Notes on Kan Extensions and Monads

Marco Paviotti

December 22, 2024

Abstract

These notes are meant to remind myself of some facts about Kan extension and monads. The first part is devoted to basic definitions about adjunctions, ends and coends, which are needed to explain the proofs later on. The second part is on monads and Kan extensions.

Contents

1 Adjunctions and Monads **2**
1.1 Examples 3

2 Limits and Colimits **3**
2.1 Preservation and Creation of (Co)Limits 4
2.2 Dependent product and sum 4
2.2.1 Powers and CoPowers 4

3 Ends and Coends **5**
3.1 Preservation of Ends 6
3.2 Natural Transformations and Ends 7

4 The Yoneda Lemma **7**
4.1 Ninja Yoneda Lemmas 8
4.2 The mini Yoneda Lemma for Type Theorists 9
4.3 Exponentials in Presheaf Categories 10

5 Kan Extensions **11**
5.1 Kan Extensions 12
5.2 Yoneda as Kan Extensions 15
5.3 CoYoneda as Kan Extensions 16
5.4 Left and Right Shifts 16

6 Distributive Laws, (Co)Algebras and Kan Extensions **16**

7 Monads from Kan Extensions **17**
7.1 The Codensity Monad 17
7.2 The Codensity Transformation 17

8 On Free Monads **17**
8.1 Free Monads and The Right Kan Extension 18
8.2 Free Monads and the Left Kan Extension 18
8.3 Algebras for the Left Kan extension 19
8.4 The Freest Monad 20
8.5 A Free Monad for Scoped Effects 20

9 (Free) Applicative Functors **21**
9.1 Functor Categories as Monoidal Categories 23
9.2 Day Convolution as a Left Kan Extension 23
9.3 Free Applicatives 24

10 On Traversals **24**
10.1 Traversals as distributive laws 24
10.2 Traversals are Coalgebras for Free Applicatives 25
10.3 Containers 27

1 Adjunctions and Monads

Given two functors $L : \mathcal{D} \to \mathcal{C}$ and $R : \mathcal{C} \to \mathcal{D}$ and *adjunction* is an isomorphism of homsets

$$\lfloor \cdot \rfloor : \mathcal{C}(LA, B) \cong \mathcal{D}(A, RB) : \lceil \cdot \rceil$$

which is furthermore natural in A and B. Here $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ are the functions witnessing the isomorphism. The adjunction is usually depicted as follows

$$\begin{tikzcd} \mathcal{C} \arrow[r, bend right, "R"'] \arrow[r, phantom, "\perp"] & \mathcal{D} \arrow[l, bend right, "L"'] \end{tikzcd}$$

We say that that L is left adjoint and R is right adjoint and it is indicated by $L \vdash R$. As a consequence of the isomorphism, for $f : LA \to B$ and $g : A \to RB$ we have that

$$\lfloor f \rfloor = g \iff f = \lceil g \rceil$$

and because the isomorphism is natural we can derive the *fusion laws*. For $a : A' \to A$, $b : B \to B'$, $f : LA \to B$ and $g : A \to RB$

$$R(b) \cdot \lfloor f \rfloor = \lfloor b \cdot f \rfloor$$
$$\lfloor f \rfloor \cdot a = \lfloor f \cdot L(a) \rfloor$$
$$b \cdot \lceil g \rceil = \lceil R(b) \cdot g \rceil$$
$$\lceil g \rceil \cdot L(a) = \lceil g \cdot a \rceil$$

We can also compute the fusion laws this way.

$$R(b) \cdot \lfloor f \rfloor \cdot a = \lfloor b \cdot f \cdot L(a) \rfloor$$
$$b \cdot \lceil g \rceil \cdot L(a) = \lceil R(b) \cdot g \cdot a \rceil$$

This is really all about adjunctions. All the other definitions and constructions are equivalent to this one. Furthermore, this material is very well covered elsewhere [2, 6, 4] so I will not be covering it further.

What is important for the sake of this notes is that an adjunction gives rise to a monad and a comonad where RL is the monad and LR is the comonad. The unit and counit of the adjunction are defined as follows

$$\eta_A = \lfloor id_{LA} \rfloor$$
$$\epsilon_B = \lceil id_{RB} \rceil$$

The join or multiplication of the monad $\mu : RLRL \to RL$ is defined as $\mu = R\epsilon_L$ and the cojoin or comultiplication $\delta : LR \to LRLR$ is defined as $\delta = L\eta_R$. The operations of the comonad are dually defined.

1.1 Examples

For example the functor $(- \times X)$ is left adjoint to the exponential $(-)^X$. In particular, the following is a natural isomorphism in A and B:

$$\hat{\ } : \mathcal{C}(A \times X, B) \cong \mathcal{D}(A, B^X) : \check{\ } \tag{1}$$

For arrows $f : A \times X \to B$ and $g : A \to B^X$ the operations $\hat{f}$ and $\check{g}$ are respectively the curry and uncurry operations in functional programming.

2 Limits and Colimits

Given two objects X and Y in a category, $X \times Y$ forms the product of X and Y. We can generalise this further. Given a functor $D : \mathcal{I} \to \mathcal{C}$ the limit $\varprojlim D$ is an universal object such that for every $I \in \mathcal{I}$, there exists a projection map $\varprojlim DI \xrightarrow{\pi_I} DI$ such that for every morphism $DI_1 \xrightarrow{f} DI_2$ we have

$$\pi_{I_2} = f \cdot \pi_{I_1}$$

and furthermore any other object such as this has a unique morphism into the limit commuting with the projections [2, 6].

The limit is right adjoint to the diagonal functor mapping every object to the constant functor and the colimit is the left adjoint to the diagonal functor.

$$\begin{tikzcd}
\mathcal{C} \arrow[r, shift right, "\Delta"'] \arrow[r, phantom, "\perp"] & \mathcal{C}^{\mathcal{I}} \arrow[l, shift right, "\varinjlim"'] \arrow[r, shift right, "\varprojlim"'] \arrow[r, phantom, "\perp"] & \mathcal{C} \arrow[l, shift right, "\Delta"']
\end{tikzcd}$$

We right down the isomorphisms

$$\mathcal{C}(\varinjlim_{I \in \mathcal{I}} A(I), B) \cong \mathcal{C}^{\mathcal{I}}(A, \Delta B)$$

$$\mathcal{C}^{\mathcal{I}}(\Delta A, B) \cong \mathcal{C}(A, \varprojlim_{I \in \mathcal{I}} B(I))$$

2.1 Preservation and Creation of (Co)Limits

A functor $H : \mathcal{C} \to \mathcal{D}$ is said to *preserve* limits if, given a diagram $F : \mathcal{I} \to \mathcal{C}$

$$H \varprojlim_{I\in\mathcal{I}} FI \cong \varprojlim_{I\in\mathcal{I}} HFI$$

In other words, H preserves limits if the limit of the diagram obtained by composition with H, namely HF, corresponds with the limit of F applied to H. In particular, such a functor preserves small limits as well. A functor that preserves small (co)limits is called *(co)continuous*.

As a prominent example, the covariant homset functor $\mathcal{C}(C, -) : \mathcal{C} \to \mathbf{Set}$ preserve limits

$$\mathcal{C}(C, \varprojlim_{I\in\mathcal{I}} FI) \cong \varprojlim_{I\in\mathcal{I}} \mathcal{C}(C, FI) \tag{2}$$

On the other hand, the contravariant homset functor, which may be written as $\mathcal{C}(-, C) = \mathcal{C}^{\mathrm{op}}(C, -) : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ carries colimits over to limits in the following sense

$$\mathcal{C}(\varinjlim_{I\in\mathcal{I}} FI, C) \cong \varprojlim_{I\in\mathcal{I}} \mathcal{C}(FI, C) \tag{3}$$

2.2 Dependent product and sum

Let us consider the set $\mathcal{I}$ (or the discrete category $\mathcal{I}$ with only identities). Then the right adjoint to the diagonal functor is called the *dependent product* $\Pi_{I\in\mathcal{I}}B(I)$ for some functor $B : \mathcal{I} \to \mathcal{C}$ and the left adjoint is called the *dependent sum* $\Sigma_{I\in\mathcal{I}}A(I)$ for some functor $A : \mathcal{I} \to \mathcal{C}$.

$$\begin{tikzcd}
\mathcal{C} \arrow[r, shift right=1.5, "\Delta"'] \arrow[r, phantom, "\perp"] & \mathcal{C}^{\mathcal{I}} \arrow[l, shift right=1.5, "\Sigma_{I\in\mathcal{I}}.(-)_I"'] \arrow[r, shift right=1.5, "\Pi_{I\in\mathcal{I}}.(-)_I"'] \arrow[r, phantom, "\perp"] & \mathcal{C} \arrow[l, shift right=1.5, "\Delta"']
\end{tikzcd}$$

The limit preservation (2) and colimit reverse (3) continues to hold for dependent products and sums.

$$\Pi_{I\in\mathcal{I}}\mathcal{C}(X, A(I)) \cong \mathcal{C}(X, \Pi_{I\in\mathcal{I}}A(I) \tag{4}$$
$$\mathcal{C}(\Sigma_{I\in\mathcal{I}}A(I), X) \cong \Pi_{I\in\mathcal{I}}\mathcal{C}(A(I), X) \tag{5}$$

2.2.1 Powers and CoPowers

Now we consider categories of constant functors $\mathcal{C}^{\mathcal{I}}$ and keep $\mathcal{I}$ as the discrete category. The limits and colimits of these functors are called powers and copowers which can be indicated by $\Sigma\mathcal{I}.A = \mathcal{I} \bullet A$ and $\Pi\mathcal{I}.B = B^{\mathcal{I}}$

$$\begin{tikzcd}
\mathcal{C} \arrow[r, shift right=1.5, "\Delta"'] \arrow[r, phantom, "\perp"] & \mathcal{C}^{\mathcal{I}} \arrow[l, shift right=1.5, "\Sigma\mathcal{I}.(-)"'] \arrow[r, shift right=1.5, "\Pi\mathcal{I}.(-)"'] \arrow[r, phantom, "\perp"] & \mathcal{C} \arrow[l, shift right=1.5, "\Delta"']
\end{tikzcd}$$

Now equations (4) and (5) in turn specialise to powers and copowers

$$\mathcal{C}(X, A)^{\mathcal{I}} \cong \mathcal{C}(X, A^{\mathcal{I}}) \tag{6}$$

$$\mathcal{C}(\mathcal{I} \bullet A, X) \cong \mathcal{C}(A, X)^{\mathcal{I}} \tag{7}$$

As a consequence of (6) and (7) we get that

$$\mathcal{C}(\mathcal{I} \bullet A, B) \cong \mathcal{C}(A, B)^{\mathcal{I}} \cong \mathcal{C}(A, B^{\mathcal{I}})$$

Now since $\mathcal{I}$ is the discrete category (it has no arrows) it can be regarded as a set! Hence, the set of natural transformation betweeen to constant functors is just the set of functions between the images of these indexed by $\mathcal{I}$

$$\mathcal{C}^{\mathcal{I}}(\Delta X, \Delta Y) \cong \mathcal{C}(X, Y)^{\mathcal{I}} \cong \mathcal{I} \to \mathcal{C}(X, Y) \tag{8}$$

Note that, since $\mathcal{I}$ and $\mathcal{C}(X, Y)$ are sets, then $\mathcal{C}(X, Y)^{\mathcal{I}}$ is the exponential object in **Set** and hence it is isomorphic to $\mathcal{I} \to \mathcal{C}(X, Y)$ which is the set of functions.

In **Set**, we have that teh copower is the product

$$\mathcal{I} \bullet A = A \times \mathcal{I} \tag{9}$$

and the power is the function space

$$B^{\mathcal{I}} = \mathcal{I} \to B \tag{10}$$

$B^{\mathcal{I}}$ is the function space $\mathcal{I} \to B$.

3 Ends and Coends

Sometimes it useful to talk about limits and colimits of diagrams that have a contravariant component additionally to a covariant one. These are functors of type $F : \mathcal{C}^{\text{op}} \times \mathcal{C} \to \mathcal{C}$. An example of such a functor is the exponential

$$F(X, Y) = Y^X$$

where X is the variable appearing in a contravariant position and Y is the variable appearing in a covariant position.

The limit of this functor (if it exists) is called *end* and the colimit is called *coend*. Consider a functor $S : \mathcal{C}^{\text{op}} \times \mathcal{C} \to \mathcal{D}$, the end of S is the limit of S when S is seeing as a diagram $S(C, C) \xrightarrow{S(f,C)} S(B, C) \xleftarrow{S(B,f)} S(B, B)$

$$\begin{tikzcd}
 & \int_X S(X,X) \arrow[dl] \arrow[dr] \arrow[dd] & \\
S(C,C) \arrow[dr, "S(f,C)"'] & & S(B,B) \arrow[dl, "S(B,f)"] \\
 & S(B,C) &
\end{tikzcd}$$

This is though not a very precise definition since the limit needs to be defined on a covariant functor. However, it can be shown [6, Chapter IX, Proposition 1], that for every category $\mathcal{C}$ and functor $S : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{D}$ there exists a category $\mathcal{C}^{\S}$ and functor $S^{\S} : \mathcal{C}^{\S} \to \mathcal{D}$ such that

$$\int_C S(C,C) \cong \varprojlim_C S^{\S}C \tag{11}$$

We refer the reader to the more in-depth presentations of this fact [6, 5].

Moreover, whenever S is "dummy" in the first variable, i.e. S factors through the second projection as in

$$\begin{tikzcd}
\mathcal{C}^{\mathrm{op}} \times \mathcal{C} \arrow[r, "\pi_2"] \arrow[dr, "S"'] & \mathcal{C} \arrow[d, "T"] \\
 & \mathcal{D}
\end{tikzcd}$$

then the end coincides with the limit of T

$$\int_{C:\mathcal{C}} S(C,C) = \varprojlim_{C:\mathcal{C}} TC \tag{12}$$

Ends behave similarly to universal quantification. For a functor $S : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \times \mathcal{D}^{\mathrm{op}} \times \mathcal{D} \to \mathcal{E}$

$$\int_{C:\mathcal{C}} \int_{D:\mathcal{D}} S(C,C,D,D) \cong \int_{D:\mathcal{C}} \int_{C:\mathcal{D}} S(C,C,D,D) \tag{13}$$

This property is known as the *exchange rule* which for integrals it corresponds to the Fubini rule [5].

3.1 Preservation of Ends

A functor $H : \mathcal{C} \to \mathcal{D}$ is said to *preserve the end* of a functor $S : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{D}$

$$H \int_{C \in \mathcal{C}} S(C,C) = \int_{C \in \mathcal{C}} HS(C,C)$$

In other words, when $w : e \overset{..}{\to} S$ is an end of S and $Hw : He \overset{..}{\to} HS$ is an end for HS.

For example, as it was the case for the limits and colimits, the homset functor preserves ends in the following way

$$\mathcal{C}(X, \int_{C:\mathcal{C}} S(C,C)) = \int_{C:\mathcal{C}} (\mathcal{C}(X, S(C,C))) \tag{14}$$

and reserves ends into coends

$$\mathcal{C}(\int^{C:\mathcal{C}} S(C,C), X) = \int_{C:\mathcal{C}} (\mathcal{C}(S(C,C)), X) \tag{15}$$

3.2 Natural Transformations and Ends

Natural transformations are examples of ends. Given two functors $F, G : \mathcal{D} \to \mathcal{C}$, the end of the homset functor $\mathcal{C}^{\mathcal{D}}(F-, G-)$ is the set of natural transformations from F to G

$$\mathrm{Nat}(F, G) = \int_{D:\mathcal{D}} \mathcal{C}(FD, GD) \tag{16}$$

4 The Yoneda Lemma

In this section we are going to assume a locally small category $\mathcal{C}$. This is because we need homsets in this category to extend to functors (in fact to profunctors) $\mathcal{C}(-,-) : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathbf{Set}$. Hence we need homsets to be actual sets. Now, instantiating the contravariant argument of the homset functor yields a covariant hom-functor $\mathcal{C}(X,-) : \mathcal{C} \to \mathbf{Set}$ which maps an arrow $f : A \to B$ to a *function*

$$\mathcal{C}(X, f) : \mathcal{C}(X, A) \to \mathcal{C}(X, B)$$

between sets of functions, in particular $\mathcal{C}(X, f)$ is defined as $g \mapsto f \circ g$. Furthermore, note that (4) is natural in X, therefore, every such natural transformation is obtained as the image of $\mathcal{C}(-, f)$ for some f hence the following isomorphism holds

$$\mathcal{C}(A, B) \cong \mathcal{C}(X, A) \to \mathcal{C}(X, B)$$

The covariant Yoneda lemma generalises this situation by abstracting out the functor $\mathcal{C}(A,-)$ into a generic functor $F : \mathcal{C} \to \mathbf{Set}$. Assume a locally small category $\mathcal{C}$, then for all covariant functors $F : \mathcal{C} \to \mathbf{Set}$, we have the following isomorphism of sets

$$FC \cong \mathcal{C}(C,-) \Rightarrow F$$

The functions witnessing the isomorphism are given by $\phi(x, g) = F(g)(x)$ and its inverse $\psi(i) = i_C(id_C)$.

The hom-functor is a very particular one enjoying a myriad of properties. In particular, notice that, since **Cat** is cartersian closed, every functor of type $\mathcal{C}^{\mathrm{op}} \times \to \mathbf{Set}$ corresponds to a functor of type $\mathcal{C} \to \mathbf{Set}^{\mathcal{C}^{\mathrm{op}}}$. In particular, the hom-functor $\mathcal{C}(-,-)$ can also be thought of a functor of this type and it is called the Yoneda embedding:

Definition 4.1 (The Yoneda Embedding)**.** The curried transpose of the hom-functor is denoted by $よ : \mathcal{C} \to \mathbf{Set}^{\mathcal{C}^{\mathrm{op}}}$, pronounced the Yoneda embedding. Also, we are going to denote $\mathcal{C}(-, C)$ for some C as $よ_C$ and $\mathcal{C}(-, f)$ for some f as $よ_f$.

Now since by definition of opposite categories $\mathcal{C}(-, C)$ is equal to $\mathcal{C}^{\mathrm{op}}(C,-)$ we can state the Yoneda lemma as follows:

Lemma 4.1. *Let $\mathcal{C}^{\mathrm{op}}$ be a locally small category and let $F : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ be a contravariant functor then the following isomorphism holds:*

$$FC \cong \mathbf{Set}^{\mathcal{C}^{\mathrm{op}}}(よ_C, F) \tag{17}$$

where $\mathbf{Set}^{\mathcal{C}^{\mathrm{op}}}(\text{よ}_C, F)$ *are the natural transformation from* $\mathcal{C}(-, C)$ *to* F.

The fact that よ is an embedding means that the y is a *fully faithful functor*, i.e. it is respectively surjective and injective on arrows. This leads to the Yoneda principle of indirect proof:

Lemma 4.2 (Yoneda Priciple)**.** *Let* $\mathcal{C}$ *be a locally small category, then for all* $X, Y : \mathcal{C}$*,*

$$X \cong Y \text{ if and only if } \text{よ}_X \cong \text{よ}_Y$$

The left-to-right is by definition of functor, the other direction falls out from functoriality of $\mathcal{C}(-,-)$ and from the fact this is a full functor.

Moreover, using the functorial action of よ and since よ is faithful we have that

$$f = g \text{ if and only if } \text{よ}_f = \text{よ}_g \tag{18}$$

The left-to-right is by the fact that the functor $\mathcal{C}(-,=)$ is well-defined (in both variables) and the other direction is from faithfullness.

4.1 Ninja Yoneda Lemmas

Using the coend calculus we can of course derive a multitude of equivalent formulations of Yoneda. Since these are instances of Yoneda in disguise they have been named Ninja Yonedas [5].

Lemma 4.3 (Ninja Yoneda)**.** *Let* $\mathcal{C}^{\mathrm{op}}$ *be a locally small category and* $F : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ *then the following isomorphisms hold:*

$$\int_{X:\mathcal{C}} \mathbf{Set}(\mathcal{C}(X, C), FX) \cong FC \qquad \int^{X:\mathcal{C}} \mathcal{C}(X, C) \times FX \cong FC$$

Proof. The proof of the left-hand side is easy application of (4)

$$\int_{X:\mathcal{C}} \mathbf{Set}(\mathcal{C}(X, C), FX \cong \mathcal{C}^{\mathrm{op}}(-, C) \Rightarrow F- \cong FC$$

The right-hand side needs some more work. We first prove the following isomorphism

$$\mathbf{Set}(\int^{X:\mathcal{C}} \mathcal{C}^{\mathrm{op}}(X, C) \times FX, Y) \cong \mathbf{Set}(FC, Y)$$

for all sets Y. Then by the Yoneda principle (18) using the homset functor $\mathbf{Set}(-, Y) : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Set}$ we derive

$$\mathbf{Set}(\int^{X:\mathcal{C}} \mathcal{C}(X, C) \times FX \cong FC$$

as wanted. We now prove (4.1):

$$
\begin{aligned}
& \mathbf{Set}(\int^{X:\mathcal{C}} \mathcal{C}(X,C) \times FX, Y) \\
& \cong \{ \text{ By application of (4) } \} \\
& \int_{X:\mathcal{C}} \mathbf{Set}(\mathcal{C}(X,C) \times FX, Y) \\
& \cong \{ \text{ Using the fact that } \mathbf{Set} \text{ is cartesian closed } \} \\
& \int_{X:\mathcal{C}} \mathbf{Set}(\mathcal{C}(X,C), \mathbf{Set}(FX, Y)) \\
& \cong \{ \text{ By applying (4) } \} \\
& \mathcal{C}(-,C) \Rightarrow \mathbf{Set}(F-, Y) \\
& \cong \{ \text{ Using the Yoneda lemma 4.1 } \} \\
& \mathbf{Set}(FC, Y)
\end{aligned}
$$

□

An immediate consequence of the Yoneda lemma is its instantiation to second-order functors.

Lemma 4.4. *Let $F : \mathcal{C}^{\mathcal{D}} \to \mathbf{Set}$ be a $\mathbf{Set}$-valued functor from the category of functors $\mathcal{D} \to \mathcal{C}$ and let $\mathcal{C}^{\mathcal{D}}$ be a locally small category and let H be an object in $\mathcal{C}^{\mathcal{D}}$. Then we can state the second-order Yoneda Lemma by just instantiating (*4.1*):*

$$GH \cong \mathcal{C}^{\mathcal{D}}(H,-) \Rightarrow G$$

By Equation(16) this equation becomes

$$GH \cong \int_F \mathcal{C}^{\mathcal{D}}(H,F) \Rightarrow GF \tag{19}$$

Let now $GH = HX$ for some $X : \mathcal{D}$, then

$$HX \cong \int_F \mathcal{C}^{\mathcal{D}}(H,F) \Rightarrow FX \tag{20}$$

This will be used for the proof of traversals in Section 10.

4.2 The mini Yoneda Lemma for Type Theorists

Say that you have a typed language with a unary constructor `R` which has the following typing rule

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathtt{R}(t) : B} \tag{21}$$

The task is to give a semantic interpretation $[\![\cdot]\!]$ for the language by induction on the typing judgment $\Gamma \vdash t : A$ such that terms are interpreted as morphisms $[\![\Gamma]\!] \xrightarrow{[\![t]\!]} [\![A]\!]$, assuming for course $[\![\cdot]\!]$ is also defined separately for contexts and types.

In order to interpret (21), as mentioned above, we do induction on the typing judgment so that, by induction, we know there exists a morphism $[\![\Gamma]\!] \xrightarrow{[\![t]\!]} [\![A]\!]$ and we have to construct a morphism $[\![\Gamma]\!] \xrightarrow{[\![\mathtt{R}(t)]\!]} [\![B]\!]$.

In the remainder of this section we remove the semantics brackets for simplicity, for example, assuming A be interpretation of $[\![A]\!]$, $t : \Gamma \to A$ the interpretation of t an so on.

In this situation it can be quite tricky sometimes to figure out what this moprhism should be since there is some plumbing needed to pass around the context. A particular instantiation of the Yoneda lemma states that given a morphism $t : \Gamma \xrightarrow{t} A$ and a morphism $R : A \to B$ there is a canonical way to construct a morphism $\Gamma \xrightarrow{R(t)} B$.

To show this we instantiate the Yoneda lemma 4.1 by setting $F = \mathcal{C}(-, B)$. Then for all objects $A : \mathcal{C}^{\mathrm{op}}$ we have

$$\mathcal{C}(A, B) \cong \mathcal{C}(-, A) \Rightarrow \mathcal{C}(-, B)$$

Let $R : A \to B$ be the interpretation of $\mathtt{R}$ then, one side of the isomorphism is $\phi(R, t) = F(t)(R) = \mathcal{C}([\![t]\!], B)(R)$. In other words, the interpretation of $R(t)$ is simply $R \circ t$.

4.3 Exponentials in Presheaf Categories

Say we are working in the presheaf category over $\mathcal{C}$, namely $\mathbf{Set}^{\mathcal{C}^{\mathrm{op}}}$.

We want to find out what the exponential in this category is. It may be tempting to define the exponential as the natural transformations, but this is not what is happening. If the exponential existed then it would be an object in $\mathbf{Set}^{\mathcal{C}^{\mathrm{op}}}$ and at this point we would be able to apply the Yoneda lemma instantiating F with A^B in (4.1). We then calculate as follows using properties of ends and exponentials:

$$\begin{aligned}
B^A(X) &\cong \{ \text{ Yoneda Lemma (4.1) } \} \\
&\quad \mathcal{C}(-, X) \Rightarrow B^A \\
&\quad \cong \{ \text{ Exponentials in functor categories } \} \\
&\quad \mathcal{C}(-, X) \times A \Rightarrow B \\
&\quad \cong \{ \text{ Ends as natural transformations (16) } \} \\
&\quad \int^{C:\mathcal{C}} \mathbf{Set}(\mathcal{C}(C, X) \times A(C), B(C))
\end{aligned}$$

Thus the exponential at stage X is the set natural transformations from the functor A to the functor B, but restricted to the components C such that there exists at least one arrow $C \to X$.

5 Kan Extensions

Consider the category $\mathcal{C}$ formed by these objects and arrows

$$A \to C \leftarrow B$$

and the category $\mathcal{D}$ formed by the following objects and arrows

$$\begin{tikzcd}
D \arrow[r] \arrow[d] & A \arrow[d] \\
B \arrow[r] & C
\end{tikzcd}$$

Clearly, $\mathcal{C}$ is a full subcategory of $\mathcal{D}$, indicated by the presence of the inclusion functor $i : \mathcal{C} \hookrightarrow \mathcal{D}$ since A, B and C contained in $\mathcal{D}$ along with their associated arrows. Now given a functor $F : \mathcal{C} \to \mathcal{E}$ how can we extend the functor to the category $\mathcal{D}$ such that this functor agrees with F on the objects and arrows that already belong to $\mathcal{C}$? Well notice that this functor needs to send D to an object GD which has arrows $GD \to FX = GX$ for each $X \in \mathcal{C}$ for which there is an arrow $D \to X$. A cone for the following diagram will do since D has an arrow into every object of $\mathcal{C}$, but since we will need to prove this object is the "maximal" one we choose the limiting cone of this diagram which is the pullback P

$$\begin{tikzcd}
P \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & FA \arrow[d] \\
FB \arrow[r] & FC
\end{tikzcd}$$

Now we need to ensure that this assignment is the "largest" one we could have picked. To do this we pick a generalised element of F, namely, for a functor $H : \mathcal{D} \to \mathcal{E}$ and for a generalised element $f : H \circ i \Rightarrow F$ we need to show there exists a unique map $H \Rightarrow G$ such that on objects in $\mathcal{C}$ this map agrees with f[1]. Now, for $X \in \mathcal{C}$, this is just f since $GX = FX$. We are left to show that there exists a map $HD \to GD = P$, but this is easy to see since HD is a cone for the functor F

$$\begin{tikzcd}
HD \arrow[r] \arrow[d] \arrow[rr, bend left, dotted] & HA \arrow[d] \arrow[rr, bend left, "f_A"] & P \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & FA \arrow[d] \\
HB \arrow[r] \arrow[rr, bend right, "f_B"'] & HC \arrow[rr, bend right, "f_C"'] & FB \arrow[r] & FC
\end{tikzcd}$$

Let us generalise this a bit further. Let $\mathcal{C} \hookrightarrow \mathcal{D}$ be a full subcategory of $\mathcal{D}$. Let $i : \mathcal{C} \to \mathcal{D}$ be the inclusion functor and let a functor $F : \mathcal{C} \to \mathcal{E}$. We want to *extend* this functor to a functor $G : \mathcal{D} \to \mathcal{E}$ such that for every object $X \in \mathcal{C}$, G sends X to FX.

[1]For the attentive reader, this conditions is the universal property of the counit required for G to be the right adjoint to the functor $- \circ i$

Since G has to be a functor we need for every morphism $Y \to Y'$ in $\mathcal{D}$ to define what the functorial action of G is. Now let us restrict to the case when Y' is an $X \in \mathcal{C}$. When $Y \in \mathcal{D}$, but $Y \notin \mathcal{C}$, for every morphism $Y \to X$ in $\mathcal{D}$, the functorial action of G has to be of type $GY \to FX$, hence we define GY has the object that has morphisms into every object FX that has a map $Y \to X$. In particular, we want to take the universal such cone which is the limit of a functor $F \circ \pi_1 : Y/i \to \mathcal{E}^{\mathcal{C}}$ where Y/i is the comma category formed of triples $(Y \in \mathcal{D}, X \in \mathcal{C}, f : Y \to iX)$ and π is the projection functor $\pi : Y/i \to \mathcal{C}$

$$GY = \varprojlim_{(Y,X,f:Y\to iX)} (F \circ \pi)(Y, X, f : Y \to iX) = \varprojlim_{(Y,X,f:Y\to X)} FX$$

By (12) and since this functor is covariant this limit is isomorphic to the following end formula

$$GY \cong \int_{(Y,X,f:Y\to X)} FX$$

which is the end of the functor $F \circ \pi \circ \pi_2$.

5.1 Kan Extensions

Consider a reindexing functor $J : \mathcal{C} \to \mathcal{D}$ and define the functor $- \circ J : \mathcal{E}^{\mathcal{D}} \to \mathcal{E}^{\mathcal{C}}$ also named App_J.

We want to find out whether App_J has a left and right adjoint. We call its left and right adjoint Lan_J and Ran_J respectively.

$$\begin{tikzcd} \mathcal{E}^{\mathcal{C}} \arrow[r, bend right, "\mathrm{Ran}_J"'] \arrow[r, phantom, "\perp"] & \mathcal{E}^{\mathcal{D}} \arrow[l, bend right, "\mathrm{App}_J"'] \arrow[r, bend right, "\mathrm{App}_J"'] \arrow[r, phantom, "\perp"] & \mathcal{E}^{\mathcal{C}} \arrow[l, bend right, "\mathrm{Lan}_J"'] \end{tikzcd}$$

Now because Lan_J is the left adjoint and Ran_J to be the right one we would expect the following to be a natural isomorphisms

$$\mathcal{E}^{\mathcal{D}}(\mathrm{Lan}_J A, B) \cong \mathcal{E}^{\mathcal{C}}(A, \mathrm{App}_J B) \qquad \mathcal{E}^{\mathcal{C}}(\mathrm{App}_J A, B) \cong \mathcal{E}^{\mathcal{D}}(A, \mathrm{Ran}_J B)$$

In the MacLane [6] he is defining the right Kan extension and the proving it is the right adjoint. Here we take a different approach. By using the Yoneda lemma we derive the right adjoint which is unique up to isomorphism so it must be the right Kan extension. This proof is taken form Hinze's work on generic programming [4]. Here I have made it a bit more precise.

$$
\begin{aligned}
\mathcal{E}^{\mathcal{C}}(\mathrm{App}_J A, B) &= \{ \text{ Homsets in the exponential category are natural transformations } \} \\
& \mathrm{Nat}(\mathrm{App}_J A, B) \\
& \cong \{ \text{ Natural transformations are ends (16) } \} \\
& \int_{X:\mathcal{C}} \mathcal{E}(AJX, BX) \\
& \cong \{ \text{ Yoneda with } \mathcal{E}(A-, BX) \text{ } \} \\
& \int_{X:\mathcal{C}} \mathrm{Nat}(\mathcal{D}(-, JX), \mathcal{E}(A-, BX)) \\
& \cong \{ \text{ by (16) } \} \\
& \int_{X:\mathcal{C}} \int_{Y:\mathcal{D}} \mathcal{D}(Y, JX) \to \mathcal{E}(AY, BX) \\
& \cong \{ \text{ by (8) } \} \\
& \int_{X:\mathcal{C}} \int_{Y:\mathcal{D}} \mathcal{E}(AY, BX)^{\mathcal{D}(Y, JX)} \\
& \cong \{ \text{ by (6) } \} \\
& \int_{X:\mathcal{C}} \int_{Y:\mathcal{D}} \mathcal{E}(AY, BX^{\mathcal{D}(Y, JX)}) \\
& \cong \{ \text{ by (13) } \} \\
& \int_{Y:\mathcal{D}} \int_{X:\mathcal{C}} \mathcal{E}(AY, BX^{\mathcal{D}(Y, JX)}) \\
& \cong \{ \text{ Homsets preserve ends (14) } \} \\
& \int_{Y:\mathcal{D}} \mathcal{E}(AY, \int_{X:\mathcal{C}} BX^{\mathcal{D}(Y, JX)}) \\
& \cong \{ \text{ Natural transformation are ends (16) } \} \\
& \mathrm{Nat}(A-, \int_{X:\mathcal{C}} BX^{\mathcal{E}(-, JX)}) \\
& \cong \{ \text{ Homsets in the exponential category are natural transformations } \} \\
& \mathcal{E}^{\mathcal{D}}(A-, \int_{X:\mathcal{C}} BX^{\mathcal{E}(-, JX)})
\end{aligned}
$$

For all functors $J : \mathcal{C} \to \mathcal{D}$, $A : \mathcal{C} \to \mathcal{E}$ and $B : \mathcal{D} \to \mathcal{E}$

$$\mathrm{Ran}_J AY = \int_{X \in \mathcal{C}} \Pi_{\mathcal{D}(Y, JX)} AX \tag{22}$$

We now compute the left Kan extension. I could not find this proof is not in [4, 6].

$$
\begin{aligned}
\mathcal{E}^{\mathcal{C}}(A, \mathrm{App}_J B) &= A \Rightarrow \mathrm{App}_J B \\
&\cong \{ \text{ Natural transformations are ends (16) } \} \\
&\int_{X:\mathcal{C}} \mathcal{E}(AX, BJX) \\
&\cong \{ \text{ by Yoneda with } \mathcal{E}(AX, B-) \ \} \\
&\int_{X:\mathcal{C}} \mathcal{D}(JX, -) \Rightarrow \mathcal{E}(AX, B-) \\
&\cong \{ \text{ Natural transformations are ends(16) } \} \\
&\int_{X:\mathcal{C}} \int_{Y:\mathcal{D}} \mathcal{D}(JX, Y) \to \mathcal{E}(AX, BY) \\
&\cong \{ \text{ The set functions space is a power (8) } \} \\
&\int_{X:\mathcal{C}} \int_{Y:\mathcal{D}} \mathcal{E}(AX, BY)^{\mathcal{D}(JX,Y)} \\
&\cong \{ \text{ Homsets revert powers into copowers (7) } \} \\
&\int_{X:\mathcal{C}} \int_{Y:\mathcal{D}} \mathcal{E}(\mathcal{D}(JX, Y) \bullet AX, BY) \\
&\cong \{ \text{ Switching over ends (13) } \} \\
&\int_{Y:\mathcal{D}} \int_{X:\mathcal{C}} \mathcal{E}(\mathcal{D}(JX, Y) \bullet AX, BY) \\
&\cong \{ \text{ Homsets reverse ends into coends (15) } \} \\
&\int_{Y:\mathcal{D}} \mathcal{E}(\int^{X:\mathcal{C}} \mathcal{D}(JX, Y) \bullet AX, BY) \\
&\cong \{ \text{ Natural transformations are ends (16) } \} \\
&\mathrm{Nat}(\int^{X:\mathcal{C}} \mathcal{D}(JX, -) \bullet AX, B) \\
&= \mathcal{E}^{\mathcal{D}}(\int^{X:\mathcal{C}} \mathcal{D}(JX, -) \bullet AX, B)
\end{aligned}
$$

Now by looking at what we have got we define the left Kan extension as a functor parametrised by A

$$\mathrm{Lan}_J AY = \int^{X \in \mathcal{C}} \mathcal{D}(JX, Y) \bullet AX \tag{23}$$

When $\mathcal{D}$ and $\mathcal{E}$ are the category **Set**, the left Kan extension can be rewritten as follows:

$$\mathrm{Lan}_J AY = \int^{X \in \mathcal{C}} (JX \to Y) \times AX \tag{24}$$

by the fact that, in **Set**, copowers are products (9) and homsets are exponentials.

5.2 Yoneda as Kan Extensions

Now that we know a bit more about Kan extensions and their definitions we can revisit the Yoneda lemma.

Lemma 5.1. *If $\mathcal{E}$ is complete and $\mathcal{C}$ is small then for all functors $F : \mathcal{C} \to \mathcal{E}$ we have the following isomorphism:*

$$F \cong \mathrm{Ran}_{\mathrm{Id}} F$$

Proof. By direct consequence of the adjunction $\mathrm{App}_{\mathrm{Id}} \dashv \mathrm{Ran}_{\mathrm{Id}}$ we have the following natural isomorphism

$$\mathcal{E}^{\mathcal{C}}(\mathrm{App}_{\mathrm{Id}} G, F) \cong \mathcal{E}^{\mathcal{C}}(G, \mathrm{Ran}_{\mathrm{Id}} F)$$

Notice that, by definition of $\mathrm{App}_{\mathrm{Id}}$, we have $\mathcal{E}^{\mathcal{C}}(\mathrm{App}_{\mathrm{Id}} G, F) = \mathcal{E}^{\mathcal{C}}(G, F)$ which means the previous statement can be rewritten as

$$\mathcal{E}^{\mathcal{C}}(G, F) \cong \mathcal{E}^{\mathcal{C}}(G, \mathrm{Ran}_{\mathrm{Id}} F)$$

Notice that the isomorphism above is natural in both G and F and, in particular, in G, hence we can rewrite the previous isomorphism as follows using the Yoneda embedding:

$$よ_F \cong よ_{\mathrm{Ran}_{\mathrm{Id}} F}$$

At this point, by the fact that the Yoneda embedding is fully faithful (Lemma 4.2) we have

$$F \cong \mathrm{Ran}_{\mathrm{Id}} F$$

concluding the proof. □

Yoneda in the Presheaves Case For presheaf categories over a small category $\mathcal{C}$ we could use Lemma 5.1 since **Set** is complete. However, it is useful to prove the result again using the ninja Yoneda lemma which holds since $\mathcal{C}$ is also locally small. We compute as follows:

$FC \cong \displaystyle\int_{X:\mathcal{C}} \mathbf{Set}(\mathcal{C}(C, X), FX)$

{ The homset in **Set** is the function space }

$\cong \displaystyle\int_{X:\mathcal{C}} \mathcal{C}(C, X) \to FX$

{ The function space is the power, with $\mathcal{C}(C, X)$ seeing as a discrete category }

$\cong \displaystyle\int_{X:\mathcal{C}} FX^{\mathcal{C}(C,X)}$

{ By (22) }

$\cong \mathrm{Ran}_{\mathrm{Id}} FC$

5.3 CoYoneda as Kan Extensions

Lemma 5.2 (CoYoneda)**.** *Let $\mathcal{E}$ be cocomplete and $\mathcal{C}$ be small. Then for all functors $F : \mathcal{C} \to \mathcal{E}$ the following isomorphism holds:*

$$F \cong \mathrm{Lan}_{\mathrm{Id}} F$$

Proof. Consider the adjunction $\mathrm{Lan}_{\mathrm{Id}} \dashv \mathrm{App}_{\mathrm{Id}}$ given by the left Kan extension. From the adjunction we know that the following isomorphism is natural in both F and G:

$$\mathcal{E}^{\mathcal{C}}(\mathrm{Lan}_{\mathrm{Id}} F, G) \cong \mathcal{E}^{\mathcal{C}}(F, \mathrm{App}_{\mathrm{Id}} G)$$

By definition of $\mathrm{App}_{\mathrm{Id}}$ we know that $\mathcal{E}^{\mathcal{C}}(F, G \circ \mathrm{Id}) = \mathcal{E}^{\mathcal{C}}(F, G)$. Hence, we rewrite the above isomorphism using the Yoneda embedding:

$$よ(\mathrm{Lan}_{\mathrm{Id}} F) \cong よ(F)$$

By (4.2) this implies

$$\mathrm{Lan}_{\mathrm{Id}} F \cong F$$

which concludes the computation. □

Remark. When $F : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ this is a again a trivial consequence of the ninja Yoneda Lemma 4.3.

5.4 Left and Right Shifts

Left and right shifts are just particular cases of Kan extensions where $\mathcal{C}$ is 1. So now the reindexing functor is just an object $D : 1 \to \mathcal{D}$. If we now start abusing some notation setting D to mean $D(*)$ we have $\mathrm{App}_D H = H \circ D = HD(*) = HD$

$$\begin{tikzcd}
\mathcal{E} \arrow[r, shift left, "\mathrm{Rsh}_D"'] \arrow[r, phantom, "\perp"] & \mathcal{E}^{\mathcal{D}} \arrow[l, shift left, "-D"'] \arrow[r, shift left, "-D"'] \arrow[r, phantom, "\perp"] & \mathcal{E} \arrow[l, shift left, "\mathrm{Lsh}_D"']
\end{tikzcd}$$

At this point the natural isomorphisms induces by the adjunctions are as follows

$$\mathcal{E}^{\mathcal{D}}(\mathrm{Lsh}_D F, G) \cong \mathcal{E}(F, GD) \qquad \mathcal{E}(FD, G) \cong \mathcal{E}^{\mathcal{D}}(F, \mathrm{Rsh}_J G)$$

What is interesting to note is that left and right Kan extensions simplify into left and right shifts

$$\mathrm{Lsh}_D F Y = \mathcal{D}(D, Y) \bullet F \qquad \mathrm{Rsh}_D G Y = G^{\mathcal{D}(Y,D)}$$

6 Distributive Laws, (Co)Algebras and Kan Extensions

Let $F : \mathcal{C} \to \mathcal{C}$ and $G : \mathcal{C} \to \mathcal{C}$ be two endofunctors.

A distributive law of the form

$$FG \Rightarrow GF$$

corresponds to an Γ-algebra $\Gamma G \Rightarrow G$ where $\Gamma : [\mathcal{C}, \mathcal{C}] \to [\mathcal{C}, \mathcal{C}]$ is defined as

$$\Gamma G = \mathrm{Lan}_F(FG)$$

The proof of this fact is immediate by realising that $FG \Rightarrow GF$ is in one-to-one correspondence with natural transformations $\mathrm{Lan}_F FG \Rightarrow F$ using the fact that Lan_F is left-adjoint to the functor $\mathrm{App}_F G = G \circ F$.

7 Monads from Kan Extensions

7.1 The Codensity Monad

The codensity monad is just the right Kan extension of J along J

$$\mathrm{Cod}\ JX = \mathrm{Ran}_J JX = \int_{Y:\mathcal{C}} JY^{\mathcal{D}(X, JY)}$$

7.2 The Codensity Transformation

If $L \dashv R$ then both $L \circ - \vdash R \circ -$ and $- \circ R \vdash - \circ L$ are adjunctions.

$$\text{If} \quad \begin{tikzcd} \mathcal{L} \arrow[r, bend right, "R"'] \arrow[r, phantom, "\perp"] & \mathcal{R} \arrow[l, bend right, "L"'] \end{tikzcd} \quad \text{then} \quad \begin{tikzcd} \mathcal{E}^{\mathcal{L}} \arrow[r, bend right, "- \circ L"'] \arrow[r, phantom, "\perp"] & \mathcal{E}^{\mathcal{R}} \arrow[l, bend right, "- \circ R"'] \end{tikzcd}$$

Because of this fact there is a natural isomorphism

$$\mathcal{E}^{\mathcal{L}}(F \circ R, G) \cong \mathcal{E}^{\mathcal{R}}(F, G \circ L)$$

Now, since $F \circ R$ is $\mathrm{App}_R F$ then $\mathcal{E}^{\mathcal{L}}(F \circ R, G) \cong \mathcal{E}^{\mathcal{R}}(F, \mathrm{Ran}_R G)$. But then we know also that

$$\mathcal{E}^{\mathcal{L}}(F, G \circ L) \cong \mathcal{E}^{\mathcal{R}}(F, \mathrm{Ran}_R G)$$

Since the Yoneda embedding $\mathcal{E}^{\mathcal{R}}(-, -) : \mathcal{C} \to \mathbf{Set}^{\mathcal{C}^{\mathrm{op}}}$ is fully faithful, by Lemma 4.2 then $G \circ L \cong \mathrm{Ran}_R G$. (But this is also the proof that adjoints are unique up-to isomorphism). Now, if $G = R$ then we get that the monad

$$R \circ L \cong \mathrm{Ran}_R R \tag{25}$$

8 On Free Monads

Given an endofunctor $F : \mathcal{C} \to \mathcal{C}$ the free monad M over F is the monad freely generated by the constructors $\eta : \mathrm{Id} \to M$ and $\mathrm{op} : FM \to M$. The fact that is a monad implies it must have a join operation as well $\mu : MM \to M$. This monad M is also indicated by F^*.

When it exists, the free monad is the least solution to the equation

$$F^*A \cong A + FF^*A$$

It can be shown that this is the free F-algebra in the sense that it is the monad arising from a free-forgetful adjunction

$$\begin{tikzcd} F\text{-Alg} \arrow[r, bend right, "U"'] \arrow[r, phantom, "\perp"] & \mathcal{C} \arrow[l, bend right, "\text{Free}"'] \arrow[loop right, "F*"] \end{tikzcd}$$

where $F^* = U\text{Free}$.

The ceiling and floor witness the natural isomorphism on both A and B

$$\lfloor\cdot\rfloor : F\text{-Alg}(\text{Free } A, B) \cong \mathcal{C}(A, UB) : \lceil\cdot\rceil$$

As usual, the unit of the monad is given by $\eta = \lfloor id_{\text{Free}A}\rfloor$ and the multiplication is derived by using the counit $\mu = U\epsilon_{\text{Free}}$. Now the map $FF^*A \to F^*A$ is the F-algebra given by Free$A$. To see this let F^*A to be UFree$A$ (it's just a name). Say that Free takes an object $A : \mathcal{C}$ an sends it to an F-algebra, Free$A = (X, \text{alg} : FX \to X)$ for some X and some algebras alg. But that means that $U\text{Free}A = X$ which implies X is F^*A and that alg is the map $\text{op} : FF^*A \to F^*A$ we were looking for.

8.1 Free Monads and The Right Kan Extension

From the previous section we know that every monad arising from an adjunction $L \dashv R$ (hence every monad!) is isomorphic to the right Kan extension of R along R, denoted by $\text{Ran}_R R$ and also known as the *codensity monad*.

Since the free monad is a monad, also the free monad can be transformed using the codensity transformation. Since the algebraically free monad factors as $F^* = U\text{Free}$ then by (25)

$$F^* \cong \text{Ran}_U U$$

By unfolding the definitions we get that

$$F^*A \cong \int_{Z:F\text{-Alg}} UZ^{\mathcal{C}(A,UZ)} \tag{26}$$

However, there is another way to transform the free monad. Assuming $F^*A \cong A + FF^*A$ we can compute as follows using the Yoneda lemma (5.1)

$$F^*A \cong A + FF^*A \cong A + (\text{Ran}_{\text{Id}}F)F^*X \cong A + \int_{X:\mathcal{C}} FX^{\mathcal{C}(F^*A,X)} \tag{27}$$

8.2 Free Monads and the Left Kan Extension

Assume a category $\mathcal{C}$, an object $A \in \mathcal{C}$ and a functor $F : \mathcal{C} \to \mathcal{C}$. Then the free monad $F^* : \mathcal{C} \to \mathcal{C}$ can be transformed as follows using the CoYoneda Lemma

(5.2)

$$F^* A \cong A + F F^* A \cong A + (\mathrm{Lan}_{\mathrm{Id}} F) F^* A \cong A + \int^{X:\mathcal{C}} \mathcal{C}(X, F^* A) \bullet F X \qquad (28)$$

In **Set** this isomorphism becomes,

$$F^* A \cong A + \int^{X:\mathbf{Set}} F X \times (X \to F^* A)$$

since in **Set** copowers are pairs and homsets are function spaces as in Section 2.2.1.

Remark. On Agda's Free Monad's Hack It is well-known that in Agda not all endofunctors in `Set` have a fixed-point and more specifically not all functors F yield a free monad F^*. Hence the following code is rejected by Agda's type system:

```
data Free (f : Set → Set) (A : Set) : Set where
  Var : A → Free f A
  Com : f(Free f A) → Free f A
```

More intuitively, if Agda allowed this data type then it would allow fixed-points for all endofunctors since F^*0 is isomorphic to the least fixed-point of the functor F, namely μF. I bet you could prove in fact that the following data type is isomorphic to `Free f ∅`:

```
data Fix (f : Set → Set) : Set where
  In : f (Fix f) → Fix f
```

This is obviously breaking Agda's consistent type system.

To circumvent this problem Agda programmers take insipiration from Equation 28 and define the free monad using the following data type instead

```
data Free (f : Set₀ → Set₀) (A : Set₀) : Set₁ where
  Var : A → Free f A
  Call : Σ Set₀  (λ X → f X × (X → Free f A)) → Free f A
```

However! this is not the equivalent of the free monad over the functor F, for all functors F. If it was then it would yield its fixed-point as well. The trick employed here is much more subtle and uses the fact that the free monad lives in higher universe. In particular, `Free f A` lives in `Set₁` while `f` is of type `Set₀ → Set₀`. Hence the program `f (Free f ∅)` is not even well-typed which means that `Free f ∅` cannot be the fixed-point of `f`.

8.3 Algebras for the Left Kan extension

Every G-algebra is isomorphic to the algebras for the left Kan extension on G.

$$\begin{aligned}
\mathcal{D}(\mathrm{Lan}_{\mathrm{Id}}GA, A) &\cong \mathcal{D}^{\mathcal{D}}(\mathrm{Lan}_{\mathrm{Id}}G, \mathrm{Rsh}_A A) \qquad \{ \text{ by } -A \dashv \mathrm{Rsh}_{\mathrm{Id}} \ \} \\
&\cong \mathcal{D}^{\mathcal{D}}(G, \mathrm{Rsh}_A A \circ \mathrm{Id}) \qquad \{ \text{ by } \mathrm{Lan}_{\mathrm{Id}} \dashv \mathrm{App}_{\mathrm{Id}} \ \} \\
&\cong \mathcal{D}^{\mathcal{D}}(G, \mathrm{Rsh}_A A) \\
&\cong \mathcal{D}(GA, A) \qquad \{ \text{ by } -A \dashv \mathrm{Rsh}_{\mathrm{Id}} \ \}
\end{aligned}$$

This proof is worth of reminding, but a simpler way to do it is to use CoYoneda (5.2) directly

$$\mathcal{D}(\mathrm{Lan}_{\mathrm{Id}}GA, A) \cong \mathcal{D}(GA, A)$$

8.4 The Freest Monad

Given a functor $J : \mathcal{C} \to \mathcal{D}$ and an endofunctor $F : \mathcal{C} \to \mathcal{D}$ the freest monad is defined as follows

$$F_J^{\mathrm{st}} A \cong JA + \mathrm{Lan}_J F(F^{\mathrm{st}} A)$$

Note that F is not an endofunctor and so the freest monad is in fact a *relative monad* [1]. The free monad over an endofunctor $F : \mathcal{C} \to \mathcal{C}$ is derivable by setting J to the identity functor

$$F^* A = F_{\mathrm{Id}}^{\mathrm{st}} A \cong \mathrm{Id}A + \mathrm{Lan}_{\mathrm{Id}} F(F^{\mathrm{st}} A) \cong A + F(F^{\mathrm{st}} A)$$

The last step is of course the coYoneda lemma (5.2). The freest monad cannot be derived from the free monad and I am not sure if it is the free monad in the sense I mentioned above.

8.5 A Free Monad for Scoped Effects

Another monad arising from Kan extensions is the one by Ghani et al. [3] and used to give semantics of scoped effect handlers [7].

The monad E is defined as follows

$$EA \cong A + \Sigma EA + \int^{X \in \mathcal{C}} \mathcal{C}(X, EA) \bullet \Gamma EX$$

where Σ is a signature functor and Γ is the signature functor for scoped effects. The interested reader can refer to the cited papers for reference. Here we are merely interested in showing that the coend computes the left Kan extension of ΓE along the identity applied to the object EA thereby yielding the following equivalent definition for E

$$EA \cong A + \Sigma EA + \Gamma EEA$$

In other words, this would show that a scoped effect needs two layers of syntax to be represented correctly. This is equivalent to saying that if we have a morphism $X \to EA$ and an object ΓEX we can compose via functoriality and obtain ΓEEA.

Here we want to make the calculations precise using the facts on Kan extensions derived above. We compute as follows:

$$\begin{aligned} EA &\cong A + \Sigma EA + \int^{X \in \mathcal{C}} \mathcal{C}(X, EA) \bullet \Gamma EX \\ &\cong \{ \text{ By Definition 23 } \} \\ &A + \Sigma EA + \mathrm{Lan}_{\mathrm{Id}}(\Gamma E)(EA) \\ &\cong \{ \text{ By Equation 5.2 } \} \\ &A + \Sigma EA + \Gamma EEA \end{aligned}$$

9 (Free) Applicative Functors

Definition 9.1 (Monoidal Category). A *monoidal* category is a category $\mathcal{C}$ equipped with a functor

$$\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$$

called the *tensor product* and an object $I \in \mathcal{C}$ called the *unit object* or *tensor unit* and such that the following arrows are natural isomorphisms in all variables:

- $(A \otimes B) \otimes C \xrightarrow{a_{A,B,C}} A \otimes (B \otimes C)$
- $I \otimes A \xrightarrow{\lambda_A} A$
- $A \otimes I \xrightarrow{\rho_A} A$

and such that the following diagrams commute for all objects involved:

$$\begin{tikzcd} (A \otimes I) \otimes B \arrow[rr, "a_{A,I,B}"] \arrow[dr, "A \otimes B"'] & & A \otimes (I \otimes B) \arrow[dl, "A \otimes \lambda_B"] \\ & A \otimes B & \end{tikzcd}$$

$$\begin{tikzcd} & (A \otimes B) \otimes (C \otimes D) \arrow[dr] & \\ ((A \otimes B) \otimes C) \otimes D \arrow[ur] \arrow[d] & & A \otimes (B \otimes (C \otimes D)) \\ (A \otimes (B \otimes C)) \otimes D \arrow[rr, "\alpha_{A,B \otimes C,D}"] & & A \otimes ((B \otimes C)) \otimes D) \arrow[u] \end{tikzcd}$$

An *applicative* functor is a functor $F : \mathcal{C} \to \mathcal{D}$ with the following morphisms

$$\begin{aligned} &\mathrm{pure}_A : A \to FA \\ &\circledast_{A,B} : F(B^A) \to FB^{FA} \end{aligned}$$

TODO (Laws).

A *lax monoidal* is a functor $F : \mathcal{C} \to \mathcal{D}$ with the following morphisms:

$$\begin{aligned} &\eta : I_{\mathcal{D}} \to FI_{\mathcal{C}} \\ &\mu_{A,B} : FA \otimes_{\mathcal{D}} FB \to F(A \otimes_{\mathcal{C}} B) \end{aligned}$$

together with the unital and associativity laws, while *strength* is a family of maps:

$$\mathrm{st}_{A,B} : A \times FB \to F(A \times B)$$

An lax closed functor $F : \mathcal{C} \to \mathcal{D}$ is a functor along with the following morphisms:

$$\begin{aligned} &\eta : I_{\mathcal{D}} \to FI_{\mathcal{C}} \\ &\circledast_{A,B} : F(B^A) \to FB^{FA} \end{aligned}$$

along with suitable equations.

Lemma 9.1. *Let $(\mathcal{C}, \otimes_{\mathcal{C}}, I_{\mathcal{C}})$ and $(\mathcal{D}, \otimes_{\mathcal{D}}, I_{\mathcal{D}})$be two monoidal closed categories and let $F : \mathcal{C} \to \mathcal{D}$ be a functor. The following statements about F are equivalent:*

1. *It is an applicative functor*
2. *It is a lax closed functor*
3. *It is a lax monoidal functor with strength*

Proof. ($1 \Rightarrow 2$) The unit map of the lax closed functor can be constructed simply by taking pure_I while the μ is the same as the one for the applicative functor. ($2 \Rightarrow 3$) The unit map is the same as the one from the lax closed structure. We construct the μ of the lax monoidal functor from the lax closed structure as follows:

$$\mu : FA \otimes_{\mathcal{D}} FB \xrightarrow{F\eta \otimes_{\mathcal{D}} FB} F(A \otimes_{\mathcal{C}} B)^B \otimes_{\mathcal{D}} FB \to F(A \otimes_{\mathcal{C}} B)$$

where the last step is the the uncurry of the lax closed structure and $\eta : A \to (A \otimes_D B)^B$ is the unit.

($3 \Rightarrow 1$). We construct the pure map as follows:

$$\mathrm{pure} : A \cong A \otimes_{\mathcal{C}} I \xrightarrow{A \otimes_{\mathcal{C}} \eta} A \otimes_{\mathcal{C}} FI \xrightarrow{\mathrm{st}_{A,I}} F(A \otimes_{\mathcal{C}} I) \cong FA$$

and the $\circledast$ as follows:

$$\circledast : FB^A \otimes_{\mathcal{D}} FA \to F(B^A \otimes_{\mathcal{C}} A) \to FB$$

□

The category of endofunctors $[\mathcal{C}, \mathcal{C}]$ is a monoidal category w.r.t. functor composition and the identity functor, this is denoted by $([\mathcal{C}, \mathcal{C}], \circ, \mathrm{Id})$. A monoid in this category is, in particular, a monad.

9.1 Functor Categories as Monoidal Categories

The category of functors on a monoidal category canonically inherits a monoidal category structure via the Day convolution product.

Definition 9.2 (Day Convolution). Let $(\mathcal{C}, \otimes, 1)$ be a small V-enriched monoidal category. Then the Day convolution tensor product on $\mathcal{V}^{\mathcal{C}}$

$$\otimes_{\text{Day}} : \mathcal{V}^{\mathcal{C}} \times \mathcal{V}^{\mathcal{C}} \to \mathcal{V}^{\mathcal{C}}$$

is defined as follows:

$$(F \otimes_{\text{Day}} G)A = \int^{XY:\mathcal{C}\times\mathcal{C}} \mathcal{C}(X \otimes_{\mathcal{C}} Y, A) \otimes_{\mathcal{V}} FX \otimes_{\mathcal{V}} GY$$

For a monoidal a small V-enriched category $(\mathcal{C}, \otimes_{\mathcal{C}}, I_{\mathcal{C}})$ the Day convolution product $\otimes_{\text{Day}}$ makes

$$([\mathcal{C}, \mathcal{V}], \otimes_{\text{Day}}, よ(I_{\mathcal{C}}))$$

a monoidal category with tensor unit $よ(I_{\mathcal{C}})$, the yoneda embedding applied to the unit $I_{\mathcal{C}}$, $\mathcal{C}(I, -) : \mathcal{C} \to \mathcal{V}$.

Lemma 9.2. *The following statements about $F : \mathcal{C} \to \mathcal{D}$ are equivalent:*

- *is lax closed with tensorial strength*
- *is a monoid in the monoidal category $(\mathcal{D}^{\mathcal{C}}, \otimes_{\text{Day}}, よ I)$*

Proof. (Sketch) An applicative functor $F : \mathcal{C} \to \mathcal{C}$ possesses the following arrows: □

9.2 Day Convolution as a Left Kan Extension

Lemma 9.3 (Day Convolution as a Left Kan Extension). *Let $(\mathcal{C}, \otimes_{\mathcal{C}}, I_{\mathcal{C}})$ be a monoidal category and let $F, G : \mathcal{C} \to \mathcal{V}$ The Day convolution is isomorphic to the left Kan extension of $F\overline{\otimes}G$ along the functor $\otimes_{\mathcal{C}} : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$:*

$$(F \otimes_{\text{Day}} G) \cong \text{Lan}_{\otimes_{\mathcal{C}}}(F\overline{\otimes}G)$$

where $(F\overline{\otimes}G)(X, Y) = FX \otimes_V GY$

Proof. By unfolding the definitions.

$$\begin{aligned}
& F \otimes_{\text{Day}} G \\
& = \{ \text{ By definition } \} \\
& \int^{X,Y:\mathcal{C}\times\mathcal{C}} \mathcal{C}(X \otimes_{\mathcal{C}} Y, -) \otimes_{\mathcal{V}} (FX \otimes_{\mathcal{V}} GY) \\
& \{ \text{ Using the definition of Lan with } (F\overline{\otimes}G)X = FX \otimes_{\mathcal{V}} GX \ \} \\
& = \text{Lan}_{\otimes_{\mathcal{C}}}(F\overline{\otimes}G)
\end{aligned}$$

□

Lemma 9.4 (Day Convolution as Products in $\mathbf{Set}^{\mathbf{Cat}^{\mathrm{op}}}$).

Definition 9.3 (Applicative Functor)*.* Let $(\mathcal{C}, \otimes_{\mathcal{C}}, 1_{\mathcal{C}})$ and $(\mathcal{D}, \otimes_{\mathcal{D}}, 1_{\mathcal{D}})$ be two monoidal categories. An applicative functor $F : \mathcal{C} \to \mathcal{D}$ is lax monoidal functor with tensorial strength.

9.3 Free Applicatives

10 On Traversals

In functional programming a *traversal* is, intuitively, a data type on which we can visit the whole data by using recursion, like for example a list or a tree. Take the following Haskell example for instance of a tree data structure: This data structure is a traversable in that, in addition to be a `Foldable`, we can take an effect F inside the list and fold the whole structure while running the effects inside it producing an effect that produces a tree:

```
sequence :: Applicative f => [f a] -> f [a]
sequence [] = pure []
sequence (x:xs) = (:) <$> x <*> (sequence xs)
```

10.1 Traversals as distributive laws

A traversable can also be described in terms of a function of the following type:

$$\mathtt{traverse}_{A,B} : \forall F : \text{Applicative}.\ (A \to FB) \times TA \to FTB$$

It can be easily proven using the theory presented here that traversals are in one-to-one correspondence with distributive laws:

$$\mathtt{sequence} : \forall F : \text{Applicative}.\ TF \Rightarrow FT$$

Let me make the definition of traversal more formal. We define the category of applicative endofunctors $\mathcal{F}$ from **Set** to **Set**. A traversal is then an endofunctor $T : \mathbf{Set} \to \mathbf{Set}$ such that there exists a family of maps:

$$\mathtt{traverse}_{A,B} : \int_{F:\mathcal{F}} \mathbf{Set}(FB^{A} \times TA, FTB)$$

We want to show this map is equivalent to a distributive law

$$\mathtt{sequence} : \int_{F:\mathcal{F}} \int_{X:\mathbf{Set}} \mathbf{Set}(TFX, FTX)$$

We compute as follows:

$$\int_{AB:\mathbf{Set}} \int_{F:\mathcal{F}} \mathbf{Set}(FB^A \times TA, FTB)$$

$$\cong \{ \text{ Commuting variables } \}$$

$$\int_{F:\mathcal{F}} \int_{A,B:\mathbf{Set}} \mathbf{Set}(FB^A \times TA, FTB)$$

$$\cong \{ \text{ Switching over the ends (13) } \}$$

$$\int_{F:\mathcal{F}} \int_{B,A:\mathbf{Set}} \mathbf{Set}(FB^A \times TA, FTB)$$

$$\cong \{ \text{ Ends into CoEnds (15) } \}$$

$$\int_{F:\mathcal{F}} \int_{B:\mathbf{Set}} \mathbf{Set}(\int^{A} FB^A \times TA, FTB)$$

$$\cong \{ \text{ By definition of Left Kan Extension (24) in } \mathbf{Set} \}$$

$$\int_{F:\mathcal{F}} \int_{B:\mathbf{Set}} \mathbf{Set}(\mathrm{Lan}_{\mathrm{Id}}\, T\, FB, FTB)$$

$$\cong \{ \text{ by CoYoneda (5.2) } \}$$

$$\int_{F:\mathcal{F}} \int_{B:\mathbf{Set}} \mathbf{Set}(TFB, FTB)$$

10.2 Traversals are Coalgebras for Free Applicatives

Define $R_{A,B}X = A \times (B \to X)$. Then we show the following result:

$$A \to FB \cong R_{A,B} \Rightarrow F \tag{29}$$

The proof is using the Yoneda Lemma:

$$\begin{aligned}
& A \to FB \\
& \cong \{ \text{ Applying Yoneda on } FB \ \} \\
& A \to \int_{X:\mathcal{D}} \mathbf{Set}(B, X) \to FX \\
& \cong \{ \text{ By (14) } \} \\
& \int_{X:\mathcal{D}} A \to \mathbf{Set}(B, X) \to FX \\
& \cong \{ \text{ By (1) } \} \\
& \int_{X:\mathcal{D}} A \times (\mathbf{Set}(B, X) \to FX) \\
& \cong \{ \text{ By Definition } \} \\
& \int_{X:\mathcal{D}} R_{A,B}X \to FX \\
& \cong \{ \text{ by (16) } \} \\
& R_{A,B} \Rightarrow F
\end{aligned}$$

We now prove that

$$\int_{F:\mathrm{App}} (A \to U(F)B) \to U(F)X \cong UR^*_{A,B}X \tag{30}$$

Here's the proof using the higher-order Yoneda lemma and the fact that $(-)^* \dashv U : \mathrm{Endo}(\mathcal{C}) \to \mathrm{App}_(\mathcal{C})$ between the endofunctor category $\mathrm{Endo}(\mathcal{C})$ into the category $\mathrm{App}(\mathcal{C})$ of applicative functors:

$$\begin{aligned}
& \int_{F:\mathrm{App}} (A \to U(F)B) \to U(F)X \\
& \cong \{ \text{ By previous lemma (29) } \} \\
& (\int_{F:\mathrm{App}} (R_{A,B} \Rightarrow U(F)) \to U(F)X \\
& \cong \{ \text{ Using the adjunction } (-)^* \dashv U \ \} \\
& (\int_{F:\mathrm{App}} (R^*_{A,B} \Rightarrow F) \to U(F)X \\
& \cong \{ \text{ By Definition of Nat. Transformations } \} \\
& (\int_{F:\mathrm{App}} \mathbf{Set}^{\mathcal{C}}(R^*_{A,B}, F) \to U(F)X \\
& \cong \{ \text{ By application of 4.4 with } GH = U(H)X,\ H = R^*_{A,B} \ \} \\
& U(R^*_{A,B})X
\end{aligned}$$

We now prove that a traversable is a *parametrised* R^*-coalgebra on the carrier TA. More precisely the statement reads:

$$\int_{F:\mathrm{App}} (A \to U(F)B) \to TA \to U(F)TB \cong TA \to U(R^*_{A,B})TB \tag{31}$$

The proof is simple:

$$\begin{aligned}
&\int_{F:\mathrm{App}} (A \to U(F)B) \to TA \to U(F)TB \\
&\cong \{ \text{ Swapping the arguments for the function space } \} \\
&TA \to \int_{F:\mathrm{App}} (A \to U(F)B) \to U(F)TB \\
&\cong \{ \text{ By previous lemma (30) } \} \\
&TA \to U(R^*_{A,B})TB
\end{aligned}$$

10.3 Containers

Consider an endofunctor $F : \mathbf{Set} \to \mathbf{Set}$. The initial F-algebra (when it exists) is denoted by $(\mu F, \mathrm{In})$ and it is the least fixed-point of F with $\mathrm{In}^{\mathrm{op}} : F\mu F \to \mu F$ witnessing the isomorphism $\mu F \cong F\mu F$.

The problem is that not every functor has a fixed-point. For this reason, when doing programming languages theory, we often restrict ourselves to polynomial functors. For families of objects $\{A_i\}_{i\in\underline{n}}$ and $\{X_i\}_{i\in\underline{n}}$, these are functors of the following form:

$$FX = A_1 + A_2 \times X + A_3 \times X^2 + A_4 \times X^3 + \cdots + A_{n+1} \times X^n$$

A way of representing a polynomial functor is by means of a container. A *container* is a pair (S, ar) where S is a object representing *shapes* and $ar : S \to \mathbb{N}$ is the arity of the shape.

The *container extension* represents the actual functor and is defined as follows:

$$[\![S, ar]\!]X = \Sigma s : S.X^{ar(s)}$$

An example is the functor $FX = 1 + A \times X$ whose least fixed-point is the type of lists over an object A. A container for this functor is formed by the shape $1 + A$ and the arity for each position in the shape $ar : 1 + A \to \mathbb{N}$ given by $ar(\mathrm{inl}(*)) = 0$ and $ar(\mathrm{inr}(a)) = 1$. The container extension is therefore the type $\Sigma s : 1 + A.X^{ar(s)}$. There are two possible pairs for this type. The first one is where the first component is $\mathrm{inl}(*)$ which and the second is a map $f : 0 \to X$ whose cardinality is in fact 1 corresponding to the first component of the functor. The second one is where the first component is $\mathrm{inr}(a)$ and the second component is a map $f : 1 \to X$ whose cardinality is actually the cardinality of X. Hence the elements of the extension of this container are the same elements of the functor $1 + A \times X$.

References

[1] Thorsten Altenkirch, James Chapman, and Tarmo Uustalu. Monads need not be endofunctors. *Log. Methods Comput. Sci.*, 11(1), 2015.

[2] Steve Awodey. *Category Theory.* Oxford University Press, Inc., USA, 2nd edition, 2010.

[3] Neil Ghani, Tarmo Uustalu, and Makoto Hamana. Explicit substitutions and higher-order syntax. *High. Order Symb. Comput.*, 19(2-3):263–282, 2006.

[4] Ralf Hinze. Generic programming with adjunctions. In Jeremy Gibbons, editor, *Generic and Indexed Programming - International Spring School, SSGIP 2010, Oxford, UK, March 22-26, 2010, Revised Lectures*, volume 7470 of *Lecture Notes in Computer Science*, pages 47–129. Springer, 2010.

[5] Fosco Loregian. *(Co)end Calculus.* Cambridge University Press, jun 2021.

[6] Saunders MacLane. *Categories for the Working Mathematician.* Springer-Verlag, New York, 1971. Graduate Texts in Mathematics, Vol. 5.

[7] Maciej Piróg, Tom Schrijvers, Nicolas Wu, and Mauro Jaskelioff. Syntax and semantics for operations with scopes. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 809–818. ACM, 2018.