a new statement of the semantic function $\mu\mathsf{S} \to \nu\mathsf{B}$.

$$\begin{array}{ccc}
\mathsf{P}\,(\mu\mathsf{S}) & \dashrightarrow & \mathsf{P}\,(\nu\mathsf{B}) \\
{\scriptstyle \lceil in \rceil}\downarrow & & \downarrow{\scriptstyle [\![\mathsf{P}^\lambda\ out]\!]} \\
\mu\mathsf{S} & \dashrightarrow & \nu\mathsf{B} \\
{\scriptstyle (\!\lfloor \mathsf{B}_\lambda\ \lceil in \rceil \rfloor\!)}\downarrow & & \downarrow{\scriptstyle out} \\
\mathsf{B}\,(\mu\mathsf{S}) & \dashrightarrow & \mathsf{B}\,(\nu\mathsf{B})
\end{array}
\qquad
\begin{array}{c}
(\!\lfloor [\![\mathsf{P}^\lambda\ out]\!] \rfloor\!) \\
\| \\
[\![(\!\lfloor \mathsf{B}_\lambda\ \lceil in \rceil \rfloor\!)]\!]
\end{array}$$

We are in a more expressive setting, yet thanks to Theorem 8.1, we can hold on to our resolution of Iniga and Finn's viewpoints.

8.3. **Creating a Distributive Law.** In Section 7 we modelled a stream program by a distributive law of type $\mathsf{S}\circ\mathsf{B} \stackrel{\cdot}{\to} \mathsf{B}\circ\mathsf{S}$. With the introduction of the free pointed functor, stream equations have become slightly more expressive. A program, such as in Example 8.1, now gives rise to a natural transformation $\rho : \mathsf{S}\circ\mathsf{B} \stackrel{\cdot}{\to} \mathsf{B}\circ\mathsf{P}$. The pointed functor appears only on the right. On the left we keep S, as a stream equation defines a constructor of S, not a variable. From $\rho : \mathsf{S}\circ\mathsf{B} \stackrel{\cdot}{\to} \mathsf{B}\circ\mathsf{P}$ we seek to *construct* a distributive law $\lambda : \mathsf{P}\circ\mathsf{B} \stackrel{\cdot}{\to} \mathsf{B}\circ\mathsf{P}$ such that

$$c \cdot \lceil a \rceil = \mathsf{B}\,\lceil a \rceil \cdot \lambda\, X \cdot \mathsf{P}\, c \quad \Longleftrightarrow \quad c \cdot a = \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\, c \ . \tag{8.11}$$

Since P is a coproduct, λ has to be defined by a case analysis. Though obvious, we will calculate λ from the specification above as this will serve nicely as a blueprint for the more demanding constructions ahead.

$$\begin{array}{ll}
 & c \cdot a = \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\, c \\
\Longleftrightarrow & \quad \{ \text{ equality of joins } \} \\
 & c \triangledown c \cdot a = c \triangledown \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\, c \\
\Longleftrightarrow & \quad \{ \lceil a \rceil \text{ respects } \eta \text{ (8.2) and } \mathsf{B} \text{ functor } \} \\
 & c \triangledown c \cdot a = \mathsf{B}\,\lceil a \rceil \cdot \mathsf{B}\,(\eta X) \cdot c \triangledown \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\, c \\
\Longleftrightarrow & \quad \{ \text{ join fusion (B.4) and functor fusion (B.5) } \} \\
 & c \cdot (id \triangledown a) = \mathsf{B}\,\lceil a \rceil \cdot (\mathsf{B}\,(\eta X) \triangledown \rho\, X) \cdot (c + \mathsf{S}\, c) \\
\Longleftrightarrow & \quad \{ \text{ definitions of } \lceil - \rceil \text{ (8.7) and } \mathsf{P} \text{ (8.6) } \} \\
 & c \cdot \lceil a \rceil = \mathsf{B}\,\lceil a \rceil \cdot (\mathsf{B}\,(\eta X) \triangledown \rho\, X) \cdot \mathsf{P}\, c
\end{array}$$

The specification (8.11) can be satisfied if we set $\lambda = \mathsf{B}\circ\eta \triangledown \rho$, which is easily seen to satisfy the coherence condition (8.1).

9. Monads over Endofunctors

With pointed functors we made a limited introduction of variables. The teacher told Finn that the *free monad* would enable him to embed streams *in* his terms. In this section we are going to build on our picture of λ-bialgebras again, augmenting pointed functors to monads.

Example 9.1. Let us look at an example comparable to those of Section 2. Here is a stream equation for the natural numbers.

$$nat = 0 \prec nat + 1$$

We need more than a single syntax constructor to represent $nat + 1$. To solve this, we build terms with variables and constructors of S.

$$\begin{aligned}\textbf{data}\ \mathsf{M}\ x &= \mathit{Var}\ x \mid \mathit{Com}\ (\mathsf{S}\ (\mathsf{M}\ x))\\ \textbf{data}\ \mathsf{S}\ x &= \mathit{One} \mid \mathit{Plus}\ (x, x) \mid \mathit{Nat}\end{aligned}$$

A system of recursion equations is now captured by a natural transformation ρ of type $\mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{M}$.

$$\begin{aligned}&\rho\ \mathit{One}\\ &\quad = \mathit{Cons}\ (1, \mathit{Com}\ \mathit{One})\\ &\rho\ (\mathit{Plus}\ (\mathit{Cons}\ (m, s), \mathit{Cons}\ (n, t)))\\ &\quad = \mathit{Cons}\ (m + n, \mathit{Com}\ (\mathit{Plus}\ (\mathit{Var}\ s, \mathit{Var}\ t)))\\ &\rho\ \mathit{Nat}\\ &\quad = \mathit{Cons}\ (0, \mathit{Com}\ (\mathit{Plus}\ (\mathit{Com}\ \mathit{Nat}, \mathit{Com}\ \mathit{One})))\end{aligned}$$

Note that we only have terms on the right-hand side. Arguments of *Cons* on the left can be embedded into variables on the right, and as shown in the case of *Nat*, we can use more than one level of syntax. Again, we shall restore symmetry later, showing how to derive a distributive law from ρ (Section 9.3). □

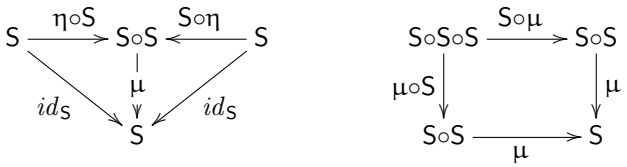
The Haskell type M is the so-called free monad of S. We will discuss monads in general and then return to the free construction in Section 9.1.

Definition 9.2. We say that S is a *monad* if it is equipped with natural transformations $\eta : \mathsf{Id} \dot{\to} \mathsf{S}$ and $\mu : \mathsf{S}\circ\mathsf{S} \dot{\to} \mathsf{S}$ such that

$$\mu \cdot \eta\circ\mathsf{S} = id_{\mathsf{S}} \quad , \tag{9.1a}$$

$$\mu \cdot \mathsf{S}\circ\eta = id_{\mathsf{S}} \quad , \tag{9.1b}$$

$$\mu \cdot \mu\circ\mathsf{S} = \mu \cdot \mathsf{S}\circ\mu \quad . \tag{9.1c}$$

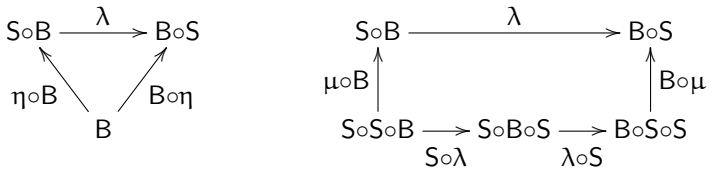


A monad extends a pointed functor with a second natural transformation $\mu : \mathsf{S}\circ\mathsf{S} \dot{\to} \mathsf{S}$. In the previous section we saw that η must be respected when constructing algebras and also by the distributive law of the λ-bialgebra; these same conditions extend to μ.

Condition 9.3. The following are the necessary coherence conditions for a distributive law $\lambda : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}$ over a monad S:

$$\lambda \cdot \eta\circ\mathsf{B} = \mathsf{B}\circ\eta \quad , \tag{9.2a}$$

$$\lambda \cdot \mu\circ\mathsf{B} = \mathsf{B}\circ\mu \cdot \lambda\circ\mathsf{S} \cdot \mathsf{S}\circ\lambda \quad . \tag{9.2b}$$

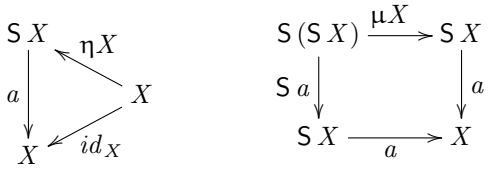


The first condition has carried over from the previous section. The second condition says that there are two ways to construct an arrow from $\mathsf{S}\circ\mathsf{S}\circ\mathsf{B}$ to $\mathsf{B}\circ\mathsf{S}$, and that these must be equal.

Condition 9.4. If we construct an algebra $\langle X, a : \mathsf{S}\,X \to X\rangle$ of a monad S, then it must respect both η and μ.

$$a \cdot \eta X = id_X \quad , \tag{9.3a}$$

$$a \cdot \mu X = a \cdot \mathsf{S}\,a \quad . \tag{9.3b}$$



In the same manner as for pointed functors, we will say that (S,η,μ)-**Alg**$(\mathscr{C})$ is the category of S-algebras that respect η and μ, a full subcategory of S-**Alg**$(\mathscr{C})$. Henceforth, we will be working with λ-bialgebras based on (S,η,μ)-algebras and B-coalgebras.

As in Section 8, the additional conditions ensure that the double isomorphism (7.7) is maintained. We have shown previously that η can be lifted to a B-coalgebra homomorphism (8.3). There is an analogous property for μ:

Property 9.5. Let $c : X \to \mathsf{B}\,X$ be a B-coalgebra, then

$$\mu X : \mathsf{S}^\lambda\,(\mathsf{S}^\lambda\,c) \to \mathsf{S}^\lambda\,c : \mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C}) \quad , \tag{9.4}$$

is the lifting of μ to a B-coalgebra homomorphism.

Proof.

$$\begin{aligned}
& \mathsf{S}^\lambda\,c \cdot \mu X \\
= \quad & \{ \text{ definition of } \mathsf{S}^\lambda \text{ (7.4) } \} \\
& \lambda\,X \cdot \mathsf{S}\,c \cdot \mu X \\
= \quad & \{ \text{ } \mu : \mathsf{S}\circ\mathsf{S} \dot{\to} \mathsf{S} \text{ is natural } \} \\
& \lambda\,X \cdot \mu(\mathsf{B}\,X) \cdot \mathsf{S}\,(\mathsf{S}\,c) \\
= \quad & \{ \text{ coherence of } \lambda \text{ with } \mu \text{ (9.2b) } \} \\
& \mathsf{B}\,(\mu X) \cdot \lambda\,(\mathsf{S}\,X) \cdot \mathsf{S}\,(\lambda\,X) \cdot \mathsf{S}\,(\mathsf{S}\,c) \\
= \quad & \{ \text{ } \mathsf{S} \text{ functor and definition of } \mathsf{S}^\lambda \text{ (7.4) } \} \\
& \mathsf{B}\,(\mu X) \cdot \lambda\,(\mathsf{S}\,X) \cdot \mathsf{S}\,(\mathsf{S}^\lambda\,c) \\
= \quad & \{ \text{ definition of } \mathsf{S}^\lambda \text{ (7.4) } \} \\
& \mathsf{B}\,(\mu X) \cdot \mathsf{S}^\lambda\,(\mathsf{S}^\lambda\,c) \quad \square
\end{aligned}$$

In other words, the lifted functor S^λ is a monad as well and we can form $(\mathsf{S}^\lambda,\eta,\mu)$-**Alg**$(\mathsf{B}$-**Coalg**$(\mathscr{C}))$.

We also have shown that B_λ preserves respect for η (8.4). Again, there is an analogous property for μ:

Property 9.6. The lifted functor B_λ preserves respect for μ.

$$\mathsf{B}_\lambda\,a \cdot \mu(\mathsf{B}\,X) = \mathsf{B}_\lambda\,a \cdot \mathsf{S}\,(\mathsf{B}_\lambda\,a) \quad \Longleftarrow \quad a \cdot \mu X = a \cdot \mathsf{S}\,a \tag{9.5}$$

Proof.

$$
\begin{aligned}
& \mathsf{B}_\lambda\ a \cdot \mu(\mathsf{B}\ X) \\
= & \quad \{ \text{ definition of } \mathsf{B}_\lambda \ (7.1) \ \} \\
& \mathsf{B}\ a \cdot \lambda\ X \cdot \mu(\mathsf{B}\ X) \\
= & \quad \{ \text{ coherence of } \lambda \text{ with } \mu \ (9.2\text{b}) \ \} \\
& \mathsf{B}\ a \cdot \mathsf{B}\ (\mu X) \cdot \lambda\ (\mathsf{S}\ X) \cdot \mathsf{S}\ (\lambda\ X) \\
= & \quad \{ \ \mathsf{B} \text{ functor and assumption } a \cdot \mu X = a \cdot \mathsf{S}\ a \ \} \\
& \mathsf{B}\ a \cdot \mathsf{B}\ (\mathsf{S}\ X) \cdot \lambda\ (\mathsf{S}\ X) \cdot \mathsf{S}\ (\lambda\ X) \\
= & \quad \{ \ \lambda : \mathsf{S} \circ \mathsf{B} \dot{\to} \mathsf{B} \circ \mathsf{S} \text{ is natural } \} \\
& \mathsf{B}\ a \cdot \lambda\ X \cdot \mathsf{S}\ (\mathsf{B}\ X) \cdot \mathsf{S}\ (\lambda\ X) \\
= & \quad \{ \ \mathsf{S} \text{ functor and definition of } \mathsf{B}_\lambda \ (7.1) \ \} \\
& \mathsf{B}_\lambda\ a \cdot \mathsf{S}\ \mathsf{B}_\lambda\ a \quad \square
\end{aligned}
$$

Thus, B_λ is an endofunctor on (S, η, μ)-$\mathbf{Alg}(\mathscr{C})$ and we can form B_λ-$\mathbf{Coalg}((\mathsf{S}, \eta, \mu)$-$\mathbf{Alg}(\mathscr{C}))$.

Summary. As before the category of bialgebras can be seen as a category of algebras over coalgebras or as a category of coalgebras over algebras.

$$
\lambda\text{-}\mathbf{Bialg}(\mathscr{C}) \cong \begin{cases} (\mathsf{S}^\lambda, \eta, \mu)\text{-}\mathbf{Alg}(\mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C})) \\ \mathsf{B}_\lambda\text{-}\mathbf{Coalg}((\mathsf{S}, \eta, \mu)\text{-}\mathbf{Alg}(\mathscr{C})) \end{cases} \tag{9.6}
$$

9.1. **Free Monad.** Let $\mathsf{S} : \mathscr{C} \to \mathscr{C}$ be an endofunctor representing our syntax. There is a canonical monad, with pleasant properties, that we can construct from S. To do so we will first describe the *free* S-algebra.

The free S-algebra over X is an algebra $\langle \mathsf{M}\ X, \mathit{com} \rangle$ equipped with an arrow $\mathit{var} : X \to \mathsf{M}\ X$. We think of elements of $\mathsf{M}\ X$ as terms built from our syntax functor S and variables drawn from X. There are two ways to construct a term: var embeds a variable into a term; and $\mathit{com} : \mathsf{S}\ (\mathsf{M}\ X) \to \mathsf{M}\ X$ constructs a composite term from a level of syntax over subterms.

If we have an algebra $a : \mathsf{S}\ X \to X$, we can evaluate a term with $(\!|a|\!) : \mathsf{M}\ X \to X$ (pronounce "eval"). Given an arrow $g : Y \to X$ to evaluate variables and an S-algebra a to evaluate composites, evaluation of terms is characterized by the uniqueness property,

$$
f = (\!|a|\!) \cdot \mathsf{M}\ g \iff f \cdot \mathit{var} = g \ \wedge \ f \cdot \mathit{com} = a \cdot \mathsf{S}\ f \ , \tag{9.7}
$$

for all $f : \mathsf{M}\ Y \to X$. The equivalence states that a compositional evaluation of a term, second conjunct, is uniquely defined by an evaluation of variables, first conjunct. (For the clued-in reader, all of this information comes from the adjunction of the free and forgetful functors between S-$\mathbf{Alg}(\mathscr{C})$ and $\mathscr{C}$.)

The initial algebra emerges as a special case: $\mu\mathsf{S} \cong \mathsf{M}\ 0$. It represents the closed terms. Modulo this isomorphism, we have $\mathit{in} = \mathit{com}_0$ and $(\!|a|\!) = (\!|a|\!) \cdot \mathsf{M}\ ¡_A$. (Again, this relation is induced by the aforementioned adjunction.)

There are two simple consequences of the uniqueness property. If we set the evaluation of variables to the identity ($g = \mathit{id}$), we get the computation laws:

$$
(\!|a|\!) \cdot \mathit{var} = \mathit{id} \ , \tag{9.8a}
$$

$$
(\!|a|\!) \cdot \mathit{com} = a \cdot \mathsf{S}\ (\!|a|\!) \ . \tag{9.8b}
$$

As var and com are the constructors of terms, we can read these as defining equations of $(\!|-|\!)$. The uniqueness property also implies that var and com are natural

in X and that $\mathbf{(-)}$ preserves naturality.

$$\mathbf{(\alpha)} : \mathsf{M}\circ\mathsf{G} \dot{\to} \mathsf{G} \quad \Longleftarrow \quad \alpha : \mathsf{S}\circ\mathsf{G} \dot{\to} \mathsf{G}$$

(Here $\mathbf{(\alpha)}$ is shorthand for $\phi\, A = \mathbf{(}\alpha\, A\mathbf{)}$.)

Proof. Let $h : A \to B$, then

$$\begin{array}{ll} & \mathsf{G}\, h \cdot \mathbf{(}\alpha\, A\mathbf{)} = \mathbf{(}\alpha\, B\mathbf{)} \cdot \mathsf{M}\,(\mathsf{G}\, h) \\ \Longleftrightarrow & \quad \{ \text{ uniqueness of eval (9.7) } \} \\ & \mathsf{G}\, h \cdot \mathbf{(}\alpha\, A\mathbf{)} \cdot \mathit{var} = \mathsf{G}\, h \quad \wedge \quad \mathsf{G}\, h \cdot \mathbf{(}\alpha\, A\mathbf{)} \cdot \mathit{com} = \alpha\, B \cdot \mathsf{S}\,(\mathsf{G}\, h \cdot \mathbf{(}\alpha\, A\mathbf{)}) \end{array}$$

The first conjunct is an immediate consequence of (9.8a). For the second conjunct we reason:

$$\begin{array}{ll} & \mathsf{G}\, h \cdot \mathbf{(}\alpha\, A\mathbf{)} \cdot \mathit{com} \\ = & \quad \{ \text{ eval computation (9.8b) } \} \\ & \mathsf{G}\, h \cdot \alpha\, A \cdot \mathsf{S}\,\mathbf{(}\alpha\, A\mathbf{)} \\ = & \quad \{ \text{ assumption: } \alpha : \mathsf{S}\circ\mathsf{G} \dot{\to} \mathsf{G} \text{ is natural } \} \\ & \alpha\, B \cdot \mathsf{S}\,(\mathsf{G}\, h) \cdot \mathsf{S}\,\mathbf{(}\alpha\, A\mathbf{)} \\ = & \quad \{ \mathsf{S} \text{ functor } \} \\ & \alpha\, B \cdot \mathsf{S}\,(\mathsf{G}\, h \cdot \mathbf{(}\alpha\, A\mathbf{)}) \quad \square \end{array}$$

Now that we have established the naturality of *var*, *com* and $\mathbf{(-)}$, we can use these to define a monad.

Definition 9.7. The free monad of the functor S is $\langle \mathsf{M}, \eta, \mu \rangle$, where $\eta = \mathit{var}$ and $\mu = \mathbf{(}\mathit{com}\mathbf{)}$.

The $\mu : \mathsf{M}\circ\mathsf{M} \dot{\to} \mathsf{M}$ of the monad flattens a term whose variables are terms. It does so by evaluating the term with the composite constructor—the action of the free algebra.

Theorem 9.1. *The category of algebras for the free monad of* S *is isomorphic to the category of* S*-algebras:*

$$(\mathsf{M}, \eta, \mu)\text{-}\mathbf{Alg}(\mathscr{C}) \cong \mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C}) \ .$$

The following definitions are the witnesses to this isomorphism.

$$\lceil \langle X, a : \mathsf{S}\, X \to X \rangle \rceil = \langle X, \mathbf{(}a\mathbf{)} : \mathsf{M}\, X \to X \rangle \qquad \lceil h \rceil = h \ , \tag{9.9}$$

$$\lfloor \langle X, b : \mathsf{M}\, X \to X \rangle \rfloor = \langle X, b \cdot \theta X : \mathsf{S}\, X \to X \rangle \qquad \lfloor h \rfloor = h \ , \tag{9.10}$$

where $\theta = \mathit{com} \cdot \mathsf{S}\circ\eta : \mathsf{S} \dot{\to} \mathsf{M}$*, which turns a level of syntax into a term. (Note that we are overloading the notation* $\lceil - \rceil$ *and* $\lfloor - \rfloor$*.) The map* $\lceil - \rceil$ *preserves and reflects homomorphisms.*

$$h \ : \ \lceil a \rceil \to \lceil b \rceil \ : \ (\mathsf{M}, \eta, \mu)\text{-}\mathbf{Alg}(\mathscr{C}) \quad \Longleftrightarrow \quad h \ : \ a \to b \ : \ \mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C}) \tag{9.11}$$

Proof.

(i) $\lfloor\lceil\langle X, a\rangle\rceil\rfloor = \langle X, a\rangle$:

$$
\begin{aligned}
& \lfloor\lceil a\rceil\rfloor \\
= & \quad \{ \text{ definitions of } \lceil-\rceil \text{ (9.9) and } \lfloor-\rfloor \text{ (9.10) } \} \\
& (\!|a|\!) \cdot \theta X \\
= & \quad \{ \text{ definition of } \theta \ \} \\
& (\!|a|\!) \cdot \mathit{com} \cdot \mathsf{S}\,(\eta X) \\
= & \quad \{ \text{ eval computation (9.8b) } \} \\
& a \cdot \mathsf{S}\,(\!|a|\!) \cdot \mathsf{S}\,(\eta X) \\
= & \quad \{ \ \mathsf{S} \text{ functor } \} \\
& a \cdot \mathsf{S}\,((\!|a|\!) \cdot \eta X) \\
= & \quad \{ \text{ eval computation (9.8a) } \} \\
& a \cdot \mathsf{S}\,\mathit{id} \\
= & \quad \{ \ \mathsf{S} \text{ functor } \} \\
& a
\end{aligned}
$$

An instance of this property is $\mathit{com}_X = \lfloor\lceil \mathit{com}_X \rceil\rfloor = (\!|\mathit{com}_X|\!) \cdot \theta(\mathsf{M}\,X)$. This says that the composite constructor com is equal to embedding the subterm arguments to the syntax constructor as variables and then flattening the term of terms.

In the opposite direction, $\lceil\lfloor\langle X, b\rangle\rfloor\rceil = \langle X, b\rangle$:

$$
\begin{aligned}
& \lceil\lfloor b\rfloor\rceil = b \\
\iff & \quad \{ \text{ definitions of } \lceil-\rceil \text{ (9.9) and } \lfloor-\rfloor \text{ (9.10) } \} \\
& (\!|b \cdot \theta X|\!) = b \\
\iff & \quad \{ \text{ uniqueness of eval (9.7) } \} \\
& b \cdot \eta X = \mathit{id} \ \ \wedge \ \ b \cdot \mathit{com} = b \cdot \theta X \cdot \mathsf{S}\,b
\end{aligned}
$$

The first conjunct follows from the fact that b respects η (9.3a). For the second conjunct we reason:

$$
\begin{aligned}
& b \cdot \theta X \cdot \mathsf{S}\,b \\
= & \quad \{ \ \theta : \mathsf{S} \dot{\to} \mathsf{M} \text{ is natural } \} \\
& b \cdot \mathsf{M}\,b \cdot \theta(\mathsf{M}\,X) \\
= & \quad \{ \ b \text{ respects } \mu \text{ (9.3b) } \} \\
& b \cdot \mu X \cdot \theta(\mathsf{M}\,X) \\
= & \quad \{ \ \mu X \cdot \theta(\mathsf{M}\,X) = \mathit{com}, \text{ see above } \} \\
& b \cdot \mathit{com} \ .
\end{aligned}
$$

(ii) $\lfloor-\rfloor$ is functorial—it maps as M-homomorphisms to S-homomorphisms—as $\theta : \mathsf{S} \to \mathsf{M}$ is natural and $\lfloor-\rfloor = \theta\text{-}\mathbf{Alg}$ (cf. Section 3.3).

(iii) $\lceil-\rceil$ maps S-homomorphisms to M-homomorphisms.

$$
\begin{aligned}
& h \cdot (\!|a|\!) = (\!|b|\!) \cdot \mathsf{M}\,h \\
\iff & \quad \{ \text{ uniqueness of eval (9.7) } \} \\
& h \cdot (\!|a|\!) \cdot \eta X = h \ \ \wedge \ \ h \cdot (\!|a|\!) \cdot \mathit{com} = b \cdot \mathsf{S}\,(h \cdot (\!|a|\!))
\end{aligned}
$$

The first conjunct is a direct consequence of computation (9.8a). For the second conjunct we reason:

$$\begin{aligned} & h \cdot (\!|a|\!) \cdot com \\ = & \quad \{ \text{ eval computation (9.8b) } \} \\ & h \cdot a \cdot \mathsf{S}\,(\!|a|\!) \\ = & \quad \{ \text{ assumption } h : a \to b : \mathsf{S}\text{-}\mathbf{Alg} \} \\ & b \cdot \mathsf{S}\,h \cdot \mathsf{S}\,(\!|a|\!) \\ = & \quad \{ \ \mathsf{S} \text{ functor } \} \\ & b \cdot \mathsf{S}\,(h \cdot (\!|a|\!)) \end{aligned}$$

(iv) Finally, $\lceil \langle A, a \rangle \rceil$ is an algebra for the monad. That $\lceil a \rceil$ respects η (9.3a), unfolds to, $(\!|a|\!) \cdot \eta X = id$, which is the first computation law (9.8a). That $\lceil a \rceil$ respects μ (9.3b), unfolds to, $(\!|a|\!) \cdot \mu X = (\!|a|\!) \cdot \mathsf{M}\,(\!|a|\!)$, and this follows from part (iii)

$$\begin{aligned} & (\!|a|\!) \cdot \mu X = (\!|a|\!) \cdot \mathsf{M}\,(\!|a|\!) \\ \Longleftarrow & \quad \{ \ \lceil - \rceil \text{ maps } \mathsf{S}\text{- to } \mathsf{M}\text{-homomorphisms and } \mu = (\!|com|\!) \ \} \\ & (\!|a|\!) \cdot com = a \cdot \mathsf{S}\,(\!|a|\!) \end{aligned}$$

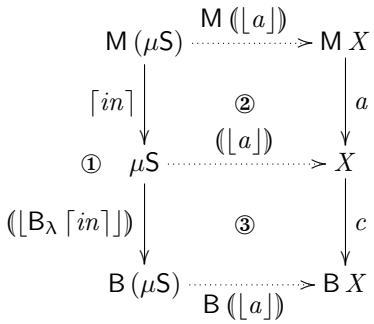
and the second computation law (9.8b). □

Summary. Taking Theorems 8.1 and 9.1 together, $\lfloor - \rfloor$ and $\lceil - \rceil$ are now casting operators between (M, η, μ)-algebras, (P, η)-algebras and S-algebras—all three categories are isomorphic.

$$\begin{array}{ccc} (\mathsf{P},\eta)\text{-}\mathbf{Alg} & & (\mathsf{M},\eta,\mu)\text{-}\mathbf{Alg} \\ \lceil - \rceil \uparrow \cong \downarrow \lfloor - \rfloor & & \lceil - \rceil \uparrow \cong \downarrow \lfloor - \rfloor \\ \mathsf{S}\text{-}\mathbf{Alg} & & \mathsf{S}\text{-}\mathbf{Alg} \end{array}$$

9.2. **Initial Object and Final Object.** Now that we have completed another round of generalization, from free pointed functors to free monads, it is appropriate to examine what the new initial and final λ-bialgebras are. Again, they can be derived from the double isomorphism (9.6), and again, we will highlight the salient details.

Superficially, the initial λ-bialgebra has not changed: it remains $\langle \mu\mathsf{S}, \lceil in \rceil, (\!|\lfloor \mathsf{B}^{\lambda}\, \lceil in \rceil \rfloor|\!) \rangle$. What *has* changed are the definitions of $\lceil - \rceil$ and $\lfloor - \rfloor$.



The usual three proof obligations are all discharged by the proofs provided in previous section. All of the proof steps have analogues in this section—in particular, Theorem 8.1 has been succeeded by Theorem 9.1.

The final λ-bialgebra is $\langle \nu\mathsf{B}, [\![\mathsf{M}^{\lambda}\, out]\!], out \rangle$; the single change is replacing P^{λ} with M^{λ}. The unique λ-bialgebra homomorphism to the final λ-bialgebra from

any λ-bialgebra $\langle X, a, C\rangle$ is still $[\![c]\!]$. Just as in Section 8.2, there is one final proof obligation: we have to show that $[\![\mathsf{M}^\lambda\ out]\!]$ is an algebra for M. Previously we showed that $[\![\mathsf{P}^\lambda\ out]\!]$ respects η, and this proof suffices to show the same of $[\![\mathsf{M}^\lambda\ out]\!]$ (9.3a). It remains to show that μ is respected (9.3b): we show that both sides of equation (9.3b) simplify to $[\![\mathsf{M}^\lambda\ (\mathsf{M}^\lambda\ out)]\!]$.

$$
\begin{array}{ll}
& [\![\mathsf{M}^\lambda\ (\mathsf{M}^\lambda\ out)]\!] = [\![\mathsf{M}^\lambda\ out]\!] \cdot \mu(\nu\mathsf{B}) \\
\Longleftarrow & \quad \{\ \text{unfold fusion (D.3)}\ \} \\
& \mu(\nu\mathsf{B}) : \mathsf{M}^\lambda\ (\mathsf{M}^\lambda\ out) \to \mathsf{M}^\lambda\ out \\
\Longleftrightarrow & \quad \{\ \mathsf{M}^\lambda \text{ is a monad (9.4)}\ \} \\
& true
\end{array}
\qquad
\begin{array}{ll}
& [\![\mathsf{M}^\lambda\ (\mathsf{M}^\lambda\ out)]\!] = [\![\mathsf{M}^\lambda\ out]\!] \cdot \mathsf{M}\ [\![\mathsf{M}^\lambda\ out]\!] \\
\Longleftarrow & \quad \{\ \text{unfold fusion (D.3)}\ \} \\
& \mathsf{M}\ [\![\mathsf{M}^\lambda\ out]\!] : \mathsf{M}^\lambda\ (\mathsf{M}^\lambda\ out) \to \mathsf{M}^\lambda\ out \\
\Longleftarrow & \quad \{\ \mathsf{M}^\lambda \text{ functor (7.6)}\ \} \\
& [\![\mathsf{M}^\lambda\ out]\!] : \mathsf{M}^\lambda\ out \to out \\
\Longleftarrow & \quad \{\ \text{type of } [\![-]\!]\ \} \\
& true
\end{array}
$$

Finally, we can give another statement of the semantic function $\mu\mathsf{S} \to \nu\mathsf{B}$, in the setting of $\lambda : \mathsf{M}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{M}$.

$$
\begin{array}{ccc}
\mathsf{M}\ (\mu\mathsf{S}) & \cdots\cdots\succ & \mathsf{M}\ (\nu\mathsf{B}) \\
{\scriptstyle \lceil in \rceil}\downarrow & (\!\lfloor [\![\mathsf{M}^\lambda\ out]\!] \rfloor\!) & \downarrow{\scriptstyle [\![\mathsf{M}^\lambda\ out]\!]} \\
\mu\mathsf{S} & \cdots\cdots \| \cdots\cdots\succ & \nu\mathsf{B} \\
{\scriptstyle (\!\lfloor \mathsf{B}_\lambda\ \lceil in \rceil \rfloor\!)}\downarrow & [\!(\!\lfloor \mathsf{B}_\lambda\ \lceil in \rceil \rfloor\!)\!] & \downarrow{\scriptstyle out} \\
\mathsf{B}\ (\mu\mathsf{S}) & \cdots\cdots\succ & \mathsf{B}\ (\nu\mathsf{B})
\end{array}
$$

We have upgraded pointed functors to monads and Theorem 9.1 ensures that Iniga and Finn still see eye to eye. However, we will need to repeat the exercise of Section 8.3.

9.3. **Creating a Distributive Law.** Given a program that is modelled by a natural transformation of type $\rho : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{M}$, we seek to derive a distributive law $\lambda : \mathsf{M}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{M}$ such that

$$
c \cdot \lceil a \rceil = \mathsf{B}\ \lceil a \rceil \cdot \lambda\ X \cdot \mathsf{M}\ c \quad \Longleftrightarrow \quad c \cdot a = \mathsf{B}\ \lceil a \rceil \cdot \rho\ X \cdot \mathsf{S}\ c \ . \tag{9.12}
$$

Let us calculate.

$$
\begin{array}{ll}
& c \cdot a = \mathsf{B}\ \lceil a \rceil \cdot \rho\ X \cdot \mathsf{S}\ c \\
\Longleftrightarrow & \quad \{\ \text{isomorphism } (\mathsf{M}, \eta, \mu)\text{-}\mathbf{Alg} \cong \mathsf{S}\text{-}\mathbf{Alg} \text{ (9.11)}\ \} \\
& c \cdot \lceil a \rceil = \lceil \mathsf{B}\ \lceil a \rceil \cdot \rho\ X \rceil \cdot \mathsf{M}\ c \\
\Longleftrightarrow & \quad \{\ \text{see below}\ \} \\
& c \cdot \lceil a \rceil = \mathsf{B}_\lambda\ \lceil a \rceil \cdot \mathsf{M}\ c \\
\Longleftrightarrow & \quad \{\ \text{definition of } \mathsf{B}_\lambda \text{ (7.1)}\ \} \\
& c \cdot \lceil a \rceil = \mathsf{B}\ \lceil a \rceil \cdot \lambda\ X \cdot \mathsf{M}\ c
\end{array}
$$

The specification (9.12) holds if $\mathsf{B}_\lambda\ \lceil a \rceil = \lceil \mathsf{B}\ \lceil a \rceil \cdot \rho\ X \rceil$. To turn this property into a definition for λ, we have to delve a bit deeper into the theory. Applegate [1965] discovered that distributive laws $\lambda : \mathsf{M}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{M}$ are in one-to-one correspondence to lifted functors $\bar{\mathsf{B}} : (\mathsf{M}, \eta, \mu)\text{-}\mathbf{Alg} \to (\mathsf{M}, \eta, \mu)\text{-}\mathbf{Alg}$, where a functor $\bar{\mathsf{B}}$ is a lifting of B if its action on carriers and homomorphisms is given by B, that is, $\mathsf{U}\circ\bar{\mathsf{B}} = \mathsf{B}\circ\mathsf{U}$.[1] It

[1] The result is actually more general, but this extra generality is not needed.

is useful to make explicit what it means for $\bar{\mathsf{B}}$ to preserve algebra homomorphisms.[2]

$$\mathsf{B}\,h \cdot \bar{\mathsf{B}}\,a = \bar{\mathsf{B}}\,b \cdot \mathsf{M}\,(\mathsf{B}\,h) \quad \Longleftarrow \quad h \cdot a = b \cdot \mathsf{M}\,h \tag{9.13}$$

This property immediately implies that $\bar{\mathsf{B}}$ takes natural algebras of type $\mathsf{M}\circ\mathsf{F} \dot{\to} \mathsf{F}$ to natural algebras of type $\mathsf{M}\circ\mathsf{B}\circ\mathsf{F} \dot{\to} \mathsf{B}\circ\mathsf{F}$.

Looking back, we note that we have already made extensive use of the correspondence in one direction, turning a distributive law into a lifting B_λ; now we need the opposite direction. Given a lifting $\bar{\mathsf{B}}$, we can construct a distributive law as follows. The uniqueness property (9.7) states that homomorphisms of type $\mathsf{M}\,X \to A$ are in one-to-one correspondence to arrows of type $X \to A$. We aim to construct $\lambda : \mathsf{M}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{M}$, so we need a natural transformation of type $\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{M}$. The composition $\mathsf{B}\circ\eta$ will do nicely. We obtain:

$$\lambda_{\bar{\mathsf{B}}} = \bar{\mathsf{B}}\,\mu \cdot \mathsf{M}\circ\mathsf{B}\circ\eta \ , \tag{9.14}$$

where $\bar{\mathsf{B}}\,\mu : \mathsf{M}\circ\mathsf{B}\circ\mathsf{M} \dot{\to} \mathsf{B}\circ\mathsf{M}$ is the M-algebra for the carrier $\mathsf{B}\circ\mathsf{M}$. We must show that $\lambda_{\bar{\mathsf{B}}}$ coheres with η and μ per equations (9.2a) and (9.2b). To reduce clutter we set $\alpha = \bar{\mathsf{B}}\,\mu$. For the first coherence condition (9.2a) we reason:

$$\begin{aligned}
& \lambda_{\bar{\mathsf{B}}} \cdot \eta\circ\mathsf{B} \\
= & \quad \{ \text{ definition of } \lambda_{\bar{\mathsf{B}}} \ (9.14) \ \} \\
& \alpha \cdot \mathsf{M}\circ\mathsf{B}\circ\eta \cdot \eta\circ\mathsf{B} \\
= & \quad \{ \text{ interchange law: } \eta : \mathsf{Id} \dot{\to} \mathsf{M} \ \} \\
& \alpha \cdot \eta\circ\mathsf{B}\circ\mathsf{M} \cdot \mathsf{B}\circ\eta \\
= & \quad \{ \ \alpha \text{ respects } \eta \ (9.3a) \ \} \\
& \mathsf{B}\circ\eta \ .
\end{aligned}$$

Turning to the second coherence condition (9.2b), the following observation is useful. If we apply the lifting property (9.13) to the third monad law (9.1c), we obtain:

$$\mathsf{B}\circ\mu \cdot \alpha\circ\mathsf{M} = \alpha \cdot \mathsf{M}\circ\mathsf{B}\circ\mu \ . \tag{9.15}$$

The calculation below simplifies both sides of (9.2b) to $\alpha \cdot \mathsf{M}\circ\lambda_{\bar{\mathsf{B}}}$.

$$\begin{aligned}
& \lambda_{\bar{\mathsf{B}}} \cdot \mu\circ\mathsf{B} \\
= & \quad \{ \text{ definition of } \lambda_{\bar{\mathsf{B}}} \ (9.14) \ \} \\
& \alpha \cdot \mathsf{M}\circ\mathsf{B}\circ\eta \cdot \mu\circ\mathsf{B} \\
= & \quad \{ \text{ interchange law: } \eta : \mathsf{Id} \dot{\to} \mathsf{M} \text{ and } \mu : \mathsf{M}\circ\mathsf{M} \dot{\to} \mathsf{M} \ \} \\
& \alpha \cdot \mu\circ\mathsf{B}\circ\mathsf{M} \cdot \mathsf{M}\circ\mathsf{M}\circ\mathsf{B}\circ\eta \\
= & \quad \{ \ \alpha \text{ respects } \mu \ (9.3b) \ \} \\
& \alpha \cdot \mathsf{M}\circ\alpha \cdot \mathsf{M}\circ\mathsf{M}\circ\mathsf{B}\circ\eta \\
= & \quad \{ \ \mathsf{M}\circ- \text{ functor } \} \\
& \alpha \cdot \mathsf{M}\circ(\alpha \cdot \mathsf{M}\circ\mathsf{B}\circ\eta) \\
= & \quad \{ \text{ definition of } \lambda_{\bar{\mathsf{B}}} \ (9.14) \ \} \\
& \underline{\alpha \cdot \mathsf{M}\circ\lambda_{\bar{\mathsf{B}}}} \\
= & \quad \{ \text{ first monad law } (9.1a) \ \} \\
& \alpha \cdot \mathsf{M}\circ\mathsf{B}\circ\mu \cdot \mathsf{M}\circ\mathsf{B}\circ\eta\circ\mathsf{M} \cdot \mathsf{M}\circ\lambda_{\bar{\mathsf{B}}} \\
= & \quad \{ \ (9.15) \ \} \\
& \mathsf{B}\circ\mu \cdot \alpha\circ\mathsf{M} \cdot \mathsf{M}\circ\mathsf{B}\circ\eta\circ\mathsf{M} \cdot \mathsf{M}\circ\lambda_{\bar{\mathsf{B}}}
\end{aligned}$$

[2]The notation is potentially misleading: $\bar{\mathsf{B}}$ is applied to the action of an algebra. It sends $a : \mathsf{M}\,X \to X$ to $\bar{\mathsf{B}}\,a : \mathsf{M}\,(\mathsf{B}\,X) \to \mathsf{B}\,X$. It is *not* the application of a functor to an arrow.

$$
\begin{aligned}
&= \quad \{ \text{ definition of } \lambda_{\bar{\mathsf{B}}} \text{ (9.14) } \} \\
&\quad \mathsf{B} \circ \mu \cdot \lambda_{\bar{\mathsf{B}}} \circ \mathsf{M} \cdot \mathsf{M} \circ \lambda_{\bar{\mathsf{B}}}
\end{aligned}
$$

The mappings $\lambda \mapsto \mathsf{B}_\lambda$ and $\bar{\mathsf{B}} \mapsto \lambda_{\bar{\mathsf{B}}}$ then establish the one-to-one correspondence between distributive laws and lifted functors.

$\lambda = \lambda_{\mathsf{B}_\lambda}$**:**

$$
\begin{aligned}
&\lambda_{\mathsf{B}_\lambda} \\
&= \quad \{ \text{ definitions of } \mathsf{B}_\lambda \text{ (7.1) and } \lambda_{\bar{\mathsf{B}}} \text{ (9.14) } \} \\
&\quad \mathsf{B} \circ \mu \cdot \lambda \circ \mathsf{M} \cdot \mathsf{M} \circ \mathsf{B} \circ \eta \\
&= \quad \{ \lambda : \mathsf{M} \circ \mathsf{B} \dot{\to} \mathsf{B} \circ \mathsf{M} \text{ is natural } \} \\
&\quad \mathsf{B} \circ \mu \cdot \mathsf{B} \circ \mathsf{M} \circ \eta \cdot \lambda \\
&= \quad \{ \mathsf{B} \text{ functor and second monad law (9.1b) } \} \\
&\quad \lambda
\end{aligned}
$$

$\bar{\mathsf{B}} = \mathsf{B}_{\lambda_{\bar{\mathsf{B}}}}$**:**

$$
\begin{aligned}
&\mathsf{B}_{\lambda_{\bar{\mathsf{B}}}}\, a \\
&= \quad \{ \text{ definitions of } \lambda_{\bar{\mathsf{B}}} \text{ (9.14) and } \mathsf{B}_\lambda \text{ (7.1) } \} \\
&\quad \mathsf{B}\, a \cdot \bar{\mathsf{B}}\, (\mu A) \cdot \mathsf{M}\, (\mathsf{B}\, (\eta A)) \\
&= \quad \{ \text{ (9.13) applied to } a \cdot \mu A = a \cdot \mathsf{M}\, a \text{—} a \text{ respects } \mu \text{ (9.3b) } \} \\
&\quad \bar{\mathsf{B}}\, a \cdot \mathsf{M}\, (\mathsf{B}\, a) \cdot \mathsf{M}\, (\mathsf{B}\, (\eta A)) \\
&= \quad \{ \mathsf{M} \text{ and } \mathsf{B} \text{ functor, and } a \text{ respects } \eta \text{ (9.3a) } \} \\
&\quad \bar{\mathsf{B}}\, a
\end{aligned}
$$

Returning to the task at hand, constructing a distributive law from ρ, we use the property $\mathsf{B}_\lambda \lceil a \rceil = \lceil \mathsf{B} \lceil a \rceil \cdot \rho\, X \rceil$ to define:

$$
\begin{aligned}
\bar{\mathsf{B}}\, \langle X, b : \mathsf{M}\, X \to X \rangle &= \langle \mathsf{B}\, X, \lceil \mathsf{B}\, b \cdot \rho\, X \rceil : \mathsf{M}\, (\mathsf{B}\, X) \to \mathsf{B}\, X \rangle \ , \\
\bar{\mathsf{B}}\, h &= \mathsf{B}\, h \ .
\end{aligned}
$$

This defines a lifting because $\lceil - \rceil = (-)$ is one. Putting things together, the distributive law $\lambda = \lambda_{\bar{\mathsf{B}}}$ expressed as a composition of natural transformations is:

$$
\lambda = (\mathsf{B} \circ \mu \cdot \rho \circ \mathsf{M}) \cdot \mathsf{M} \circ \mathsf{B} \circ \eta \ . \tag{9.16}
$$

9.4. **Semantics of Open Terms.** Before we left Iniga and Finn, we heard that Finn could define $\underline{head} : \mu\mathsf{S} \to \mathbb{N}$ and $tail : \mu\mathsf{S} \to \mu\mathsf{S}$, but he had a problem defining the function $\underline{plus} : \nu\mathsf{B} \times \nu\mathsf{B} \to \nu\mathsf{B}$, the semantic counterpart to the syntax constructor *Plus*, as he had no means to embed streams in terms. The teacher promised that the free monad was the right structure to address this, and we have seen that now. So let us specifically address Finn's problem; his $\underline{plus}$ is defined as,

$$
\underline{plus}\, (s, t) = evaluate\, (Plus\, (Var\, s, Var\, t)) \ ,
$$

where $evaluate : \mathsf{M}\, (\nu\mathsf{B}) \to \nu\mathsf{B}$.

To give a definition to stream operators such as $\underline{plus}$, we need to define an evaluation function for open terms, that is, an arrow of type $\mathsf{M}\, (\nu\mathsf{B}) \to \nu\mathsf{B}$. The task is clear: we need to define a bialgebra structure on $\mathsf{M}\, (\nu\mathsf{B})$. Here is a handy lemma:

Lemma 9.2. *If* $\langle X, d \rangle$ *is a* B*-coalgebra, then* $\langle \mathsf{M}\, X, \mu X, \mathsf{M}^\lambda\, d \rangle$ *is a* λ*-bialgebra. (cf. Theorem 7.2, Turi and Plotkin [1997])*

Proof. We have three proof obligations.

(i) $\langle \mathsf{M}\,X, \mu X\rangle$ has to be an algebra for the monad M. This is immediate as the instantiated coherence properties for the algebra, (9.3a) and (9.3b), become the monad laws (9.1b) and (9.1c).
(ii) That $\langle \mathsf{M}\,X, \mathsf{M}^{\lambda}\,d\rangle$ is a coalgebra for the endofunctor B follows from the definition of M^{λ}.
(iii) Finally, the pentagonal law (5.1) is satisfied:

$$\begin{aligned}
& \mathsf{B}\,(\mu X)\cdot\lambda\,(\mathsf{M}\,X)\cdot\mathsf{M}\,(\mathsf{M}^{\lambda}\,d)\\
= \quad & \{\ \text{definition of } \mathsf{M}^{\lambda}\,d\ \}\\
& \mathsf{B}\,(\mu X)\cdot\lambda\,(\mathsf{M}\,X)\cdot\mathsf{M}\,(\lambda X\cdot\mathsf{M}\,d)\\
= \quad & \{\ \mathsf{M}\ \text{functor}\ \}\\
& \mathsf{B}\,(\mu X)\cdot\lambda\,(\mathsf{M}\,X)\cdot\mathsf{M}\,(\lambda X)\cdot\mathsf{M}\,(\mathsf{M}\,d)\\
= \quad & \{\ \text{coherence property of } \lambda \text{ with } \mu\ (9.2b)\ \}\\
& \lambda X\cdot\mu(\mathsf{B}\,X)\cdot\mathsf{M}\,(\mathsf{M}\,d)\\
= \quad & \{\ \mu : \mathsf{M}\circ\mathsf{M}\dot{\to}\mathsf{M}\ \text{is natural}\ \}\\
& \lambda X\cdot\mathsf{M}\,d\cdot\mu X\\
= \quad & \{\ \text{definition of } \mathsf{M}^{\lambda}\,d\ \}\\
& \mathsf{M}^{\lambda}\,d\cdot\mu X \quad \square
\end{aligned}$$

Returning to the original problem of giving a semantics to open terms, we can use the λ-bialgebra $\langle \mathsf{M}\,(\nu\mathsf{B}), \mu(\nu\mathsf{B}), \mathsf{M}^{\lambda}\,out\rangle$ and the evaluation function is $[\![\mathsf{M}^{\lambda}\,out]\!] : \mathsf{M}\,(\nu\mathsf{B}) \to \nu\mathsf{B}$.

9.5. **Embedding Behaviour into Syntax.** In this section we need to construct the free monad for different syntax functors S, so we replace the notation M by the more informative S^*. The mapping $(-)^*$ is actually a higher-order functor whose arrow part takes a natural transformation $\alpha : \mathsf{S} \dot{\to} \mathsf{T}$ to a natural transformation $\alpha^* : \mathsf{S}^* \dot{\to} \mathsf{T}^*$. (The action on arrows is defined $\alpha^* = (\alpha\text{-}\mathbf{Alg}\ com_{\mathsf{T}})_{\mathsf{S}} \cdot \mathsf{S}^*\ var_{\mathsf{T}}$.) Think of α^* as a term converter.

Example 9.8. Consider the following alternative definition of *fib*.

$$fib = 0 \prec (1 \prec fib) + fib$$

Note is that there is a nested occurrence of $\prec$. $\square$

We can support nested stream constructors if we embed behaviour into syntax. The new syntax-with-behaviour functor is $\mathsf{T}\,X = \mathsf{B}\,X + \mathsf{S}\,X$, or in Haskell,

$$\mathbf{data}\ \mathsf{T}\ x = FBy\ (\mathbb{N}, x)\ |\ \ldots\ ,$$

where *FBy* is pronounced "followed by", and is the syntactic counterpart of the semantic *Cons*. Given a system of recursion equations $\rho : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{T}^*$, we can construct a symmetric system σ as,

$$\sigma = \mathsf{B}\circ inl^* \cdot \mathsf{B}\circ\theta \triangledown \rho = \mathsf{B}\circ(\theta \cdot inl) \triangledown \rho : \mathsf{T}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{T}^*\ ,$$

where $inl : \mathsf{B} \dot{\to} \mathsf{T}$ and $\theta : \mathsf{T} \dot{\to} \mathsf{T}^*$. A distributive law $\lambda : \mathsf{T}^*\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{T}^*$ can then be created following the recipe of Section 9.3.

Embedding behaviour into syntax is a special case of extending a system of recursion equations. Specifically, given a base system $\rho_1 : \mathsf{S}_1\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}_1^*$ and an extension $\rho_2 : \mathsf{S}_2\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}^*$, where $\mathsf{S}\,X = \mathsf{S}_1\,X + \mathsf{S}_2\,X$, we can form a combined system as follows:

$$\rho = \mathsf{B}\circ inl^* \cdot \rho_1 \triangledown \rho_2 : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}^*\ .$$

The idea is that ρ_2 can use the operators of S_1 *and* S_2 to define the operators of S_2. Setting $\mathsf{S}_1 = \mathsf{B}$ and $\rho_1 = \mathsf{B}\circ\theta$ the embedding above emerges as a special case.

A minor, but perhaps less interesting variation is the merge of two independent systems of recursion equations,

$$\rho = \mathsf{B}\circ inl^* \cdot \rho_1 \triangledown \mathsf{B}\circ inr^* \cdot \rho_2 \ ,$$

where $\rho_1 : \mathsf{S}_1\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}_1^*$ and $\rho_2 : \mathsf{S}_2\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}_2^*$.

The embedding makes the constructors of B available in the syntax. Often, one also wishes to embed an element of $\nu\mathsf{B}$: consider the equation $x = 0 \prec \mathit{even}\, \mathit{fib}' + x$ of Section 2. The stream $\mathit{even}\, \mathit{fib}'$ is defined by a previous system, in fact, two systems; we wish to reuse it at this point. This can be accommodated by setting $\mathsf{S}_1\, X = \nu\mathsf{B}$ and $\rho_1 = \mathsf{B}\circ \mathit{com} \cdot \mathit{out}$. Here S_1 is a constant functor—elements of $\nu\mathsf{B}$ are embedded as constants. On a final note, merging the systems for fib', even and x is *not* an option as even uses a different definitional style and, as we have pointed out, we cannot mix styles. Of course, we have to show that even is uniquely defined and this is what we do in Section 13.

9.6. **Proving The Unique Fixed-Point Principle Correct.** Let us now return to our original problem of proving the unique fixed-point principle correct. Also, a brief summary is perhaps not amiss. A system of recursion equations is modelled by a natural transformation $\rho : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}^*$, where S is the syntax functor and B the behaviour functor. The type of ρ captures the slogan *consume at most one, produce at least one*. Using the trick of embedding behaviour into syntax we can consume nothing (the argument is reassembled on the right) and we can produce more than one. Systems of this form are quite liberal; most, but not all of the examples in the literature satisfy the restrictions.

A solution of a system modelled by ρ consists of an S-algebra and a B-coalgebra over a common carrier that satisfies:

$$c \cdot a = \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\, c \ .$$

We can now replay the calculations of Section 4. If the coalgebra is final, then a is uniquely determined, which establishes the UFP:

$$\begin{aligned}
& \mathit{out} \cdot a = \mathsf{B}\,\lceil a \rceil \cdot \rho\,(\nu\mathsf{B}) \cdot \mathsf{S}\, \mathit{out} \\
\iff & \quad \{\ \lambda \text{ given by (9.16) which satisfies (9.12)}\ \} \\
& \mathit{out} \cdot \lceil a \rceil = \mathsf{B}\,\lceil a \rceil \cdot \lambda\,(\nu\mathsf{B}) \cdot \mathsf{M}\, \mathit{out} \\
\iff & \quad \{\ \text{definition of } \mathsf{M}^\lambda\ \} \\
& \mathit{out} \cdot \lceil a \rceil = \mathsf{B}\,\lceil a \rceil \cdot \mathsf{M}^\lambda\, \mathit{out} \\
\iff & \quad \{\ \text{uniqueness of unfold (3.2)}\ \} \\
& \lceil a \rceil = [\![\mathsf{M}^\lambda\, \mathit{out}]\!] \\
\iff & \quad \{\ \text{isomorphism } (\mathsf{M},\eta,\mu)\text{-}\mathbf{Alg} \cong \mathsf{S}\text{-}\mathbf{Alg} \text{ (9.1)}\ \} \\
& a = \lfloor [\![\mathsf{M}^\lambda\, \mathit{out}]\!] \rfloor \ .
\end{aligned}$$

Conversely, if the algebra is initial, then c is fixed: $c = (\!(\lfloor \mathsf{B}_\lambda\, \lceil \mathit{in} \rceil \rfloor)\!)$. Since the data defines initial and final objects in λ-$\mathbf{Bialg}(\mathscr{C})$, we can furthermore conclude that the two ways of defining the semantic function of type $\mu\mathsf{S} \to \nu\mathsf{B}$ coincide: $(\!(a)\!) = [\![c]\!]$.

10. Endofunctors over Copointed Functors

In Section 9.5 we demonstrated how to embed behaviour into syntax, which enabled us to syntactically construct behaviour on the right-hand side of a stream equation. In the case of Example 9.8 we defined a stream constant and needed a

nested occurrence of $\prec$. However, often we wish to construct behaviour on right-hand side because we in fact need to *reconstruct* behaviour that we have deconstructed on the left-hand side. In essence, we want to use a whole stream, not its tail.

Example 10.1. Let us first consider the stream operator that alternates between two streams:

$$alternate\ (Cons\ m\ s)\ (Cons\ \_\ t) = m \prec alternate\ t\ s\ .$$

For all streams s, $alternate\ s\ s = s$. In this definition it is the tails of the input streams that are used in the recursive call. Now consider the stream operator that interleaves two streams:

$$interleave\ (Cons\ m\ s)\ (Cons\ n\ t) = m \prec interleave\ (n \prec t)\ s\ .$$

The result of $interleave\ (0\ 2\ 4\ldots)\ (1\ 3\ 5\ldots)$ is $0\ 1\ 2\ 3\ldots$, the natural numbers. In this definition we are unnecessarily deconstructing the second parameter into its head and tail, we simply need the whole stream. A more natural definition is:

$$interleave\ (Cons\ m\ s)\ t = m \prec interleave\ t\ s\ .$$

A related issue is that sometimes we want the head, tail, *and* the whole stream. Consider the stream operator that performs an ordered merge of two streams:

$$\begin{aligned} &merge\ s@(Cons\ m\ s')\ t@(Cons\ n\ t') \\ &\quad = \mathbf{if}\ m \leqslant n\ \mathbf{then}\ m \prec merge\ s'\ t\ \mathbf{else}\ n \prec merge\ s\ t'\ . \end{aligned}$$

This definition uses Haskell's *as-patterns*, where $var@pat$ gives the name var to the value being matched by pat. We can model as-patterns in our categorical setting with *copointed* functors.

$$\mathbf{data}\ \mathsf{C}\ x = As\ x\ (\mathsf{B}\ x)$$

Revisiting our natural transformations of type $\mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}$, if we replace B on the left-hand side with C, then we are able to construct a natural transformation to model *interleave* and *merge* without the need to reconstruct behaviour on the right-hand side.

$$\begin{aligned} &\rho :: \mathsf{S}\ (\mathsf{C}\ x) \to \mathsf{B}\ (\mathsf{S}\ x) \\ &\rho\ (Interleave\ (As\ \_\ (Cons\ (m, s)), As\ t\ \_)) \\ &\quad = Cons\ (m, Interleave\ (t, s)) \\ &\rho\ (Merge\ (As\ s\ (Cons\ (m, s')), As\ t\ (Cons\ (n, t')))) \\ &\quad = Cons\ (\mathbf{if}\ m \leqslant n\ \mathbf{then}\ m\ \mathbf{else}\ n, \\ &\qquad\qquad Merge\ (\mathbf{if}\ m \leqslant n\ \mathbf{then}\ s'\ \mathbf{else}\ s, \\ &\qquad\qquad\qquad \mathbf{if}\ m \leqslant n\ \mathbf{then}\ t\ \ \mathbf{else}\ t')) \end{aligned}$$

Our model of *interleave* is using a degenerative form of the as-pattern, as we have no need to pattern match. We have not generalized the B on the right-hand side: it would not make sense to label the produced behaviour. Copointed functors give us flexibility: before we could only say that a stream equation "consumes exactly one", now we can say that a stream equation "consumes *at most* one". □

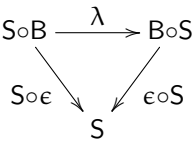
The Haskell type C is the so-called *cofree copointed* functor of B. We will discuss copointed functors in general and then return to the cofree construction in Section 10.1.

Definition 10.2. We say that B is *copointed* if it is equipped with a natural transformation $\epsilon : \mathsf{B} \dot{\to} \mathsf{Id}$.

We are going to build on the picture laid out in Section 7 by replacing plain endofunctor B with a copointed functor. The extra structure that we have introduced with ϵ has two implications: first with regards to the distributive law λ, and second with regards to constructing coalgebras of copointed functors.

Condition 10.3. A distributive law $\lambda : \mathsf{S} \circ \mathsf{B} \dot{\to} \mathsf{B} \circ \mathsf{S}$ for a pointed functor B has an additional coherence condition to satisfy:

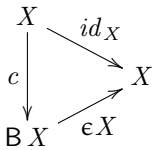
$$\epsilon \circ \mathsf{S} \cdot \lambda = \mathsf{S} \circ \epsilon \ , \tag{10.1}$$



The condition says that there are two ways to construct an arrow from syntax over behaviour to syntax, and that these must be equal.

Condition 10.4. If we construct a coalgebra $\langle X, c : X \to \mathsf{B}\, X \rangle$ of a copointed functor B, then it must respect ϵ:

$$\epsilon X \cdot c = id_X \ , \tag{10.2}$$



For full specificity we will say that (B, ϵ)-**Coalg**$(\mathscr{C})$ is the category of B-coalgebras that respect ϵ (a full subcategory of B-**Coalg**$(\mathscr{C})$). Henceforth, we will be working with λ-bialgebras based on S-algebras and (B, ϵ)-coalgebras.

Just as in Sections 8 and 9, the additional conditions 10.3 and 10.4 ensure that the double isomorphism (7.7) is maintained for copointed functors. In Section 8 we saw that η could be lifted to a B-coalgebra homomorphism, there is a dual property for ϵ.

Property 10.5. Let $a : \mathsf{S}\, X \to X$ be an S-algebra, then

$$\epsilon X : \mathsf{B}_\lambda\, a \to a : \mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C}) \ , \tag{10.3}$$

is the lifting of ϵ to a S-algebra homomorphism. In other words, the lifted functor B_λ is copointed and we can form $(\mathsf{B}_\lambda, \epsilon)$-**Coalg**$(\mathsf{S}$-**Alg**$(\mathscr{C}))$.

Property 10.6. We have shown previously that S^λ preserves composition, the identity, and homomorphisms (7.6). Now, S^λ as an endofunctor on (B, ϵ)-**Coalg**$(\mathscr{C})$ must preserve respect for ϵ.

$$\epsilon(\mathsf{S}\, X) \cdot \mathsf{S}^\lambda\, c = id_{\mathsf{S}\, X} \iff \epsilon X \cdot c = id_X \tag{10.4}$$

In other words, we can form S^λ-**Alg**$((\mathsf{B}, \epsilon)$-**Coalg**$(\mathscr{C}))$.

Summary. Properties 10.5 and 10.6 imply that the double isomorphism (7.7) continues to hold in this new setting.

$$\lambda\text{-}\mathbf{Bialg}(\mathscr{C}) \cong \begin{cases} \mathsf{S}^\lambda\text{-}\mathbf{Alg}((\mathsf{B}, \epsilon)\text{-}\mathbf{Coalg}(\mathscr{C})) \\ (\mathsf{B}_\lambda, \epsilon)\text{-}\mathbf{Coalg}(\mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C})) \end{cases} \tag{10.5}$$

10.1. **Cofree Copointed Functor.** Let $\mathsf{B} : \mathscr{C} \to \mathscr{C}$ be an endofunctor. There is a canonical copointed functor, with pleasant properties, that we can construct from B. This is the *cofree* copointed functor of B, the categorical version of the Haskell type C we saw in Example 10.1,

$$\mathsf{C}\, X = X \times \mathsf{B}\, X \ . \tag{10.6}$$

The natural transformation $\epsilon : \mathsf{C} \dot{\to} \mathsf{Id}$ that equips the cofree copointed functor is simply $\epsilon = outl$.

Just as when we introduced the free pointed functor P of S, our λ-bialgebras now have C-coalgebras, and these coalgebras are isomorphic to the B-coalgebras.

Theorem 10.1. *The category of coalgebras for the cofree copointed functor is isomorphic to the category of* B*-coalgebras:*

$$(\mathsf{C},\epsilon)\text{-}\mathbf{Coalg}(\mathscr{C}) \cong \mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C}) \ .$$

The following definitions are the witnesses to this isomorphism.

$$\lceil \langle X, c : X \to \mathsf{B}\, X\rangle \rceil = \langle X, id \vartriangle c : X \to \mathsf{C}\, X\rangle \qquad \lceil h \rceil = h \tag{10.7}$$

$$\lfloor \langle X, d : X \to \mathsf{C}\, X\rangle \rfloor = \langle X, outr \cdot d : X \to \mathsf{B}\, X\rangle \qquad \lfloor h \rfloor = h \tag{10.8}$$

In particular, $\lceil - \rceil$ *preserves and reflects homomorphisms.*

$$h : \lceil c \rceil \to \lceil d \rceil : (\mathsf{C},\epsilon)\text{-}\mathbf{Coalg}(\mathscr{C}) \iff h : c \to d : \mathsf{B}\text{-}\mathbf{Coalg}(\mathscr{C}) \tag{10.9}$$

Proof Outline. The proof is the dual of that for Theorem 8.1. We will simply highlight the four components of the proof.

(i) $\lfloor \lceil \langle X, c\rangle \rceil \rfloor = \langle X, c\rangle$ and $\lceil \lfloor \langle X, d\rangle \rfloor \rceil = \langle X, d\rangle$.
(ii) $\lfloor - \rfloor$ is functorial—it maps C-homomorphisms to B-homomorphisms—as $outr : \mathsf{C} \dot{\to} \mathsf{B}$ is natural and $\lfloor - \rfloor = outr\text{-}\mathbf{Coalg}$ (cf. Section 3.3).
(iii) Furthermore, $\lceil - \rceil$ maps B-homomorphisms to C-homomorphisms.
(iv) $\lceil \langle X, c\rangle \rceil$ is a coalgebra for the copointed functor (10.2). □

Summary. Theorem 8.1 established the isomorphism between (P,η)-algebras and S-algebras. Dually, Theorem 10.1 has established the isomorphism between (C,ϵ)-coalgebras and B-coalgebras, with $\lfloor - \rfloor$ and $\lceil - \rceil$ as the (overloaded) casting operators between them.

$$\begin{array}{c} (\mathsf{C},\epsilon)\text{-}\mathbf{Coalg} \\ \lceil - \rceil \uparrow \cong \downarrow \lfloor - \rfloor \\ \mathsf{B}\text{-}\mathbf{Coalg} \end{array}$$

10.2. **Initial Object and Final Object.** We have begun to explore the spine of the second dimension of our design space: we have moved from an endofunctor B to the cofree copointed functor. At this point we should examine what the new initial and final λ-bialgebras are. The objects can be derived from the double isomorphism (10.5), and here we will only provide the dual statements of Section 8.2.

The initial λ-bialgebra is $\langle \mu\mathsf{S}, in, (\!|\mathsf{C}_\lambda\ in|\!)\rangle$. The unique λ-bialgebra homomorphism from the initial λ-bialgebra to any λ-bialgebra $\langle X, a, c\rangle$ is $(\!|a|\!)$.

$$\begin{array}{ccc} \mathsf{S}\,(\mu\mathsf{S}) & \xrightarrow{\mathsf{S}\,(\!|a|\!)} & \mathsf{S}\,X \\ {\scriptstyle in}\downarrow & & \downarrow{\scriptstyle a} \\ \mu\mathsf{S} & \xrightarrow{(\!|a|\!)} & X \\ {\scriptstyle (\!|\mathsf{C}_\lambda\ in|\!)}\downarrow & & \downarrow{\scriptstyle c} \\ \mathsf{C}\,(\mu\mathsf{S}) & \xrightarrow[\mathsf{C}\,(\!|a|\!)]{} & \mathsf{C}\,X \end{array}$$

There is one proof obligation: we have to show that $(\!|\mathsf{C}_\lambda\ in|\!)$ is a coalgebra for C, that is, it respects ϵ (10.4). This is simply the dual proof of the dual obligation found in Section 8.2, following from Property 10.5.

The final λ-bialgebra is $\langle \nu\mathsf{B}, [\![\lfloor \mathsf{S}^{\lambda}\, \lceil out \rceil \rfloor]\!], \lceil out \rceil \rangle$. The unique λ-bialgebra homomorphism to the final λ-bialgebra from any λ-bialgebra $\langle X, a, c\rangle$ is $[\![\lfloor c \rfloor]\!]$.

$$\begin{array}{ccc}
\mathsf{S}\,X & \xrightarrow{\mathsf{S}\,[\![\lfloor c \rfloor]\!]} & \mathsf{S}\,(\nu\mathsf{B}) \\
{\scriptstyle a}\downarrow & ② & \downarrow{\scriptstyle [\![\lfloor \mathsf{S}^{\lambda}\, \lceil out \rceil \rfloor]\!]} \\
X & \xrightarrow[[\![\lfloor c \rfloor]\!]]{} & \nu\mathsf{B} \quad ① \\
{\scriptstyle c}\downarrow & ③ & \downarrow{\scriptstyle \lceil out \rceil} \\
\mathsf{C}\,X & \xrightarrow[\mathsf{C}\,[\![\lfloor c \rfloor]\!]]{} & \mathsf{C}\,(\nu\mathsf{B})
\end{array}$$

There are three proof obligations.

① $\langle \nu\mathsf{B}, [\![\lfloor \mathsf{S}^{\lambda}\, \lceil out \rceil \rfloor]\!], \lceil out \rceil \rangle$ is a λ-bialgebra, in that it satisfies the pentagonal law (5.1).

$$\lceil out \rceil \cdot [\![\lfloor \mathsf{S}^{\lambda}\, \lceil out \rceil \rfloor]\!] = \mathsf{C}\,[\![\lfloor \mathsf{S}^{\lambda}\, \lceil out \rceil \rfloor]\!] \cdot \lambda\,(\nu\mathsf{B}) \cdot \mathsf{S}\,\lceil out \rceil$$

② $[\![\lfloor c \rfloor]\!]$ is a S-algebra homomorphism.

$$[\![\lfloor c \rfloor]\!] : a \to [\![\lfloor \mathsf{S}^{\lambda}\, \lceil out \rceil \rfloor]\!] : \mathsf{S}\text{-}\mathbf{Alg}$$

③ $[\![\lfloor c \rfloor]\!]$ is a C-coalgebra homomorphism.

$$[\![\lfloor c \rfloor]\!] : c \to \lceil out \rceil : \mathsf{C}\text{-}\mathbf{Coalg}$$

Again, these obligations are dispatched by the dual proofs of those in Section 8.2.

$$\begin{array}{ccc}
\mathsf{S}\,(\mu\mathsf{S}) & \xrightarrow{\qquad} & \mathsf{S}\,(\nu\mathsf{B}) \\
{\scriptstyle in}\downarrow & (\!([\![\lfloor \mathsf{S}^{\lambda}\, \lceil out \rceil \rfloor]\!])\!) & \downarrow{\scriptstyle [\![\lfloor \mathsf{S}^{\lambda}\, \lceil out \rceil \rfloor]\!]} \\
\mu\mathsf{S} & \xrightarrow[[\![\lfloor (\!|\mathsf{C}_{\lambda}\ in|\!) \rfloor]\!]]{} & \nu\mathsf{B} \\
{\scriptstyle (\!|\mathsf{C}_{\lambda}\ in|\!)}\downarrow & & \downarrow{\scriptstyle \lceil out \rceil} \\
\mathsf{C}\,(\mu\mathsf{S}) & \xrightarrow{\qquad} & \mathsf{C}\,(\nu\mathsf{B})
\end{array}$$

Theorem 10.1, like Theorems 8.1 and 9.1, continues to preserve the resolution of Iniga and Finn's viewpoints. This section has been the dual of Section 8. In Sections 8 and 9 we explored a single dimension of the design space—augmenting the syntax functor S. In this section we have made a first step in a second dimension—augmenting the behaviour functor B. Like the sections before, we now need to create a distributive law.

10.3. **Creating a Distributive Law.** A program, such as in Example 10.1, gives rise to a natural transformation $\rho : \mathsf{S}\circ\mathsf{C} \dot{\to} \mathsf{B}\circ\mathsf{S}$, where the copointed functor appears only on the left, as that is where pattern matching is confined to. From ρ we must create a distributive law $\lambda : \mathsf{S}\circ\mathsf{C} \dot{\to} \mathsf{C}\circ\mathsf{S}$ such that

$$\lceil c \rceil \cdot a = \mathsf{C}\,a \cdot \lambda\,X \cdot \mathsf{S}\,\lceil c \rceil \quad \Longleftrightarrow \quad c \cdot a = \mathsf{B}\,a \cdot \rho\,X \cdot \mathsf{S}\,\lceil c \rceil \ . \tag{10.10}$$

This is the dual situation to that of Section 8.3, where $\lambda = \mathsf{B}\circ\eta \triangledown \rho$; here this becomes $\lambda = \mathsf{S}\circ\epsilon \vartriangle \rho$, and satisfies Condition 10.3 by the dual reasoning.

11. Pointed Functors over Copointed Functors

In this section we will briefly combine pointed and copointed functors from Section 8 and 10. Let us begin with an example.

Example 11.1. The following is the definition of the stream operator that duplicates the head of a stream.

$$dup\ s@(Cons\ m\ \_) = m \prec s$$

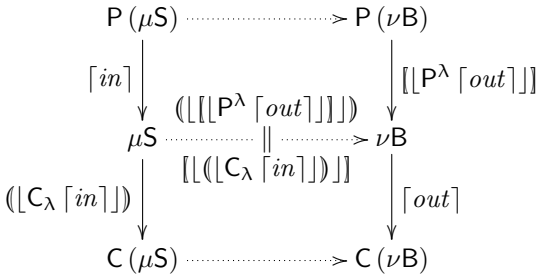
While a simple definition, it makes use of the features that (co)free (co)pointed functors introduce. Thus, the *dup* recursion equation is captured by a natural transformation:

$$\begin{aligned}&\rho :: \mathsf{S}\ (\mathsf{C}\ x) \to \mathsf{B}\ (\mathsf{P}\ x)\\ &\rho\ (Dup\ (As\ s\ (Cons\ (m, \_)))) = Cons\ (m, Var\ s) \quad \square\end{aligned}$$

Properties 8.6 and 10.6 show that B_λ preserves η and S^λ preserves ϵ. Taken together, Properties 8.5 and 10.6 say that η lifts to a (B, ϵ)-coalgebra homomorphism. Likewise, Properties 10.5 and 8.6 say that ϵ lifts to an (S, η)-algebra homomorphism. Therefore all four properties conspire to preserve the desired double isomorphism.

$$\lambda\text{-}\mathbf{Bialg}(\mathscr{C}) \cong \begin{cases}(\mathsf{S}^\lambda, \eta)\text{-}\mathbf{Alg}((\mathsf{B}, \epsilon)\text{-}\mathbf{Coalg}(\mathscr{C}))\\ (\mathsf{B}_\lambda, \epsilon)\text{-}\mathbf{Coalg}((\mathsf{S}, \eta)\text{-}\mathbf{Alg}(\mathscr{C}))\end{cases} \tag{11.1}$$

11.1. **Initial and Final Objects.** In the following diagram we can see the initial and final λ-bialgebras in our new setting where $\lambda : \mathsf{P}\circ\mathsf{C} \dot{\to} \mathsf{C}\circ\mathsf{P}$, and also the dual solutions to the unique semantic function.



It is important to note that the casting operators $\lfloor - \rfloor$ and $\lceil - \rceil$ are overloaded in this diagram. On the left-hand side of the diagram we are casting between (P, η)-algebras and S-algebras (Theorem 8.1), and on the right-hand side, (C, ϵ)-coalgebras and B-coalgebras (Theorem 10.1).

11.2. **Creating a Distributive Law.** A program, such as in Example 11.1, gives rise to a natural transformation $\rho : \mathsf{S}\circ\mathsf{C} \dot{\to} \mathsf{B}\circ\mathsf{P}$, where the copointed functor appears only the left and the pointed functor only on the right. Given ρ we must create a distributive law $\lambda : \mathsf{P}\circ\mathsf{C} \dot{\to} \mathsf{C}\circ\mathsf{P}$. We can do so by combining Sections 8.3 and 10.3. As there are two orderings by which we can combine these, there are two different formulations of λ, and thus two different intermediate steps. From $\rho : \mathsf{S}\circ\mathsf{C} \dot{\to} \mathsf{B}\circ\mathsf{P}$ we can either create $\rho^\mathsf{P} : \mathsf{P}\circ\mathsf{C} \dot{\to} \mathsf{B}\circ\mathsf{P}$, following Section 8.3, or $\rho^\mathsf{C} : \mathsf{S}\circ\mathsf{C} \dot{\to} \mathsf{C}\circ\mathsf{P}$, following Section 10.3. Let us take the former. The natural transformation ρ^P is related to ρ by the equivalence:

$$c \cdot \lceil a \rceil = \mathsf{B}\lceil a \rceil \cdot \rho^\mathsf{P}\ X \cdot \mathsf{P}\lceil c \rceil \quad \Longleftrightarrow \quad c \cdot a = \mathsf{B}\lceil a \rceil \cdot \rho\ X \cdot \mathsf{S}\lceil c \rceil\ . \tag{11.2}$$

The calculation of ρ^P is not completely identical to that in Section 8.3, as there is now an occurrence of C on the left-hand side, so there is (literally) an extra step.

Let us replay the calculation.

$$
\begin{aligned}
& c \cdot a = \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\,\lceil c \rceil \\
= \quad & \{ \text{ equality of joins } \} \\
& c \triangledown c \cdot a = c \triangledown \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\,\lceil c \rceil \\
= \quad & \{ \ (\mathsf{C}, \epsilon)\text{-}\mathbf{Coalg} \cong \mathsf{B}\text{-}\mathbf{Coalg}\colon\ c = \lfloor \lceil c \rceil \rfloor = \mathit{outr} \cdot \lceil c \rceil \ (10.8) \ \} \\
& c \triangledown c \cdot a = \mathit{outr} \cdot \lceil c \rceil \triangledown \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\,\lceil c \rceil \\
= \quad & \{ \ \mathsf{B} \text{ functor and } \lceil a \rceil : (\mathsf{P}, \eta)\text{-}\mathbf{Alg} \ (8.2) \ \} \\
& c \triangledown c \cdot a = \mathsf{B}\,\lceil a \rceil \cdot \mathsf{B}\,(\eta X) \cdot \mathit{outr} \cdot \lceil c \rceil \triangledown \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\,\lceil c \rceil \\
= \quad & \{ \text{ join fusion } (B.4) \text{ and functor fusion } (B.5) \ \} \\
& c \cdot (\mathit{id} \triangledown a) = \mathsf{B}\,\lceil a \rceil \cdot (\mathsf{B}\,(\eta X) \cdot \mathit{outr} \triangledown \rho\, X) \cdot (\lceil c \rceil + \mathsf{S}\,\lceil c \rceil) \\
= \quad & \{ \text{ definitions of } \lceil - \rceil \ (8.7) \text{ and } \mathsf{P} \ (8.6) \ \} \\
& c \cdot \lceil a \rceil = \mathsf{B}\,\lceil a \rceil \cdot (\mathsf{B}\,(\eta X) \cdot \mathit{outr} \triangledown \rho\, X) \cdot \mathsf{P}\,\lceil c \rceil
\end{aligned}
$$

The second step was the key addition; in this context, *outr* is the natural transformation $\mathsf{P} \dot{\to} \mathsf{S}$. The specification (11.2) can be satisfied if we set $\rho^{\mathsf{P}} = \mathsf{B} \circ \eta \cdot \mathit{outr} \triangledown \rho$. Dually, for a corresponding specification, $\rho^{\mathsf{C}} = \mathit{inr} \cdot \mathsf{S} \circ \epsilon \vartriangle \rho$. Continuing from ρ^{P}, we can construct $\lambda = \mathsf{P} \circ \epsilon \vartriangle (\mathsf{B} \circ \eta \cdot \mathit{outr} \triangledown \rho)$, following Section 10.3 with P in place of S, or $\lambda = \mathsf{C} \circ \eta \triangledown (\mathit{inr} \cdot \mathsf{S} \circ \epsilon \vartriangle \rho)$ from ρ^{C}, following Section 8.3 with C in place of B.

A distributive law $\lambda : \mathsf{P} \circ \mathsf{C} \dot{\to} \mathsf{C} \circ \mathsf{P}$ must satisfy Conditions 8.3 and 10.3. Let us take the first construction $\lambda = \mathsf{P} \circ \epsilon \vartriangle (\mathsf{B} \circ \eta \cdot \mathit{outr} \triangledown \rho)$ and show that Condition 8.3 is satisfied.

$$
\begin{aligned}
& (\mathsf{P} \circ \epsilon \vartriangle (\mathsf{B} \circ \eta \cdot \mathit{outr} \triangledown \rho)) \cdot \eta \circ \mathsf{C} \\
= \quad & \{ \text{ split fusion } (A.4) \ \} \\
& \mathsf{P} \circ \epsilon \cdot \eta \circ \mathsf{C} \vartriangle (\mathsf{B} \circ \eta \cdot \mathit{outr} \triangledown \rho) \cdot \eta \circ \mathsf{C} \\
= \quad & \{ \ \eta = \mathit{inl} \text{ and join computation } (B.3a) \ \} \\
& \mathsf{P} \circ \epsilon \cdot \eta \circ \mathsf{C} \vartriangle \mathsf{B} \circ \eta \cdot \mathit{outr} \\
= \quad & \{ \text{ definition of } \mathsf{P} \ (8.6) \ \} \\
& (\epsilon + \mathsf{S} \circ \epsilon) \cdot \eta \circ \mathsf{C} \vartriangle \mathsf{B} \circ \eta \cdot \mathit{outr} \\
= \quad & \{ \ \eta = \mathit{inl} \text{ and } \mathit{inl} \text{ is natural } (B.6a) \ \} \\
& \eta \cdot \epsilon \vartriangle \mathsf{B} \circ \eta \cdot \mathit{outr} \\
= \quad & \{ \ \epsilon = \mathit{outl} \text{ and split functor fusion } (A.5) \ \} \\
& (\eta \times \mathsf{B} \circ \eta) \cdot (\mathit{outl} \vartriangle \mathit{outr}) \\
= \quad & \{ \text{ definition of } \mathsf{C} \ (10.6) \text{ and split reflection } (A.2) \ \} \\
& \mathsf{C} \circ \eta
\end{aligned}
$$

Condition 10.3 follows immediately from split computation (A.3a). Dually, for the construction $\lambda = \mathsf{C} \circ \eta \triangledown (\mathit{inr} \cdot \mathsf{S} \circ \epsilon \vartriangle \rho)$, Condition 8.3 is trivially satisfied and the dual of the proof above satisfies Condition 10.3.

12. Monads over Copointed Functors

In Section 9 we saw that we could model recursion equations that needed full terms on the right-hand side, and using the technique in Section 9.5, we could precisely model equations that "consume at most one, and produce at least one". This was in the case of a distributive law of a monad over a plain endofunctor. The nature of "consuming at most one" can be made more explicit by the use of

copointed functors, from Section 10. In this section we will combine monads and copointed functors from Sections 9 and 10. Let us begin with two examples.

Example 12.1. The Thue–Morse sequence, or Prouhet–Thue–Morse sequence, is, like π, a ubiquitous mathematical object [Allouche and Jeffery, 1999]. It is a binary sequence, the first 15 digits of which are:

$$t = 0\,1\,1\,0\,1\,0\,0\,1\,1\,0\,0\,1\,0\,1\,1\ldots$$

The n^{th} element of the sequence can be computed directly. First, n is written in binary; if there are an odd number of ones in the binary representation, then $t_n = 1$, otherwise $t_n = 0$. The sequence can also be expressed as a recurrence relation,

$$t_0 = 0, \quad t_{2n+1} = 1 - t_n, \quad t_{2n+2} = t_{n+1} \ .$$

Hinze [2010] shows that a recurrence of the form $t_0 = k$, $t_{2n+1} = f(t_n)$ and $t_{2n+2} = g(t_n)$ corresponds to the stream $t = k \prec \mathit{interleave}\ (\mathit{map}\ f\ t)\ (\mathit{map}\ g\ t)$. (The definition of *interleave* is given in Example 10.1.) The clause $t_{2n+2} = t_{n+1}$ in the definition does not immediately match this pattern, however, this can be overcome by using *tail* to transform t_n into t_{n+1}. Therefore, we can give the following stream as a specification of the Thue–Morse sequence,

$$\mathit{thue} = 0 \prec \mathit{interleave}\ (\mathit{inv}\ \mathit{thue})\ (\mathit{tail}\ \mathit{thue}) \ ,$$

where *inv* is the specialization of *map* $(1-)$,

$$\mathit{inv}\ (\mathit{Cons}\ m\ s) = (1 - m) \prec \mathit{inv}\ s \ .$$

The occurrence is of *tail* is a problem as it is a stream function that consumes one element but does not produce any. If we unroll the definitions, we can mechanically derive an implementation that is in a compliant form.

$$\begin{aligned} \mathit{thue} &= 0 \prec \mathit{thue}' \\ \mathit{thue}' &= 1 \prec \mathit{interleave}\ \mathit{thue}'\ (\mathit{inv}\ \mathit{thue}') \end{aligned}$$

Now we can capture the recursion equations *interleave*, *inv*, *thue* and thue' by a natural transformation:

$$\begin{aligned} &\rho :: \mathsf{S}\ (\mathsf{C}\ x) \to \mathsf{B}\ (\mathsf{M}\ x) \\ &\rho\ (\mathit{Interleave}\ (\mathit{As}\ \_\ (\mathit{Cons}\ (m, s)), \mathit{As}\ t\ \_)) \\ &\quad = \mathit{Cons}\ (m, \mathit{Com}\ (\mathit{Interleave}\ (\mathit{Var}\ t, \mathit{Var}\ s))) \\ &\rho\ (\mathit{Inv}\ (\mathit{As}\ \_\ (\mathit{Cons}\ (m, s)))) \\ &\quad = \mathit{Cons}\ (1 - m)\ (\mathit{Com}\ (\mathit{Inv}\ (\mathit{Var}\ s))) \\ &\rho\ \mathit{Thue} \\ &\quad = \mathit{Cons}\ 0\ (\mathit{Com}\ \mathit{Thue}') \\ &\rho\ \mathit{Thue}' \\ &\quad = \mathit{Cons}\ 1\ (\mathit{Com}\ (\mathit{Interleave}\ (\mathit{Com}\ \mathit{Thue}', \\ &\qquad\qquad\qquad\qquad \mathit{Com}\ (\mathit{Inv}\ (\mathit{Com}\ \mathit{Thue}'))))) \end{aligned}$$

Example 12.2. The *regular numbers* are the numbers that evenly divide powers of 60, and can be characterized as the numbers that only have 2, 3 or 5 as prime factors. They are also called the Hamming numbers, after the Turing award winner Richard Hamming, who posed the problem of generating these numbers in ascending order. Dijkstra [1981] presented a solution à la SASL, attributed to J. L. A. van de Snepscheut, and proved its correctness. Here we will replicate the same solution, which is in fact a slightly simplified version for numbers that only have 2 and 3 as prime factors. The stream is,

$$\mathit{ham} = 1 \prec \mathit{merge}\ (\mathit{times}\ 2\ \mathit{ham}, \mathit{times}\ 3\ \mathit{ham}) \ ,$$

where the definition of *merge* is given in Example 10.1 and *times* is,

$$\mathit{times}\ n\ (\mathit{Cons}\ m\ s) = n \times m \prec \mathit{times}\ n\ s \ .$$

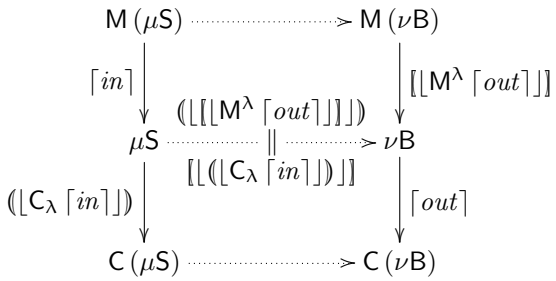
Again, we can capture the recursion equations *merge*, *times* and *ham* by a natural transformation:

$$
\begin{aligned}
&\rho :: \mathsf{S}\ (\mathsf{C}\ x) \to \mathsf{B}\ (\mathsf{M}\ x)\\
&\rho\ (\mathit{Merge}\ (\mathit{As}\ s\ (\mathit{Cons}\ (m, s')), \mathit{As}\ t\ (\mathit{Cons}\ (n, t'))))\\
&\quad = \mathit{Cons}\ (\mathbf{if}\ m \leqslant n\ \mathbf{then}\ m\ \mathbf{else}\ n,\\
&\qquad\qquad \mathit{Com}\ (\mathit{Merge}\ (\mathit{Var}\ (\mathbf{if}\ m \leqslant n\ \mathbf{then}\ s'\ \mathbf{else}\ s),\\
&\qquad\qquad\qquad\qquad \mathit{Var}\ (\mathbf{if}\ m \leqslant n\ \mathbf{then}\ t\ \ \mathbf{else}\ t'))))\\
&\rho\ (\mathit{Times}\ n\ (\mathit{As}\ \_\ (\mathit{Cons}\ (m, s))))\\
&\quad = \mathit{Cons}\ (n \times m, \mathit{Com}\ (\mathit{Times}\ n\ (\mathit{Var}\ s)))\\
&\rho\ \mathit{Ham}\\
&\quad = \mathit{Cons}\ (1, \mathit{Com}\ (\mathit{Merge}\ (\mathit{Com}\ (\mathit{Times}\ 2\ (\mathit{Com}\ \mathit{Ham})),\\
&\qquad\qquad\qquad\qquad \mathit{Com}\ (\mathit{Times}\ 3\ (\mathit{Com}\ \mathit{Ham})))))\quad \square
\end{aligned}
$$

Properties 8.6 and 9.6 show that B_λ preserves η and μ and Property 10.6 shows that S^λ preserves ϵ. Taken together, Properties 8.5, 9.5 and 10.6 say that η and μ lift to (B, ϵ)-coalgebra homomorphisms. Likewise, Properties 10.5, 8.6 and 9.6 say that ϵ lifts to an (S, η, μ)-algebra homomorphism. Therefore all six properties conspire to preserve the desired double isomorphism.

$$
\lambda\text{-}\mathbf{Bialg}(\mathscr{C}) \cong \begin{cases} (\mathsf{S}^\lambda, \eta, \mu)\text{-}\mathbf{Alg}((\mathsf{B}, \epsilon)\text{-}\mathbf{Coalg}(\mathscr{C})) \\ (\mathsf{B}_\lambda, \epsilon)\text{-}\mathbf{Coalg}((\mathsf{S}, \eta, \mu)\text{-}\mathbf{Alg}(\mathscr{C})) \end{cases} \tag{12.1}
$$

12.1. **Initial and Final Objects.** The following diagram is only a minor revision of that in Section 11.1. In it we can see the initial and final λ-bialgebras in the setting of monads over copointed functors, and also the dual solutions to the unique semantic function.



The diagram has only 'syntactically' changed by replacing P with M, however, the distributive law is now $\lambda : \mathsf{M}\circ\mathsf{C} \to \mathsf{C}\circ\mathsf{M}$, and the casting operators $\lfloor - \rfloor$ and $\lceil - \rceil$ are again overloaded. On the left-hand side of the diagram we are casting between (M, η, μ)-algebras and S-algebras (Theorem 9.1), and on the right-hand side, (C, ϵ)-coalgebras and B-coalgebras (Theorem 10.1).

12.2. **Creating a Distributive Law.** A distributive law that comprises of a monad over a copointed functor is a sweet spot in the design space. Given a natural transformation $\rho : \mathsf{S}\circ\mathsf{C} \to \mathsf{B}\circ\mathsf{M}$, such as from from Examples 12.1 and 12.2, we can create a distributive law $\lambda : \mathsf{M}\circ\mathsf{C} \to \mathsf{C}\circ\mathsf{M}$. Just as in Section 11.2, an intermediate step is to turn ρ into either $\rho^{\mathsf{M}} : \mathsf{M}\circ\mathsf{C} \to \mathsf{B}\circ\mathsf{M}$ or $\rho^{\mathsf{C}} : \mathsf{S}\circ\mathsf{C} \to \mathsf{C}\circ\mathsf{M}$. In this case we will choose the latter, and ρ^{C} is related to ρ by the equivalence:

$$
\lceil c \rceil \cdot a = \mathsf{C}\,\lceil a \rceil \cdot \rho^{\mathsf{C}}\, X \cdot \mathsf{S}\,\lceil c \rceil \iff c \cdot a = \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\,\lceil c \rceil\ . \tag{12.2}
$$

The calculation of ρ^{C} is the dual of the calculation of ρ^{P} in Section 11.2, and the extension of the calculation omitted in Section 10.3. We will include it here for

completeness.

$$
\begin{aligned}
& c \cdot a = \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\,\lceil c \rceil \\
\iff & \quad \{ \text{ equality of splits } \} \\
& a \vartriangle c \cdot a = a \vartriangle \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\,\lceil c \rceil \\
\iff & \quad \{ \ (\mathsf{M}, \mu, \eta)\text{-}\mathbf{Alg} \cong \mathsf{S}\text{-}\mathbf{Alg}\text{: } a = \lfloor \lceil a \rceil \rfloor = \lceil a \rceil \cdot \theta X \ (9.10) \ \} \\
& a \vartriangle c \cdot a = \lceil a \rceil \cdot \theta X \vartriangle \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\,\lceil c \rceil \\
\iff & \quad \{ \ \mathsf{S} \text{ functor and } \lceil c \rceil : (\mathsf{C}, \epsilon)\text{-}\mathbf{Coalg} \ (10.2) \ \} \\
& a \vartriangle c \cdot a = \lceil a \rceil \cdot \theta X \cdot \mathsf{S}\,(\epsilon X) \cdot \mathsf{S}\,\lceil c \rceil \vartriangle \mathsf{B}\,\lceil a \rceil \cdot \rho\, X \cdot \mathsf{S}\,\lceil c \rceil \\
\iff & \quad \{ \text{ split fusion } (A.4) \text{ and functor fusion } (A.5) \ \} \\
& (id \vartriangle c) \cdot a = (\lceil a \rceil \times \mathsf{B}\,\lceil a \rceil) \cdot (\theta X \cdot \mathsf{S}\,(\epsilon X) \vartriangle \rho\, X) \cdot \mathsf{S}\,\lceil c \rceil \\
\iff & \quad \{ \text{ definitions of } \lceil - \rceil \ (10.7) \text{ and } \mathsf{C} \ (10.6) \ \} \\
& \lceil c \rceil \cdot a = \mathsf{C}\,\lceil a \rceil \cdot (\theta X \cdot \mathsf{S}\,(\epsilon X) \vartriangle \rho\, X) \cdot \mathsf{S}\,\lceil c \rceil
\end{aligned}
$$

Therefore, the specification (12.2) can be satisfied if we set $\rho^{\mathsf{C}} = \theta \cdot \mathsf{S} \circ \epsilon \vartriangle \rho$.

Now that we have a natural transformation $\rho^{\mathsf{C}} : \mathsf{S} \circ \mathsf{C} \to \mathsf{C} \circ \mathsf{M}$ from $\rho : \mathsf{S} \circ \mathsf{C} \to \mathsf{B} \circ \mathsf{M}$, we will continue and create the distributive law $\lambda : \mathsf{M} \circ \mathsf{C} \to \mathsf{C} \circ \mathsf{M}$ by following the template of Section 9.3. First we define a lifting of C to $\mathsf{C}_\rho : (\mathsf{M}, \eta, \mu)\text{-}\mathbf{Alg} \to (\mathsf{M}, \eta, \mu)\text{-}\mathbf{Alg}$,

$$\mathsf{C}_\rho\, a = \mathbf{(}\mathsf{C}\, a \cdot \rho^{\mathsf{C}}\, A\mathbf{)} \ , \tag{12.3}$$

and then with this, the distributive law,

$$\lambda = \mathsf{C}_\rho\, \mu \cdot \mathsf{M} \circ \mathsf{C} \circ \eta \ . \tag{12.4}$$

As a result of following the construction of Section 9.3, λ satisfies Condition 9.3 and $\mathsf{C}_\rho\, \mu$ satisfies Condition 9.4, that is, λ coheres with η and μ, and $\mathsf{C}_\rho\, \mu$ is an M-algebra that respects η and μ.

As λ contains the copointed functor C, it must also cohere with ϵ, Condition 10.3. Before we get to the proof, we will first show that C_ρ is copointed (cf. Property 10.5). Let $a : \mathsf{M}\, X \to X$ be an M-algebra, then $\epsilon X : \mathsf{C}_\rho\, a \to a : \mathsf{M}\text{-}\mathbf{Alg}$ is the lifting of ϵ to an M-algebra homomorphism.

Proof.

$$
\begin{aligned}
& \epsilon X \cdot \mathsf{C}_\rho\, a = a \cdot \mathsf{M}\,(\epsilon X) \\
\iff & \quad \{ \ (\mathsf{M}, \eta, \mu)\text{-}\mathbf{Alg} \cong \mathsf{S}\text{-}\mathbf{Alg} \ (9.11) \ \} \\
& \epsilon X \cdot \lfloor \mathsf{C}_\rho\, a \rfloor = \lfloor a \rfloor \cdot \mathsf{S}\,(\epsilon X) \\
\iff & \quad \{ \text{ definitions of } \mathsf{C}_\rho \ (12.3) \text{ and } \lfloor - \rfloor \ (9.10) \ \} \\
& \epsilon X \cdot \mathsf{C}\, a \cdot \rho^{\mathsf{C}}\, X = a \cdot \theta X \cdot \mathsf{S}\,(\epsilon X) \\
\iff & \quad \{ \ \epsilon : \mathsf{C} \dot{\to} \mathsf{Id} \text{ is natural } \} \\
& a \cdot \epsilon(\mathsf{M}\, X) \cdot \rho^{\mathsf{C}}\, X = a \cdot \theta X \cdot \mathsf{S}\,(\epsilon X) \\
\iff & \quad \{ \ \epsilon = \mathit{outl}, \text{ definition of } \rho^{\mathsf{C}} \text{ and split computation } (A.3a) \ \} \\
& a \cdot \theta X \cdot \mathsf{S}\,(\epsilon X) = a \cdot \theta X \cdot \mathsf{S}\,(\epsilon X) \quad \square
\end{aligned}
$$

If we instantiate this property with the M-algebra $\langle \mathsf{M}\, X, \mu X \rangle$, we have:

$$\epsilon(\mathsf{M}\, X) \cdot \mathsf{C}_\rho\, (\mu X) = \mu X \cdot \mathsf{M}\,(\epsilon(\mathsf{M}\, X)) \ . \tag{12.5}$$

We are now in a position to prove that λ satisfies Condition 10.3.

Proof.

$$
\begin{aligned}
& \epsilon(\mathsf{M}\,X) \cdot \lambda\,X \\
= & \quad \{ \text{ definition of } \lambda \text{ (12.4) } \} \\
& \epsilon(\mathsf{M}\,X) \cdot \mathsf{C}_\rho\,(\mu X) \cdot \mathsf{M}\,(\mathsf{C}\,(\eta X)) \\
= & \quad \{ \ \epsilon(\mathsf{M}\,X) \text{ is an } \mathsf{M}\text{-homomorphism (12.5) } \} \\
& \mu X \cdot \mathsf{M}\,(\epsilon(\mathsf{M}\,X)) \cdot \mathsf{M}\,(\mathsf{C}\,(\eta X)) \\
= & \quad \{ \ \mathsf{M} \text{ functor and } \epsilon : \mathsf{C} \dot{\to} \mathsf{Id} \text{ is natural } \} \\
& \mu X \cdot \mathsf{M}\,(\eta X) \cdot \mathsf{M}\,(\epsilon X) \\
= & \quad \{ \text{ second monad law (9.1b) } \} \\
& \mathsf{M}\,(\epsilon X) \quad \square
\end{aligned}
$$

12.3. **Unique Fixed-Point Principle.** Let us briefly revisit the results of Section 9.6, where recursion equations were modelled by a natural transformation $\rho : \mathsf{S} \circ \mathsf{B} \to \mathsf{B} \circ \mathsf{M}$. Our natural transformation is now $\rho : \mathsf{S} \circ \mathsf{B} \to \mathsf{B} \circ \mathsf{M}$. A solution of a system modelled by ρ consists of an S-algebra and a B-coalgebra over a common carrier X that satisfies:

$$c \cdot a = \mathsf{B}\,\lceil a \rceil \cdot \rho\,X \cdot \mathsf{S}\,\lceil c \rceil$$

If the coalgebra is final, then a is uniquely determined. The following calculation gives that solution and establishes the UFP.

$$
\begin{aligned}
& \mathit{out} \cdot a = \mathsf{B}\,\lceil a \rceil \cdot \rho\,(\nu \mathsf{B}) \cdot \mathsf{S}\,\lceil \mathit{out} \rceil \\
\iff & \quad \{ \ \rho^{\mathsf{C}} \text{ which satsifies (12.2) } \} \\
& \lceil \mathit{out} \rceil \cdot a = \mathsf{C}\,\lceil a \rceil \cdot \rho^{\mathsf{C}}\,X \cdot \mathsf{S}\,\lceil \mathit{out} \rceil \\
\iff & \quad \{ \ \lambda \text{ given by (12.4) which satisfies (9.12) } \} \\
& \lceil \mathit{out} \rceil \cdot \lceil a \rceil = \mathsf{C}\,\lceil a \rceil \cdot \lambda\,(\nu \mathsf{B}) \cdot \mathsf{M}\,\lceil \mathit{out} \rceil \\
\iff & \quad \{ \text{ definition of } \mathsf{M}^\lambda \text{ (7.4) } \} \\
& \lceil \mathit{out} \rceil \cdot \lceil a \rceil = \mathsf{C}\,\lceil a \rceil \cdot \mathsf{M}^\lambda\,\lceil \mathit{out} \rceil \\
\iff & \quad \{ \ (\mathsf{C}, \epsilon)\textbf{-Coalg} \cong \mathsf{B}\textbf{-Coalg (10.9) } \} \\
& \mathit{out} \cdot \lceil a \rceil = \mathsf{B}\,\lceil a \rceil \cdot \lfloor \mathsf{M}^\lambda\,\lceil \mathit{out} \rceil \rfloor \\
\iff & \quad \{ \text{ uniqueness of unfold (3.2) } \} \\
& \lceil a \rceil = [\![\lfloor \mathsf{M}^\lambda\,\lceil \mathit{out} \rceil \rfloor]\!] \\
\iff & \quad \{ \ (\mathsf{M}, \eta, \mu)\textbf{-Alg} \cong \mathsf{S}\textbf{-Alg (9.11) } \} \\
& a = \lfloor [\![\lfloor \mathsf{M}^\lambda\,\lceil \mathit{out} \rceil \rfloor]\!] \rfloor
\end{aligned}
$$

Conversely, if the algebra is initial, then c is uniquely determined: $c = \lfloor (\!| \lfloor \mathsf{C}_\lambda\,\lceil \mathit{in} \rceil \rfloor |\!) \rfloor$. These make up the initial and final objects in $\lambda\textbf{-Bialg}(\mathscr{C})$, as seen in Section 12.1.

13. Endofunctors over Comonads

In Section 10, with the introduction of copointed functors, we saw how to model recursion equations that consume either one level of behaviour or none at all. This nicely captured stream functions such as *interleave* and *merge* (Example 10.1). In this section we will augment copointed functors to comonads, in the same way that we did for pointed functors to monads in Section 9.

Example 13.1. Consider the stream functions *even* and *odd*, which preserve only the even and odd indexed elements of a stream, respectively.

$$\begin{aligned} even\ (Cons\ m\ (Cons\ \_\ s)) &= m \prec even\ s \\ odd\ \ (Cons\ \_\ \ (Cons\ n\ s)) &= n\ \prec odd\ \ s \end{aligned}$$

Here we have nested pattern matching on the left-hand side of the equations. In both cases we need to examine two levels of behaviour: so that one can be kept and the other discarded. We can model this nested pattern matching in our categorical setting with *comonads*.

$$\mathbf{data}\ \mathsf{N}\ x = Root\ x\ (\mathsf{B}\ (\mathsf{N}\ x))$$

We can read $Root\ s\ b$ as $s@b$—a Haskell as-pattern—where b may contain further patterns. If we use N to replace B on the left-hand side, then we are able to construct a natural transformation to model *even* and *odd*.

$$\begin{aligned} &\rho :: \mathsf{S}\ (\mathsf{N}\ x) \to \mathsf{B}\ (\mathsf{S}\ x) \\ &\rho\ (Even\ (Root\ \_\ (Cons\ (m, Root\ \_\ (Cons\ (\_, Root\ s\ \_))))))) \\ &\quad = Cons\ (m, Even\ s) \\ &\rho\ (Odd\ \ (Root\ \_\ (Cons\ (\_,\ \ Root\ \_\ (Cons\ (n, Root\ s\ \_))))))) \\ &\quad = Cons\ (n,\ \ Odd\ \ s) \end{aligned}$$

These are examples of a sampling functions [Niqui and Rutten, 2010], functions that consume more than they produce. Another example from Niqui and Rutten is the family of all drop operators.

Example 13.2. For $l \geq 2$ and $0 \leq i < l$, the family of drop operators,

$$drop_l^i :: \mathsf{Stream}\ x \to \mathsf{Stream}\ x\ \ ,$$

drops the i-th element from each input block of size l. The *even* and *odd* operators are members of this family: $even = drop_2^1$ and $odd = drop_2^0$. Niqui and Rutten give the following definition.

$$\begin{aligned} head\ (drop_l^{i+1} s) &= head\ s \\ tail\ \ (drop_l^{i+1} s) &= drop_l^i (tail\ s) \\ head\ (drop_l^0\ \ s) &= head\ (tail\ s) \\ tail\ \ (drop_l^0\ \ s) &= drop_l^{l-2} (tail\ (tail\ s)) \end{aligned}$$

This is simple to translate into our definitional style.

$$\begin{aligned} drop_l^{i+1} (Cons\ m\ s) &= m \prec drop_l^i\ \ s \\ drop_l^0\ \ (Cons\ \_\ (Cons\ n\ s)) &= n\ \prec drop_l^{l-2} s \end{aligned}$$

And as per usual, this is straightforward to turn into a natural transformation.

$$\begin{aligned} &\rho :: \mathsf{S}\ (\mathsf{N}\ x) \to \mathsf{B}\ (\mathsf{S}\ x) \\ &\rho\ (Drop_l^{i+1} (Root\ \_\ (Cons\ (m, Root\ s\ \_)))) \\ &\quad = Cons\ (m, Drop_l^i\ \ s) \\ &\rho\ (Drop_l^0\ \ (Root\ \_\ (Cons\ (\_, Root\ \_\ (Cons\ (n, Root\ s\ \_)))))) \\ &\quad = Cons\ (n,\ \ Drop_l^{l-2} s) \end{aligned}$$

Example 13.3. Another informative example is the *finite difference* or *forward difference* stream operator [Hinze, 2010]. It has a simple, non-recursive definition.

$$diff\ \ s = tail\ s - s$$

If we are to express this as a natural transformation, we need to draw out the head of the resulting stream. Rather than letting the lifted subtraction do the work, we will turn *diff* into a recursive definition.

$$diff\ \ (Cons\ m\ s@(Cons\ n\ \_)) = n - m \prec diff\ \ s$$

The use of the inner as-pattern makes the 'lookahead' nature of *diff* clear. The names m and s are bound to the head and tail of the input stream, and n is bound to the head of the tail s. This inspection of the input stream is unlike any of the previous examples that we have seen. The natural transformation now follows immediately:

$$\begin{aligned}&\rho :: \mathsf{S}\ (\mathsf{N}\ x) \to \mathsf{B}\ (\mathsf{S}\ x)\\ &\rho\ (\mathit{Diff}\ (\mathit{Root}\ \_\ (\mathit{Cons}\ (m, \mathit{Root}\ s\ (\mathit{Cons}\ (n, \_))))))\\ &\quad = \mathit{Cons}\ (n - m, \mathit{Diff}\ s) \qquad \square\end{aligned}$$

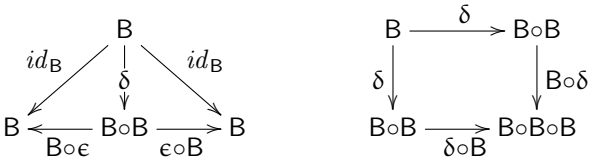
The Haskell type N is the so-called *cofree comonad* of B. We will discuss comonads in general and then return to the cofree construction in Section 13.1.

Definition 13.4. We say that B is a *comonad* if it is equipped with natural transformations $\epsilon : \mathsf{B} \dot{\to} \mathsf{Id}$ and $\delta : \mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{B}$ such that

$$\mathsf{B}\circ\epsilon \cdot \delta = id_{\mathsf{B}} \ , \tag{13.1a}$$

$$\epsilon\circ\mathsf{B} \cdot \delta = id_{\mathsf{B}} \ , \tag{13.1b}$$

$$\mathsf{B}\circ\delta \cdot \delta = \delta\circ\mathsf{B} \cdot \delta \ . \tag{13.1c}$$

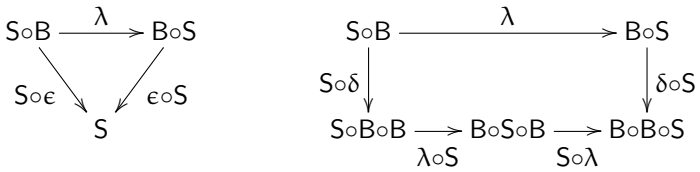


A comonad extends a copointed functor with a second natural transformation $\delta : \mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{B}$. When we introduced copointed functors in Section 10, we saw that $\epsilon : \mathsf{B} \dot{\to} \mathsf{Id}$ must be respected when constructing coalgebras and also by the distributive law of the λ-bialgebra; these same conditions extend to δ.

Condition 13.5. The following are the necessary coherence properties for a distributive law $\lambda : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}$:

$$\epsilon\circ\mathsf{S} \cdot \lambda = \mathsf{S}\circ\epsilon \ , \tag{13.2a}$$

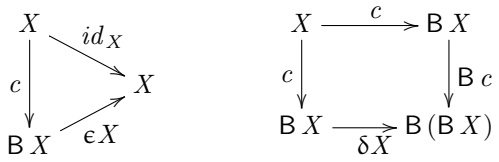
$$\delta\circ\mathsf{S} \cdot \lambda = \mathsf{S}\circ\lambda \cdot \lambda\circ\mathsf{S} \cdot \mathsf{S}\circ\delta \ . \tag{13.2b}$$



Condition 13.6. If we construct a coalgebra of comonad, then it must respect ϵ and δ. Let $\langle X, c : X \to \mathsf{B}\ X\rangle$ be an B-coalgebra, then

$$\epsilon X \cdot c = id_X \ , \tag{13.3a}$$

$$\delta X \cdot c = \mathsf{B}\ c \cdot c \ . \tag{13.3b}$$



Property 13.7. In Section 10 we showed that ϵ can be lifted to an S-algebra homomorphism (10.3). There is an analogous property for δ. Let $a : \mathsf{S}\,X \to X$ be an S-algebra, then

$$\delta X : \mathsf{B}_\lambda\ a \to \mathsf{B}_\lambda\,(\mathsf{B}_\lambda\ a) : \mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C})\ , \tag{13.4}$$

is the lifting of δ to an S-algebra homomorphism. In other words, B_λ is a comonad and we can form $(\mathsf{B}_\lambda, \epsilon, \delta)$-$\mathbf{Coalg}(\mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C}))$.

Property 13.8. We have shown previously that S^λ preserves composition, the identity, homomorphisms (7.6), and ϵ (10.4). Now, S^λ as an endo functor on $(\mathsf{B}, \epsilon, \delta)$-$\mathbf{Coalg}(\mathscr{C})$ must preserve δ.

$$\delta(\mathsf{S}\,X) \cdot \mathsf{S}^\lambda\ c = \mathsf{B}\,(\mathsf{S}^\lambda\ c) \cdot \mathsf{S}^\lambda\ c \quad \Longleftarrow \quad \delta X \cdot c = \mathsf{B}\,c \cdot c \tag{13.5}$$

In other words, we can form S^λ-$\mathbf{Alg}((\mathsf{B}, \epsilon, \delta)$-$\mathbf{Coalg}(\mathscr{C}))$.

Summary.

$$\lambda\text{-}\mathbf{Bialg}(\mathscr{C}) \cong \begin{cases} \mathsf{S}^\lambda\text{-}\mathbf{Alg}((\mathsf{B}, \epsilon, \delta)\text{-}\mathbf{Coalg}(\mathscr{C})) \\ (\mathsf{B}_\lambda, \epsilon, \delta)\text{-}\mathbf{Coalg}(\mathsf{S}\text{-}\mathbf{Alg}(\mathscr{C})) \end{cases}$$

13.1. **Cofree Comonad.** Let $\mathsf{B} : \mathscr{C} \to \mathscr{C}$ be an endofunctor representing our behaviour. There is a canonical comonad, with pleasant properties, that we can construct from B. To do so we will first describe the *cofree* B-coalgebra.

The cofree B-coalgebra over X is $\langle \mathsf{N}\,X, \mathit{des}\rangle$ equipped with an arrow $\mathit{lab} : \mathsf{N} \dot{\to} \mathsf{Id}$. We think of elements of $\mathsf{N}\,X$ as trees built from our behaviour functor B, where each node of a tree is labelled with an element of X. There are two ways to destruct a behaviour tree: the natural transformation *lab* extracts the (root) label of the tree; and $\mathit{des} : \mathsf{N} \dot{\to} \mathsf{B}\circ\mathsf{N}$ destructs a tree into one level of behaviour over subtrees—a B structure of behaviour trees.

If we have a coalgebra $c : X \to \mathsf{B}\,X$, we can build a behaviour tree $\mathsf{N}\,X$ with $\llbracket c \rrbracket : X \to \mathsf{N}\,X$ (pronounced "trace"). Given an arrow $f : X \to Y$, a relabelling, and a B-coalgebra c to build behaviour, a building of trees is characterized by the uniqueness property,

$$\mathsf{N}\,f \cdot \llbracket c \rrbracket = g \quad \Longleftrightarrow \quad f = \mathit{lab}\ Y \cdot g \ \wedge\ \mathsf{B}\,g \cdot c = \mathit{des}\ Y \cdot g\ , \tag{13.6}$$

for all tree builders $g : X \to \mathsf{N}\ Y$. The equivalence states that a compositional building of a tree, second conjunct, is uniquely defined by a relabelling, first conjunct. (For the clued-in reader, all of this information comes from the adjunction of the free and forgetful functors between B-$\mathbf{Coalg}(\mathscr{C})$ and $\mathscr{C}$.)

There are two simple consequences of the uniqueness property. If we use the identity as a relabelling ($f = \mathit{id}$), we get the computation laws:

$$\mathit{lab}\ X \cdot \llbracket c \rrbracket = \mathit{id}\ , \tag{13.7a}$$

$$\mathit{des}\ X \cdot \llbracket c \rrbracket = \mathsf{B}\,\llbracket c \rrbracket \cdot c\ . \tag{13.7b}$$

As *lab* and *des* are the destructors of trees, we can read these as defining equations of $\llbracket - \rrbracket$. The uniqueness property also implies that *lab* and *des* are natural in X and that $\llbracket - \rrbracket$ preserves naturality.

$$\llbracket \alpha \rrbracket : \mathsf{G} \dot{\to} \mathsf{N}\circ\mathsf{G} \quad \Longleftarrow \quad \alpha : \mathsf{G} \dot{\to} \mathsf{B}\circ\mathsf{G}$$

(Here $\llbracket \alpha \rrbracket$ is shorthand for $\phi\,A = \llbracket \alpha\,A \rrbracket$.) The proof is the dual of the proof that $(\!|-|\!)$ preserves naturality in Section 9.1.

Definition 13.9. The cofree comonad is $\langle \mathsf{N}, \epsilon, \delta\rangle$, where $\epsilon = \mathit{lab}$ and $\delta = \llbracket \mathit{des} \rrbracket$.

The $\delta : \mathsf{N} \dot{\to} \mathsf{N}\circ\mathsf{N}$ of the comonad relabels every subtree with itself. It does so by building a tree using *des*, the action of the cofree comonad.

Theorem 13.1. *The category of coalgebras for the cofree comonad of* B *is isomorphic to the category of* B*-coalgebras:*

$$(\mathsf{N}, \epsilon, \delta)\text{-}\mathbf{Coalg} \cong \mathsf{B}\text{-}\mathbf{Coalg} \ .$$

The following definitions are the witnesses to this isomorphism.

$$\begin{aligned} \lceil \langle X, c : X \to \mathsf{B}\, X \rangle \rceil &= \langle X, [\![a]\!] : X \to \mathsf{N}\, X \rangle & \lceil h \rceil &= h \\ \lfloor \langle X, d : X \to \mathsf{N}\, X \rangle \rfloor &= \langle X, \phi X \cdot d : X \to \mathsf{B}\, X \rangle & \lfloor h \rfloor &= h \ , \end{aligned}$$

where $\phi = \mathsf{B} \circ \epsilon \cdot \mathit{des} : \mathsf{N} \dot{\to} \mathsf{B}$*, which destructs a behaviour tree into a level of behaviour, the arguments of which are the labels of the subtrees. Furthermore,* $\lceil - \rceil$ *and* $\lfloor - \rfloor$ *are functorial.*

$$h : \lceil c \rceil \to \lceil d \rceil : (\mathsf{N}, \epsilon, \delta)\text{-}\mathbf{Coalg} \iff h : c \to d : \mathsf{B}\text{-}\mathbf{Coalg} \tag{13.8}$$

Proof Outline. The proof is the dual of that for Theorem 9.1. We will simply highlight the four components of the proof.

(i) $\lfloor \lceil \langle X, c \rangle \rceil \rfloor = \langle X, c \rangle$ and $\lceil \lfloor \langle X, d \rangle \rfloor \rceil = \langle X, d \rangle$.
(ii) $\lfloor - \rfloor$ is functorial—it maps N-homomorphisms to B-homomorphisms—as $\phi : \mathsf{N} \dot{\to} \mathsf{B}$ is natural and $\lfloor - \rfloor = \phi\text{-}\mathbf{Coalg}$ (cf. Section 3.3).
(iii) $\lceil - \rceil$ maps B-homomorphisms to N-homomorphisms.
(iv) $\lceil \langle X, c \rangle \rceil$ is a coalgebra from the comonad (13.3a and 13.3b). □

Summary. Taking Theorems 10.1 and 13.1 together, $\lfloor - \rfloor$ $\lceil - \rceil$ are now casting operators between $(\mathsf{N}, \epsilon, \delta)$-coalgebras, (C, ϵ)-coalgebras and B-coalgebras—all three categories are isomorphic.

$$\begin{array}{ccc} (\mathsf{C}, \epsilon)\text{-}\mathbf{Coalg} & & (\mathsf{N}, \epsilon, \delta)\text{-}\mathbf{Coalg} \\ \lceil - \rceil \uparrow \cong \downarrow \lfloor - \rfloor & & \lceil - \rceil \uparrow \cong \downarrow \lfloor - \rfloor \\ \mathsf{B}\text{-}\mathbf{Coalg} & & \mathsf{B}\text{-}\mathbf{Coalg} \end{array}$$

13.2. **Initial and Final Objects.** In the following diagram we can see the initial and final λ-bialgebra in our new setting where $\lambda : \mathsf{S} \circ \mathsf{N} \dot{\to} \mathsf{N} \circ \mathsf{S}$. The initial λ-bialgebra is $\langle \mu\mathsf{S}, \mathit{in}, (\!|\mathsf{N}_\lambda\ \mathit{in}|\!) \rangle$. The unique λ-bialgebra homomorphism from the initial λ-bialgebra to any λ-bialgebra $\langle X, a, c \rangle$ is $(\!|a|\!)$. The final λ-bialgebra is $\langle \nu\mathsf{B}, [\![\lfloor \mathsf{S}^\lambda \lceil \mathit{out} \rceil \rfloor]\!], \lceil \mathit{out} \rceil \rangle$. The unique λ-bialgebra homomorphism to the final λ-bialgebra from any λ-bialgebra $\langle X, a, c \rangle$ is $[\![\lfloor c \rfloor]\!]$. The center of the diagram displays the dual solutions to the unique semantic function.

$$\begin{array}{ccc} \mathsf{S}\,(\mu\mathsf{S}) & \dashrightarrow & \mathsf{S}\,(\nu\mathsf{B}) \\ {\scriptstyle \mathit{in}}\downarrow & & \downarrow{\scriptstyle [\![\lfloor \mathsf{S}^\lambda \lceil \mathit{out} \rceil \rfloor]\!]} \\ \mu\mathsf{S} & \dashrightarrow \parallel \dashrightarrow & \nu\mathsf{B} \\ {\scriptstyle (\!|\mathsf{N}_\lambda\ \mathit{in}|\!)}\downarrow & [\![\lfloor (\!|\mathsf{N}_\lambda\ \mathit{in}|\!) \rfloor]\!] & \downarrow{\scriptstyle \lceil \mathit{out} \rceil} \\ \mathsf{N}\,(\mu\mathsf{S}) & \dashrightarrow & \mathsf{N}\,(\nu\mathsf{B}) \end{array}$$

On the left-hand side of the diagram there is a proof obligation that $(\!|\mathsf{N}_\lambda\ \mathit{in}|\!)$ must be an $(\mathsf{N}, \epsilon, \delta)$-coalgebra respecting ϵ (13.3a) and δ (13.3b). This is dispatched by the duals of the proofs that $[\![\mathsf{P}^\lambda\ \mathit{out}]\!]$ respects η (Section 8.2) and that $[\![\mathsf{M}^\lambda\ \mathit{out}]\!]$ respects μ (Section 9.2).

On the right-hand side of the diagram there are the same proof obligations that arose in Section 10, namely that $\langle \nu\mathsf{B}, [\![\lfloor \mathsf{S}^\lambda \lceil \mathit{out} \rceil \rfloor]\!], \lceil \mathit{out} \rceil \rangle$ is a λ-bialgebra and that $[\![\lfloor c \rfloor]\!]$ is both a S-algebra and N-coalgebra homomorphism. (The definitions of $\lfloor - \rfloor$

and $\lceil - \rceil$ have changed according to Theorem 13.1.) Again, these are dispatched by the dual proofs of those in Section 8.2.

13.3. **Creating a Distributive Law.** Given a program that is modelled by a natural transformation of type $\rho : \mathsf{S} \circ \mathsf{N} \dot{\to} \mathsf{B} \circ \mathsf{S}$, we seek to derive a distributive law $\lambda : \mathsf{S} \circ \mathsf{N} \dot{\to} \mathsf{N} \circ \mathsf{S}$ such that

$$\lceil c \rceil \cdot a = \mathsf{N}\, a \cdot \lambda\, X \cdot \mathsf{S} \lceil c \rceil \quad \Longleftrightarrow \quad c \cdot a = \mathsf{B}\, a \cdot \rho\, X \cdot \mathsf{S} \lceil c \rceil \ .$$

As stated in Section 9.3, distributive laws of type $\lambda : \mathsf{S} \circ \mathsf{N} \dot{\to} \mathsf{N} \circ \mathsf{S}$ are in one-to-one correspondence to lifted functors $\bar{\mathsf{S}} : (\mathsf{N}, \epsilon, \delta)\text{-}\mathbf{Coalg} \to (\mathsf{N}, \epsilon, \delta)\text{-}\mathbf{Coalg}$. Given a lifting $\bar{\mathsf{S}}$, we can construct a distributive law by following the dual of Section 9.3:

$$\lambda_{\bar{\mathsf{S}}} = \mathsf{N} \circ \mathsf{S} \circ \epsilon \cdot \bar{\mathsf{S}}\, \delta \ ,$$

where $\bar{\mathsf{S}}\, \delta : \mathsf{S} \circ \mathsf{N} \dot{\to} \mathsf{N} \circ \mathsf{S} \circ \mathsf{N}$ is the N-coalgebra for the carrier $\mathsf{S} \circ \mathsf{N}$. By duality, with reference to Section 9.3, $\lambda_{\bar{\mathsf{S}}}$ coheres with ϵ and δ per equations (13.2a) and (13.2b).

What is left is to define the lifting of S. Again by duality:

$$\begin{aligned} \bar{\mathsf{S}}\, \langle X, c : X \to \mathsf{N}\, X \rangle &= \langle \mathsf{S}\, X, \lceil \rho\, X \cdot \mathsf{S}\, c \rceil : \mathsf{S}\, X \to \mathsf{N}\, (\mathsf{S}\, X) \rangle \ , \\ \bar{\mathsf{S}}\, h &= \mathsf{S}\, h \ . \end{aligned}$$

This is a lifting on N-coalgebras as $\lceil - \rceil = \lfloor - \rfloor$ is one from B-coalgebras. Therefore with $\lambda = \lambda_{\bar{\mathsf{S}}}$, the distributive law is the dual of (9.16), as expected:

$$\lambda = \mathsf{N} \circ \mathsf{S} \circ \epsilon \cdot \lfloor \rho \circ \mathsf{N} \cdot \mathsf{S} \circ \delta \rfloor \ .$$

14. Pointed Functors over Comonads

In this section we will briefly combine pointed functors and comonads from Sections 8 and 13. Let us begin with a brief example.

Example 14.1. The following is the usual definition of *tail*:

$$tail\ (Cons\ m\ s) = s$$

This definition cannot be immediately used: the type of this definition of tail is $\mathsf{S}\ X \to X$ and we need a natural transformation that we can turn into a distributive law. We need $\prec$ to appear on the right-hand side, which forces us to pattern match more deeply on the left-hand side:

$$tail\ (Cons\ \_\ (Cons\ n\ s)) = n \prec s$$

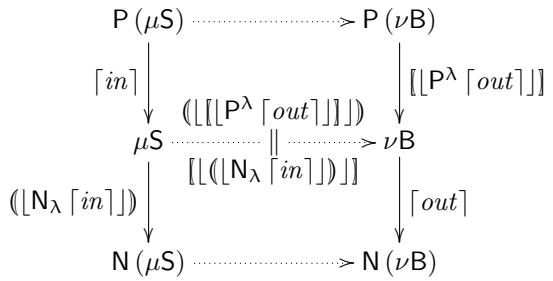
Now we can capture this recursion equation by a natural transformation:

$$\begin{aligned} &\rho :: \mathsf{S}\ (\mathsf{N}\ x) \to \mathsf{B}\ (\mathsf{P}\ x) \\ &\rho\ (Tail\ (Root\ \_\ (Cons\ (\_, Root\ \_\ (Cons\ (n, s))))))) = \\ &\quad Cons\ (n, Var\ s) \qquad \square \end{aligned}$$

Properties 8.6, 10.6 and 13.8 show that B_λ preserves respect for η and S^λ preserves respect for ϵ and δ. Taken together, Properties 8.5, 10.6 and 13.8 say that η lifts to a $(\mathsf{B}, \epsilon, \delta)$-coalgebra homomorphism: S^λ is pointed. Likewise, Properties 10.5, 13.7 and 8.6 say that ϵ and δ lifts to an (S, η)-algebra homomorphisms: B_λ is a comonad. Therefore all six properties conspire to preserve the desired double isomorphism.

$$\lambda\text{-}\mathbf{Bialg}(\mathscr{C}) \cong \begin{cases} (\mathsf{S}^\lambda, \eta)\text{-}\mathbf{Alg}((\mathsf{B}, \epsilon, \delta)\text{-}\mathbf{Coalg}(\mathscr{C})) \\ (\mathsf{B}_\lambda, \epsilon, \delta)\text{-}\mathbf{Coalg}((\mathsf{S}, \eta)\text{-}\mathbf{Alg}(\mathscr{C})) \end{cases} \tag{14.1}$$

14.1. **Initial and Final Objects.** In the following diagram we can see the initial and final λ-bialgebras in our new setting where $\lambda : \mathsf{P} \circ \mathsf{N} \dot{\to} \mathsf{N} \circ \mathsf{P}$, and also the dual solutions to the unique semantic function.



As in previous sections, the casting operators $\lfloor - \rfloor$ and $\lceil - \rceil$ are overloaded in this diagram. On the left-hand side of the diagram we are casting between (P, η)-algebras and S-algebras (Theorem 8.1), and on the right-hand side, $(\mathsf{N}, \epsilon, \delta)$-coalgebras and B-coalgebras (Theorem 13.1).

14.2. **Creating a Distributive Law.** Given a program that is modelled by a natural transformation of type $\rho : \mathsf{S} \circ \mathsf{N} \dot{\to} \mathsf{B} \circ \mathsf{P}$, we seek to derive a distributive law $\lambda : \mathsf{P} \circ \mathsf{N} \dot{\to} \mathsf{N} \circ \mathsf{P}$. This derivation is dual to that in Section 12.2.

The first step is to define an intermediate natural transformation of type $\rho^{\mathsf{P}} : \mathsf{P} \circ \mathsf{N} \dot{\to} \mathsf{B} \circ \mathsf{P}$. This the dual of ρ^{P} from Section 12.2 and similar to $\rho^{\mathsf{P}} : \mathsf{P} \circ \mathsf{C} \dot{\to} \mathsf{B} \circ \mathsf{P}$ from Section 11.2:

$$\rho^{\mathsf{P}} = \mathsf{B} \circ \eta \cdot \phi \triangledown \rho \ .$$

Now, following the duality and the lifting technique of Section 13.3, we can define a lifting of P to $\mathsf{P}^{\rho} : (\mathsf{N}, \epsilon, \delta)\text{-}\mathbf{Coalg} \to (\mathsf{N}, \epsilon, \delta)\text{-}\mathbf{Coalg}$,

$$\mathsf{P}^{\rho}\ c = [\![\rho^{\mathsf{P}}\ X \cdot \mathsf{P}\ c]\!] \ ,$$

and then with this, the distributive law,

$$\lambda = \mathsf{N} \circ \mathsf{P} \circ \epsilon \cdot [\![\rho^{\mathsf{P}} \circ \mathsf{N} \cdot \mathsf{P} \circ \delta]\!] \ .$$

As a result of following the construction in Section 13.3, λ satisfies Condition 13.5 and $\mathsf{P}^{\rho}\ \delta$ satisfies Condition 13.6, that is, λ coheres with ϵ and δ and $\mathsf{P}^{\rho}\ \delta$ is an N-coalgebra that respects ϵ and δ.

As λ contains the pointed functor P, it must also cohere with η, Condition 8.1. Again, following the development in Section 12.2, just as C_{ρ} was copointed, dually P^{ρ} is pointed (cf. Property 8.5). From this the dual of the proof in Section 12.2 follows to show that λ coheres with η.

15. Monads over Comonads

15.1. **Creating a Distributive Law.**

16. Summary

Summary: feature matrix in Table 1.
Summary: creating distributive laws in Table 2.

left-hand side		
B	C	N
constructor	@ notation	nested patterns

right-hand side		
S	P	M
constructor	constructor or variable	nested terms with variables

TABLE 1. Feature Matrix ($\rho : \mathsf{S} \circ \mathit{lhs} \dot{\to} \mathsf{B} \circ \mathit{rhs}$)

λ	S	P	M
B	$\surd$	Section 8.3 $\rho : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{P}$ $\lambda = \mathsf{B}\circ\eta \triangledown \rho$ $\lambda : \mathsf{P}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{P}$	Section 9.3 $\rho : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{M}$ lifting $\lambda : \mathsf{M}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{M}$
C	Section 10.3 $\rho : \mathsf{S}\circ\mathsf{C} \dot{\to} \mathsf{B}\circ\mathsf{S}$ $\lambda = \mathsf{S}\circ\epsilon \vartriangle \rho$ $\lambda : \mathsf{S}\circ\mathsf{C} \dot{\to} \mathsf{C}\circ\mathsf{S}$	Section 11.2 $\rho : \mathsf{S}\circ\mathsf{C} \dot{\to} \mathsf{B}\circ\mathsf{P}$ combine 8.3 & 10.3 $\lambda : \mathsf{P}\circ\mathsf{C} \dot{\to} \mathsf{C}\circ\mathsf{P}$	Section 12.2 $\rho : \mathsf{S}\circ\mathsf{C} \dot{\to} \mathsf{B}\circ\mathsf{M}$ $\rho' : \mathsf{S}\circ\mathsf{C} \dot{\to} \mathsf{C}\circ\mathsf{M}$ $\lambda : \mathsf{M}\circ\mathsf{C} \dot{\to} \mathsf{C}\circ\mathsf{M}$
N	Section 13.3 $\rho : \mathsf{S}\circ\mathsf{N} \dot{\to} \mathsf{B}\circ\mathsf{S}$ colifting (dual) $\lambda : \mathsf{S}\circ\mathsf{N} \dot{\to} \mathsf{N}\circ\mathsf{S}$	Section 14.2 $\rho : \mathsf{S}\circ\mathsf{N} \dot{\to} \mathsf{B}\circ\mathsf{P}$ $\rho' : \mathsf{P}\circ\mathsf{N} \dot{\to} \mathsf{B}\circ\mathsf{P}$ $\lambda : \mathsf{P}\circ\mathsf{N} \dot{\to} \mathsf{N}\circ\mathsf{P}$	Section 15.1 ? $\lambda : \mathsf{M}\circ\mathsf{N} \dot{\to} \mathsf{N}\circ\mathsf{M}$

TABLE 2. Creating Distributive Laws

17. Related Work

The theoretical foundations of our work exist in the literature, originally in Turi and Plotkin [1997] and refined in Lenisa et al. [2000]. We see our work as an application of, and an exercise in, this theory.

The work that is closest in spirit to ours is Bartels [2003]. It is centered around the coinduction proof principle, in contrast to the UFP. Bartels looks at two out of the nine points that we have identified, the simplest $\lambda : \mathsf{S}\circ\mathsf{B} \dot{\to} \mathsf{B}\circ\mathsf{S}$, and our sweet spot $\lambda : \mathsf{M}\circ\mathsf{C} \dot{\to} \mathsf{C}\circ\mathsf{M}$, but for space reasons does not explore any others. Bartels introduces a construction *homomorphism up-to*, which is a homomorphism from a coalgebra to a bialgebra, and uses it as a definitional principle. We simply use bialgebra homomorphisms, following the original theory of Turi and Plotkin [1997], which nicely exhibits the duality of Iniga and Finn's viewpoints.

Rutten and Silva present two coinductive calculi, one for streams [Rutten, 2003] and one for binary trees [Silva and Rutten, 2010], also using coinduction as a proof principle. They have a uniqueness proof for each: Theorem 3.1 and Appendix A in Rutten [2003]; and Theorem 2 in Silva and Rutten [2010]. Our approach treats streams and infinite trees, and *behaviour* in general, in a datatype generic way—the same proofs apply, only varying in the chosen functors for syntax and behaviour. Moreover, we emphasize a compositional, functional style.

Our task of determining that a recursion equation has a unique solution is related to the task of determining that corecursive definitions are productive [Sijtsma, 1989]. This is crucial in dependently typed programming and proof languages, where the logical consistency of the system requires it. In Coq this is enforced by

the guardedness condition [Giménez, 1995], which is particularly conservative: it has no means to propagate information through function calls, so corecursive calls are forbidden to appear anywhere other than as a direct argument of a constructor. Compositionality is the first casualty. The situation is similar in Agda [Abel and Altenkirch, 2002].

Hughes et al. [1996] were the first to talk about the notion of *sized-types*, and used it as part of a type-based analysis that guarantees termination and liveness of embedded functional programs. Following this, there have been a whole host of proposed type systems incorporating size annotations. MiniAgda [Mehltretter, 2007, Abel, 2010] is a tangible implementation of a dependently typed core language with sized types, able to track the productivity of corecursive definitions. Type signatures are mandatory and contain sizes explicitly, which is in contrast to our ρ functions, the naturality of which is easy to infer.

Specific to streams, Endrullis et al. [2008] introduce what they call *data-oblivious* productivity: productivity that can be decided without inspecting the stream elements. They present three classes of stream specifications. Their analysis is provably optimal for the *flat* class, where stream functions cannot contain nested function applications. Our slogan "consume at most one, produce at least one" corresponds to their *friendly nesting* class. A competing approach appears in Zantema [2010], who reduces the determination of uniqueness to the termination of a term rewriting system (TRS). A stream specification has a unique solution if its *observational variant* TRS is terminating, a TRS that is very like Rutten's stream definitions.

Acknowledgements. Ralf would like to thank Jan Rutten for pointing him to distributive laws and bialgebras.

References

A. Abel. MiniAgda: Integrating Sized and Dependent Types. *Electronic Proceedings in Theoretical Computer Science*, 43:14–28, 2010.

A. Abel and T. Altenkirch. A predicative analysis of structural recursion. *JFP*, 12 (1):1–41, 2002.

J.-P. Allouche and S. Jeffery. The ubiquitous Prouhet–Thue–Morse sequence. In C. Ding, T. Helleseth, and H. Niederreiter, editors, *Sequences and Their Applications*, SETA '98, pages 1–16. Springer, 1999.

H. Applegate. *Acyclic models and resolvent functors*. PhD thesis, Columbia University, 1965.

F. Bartels. Generalised coinduction. *Mathematical Structures in Computer Science*, 13(2):321–348, 2003.

R. S. Bird and O. De Moor. *Algebra of Programming*, volume 100 of *International Series in Computing Science*. Prentice Hall, 1997.

E. W. Dijkstra. Hamming's exercise in SASL. Personal Note EWD792, 1981.

J. Endrullis, C. Grabmayer, and D. Hendriks. Data-oblivious stream productivity. In *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 5330 of *LNCS*, pages 79–96. Springer, 2008.

P. Freyd. Remarks on algebraically compact categories. In M. Fourman, P. Johnstone, and A. Pitts, editors, *Applications of Categories in Computer Science*, number 177 in London Mathematical Society Lecture Note Series, pages 95–107. Cambridge University Press, 1992.

E. Giménez. Codifying guarded definitions with recursive schemes. In *Types for Proofs and Programs*, volume 996 of *LNCS*, pages 39–59. Springer, 1995.

R. Hinze. The Bird tree. *JFP*, 19(5):491–508, 2009.

R. Hinze. Concrete stream calculus—an extended study. *JFP*, 20(5–6):463–535, 2010.
J. Hughes, L. Pareto, and A. Sabry. Proving the correctness of reactive systems using sized types. In *POPL*, pages 410–423. ACM, 1996.
M. Lenisa, J. Power, and H. Watanabe. Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads. *Electronic Notes in Theoretical Computer Science*, 33:230–260, 2000.
C. McBride and R. Paterson. Applicative programming with effects. *JFP*, 18(1): 1–13, 2008.
K. Mehltretter. Termination checking for a dependently typed language. Master's thesis, LMU Munich, 2007.
M. Niqui and J. J. M. M. Rutten. Sampling, splitting and merging in coinductive stream calculus. In *MPC*, volume 6120 of *LNCS*, pages 310–330. Springer, 2010.
J. J. M. M. Rutten. Fundamental study: Behavioural differential equations: A coinductive calculus of streams, automata, and power series. *Theoretical Computer Science*, 308:1–53, 2003.
B. A. Sijtsma. On the productivity of recursive list definitions. *ACM Trans. Program. Lang. Syst.*, 11(4):633–649, 1989.
A. Silva and J. J. M. M. Rutten. A coinductive calculus of binary trees. *Information and Computation*, 208:578–593, 2010.
D. Turi and G. Plotkin. Towards a mathematical operational semantics. *Logic in Computer Science*, page 280, 1997.
H. Zantema. Well-definedness of streams by transformation and termination. *Logical Methods in Computer Science*, 6(3:21), 2010.

Appendix A. Product Laws

Uniqueness.

$$f_1 = outl \cdot g \;\wedge\; f_2 = outr \cdot g \iff f_1 \vartriangle f_2 = g \tag{A.1}$$

Reflection law.

$$outl \vartriangle outr = id \tag{A.2}$$

Computation law.

$$f_1 = outl \cdot (f_1 \vartriangle f_2) \tag{A.3a}$$

$$f_2 = outr \cdot (f_1 \vartriangle f_2) \tag{A.3b}$$

Fusion law.

$$(f_1 \vartriangle f_2) \cdot h = f_1 \cdot h \vartriangle f_2 \cdot h \tag{A.4}$$

Functor fusion law.

$$(k_1 \times k_2) \cdot (f_1 \vartriangle f_2) = k_1 \cdot f_1 \vartriangle k_2 \cdot f_2 \tag{A.5}$$

Projections are natural.

$$k_1 \cdot outl = outl \cdot (k_1 \times k_2) \tag{A.6a}$$

$$k_2 \cdot outr = outr \cdot (k_1 \times k_2) \tag{A.6b}$$

Appendix B. Coproduct Laws

Uniqueness.

$$f = g_1 \triangledown g_2 \iff f \cdot inl = g_1 \;\wedge\; f \cdot inr = g_2 \tag{B.1}$$

Reflection law.

$$id = inl \triangledown inr \tag{B.2}$$

Computation law.

$$(g_1 \triangledown g_2) \cdot inl = g_1 \tag{B.3a}$$

$$(g_1 \triangledown g_2) \cdot inr = g_2 \tag{B.3b}$$

Fusion law.

$$k \cdot (g_1 \triangledown g_2) = k \cdot g_1 \triangledown k \cdot g_2 \tag{B.4}$$

Functor fusion law.

$$(g_1 \triangledown g_2) \cdot (h_1 + h_2) = g_1 \cdot h_1 \triangledown g_2 \cdot h_2 \tag{B.5}$$

Injections are natural.

$$(h_1 + h_2) \cdot inl = inl \cdot h_1 \tag{B.6a}$$

$$(h_1 + h_2) \cdot inr = inr \cdot h_2 \tag{B.6b}$$

Appendix C. Initial Algebra Laws

Reflection law.

$$id = (\!|in|\!) \tag{C.1}$$

Computation law.

$$(\!|a|\!) \cdot in = a \cdot \mathsf{F}\,(\!|a|\!) \tag{C.2}$$

Fusion law.

$$h \cdot (\!|a|\!) = (\!|b|\!) \quad \Longleftarrow \quad h \cdot a = b \cdot \mathsf{F}\,h \tag{C.3}$$

Appendix D. Final Coalgebra Laws

Reflection law.

$$[\!|out|\!] = id \tag{D.1}$$

Computation law.

$$\mathsf{F}\,[\!|c|\!] \cdot c = out \cdot [\!|c|\!] \tag{D.2}$$

Fusion law.

$$[\!|d|\!] = [\!|c|\!] \cdot h \quad \Longleftarrow \quad \mathsf{F}\,h \cdot d = c \cdot h \tag{D.3}$$

Appendix E. Free Monad Laws

Uniqueness property. Let us remind ourselves of the uniqueness property. Given an algebra $a : \mathsf{S}\,X \to X$ and an arrow $g : Y \to X$,

$$f = (\!|a|\!) \cdot \mathsf{M}\,g \quad \Longleftrightarrow \quad f \cdot var = g \;\wedge\; f \cdot com = a \cdot \mathsf{S}\,f \;\;,$$

for all $f : \mathsf{M}\;Y \to X$.

Relation to the initial algebra. The carrier of the initial S-algebra is isomorphic to the free S-algebra over 0 (the initial object in $\mathscr{C}$), $\mu\mathsf{S} \cong \mathsf{M}\,0$. Modulo this isomorphism, we have the following equalities: $in = com\,0$, and $(\!|a|\!) = (\!|a|\!) \cdot \mathsf{M}\,¡_X$. (Note that $\eta 0 = ¡_{\mathsf{M}\,0}$.)

Reflection law.

$$id = (\!|com|\!) \cdot \mathsf{M}\,(var\;Y) \tag{E.1}$$

Since $\mu = (\!|com|\!)$ and $\eta = var$, this amounts to the second monad law (9.1b).

Computation law.

$$(a) \cdot \mathsf{M}\, g \cdot var\ Y \ = g \tag{E.2a}$$

$$(a) \cdot \mathsf{M}\, g \cdot com\ Y = a \cdot \mathsf{S}\,((a) \cdot \mathsf{M}\, g) \tag{E.2b}$$

Setting $Y = \mathsf{M}\, X$ and $g = id_{\mathsf{M}\, X}$, (E.2a) specializes to the first monad law (9.1a).

Fusion law.

$$h \cdot (a) \cdot \mathsf{M}\, g = (a') \cdot \mathsf{M}\, g' \iff h \cdot g = g' \ \wedge\ h \cdot a = a' \cdot \mathsf{S}\, h \tag{E.3}$$

The fusion law implies the third monad law (9.1c).

$$\begin{array}{ll} & (com) \cdot (com) = (com) \cdot \mathsf{M}\,(com) \\ \Longleftarrow & \{\ \text{fusion (E.3)}\ \} \\ & (com) = (com) \ \wedge\ (com) \cdot com = com \cdot \mathsf{S}\,(com) \\ \Longleftrightarrow & \{\ \text{computation (9.8b)}\ \} \\ & true \end{array}$$

Appendix F. Cofree Comonad Laws

Uniqueness property. Let us remind ourselves of the uniqueness property. Given a coalgebra $c : X \to \mathsf{B}\, X$ and an arrow $f : X \to Y$,

$$\mathsf{N}\, f \cdot [\![c]\!] = g \iff f = lab\ Y \cdot g \ \wedge\ \mathsf{B}\, g \cdot c = des\ Y \cdot g \ ,$$

for all $g : X \to \mathsf{N}\, Y$.

Relation to the final coalgebra. The carrier of the final B-coalgebra is isomorphic to the free B-coalgebra over 1 (the final object in $\mathscr{C}$), $\nu\mathsf{B} \cong \mathsf{N}\, 1$. Modulo this isomorphism we have the following equalities: $out = des\, 1$, and $[\![c]\!] = \mathsf{M}\, !_X \cdot [\![c]\!]$. (Note that $\epsilon 1 = !_{\mathsf{N}\, 1}$.)

Reflection law.

$$\mathsf{N}\,(lab\ Y) \cdot [\![des]\!] = id \tag{F.1}$$

Since $\delta = [\![des]\!]$ and $\epsilon = lab$, this amounts to the first comonad law (13.1a).

Computation law.

$$f = lab\ Y \cdot \mathsf{N}\, f \cdot [\![c]\!] \tag{F.2a}$$

$$\mathsf{B}\,(\mathsf{N}\, f \cdot [\![c]\!]) \cdot c = des\ Y \cdot \mathsf{N}\, f \cdot [\![c]\!] \tag{F.2b}$$

Setting $Y = \mathsf{N}\, X$ and $f = id_{\mathsf{N}\, X}$, (F.2a) specializes to the second comonad law (13.1b).

Fusion law.

$$\mathsf{N}\, f' \cdot [\![c']\!] = \mathsf{N}\, f \cdot [\![c]\!] \cdot h \iff f' = f \cdot h \ \wedge\ \mathsf{B}\, h \cdot c' = c \cdot h \tag{F.3}$$

The fusion law implies the third comonad law (13.1c).

$$\begin{array}{ll} & \mathsf{N}\,[\![des]\!] \cdot [\![des]\!] = [\![des]\!] \cdot [\![des]\!] \\ \Longleftarrow & \{\ \text{fusion (F.3)}\ \} \\ & [\![des]\!] = [\![des]\!] \ \wedge\ \mathsf{B}\,[\![des]\!] \cdot des = des \cdot [\![des]\!] \\ \Longleftrightarrow & \{\ \text{computation (13.7b)}\ \} \\ & true \end{array}$$

Appendix G. Lifting

The underlying or forgetful functor $\mathsf{U} : \mathsf{F}\text{-}\mathbf{Alg}(\mathscr{C}) \to \mathscr{C}$ is defined

$$\mathsf{U}\,\langle A, a\rangle = A \ , \qquad \mathsf{U}\,h = h \ .$$

A functor $\bar{\mathsf{H}} : \mathsf{F}\text{-}\mathbf{Alg}(\mathscr{C}) \to \mathsf{G}\text{-}\mathbf{Alg}(\mathscr{D})$ is a *lifting of* $\mathsf{H} : \mathscr{C} \to \mathscr{D}$ if $\mathsf{U} \circ \bar{\mathsf{H}} = \mathsf{H} \circ \mathsf{U}$.

$$\begin{array}{ccc} \mathsf{F}\text{-}\mathbf{Alg}(\mathscr{C}) & \xrightarrow{\bar{\mathsf{H}}} & \mathsf{G}\text{-}\mathbf{Alg}(\mathscr{D}) \\ \mathsf{U}\downarrow & & \downarrow\mathsf{U} \\ \mathscr{C} & \xrightarrow[\mathsf{H}]{} & \mathscr{D} \end{array}$$

Given a natural transformation $\lambda : \mathsf{G} \circ \mathsf{H} \dot{\to} \mathsf{H} \circ \mathsf{F}$, we can define a lifting $\mathsf{H}_\lambda : \mathsf{F}\text{-}\mathbf{Alg}(\mathscr{C}) \to \mathsf{G}\text{-}\mathbf{Alg}(\mathscr{D})$ of H as follows:

$$\mathsf{H}_\lambda\,\langle X, a : \mathsf{F}\,X \to X\rangle = \langle \mathsf{H}\,X, \mathsf{H}\,a \cdot \lambda\,X : \mathsf{G}\,(\mathsf{H}\,X) \to \mathsf{H}\,X\rangle \ , \tag{G.1}$$

$$\mathsf{H}_\lambda\,h = \mathsf{H}\,h \ . \tag{G.2}$$

Since H_λ's action on carriers and homomorphisms is given by H, it preserves identity and composition. It remains to show that it takes F-homomorphisms to G-homomorphisms.

$$\mathsf{H}\,h : \mathsf{H}_\lambda\,a \to \mathsf{H}_\lambda\,b : \mathsf{G}\text{-}\mathbf{Alg}(\mathscr{D}) \quad \Longleftarrow \quad h : a \to b : \mathsf{F}\text{-}\mathbf{Alg}(\mathscr{C}) \ ,$$

where $a : \mathsf{F}\,X \to X$ and $b : \mathsf{F}\,Y \to Y$. We reason.

$$\begin{array}{ll} & \mathsf{H}\,h \cdot \mathsf{H}_\lambda\,a \\ = & \quad \{\ \text{definition of } \mathsf{H}_\lambda\ (\text{G.1})\ \} \\ & \mathsf{H}\,h \cdot \mathsf{H}\,a \cdot \lambda\,X \\ = & \quad \{\ \mathsf{H} \text{ functor and assumption } h : a \to b : \mathsf{F}\text{-}\mathbf{Alg}(\mathscr{C})\ \} \\ & \mathsf{H}\,b \cdot \mathsf{H}\,(\mathsf{F}\,h) \cdot \lambda\,X \\ = & \quad \{\ \lambda : \mathsf{G} \circ \mathsf{H} \dot{\to} \mathsf{H} \circ \mathsf{F} \text{ is natural}\ \} \\ & \mathsf{H}\,b \cdot \lambda\,Y \cdot \mathsf{G}\,(\mathsf{H}\,h) \\ = & \quad \{\ \text{definition of } \mathsf{H}_\lambda\ (\text{G.1})\ \} \\ & \mathsf{H}_\lambda\,b \cdot \mathsf{G}\,(\mathsf{H}\,h) \end{array}$$

The functor $\alpha\text{-}\mathbf{Alg}$ emerges as a special case with $\mathsf{H} = \mathsf{Id}$ and $\lambda = \alpha$. Also, S_λ is an instance of the scheme with $\mathsf{F} = \mathsf{G}$, which consequently restricts H to endofunctors.

The construction dualizes to categories of coalgebras.

(R. Hinze) Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, England
E-mail address, R. Hinze: ralf.hinze@cs.ox.ac.uk
URL: http://www.cs.ox.ac.uk/people/ralf.hinze/

(D. James) Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, England
E-mail address, D. James: daniel.james@cs.ox.ac.uk
URL: http://www.cs.ox.ac.uk/people/daniel.james/