

Category Theory with Strings

Shunsuke Sogame

March 24, 2016

1 Introduction

This is a supplemental document for introductory books of category theory ([1], [2], [4], [9]) using *string diagrams*. Don't trust my poor mathematics. Any correction is welcome at github.com/okomok/strcat.

2 Preliminaries

2.1 Lambda Expressions

Definition 2.1 (Lambda Expression) Following famous symbols like Σ , define $\Lambda_x y$ as an anonymous function $x \mapsto y$. We casually call any form of anonymous functions a *lambda expression*.

Definition 2.2 (Lambda-Tasted Form) Given a function Γ whose domain is a set of functions, you can choose a short form of $\Gamma(\Lambda_x y)$ from the following *lambda-tasted* forms

1. $\Gamma_x y$
2. $\Gamma x.y$
3. $(\Gamma x)(y)$
4. Γxy

Definition 2.3 (Placeholder Expression) For simple lambda expressions, you may use *placeholders* for example,

$$?+1 := \Lambda_n n+1$$

Placeholder symbols can vary: $?$, $-$, 1 , etc.

2.2 Universality

Definition 2.4 (Predicate) We call a Boolean-valued function a *predicate*.

Definition 2.5 (Universal Quantifier) Given a predicate P , we define a Boolean value $\forall P$ as “anything satisfies P ”.

Definition 2.6 (Existential Quantifier) Given a predicate P , we define a Boolean value $\exists P$ as “something satisfies P ”.

Definition 2.7 (Uniqueness) Given a predicate P , a predicate $!P$ is defined by

$$!P(a) := P(a) \wedge (\forall a')(P(a') \implies a = a')$$

using the third lambda-tasted form, meaning that “ a is the unique thing that satisfies P ”.

Definition 2.8 (Unique Existential Quantifier) The *unique existential quantifier* $\exists!$ is defined as $\exists \circ !$, where \circ is the function composition. Spelling out the detail,

$$(\exists!a)(P(a)) = (\exists a)(!P(a))$$

meaning that “there exists a unique thing that satisfies P ”.

Remark 2.9 On the other hand, $(\exists a)(!P(a) \wedge Q(a))$ states “there exists a unique a that satisfies P . Furthermore, the a satisfies Q ”.

Definition 2.10 Given a predicate P and a set X ,

$$\begin{aligned} (\forall x \in X)(P(x)) &:= (\forall x)(x \in X \implies P(x)) \\ (\exists x \in X)(P(x)) &:= (\exists x)(x \in X \wedge P(x)) \end{aligned}$$

Definition 2.11 (Universality) Given a binary predicate P , we boldly call a statement of the form

$$(\forall x \in X)(\exists!y \in Y)(P(x, y))$$

the *universality* of P .

Proposition 2.12 (Functional Universality) Given a binary predicate P ,

$$\begin{aligned} &(\forall x \in X)(\exists!y \in Y)(P(x, y)) \\ \iff &(\exists f : X \rightarrow Y)(\forall x \in X)(\forall y \in Y)(P(x, y) \iff y = f(x)) \end{aligned}$$

PROOF. (\implies) by the axiom of choice. (\impliedby) immediate. \square

Definition 2.13 (Functional Bijectivity) Given a function $g : Y \rightarrow X$, the statement

$$(\exists f : X \rightarrow Y)(\forall x \in X)(\forall y \in Y)(x = g(y) \iff y = f(x))$$

is known as the *bijectivity* of f and g . This is a special case of universality where $P(x, y)$ is $x = g(y)$.

2.3 Families

Syntax of function applications is world-standard:

$$f(x)$$

but sometimes you might want cuter syntax like that

$$\langle x \rangle$$

Definition 2.14 (Family) A *family declaration* is a way to provide a function with arbitrary application syntax. Its usage is clear from an example

$$(\langle x \rangle \in Y)_{x \in X}$$

We call it a *family* of Y . Furthermore, a function body can be placed like that

$$(\langle x \rangle := x^2 \in Y)_{x \in X}$$

Example 2.15 (Subscript) The most-used family declaration is the subscript style $(a_i)_i$. You can view a tuple (a_1, a_2, \dots, a_n) to be an abbreviation of $(a_i)_{i \in \{1, 2, \dots, n\}}$. Subscripts are often omitted.

Families can do more.

Definition 2.16 (Dependent Function) Let F a set-valued function. A family

$$(f(x) \in F(x))_{x \in X}$$

defines a function

$$f : X \rightarrow \bigcup_{x \in X} F(x)$$

such that

$$(\forall x \in X)(f(x) \in F(x))$$

We call such f a *dependent function*, for the $F(x)$ depends on x . In case F is a constant function, f is a normal function $X \rightarrow Y$.

2.4 Coherence

It is sure you write

$$3 + 1 + 2$$

rather than

$$(3 + (0 + 1)) + 2$$

because you know the arithmetic laws

$$\begin{aligned} x + (y + z) &= (x + y) + z \\ 0 + x &= x = x + 0 \end{aligned}$$

disambiguate expressions that are not parenthesized. Informally laws to introduce natural syntax are called *coherence conditions* or shortly *coherence*.

3 Categories

3.1 The Definition

Definition 3.1 (Category) A *category* \mathcal{C} consists of

1. *objects*: a class $\text{Ob}(\mathcal{C})$
2. *morphisms* or *hom-sets*: a family of sets $(\mathcal{C}(A, B))_{A, B \in \text{Ob}(\mathcal{C})}$
3. *compositions*: a family of functions

$$(\circ : \mathcal{C}(B, C) \times \mathcal{C}(A, B) \rightarrow \mathcal{C}(A, C))_{A, B, C \in \text{Ob}(\mathcal{C})}$$

4. *identities* or *units*: a family of morphisms

$$(\text{id}_A \in \mathcal{C}(A, A))_{A \in \text{Ob}(\mathcal{C})}$$

satisfying the following coherence conditions

1. *associativity*: for any $f \in \mathcal{C}(A, B)$, $g \in \mathcal{C}(B, C)$, and $h \in \mathcal{C}(C, D)$,

$$h \circ (g \circ f) = (h \circ g) \circ f$$

2. *unitality*: for any $f \in \mathcal{C}(A, B)$,

$$\text{id}_B \circ f = f = f \circ \text{id}_A$$

A morphism $f \in \mathcal{C}(A, B)$ is often denoted as $f : A \rightarrow B$.

3.2 String Diagrams

From now on, we will introduce *string diagrams* to complement (or hopefully replace) commutative diagrams.

Given a category \mathcal{C} , an object A is depicted as an optionally-tagged string

$$\begin{array}{c} \mathcal{C} \\ | \\ A \\ | \end{array}$$

A morphism $f : A \rightarrow B$ is depicted as a node

$$\begin{array}{c} B \\ | \\ \boxed{f} \\ | \\ A \end{array}$$

The composition joins two strings.

$$\begin{array}{c} C \\ | \\ \boxed{g} \\ | \\ B \\ | \\ \boxed{f} \\ | \\ A \end{array} := g \circ f$$

Identity morphisms are indistinguishable from objects.

$$\begin{array}{c} | \\ A \end{array} := \text{id}_A$$

Check these diagrams create no ambiguity thanks to the coherence.

Definition 3.2 (Isomorphism) We call a pair of morphisms

$$\begin{aligned} f &: A \rightarrow B \\ g &: B \rightarrow A \end{aligned}$$

an *isomorphism* or shortly *iso* provided that the *invertibility*

$$\begin{array}{c} A \\ | \\ \boxed{g} \\ | \\ B \\ | \\ \boxed{f} \\ | \\ A \end{array} = \begin{array}{c} | \\ A \end{array} \quad \text{and} \quad \begin{array}{c} B \\ | \\ \boxed{f} \\ | \\ A \\ | \\ \boxed{g} \\ | \\ B \end{array} = \begin{array}{c} | \\ B \end{array}$$

is satisfied. A morphism that is a part of an isomorphism is called *invertible*.

Remark 3.3 Most people call each morphism of this pair an isomorphism.

Definition 3.4 (Functional Box) Given categories \mathcal{C} and \mathcal{C}' , a function

$$h : \mathcal{C}(A, B) \rightarrow \mathcal{C}'(A', B')$$

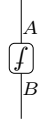
is depicted as a *functional box*

$$\begin{array}{c} B' \\ | \\ \boxed{\begin{array}{c} B \\ | \\ \boxed{?} \\ | \\ A \end{array}} \\ | \\ A' \end{array} \quad h$$

Definition 3.5 (Opposite Category) Given a category \mathcal{C} and a morphism



you can construct a category with strings upside down:



which is denoted as \mathcal{C}^{op} , the *opposite category* of \mathcal{C} .

Definition 3.6 (Discrete Category) A category \mathcal{C} such that

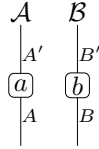
$$\begin{aligned} A = B &\implies \mathcal{C}(A, B) = \{\text{id}_A\} \\ A \neq B &\implies \mathcal{C}(A, B) = \emptyset \end{aligned}$$

is called a *discrete category*. Any set can be represented as a discrete category.

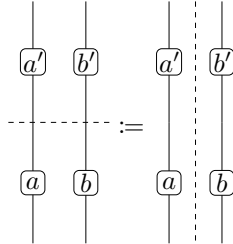
Definition 3.7 (Product Category) Given two categories \mathcal{A} and \mathcal{B} , the *product category*

$$\mathcal{A} \times \mathcal{B}$$

is depicted as parallel strings



The composition, which joins parallel strings, is defined by



Identity morphisms are trivially



By these definitions,

$$\begin{array}{c} | \\ \boxed{a} \\ | \end{array} = \begin{array}{c} | \\ \boxed{b} \\ | \end{array}$$

4 Functors

4.1 The Definition

Definition 4.1 (Functor) A *functor* $F : \mathcal{C} \rightarrow \mathcal{D}$ consists of

1. *domain*: a category \mathcal{C}
2. *codomain*: a category \mathcal{D}
3. a family of objects $(FA \in \text{Ob}(\mathcal{D}))_{A \in \text{Ob}(\mathcal{C})}$
4. families of morphisms

$$\left((F(f) \in \mathcal{D}(FA, FB))_{f \in \mathcal{C}(A, B)} \right)_{A, B \in \text{Ob}(\mathcal{C})}$$

satisfying the *functoriality*:

1. *composition-compatibility*: for any $f \in \mathcal{C}(A, B)$ and $g \in \mathcal{C}(B, C)$,

$$F(g \circ f) = F(g) \circ F(f)$$

2. *unit-compatibility*: for any $A \in \text{Ob}(\mathcal{C})$,

$$F(\text{id}_A) = \text{id}_{FA}$$

Definition 4.2 (Infrafunctor) An *infrafunctor* is a functor without the requirement of functoriality.

4.2 Functorial Tubes

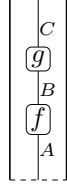
In string diagrams, a functor can be depict as a *tube* defined by

$$\left[\begin{array}{c} \text{---} \\ \text{---} \end{array} \right] \begin{array}{c} B \\ \boxed{f} \\ A \end{array} \text{---} := \begin{array}{c} \text{---} \\ \text{---} \end{array} \begin{array}{c} B \\ \boxed{f} \\ A \end{array} \begin{array}{c} \text{---} \\ \text{---} \end{array} \begin{array}{c} FB \\ FA \end{array}$$

Placeholders make it simple:

$$\left[\begin{array}{c} \text{---} \\ \text{---} \end{array} \right] \begin{array}{c} \boxed{?} \end{array} \text{---}$$

One can check the functoriality ensures any tube like



be unambiguous. “Join then tube” is the same as “tube then join”.

Proposition 4.3 Any functor preserves isomorphisms meaning that

$$\left(\begin{array}{c|c} B & A \\ \hline \boxed{f} & \boxed{g} \\ \hline A & B \end{array} \right) : \text{iso} \implies \left(\begin{array}{c|c} B & A \\ \hline \boxed{f} & \boxed{g} \\ \hline A & B \end{array} \right) : \text{iso}$$

PROOF. Immediate by functoriality, which inheres in tubes. \square

Definition 4.4 (Composite Functor) For any two functors

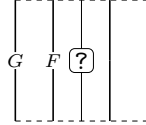
$$F : \mathcal{A} \rightarrow \mathcal{B}$$

$$G : \mathcal{B} \rightarrow \mathcal{C}$$

the *composite functor* of F and G

$$G \circ F : \mathcal{A} \rightarrow \mathcal{C}$$

is defined as



Definition 4.5 (Identity Functor) Given a category \mathcal{C} , the *identity functor* on \mathcal{C}

$$\text{Id}_{\mathcal{C}} : \mathcal{C} \rightarrow \mathcal{C}$$

is defined by

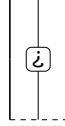
$$\left(\begin{array}{c|c} & \\ \hline \text{Id} & \boxed{?} \\ \hline & \end{array} \right) := \boxed{?}$$

Definition 4.6 (Contravariant Functor) A functor whose domain is an opposite category

$$F : \mathcal{C}^{\text{op}} \rightarrow \mathcal{D}$$

is called *contravariant*, while a normal functor is called *covariant*.

A contravariant functor is depicted as



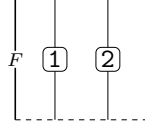
Definition 4.7 (Variant) Given a statement regarding functors, you can obtain a corresponding one regarding contravariant functors and vice versa. We call such a statement the *variant* of the original one.

Definition 4.8 (Binary Functor) A functor whose domain is a product category

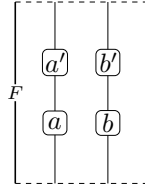
$$F : \mathcal{A} \times \mathcal{B} \rightarrow \mathcal{C}$$

is called a *binary functor* or *bifunctor*.

With numbered placeholders, it is depicted as



Spelling out the definition of functoriality, one can check a diagram like

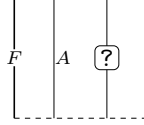


is unambiguous.

Definition 4.9 (Partial Application) Given a binary functor $F : \mathcal{A} \times \mathcal{B} \rightarrow \mathcal{C}$, a *partially applied* functor

$$\begin{aligned} \Lambda_B F(A, B) : \mathcal{B} &\rightarrow \mathcal{C} \text{ or shortly} \\ F(A, ?) : \mathcal{B} &\rightarrow \mathcal{C} \end{aligned}$$

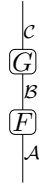
is defined as



The definition of $F(?, B)$ is an exercise.

Definition 4.10 (Small Category) A category \mathcal{C} is called *small* when $\text{Ob}(\mathcal{C})$ is a set.

Definition 4.11 (Category of Small Categories) The *category of small categories* \mathbf{Cat} is the category whose objects are all small categories and whose morphisms are functors:



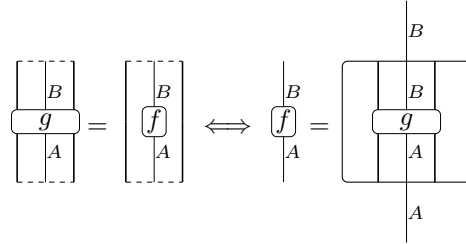
where composite functors join the strings.

Definition 4.12 (Full and Faithful Functor) A functor $F : \mathcal{C} \rightarrow \mathcal{D}$ is called *full and faithful* if for each object A and B in \mathcal{C} , the family

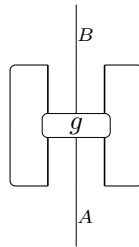
$$(F(f) : FA \rightarrow FB)_{f:A \rightarrow B}$$

is bijective.

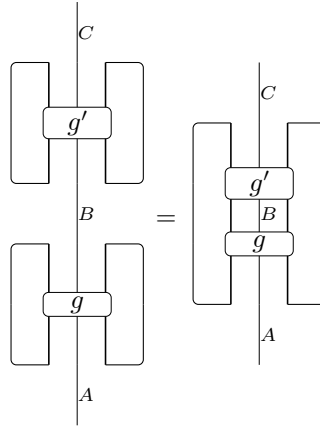
In other words, there is a functional box such that



One can make this box better-looking

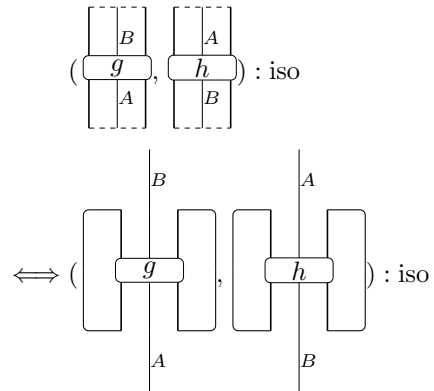


Proposition 4.13 This box has a functoriality-like property:



Combined with proposition 4.3,

Proposition 4.14



5 Natural Transformations

5.1 The Definition

Definition 5.1 (Naturality) Given two infrafunctors

$$F, G : \mathcal{C} \rightarrow \mathcal{D}$$

a family of morphisms

$$(\tau_A \in \mathcal{D}(FA, GA))_{A \in \text{Ob}(\mathcal{C})}$$

is called *natural* when for any $f \in \mathcal{C}(A, B)$,

$$\tau_B \circ F(f) = G(f) \circ \tau_A$$

In case parentheses are cumbersome, you can say “ τ_A is *natural in A*”.

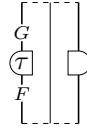
Definition 5.2 (Natural Transformation) Furthermore, in particular case F and G are functorial (then they are functors), τ is denoted as a *natural transformation* $\tau : F \rightarrow G$.

Remark 5.3 In this document, naturality is explicitly defined to be orthogonal to functoriality.

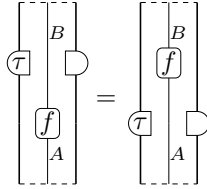
Proposition 5.4 Let $F : \mathcal{C} \rightarrow \mathcal{D}$ be an infrafunctor. Recall this is by definition a family of functions $(F_{A,B} : \mathcal{C}(A, B) \rightarrow \mathcal{D}(FA, FB))_{A,B}$. Then $F_{A,B}$ is natural in A or B if and only if F is composition-compatible.

5.2 Natural Connectors

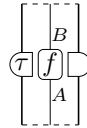
In our diagrams, a natural transformation is a *connector* of two tubes



because the naturality states a node can travel between tubes



This inspires us to assign



Definition 5.5 (Vertical Composition) Given three functors

$$F, G, H : \mathcal{C} \rightarrow \mathcal{D}$$

and two natural transformations

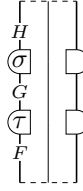
$$\tau : F \rightarrow G$$

$$\sigma : G \rightarrow H$$

the *vertical composition* of τ and σ

$$\sigma \circ \tau : F \rightarrow H$$

is defined as



Definition 5.6 (Horizontal Composition) Given four functors

$$F, G : \mathcal{A} \rightarrow \mathcal{B}$$

$$H, K : \mathcal{B} \rightarrow \mathcal{C}$$

and two natural transformations

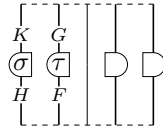
$$\tau : F \rightarrow G$$

$$\sigma : H \rightarrow K$$

the *horizontal composition* of τ and σ

$$\sigma \tau : H \circ F \rightarrow K \circ G$$

is defined as



You can easily check the naturality. Travel by car ferry.

Definition 5.7 (Identity Natural Transformation) Given a functor

$$F : \mathcal{C} \rightarrow \mathcal{D}$$

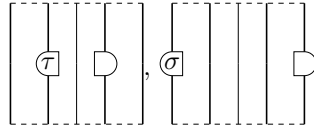
the *identity natural transformation*

$$\text{id}_F : F \rightarrow F$$

is defined by

$$\boxed{\text{id}} := \boxed{\quad}$$

Definition 5.8 (Whiskering) A *whiskering* is a horizontal composition with identity natural transformations:



Definition 5.9 (Natural Isomorphism) We call a pair of natural transformations

$$\begin{aligned} \tau : F &\rightarrow G \\ \sigma : G &\rightarrow F \end{aligned}$$

a *natural isomorphism* or shortly *natural iso* provided that the *invertibility*

$$\begin{aligned} &\boxed{\begin{array}{c} \sigma \\ \tau \end{array}} = \boxed{\quad} \\ &\boxed{\begin{array}{c} \tau \\ \sigma \end{array}} = \boxed{\quad} \end{aligned}$$

is satisfied.

Remark 5.10 Most people call each morphism of this pair a natural isomorphism.

Proposition 5.11 For any natural transformation τ ,

$$(\forall A)(\tau_A : \text{invertible})$$

is enough to build the other natural σ .

Definition 5.12 (Functor Category) Given a small category \mathcal{C} and a category \mathcal{D} , the functor category $[\mathcal{C}, \mathcal{D}]$ is a category whose objects are functors from \mathcal{C} to \mathcal{D} and whose morphisms are natural transformations:



where the vertical composition joins the strings.

Definition 5.13 For the later use, define a lambda-tasted form for a set of natural transformations by

$$\text{Nat}_X(L, R) := [\mathcal{C}, \mathcal{D}](\Lambda_X L, \Lambda_X R)$$

6 Category of Sets

6.1 The Definition

Definition 6.1 (Category of Sets) The *category of sets* **Set** is a category whose objects are sets and whose morphisms are functions:

$$\begin{array}{c} | \\ Z \\ \boxed{g} \\ | \\ Y \\ \boxed{f} \\ | \\ X \end{array}$$

where strings are joined by the function composition.

A category is essentially one-dimensional so far: the vertical composition only. Here we introduce the horizontal composition for **Set**.

Definition 6.2 (Monoidal Category of Sets) Parallel strings are defined by

$$\begin{array}{c} | \\ X \end{array} \begin{array}{c} | \\ X' \end{array} := \begin{array}{c} | \\ X \times X' \end{array}$$

The *horizontal composition* of functions is defined by

$$\begin{array}{c} | \\ Y \\ \boxed{f} \\ | \\ X \end{array} \begin{array}{c} | \\ Y' \\ \boxed{f'} \\ | \\ X' \end{array} := \Lambda_{x,x'}(f(x), f'(x'))$$

Strings for the singleton set $\{*\}$ is omitted so that an element of a set is represented as

$$\begin{array}{c} | \\ X \\ \boxed{x} \end{array}$$

One can check any string diagram built upon these definitions is unambiguous due to the trivial bijections

$$\begin{aligned} X \times (X' \times X'') &\cong (X \times X') \times X'' \\ X \times \{*\} &\cong X \end{aligned}$$

Informally such two-dimensional categories are called *monoidal*.

6.2 Hom-Set Bands

Given a category \mathcal{C} , a special string, a *band*, is introduced for hom-sets:

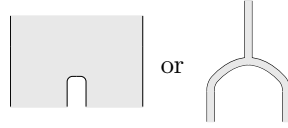
$$\begin{array}{c} \boxed{B \quad A} \\ | \end{array} := \begin{array}{c} | \\ \mathcal{C}(A, B) \end{array}$$

A space-saving form is depicted as



Remark 6.3 Note that the order of objects is flipped. This is resulting from an unfortunate convention that one write “ $b = h(a)$ ” but not “ $h : B \leftarrow A$ ”. By the way, “ B^A ” is fine.

The composition of morphisms can be depicted as



Identity morphisms can be depicted as



As an exercise, write down the associativity and unitality using these diagrams.

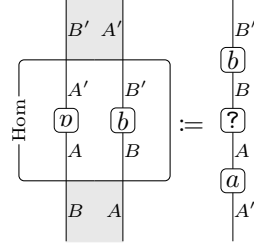
Definition 6.4 (Hom-Functor) Hom-sets can be extended to the *Hom-functor*

$$\Lambda_{A,B}\mathcal{C}(A, B) : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathbf{Set} \text{ or shortly}$$

$$\mathcal{C}(-, +) : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathbf{Set} \text{ or shortly}$$

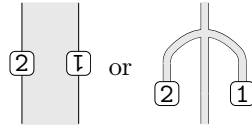
$$\text{Hom}_{\mathcal{C}} : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathbf{Set}$$

defined by



where the world in the box is product category $\mathcal{C}^{\text{op}} \times \mathcal{C}$.

This definition inspires us to depict hom-functors as

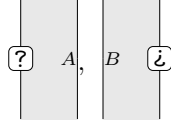


that looks topologically equivalent.

Definition 6.5 (Unary Hom-Functor) Due to definition 4.9,

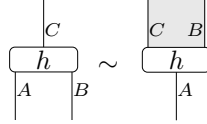
$$\begin{aligned}\mathcal{C}(A, +) &: \mathcal{C} \rightarrow \mathbf{Set} \\ \mathcal{C}(-, B) &: \mathcal{C}^{\text{op}} \rightarrow \mathbf{Set}\end{aligned}$$

are respectively depicted as



Definition 6.6 (Currying) In particular case $\mathcal{C} = \mathbf{Set}$, the *curry bijection* or shortly *currying* is defined by

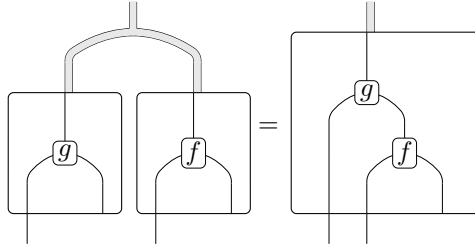
$$\begin{aligned}\mathbf{Set}(A \times B, C) &\cong \mathbf{Set}(A, \mathbf{Set}(B, C)) \\ h &\mapsto (a \mapsto b \mapsto h(a, b)) \\ ((a, b) \mapsto h(a, b)) &\leftarrow h\end{aligned}$$



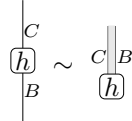
We don't distinguish these two diagrams because the following two propositions ensure “move the right-side leg up and down” works correct.

Proposition 6.7 Currying is natural in all three variables.

Proposition 6.8 Currying merges:



Definition 6.9 (Naming) In case A is the singleton set, which is omitted in diagrams, a currying



is trivial. We call it a *naming* ([3]).

Proposition 6.10 Currying *preserves naturality*, meaning that given a family of functions $(f_X : FX \times B \rightarrow GX)_X$ with infrafunctors F and G , f_X is natural in X if and only if $\text{curry}(f_X)$ is. So does uncurrying.

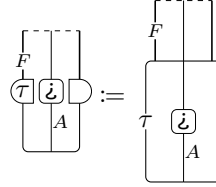
Remark 6.11 In general, natural bijections have similar properties so that you don't bother with proof of naturality ([7]).

7 The Yoneda Lemma

Definition 7.1 Given a functor $F : \mathcal{C}^{\text{op}} \rightarrow \mathbf{Set}$ and an object A in \mathcal{C} , a natural transformation of the form

$$(\tau_X : \mathcal{C}(X, A) \rightarrow FX)_X$$

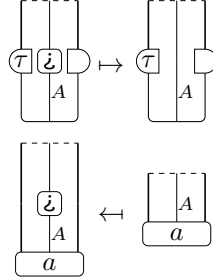
can be depicted as



owing to the naturality.

Definition 7.2 (Yoneda Bijection) The *Yoneda bijection* is defined by

$$\text{Nat}_X(\mathcal{C}(X, A), FX) \cong FA$$

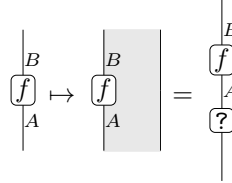


Lemma 7.3 (Yoneda Lemma) The Yoneda bijection is actually bijective and natural in F and A .

PROOF. Now the proof is on my soul trivial! \square

Definition 7.4 (Yoneda Embedding) The *Yoneda embedding* is defined by

$$\Lambda_A \Lambda_X \mathcal{C}(X, A) : \mathcal{C} \rightarrow [\mathcal{C}^{\text{op}}, \mathbf{Set}]$$



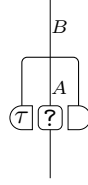
using the diagram of hom-functors. In short,



Definition 7.5 A natural transformation of the form

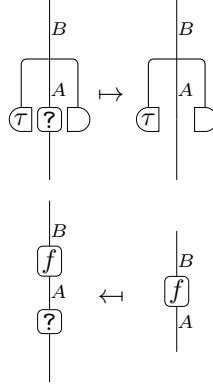
$$(\tau_X : \mathcal{C}(X, A) \rightarrow \mathcal{C}(X, B))_X$$

can be depicted as



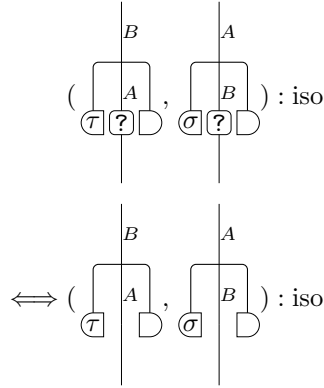
Definition 7.6 (Yoneda Embedding Bijection) In special case $F := \mathcal{C}(-, B)$, the Yoneda bijection is expanded to

$$\text{Nat}_X(\mathcal{C}(X, A), \mathcal{C}(X, B)) \cong \mathcal{C}(A, B)$$



The second mapping is the Yoneda embedding so that it is full and faithful. Combined with proposition 4.14,

Proposition 7.7 (Yoneda Principle)

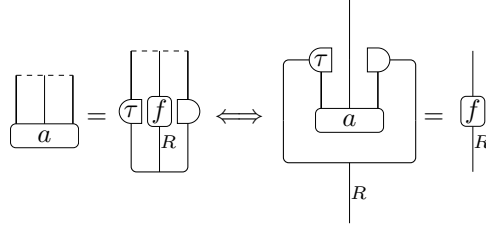


8 Representations

Definition 8.1 (Representation) Given a functor $H : \mathcal{C} \rightarrow \mathbf{Set}$, a *representation* of H is a pair of

1. an object R in \mathcal{C}
2. a natural bijection $(\tau_X : HX \cong \mathcal{C}(R, X))_X$

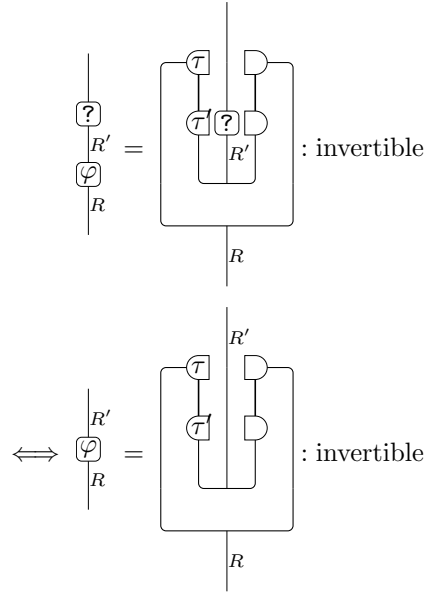
This bijectivity can be expressed using the weird boxes



thanks to the naturality. The following proposition allows us to call it *the* representation of H , denoted as $\text{rep}H$.

Proposition 8.2 (Uniqueness of Representations) Representations are unique up to unique isomorphism.

PROOF. Let (R', τ') be another representation. By the variant of proposition 7.7,

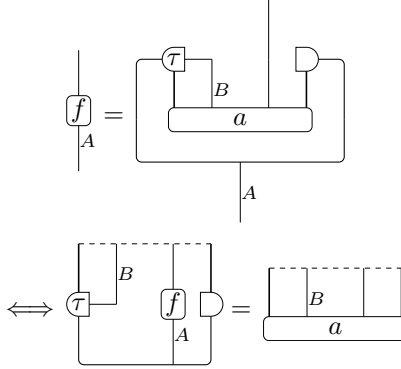


□

Definition 8.3 Given a functor $H : \mathcal{B}^{\text{op}} \times \mathcal{A} \rightarrow \mathbf{Set}$, a natural bijection of the form

$$(\tau_X : H(B, X) \cong \mathcal{A}(A, X))_X$$

can be expressed by



Proposition 8.4 (Parameterized Representations) Let $H : \mathcal{B}^{\text{op}} \times \mathcal{A} \rightarrow \mathbf{Set}$ be a functor. Given a family of objects $(SB)_B$ and a family of representations

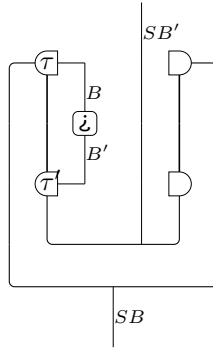
$$((\tau_X^B : H(B, X) \cong \mathcal{A}(SB, X))_X)_B$$

there exists a unique family

$$(S(f) \in \mathcal{A}(SB, SB'))_{f \in \mathcal{B}(B, B')}$$

such that τ is natural in B . Furthermore, S is functorial.

PROOF. Define S as



□

9 Limits

Definition 9.1 (Cone) Given a functor $F : \mathcal{A} \rightarrow \mathcal{B}$, a cone of F consists of

1. an object B in \mathcal{B}
2. a natural transformation $(v_X : B \rightarrow FX)_X$

Definition 9.2 (Conicality) We may explicitly call naturality of cones *conicality*, which can be expressed as

like a magical box any morphism can appear from.

Remark 9.3 Vertical and horizontal composition preserve conicality, a special case of naturality.

Definition 9.4 (Limit) Given a functor $F : \mathcal{A} \rightarrow \mathcal{B}$, a limit of F is a pair of

1. an object in \mathcal{B} denoted as $\lim F$
2. a natural bijection $(\mathcal{B}(B, \lim F) \cong \text{Nat}_X(B, FX))_B$

Definition 9.5 (Limiting Cone) The limit bijectivity, thanks to its naturality, can be expressed as

where $\boxed{\lim}$ is a cone called a *limiting cone* of F .

The following proposition justifies the notation $\lim F$, *the* limit of F .

Proposition 9.6 Limits are unique up to isomorphism.

PROOF. Immediate by proposition 8.2, because a limit is nothing but a contravariant representation

$$\text{rep}_B \text{Nat}_X(B, FX)$$

□

Proposition 9.7 A limiting cone is *monic* meaning that

PROOF. Immediate by the limit bijectivity. \square

Definition 9.8 (Product) In particular case the domain of a functor $F : \mathcal{A} \rightarrow \mathcal{B}$ is discrete, the limit of F is called the *product* of F , denoted as $\prod F$.

Definition 9.9 (Projection) Spelling out the product bijectivity,

where $\boxed{\pi}$ is called the *projection* of F .

Remark 9.10 Conicality has no concern here, because any family of the form

$$(v_X : B \rightarrow FX)_{X \in \text{Ob}(\mathcal{A})}$$

is always natural in case \mathcal{A} is discrete.

Example 9.11 (Products in Sets) In case F is a functor $X \rightarrow \mathcal{S}et$ with a set X (as a discrete category), the product of F is a set of dependent functions.

$$\prod_x F(x) \cong \{f \mid (f(x) \in F(x))_x\}$$

Definition 9.12 (Dual) Given a statement containing string diagrams, by flipping the diagrams upside down, a corresponding statement is obtained. It is called the *dual* of the original one.

Definition 9.13 (Coproduct) A *coproduct* is a structure obtained from the product bijectivity flipped.

Remark 9.14 Informally the dual makes a codomain opposite, while the variant does for a domain.

Definition 9.15 (Preservation of Limits) Given a functor $F : \mathcal{A} \rightarrow \mathcal{B}$ and a limiting cone of F

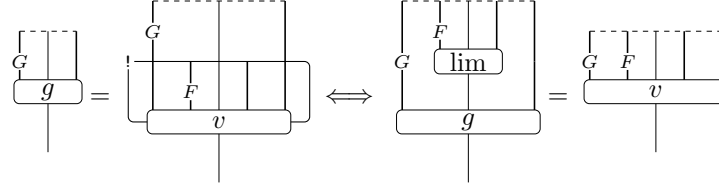
$$(\lim_X : \lim F \rightarrow FX)_X$$

a functor $G : \mathcal{B} \rightarrow \mathcal{C}$ *preserves limits* of F provided that

$$(G(\lim_X) : G\lim F \rightarrow GFX)_X$$

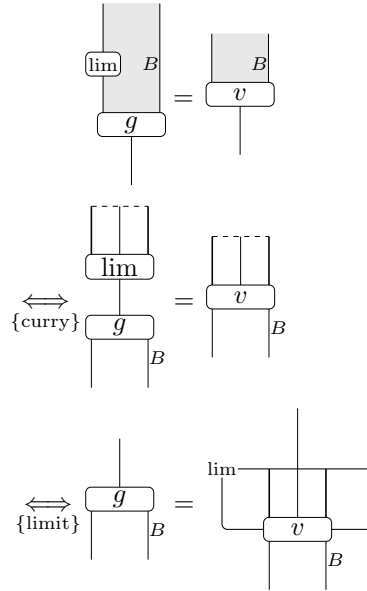
is a limiting cone of $G \circ F$.

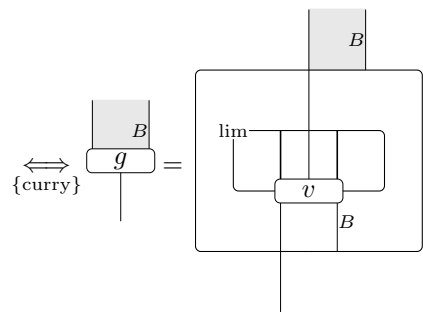
In diagrams, G is such that there exists some box $!$ satisfying



Proposition 9.16 (HFPL) Hom-functors preserve limits, meaning that given a functor $F : \mathcal{A} \rightarrow \mathcal{B}$ and an object B in \mathcal{B} , the covariant hom-functor $\mathcal{B}(B, +) : \mathcal{B} \rightarrow \mathbf{Set}$ preserves limits of F .

PROOF.





□

10 Adjunctions

Definition 10.1 (Adjunction) Given two categories \mathcal{C} and \mathcal{D} , an *adjunction*

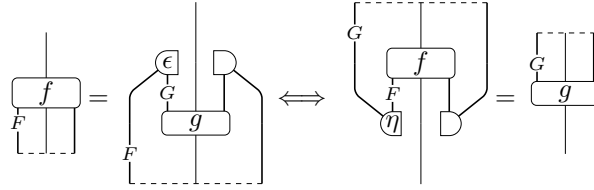
$$F \dashv G$$

consists of

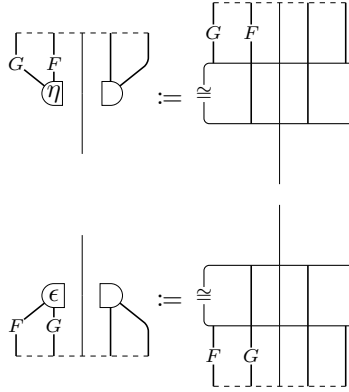
1. *left adjoint*: a functor $F : \mathcal{C} \rightarrow \mathcal{D}$
2. *right adjoint*: a functor $G : \mathcal{D} \rightarrow \mathcal{C}$
3. *adjunct*: a natural bijection

$$(\mathcal{D}(FC, D) \cong \mathcal{C}(C, GD))_{C,D}$$

A nice consequence is that this bijectivity needs no boxes, expressed by natural transformations only.



where



called the *unit* and *counit* respectively.

Proposition 10.2 Given a functor $G : \mathcal{D} \rightarrow \mathcal{C}$, a family of natural bijections

$$((\mathcal{C}(C, GD) \cong \mathcal{D}(F_c, D))_D)_C$$

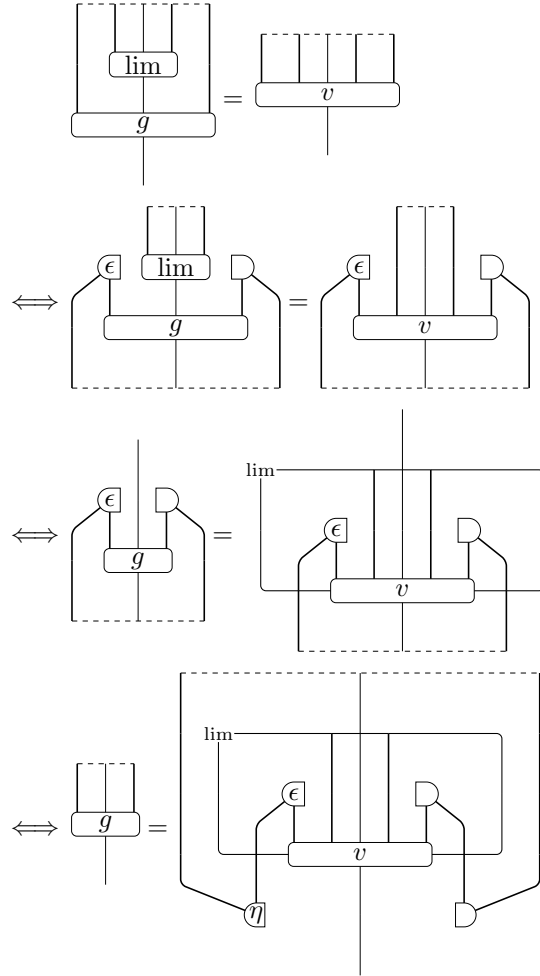
is enough to construct the adjunction $F \dashv G$.

PROOF. Immediate by proposition 8.4 with $H(C, D) := \mathcal{C}(C, GD)$. \square

Proposition 10.3 (RAPL) Right adjoints preserve limits, meaning that given an adjunction $F \dashv (G : \mathcal{D} \rightarrow \mathcal{C})$ and a functor $T : \mathcal{B} \rightarrow \mathcal{D}$,

$$\begin{aligned} & (\lim_X : \lim T \rightarrow TX)_X : \text{limiting cone} \\ \implies & (G(\lim_X) : G\lim T \rightarrow GTX)_X : \text{limiting cone} \end{aligned}$$

PROOF.



□

11 Monads

11.1 The Definition

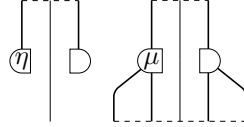
Definition 11.1 (Monad) Given a category \mathcal{C} , a *monad* on \mathcal{C} consists of

1. a functor $T : \mathcal{C} \rightarrow \mathcal{C}$
2. *unit*: a natural transformation $\eta : \text{Id}_T \rightarrow T$
3. *multiplication*: a natural transformation $\mu : T \circ T \rightarrow T$

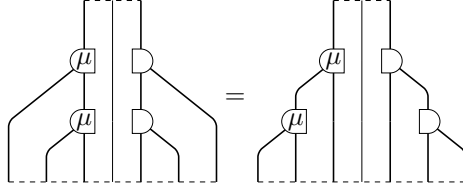
satisfying the coherence conditions

1. *associativity*: $\mu \circ T\mu = \mu \circ \mu T$
2. *unitality*: $\mu \circ T\eta = \text{Id}_T = \mu \circ \eta T$

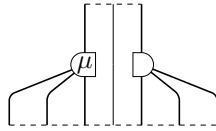
A unit and multiplication are depicted respectively as



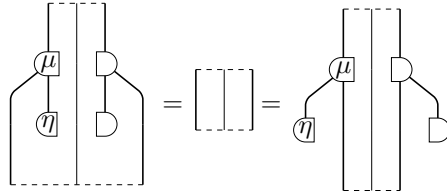
The associativity is depicted as



This inspires us to assign



The unitality is



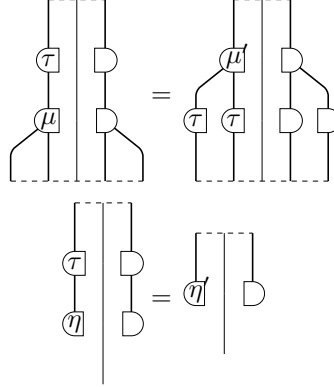
Definition 11.2 (Monad Morphism) Given a category \mathcal{C} , a *monad morphism* consists of

1. *domain*: a monad (T, η, μ) on \mathcal{C}
2. *codomain*: a monad (T', η', μ') on \mathcal{C}
3. a natural transformation $\tau : T \rightarrow T'$

satisfying the coherence conditions

1. *multiplication-compatibility*: $\tau \circ \mu = \mu' \circ \tau \tau$
2. *unit-compatibility*: $\tau \circ \eta = \eta'$

The coherence is depicted as

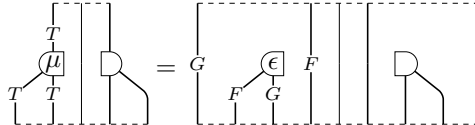


Definition 11.3 (Category of Monads) Given a category \mathcal{C} , the *category of monads* $\mathbf{Mnd}(\mathcal{C})$ is a category whose objects are monads and whose morphisms are monad morphisms.

Definition 11.4 (Monad-Associated Adjunction) Given a monad (T, η, μ) , we say an adjunction $F \dashv G$ is *T-associated* provided that

1. $T = G \circ F$
2. $\mu = G\epsilon F$

This condition can be depicted as

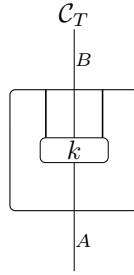


11.2 Kleisli Categories

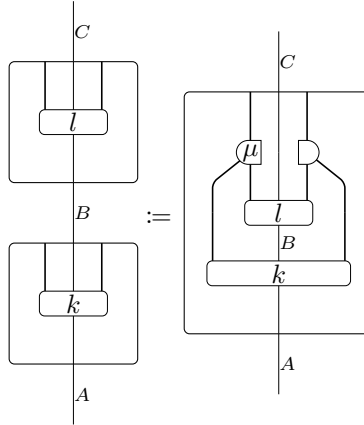
Definition 11.5 (Kleisli Category) Given a monad (T, η, μ) on \mathcal{C} , the *Kleisli category* of T , denoted as \mathcal{C}_T , is a category consisting of

1. $\text{Ob}(\mathcal{C}_T) := \text{Ob}(\mathcal{C})$
2. $\mathcal{C}_T(A, B) := \mathcal{C}(A, TB)$
3. $l \circ k := \mu \circ T(l) \circ k$
4. $\text{id}_A := \eta_A$

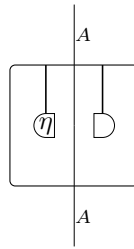
In diagrams, a morphism in \mathcal{C}_T is depicted as a *Kleisli box*



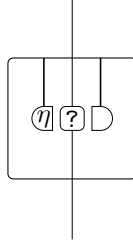
The composition is defined by



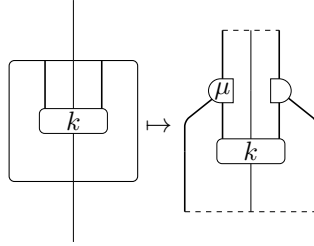
Identity morphisms are defined by



Definition 11.6 (Kleisli Adjunction) Define a functor $L : \mathcal{C} \rightarrow \mathcal{C}_T$ as



$K : \mathcal{C}_T \rightarrow \mathcal{C}$ as



then they constitute the *Kleisli adjunction* $L \dashv K$ whose adjoint is the Kleisli boxing. This adjunction is T -associated.

11.3 EM Categories

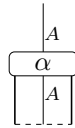
Definition 11.7 (Monad Algebra) Given a monad (T, η, μ) on \mathcal{C} , a *monad algebra*, denoted as T -algebra, consists of

1. an object $A \in \mathcal{C}$
2. a morphism $\alpha : TA \rightarrow A$

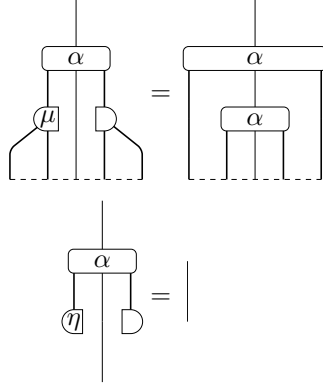
satisfying the coherence

1. *associativity*: $\alpha \circ \mu = \alpha \circ T(\alpha)$
2. *unitality*: $\alpha \circ \eta = \text{id}$

A T -algebra is depicted as



The coherence can be depicted as

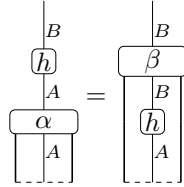


Definition 11.8 (EM Category) Given a monad (T, η, μ) , the *Eilenberg-Moore (EM) category* of T , denoted as \mathcal{C}^T , is a category whose objects are T -algebras and whose morphisms are those of the form $h : A \rightarrow B$ such that

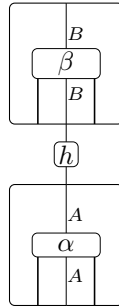
$$h \circ \alpha = \beta \circ T(h)$$

where (A, α) and (B, β) are T -algebras.

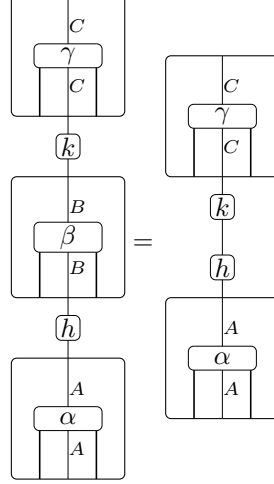
This condition is depicted as



A morphism in \mathcal{C}^T is by compromise depicted as

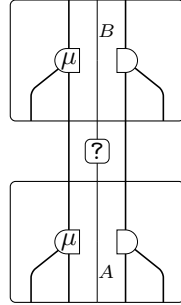


Boxes are objects. The composition can be depicted as

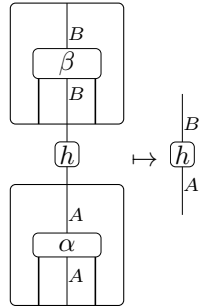


A diagram for an identity morphism is left as an exercise.

Definition 11.9 (EM Adjunction) Given a monad (T, η, μ) on \mathcal{C} , define a functor $M : \mathcal{C} \rightarrow \mathcal{C}^T$ as

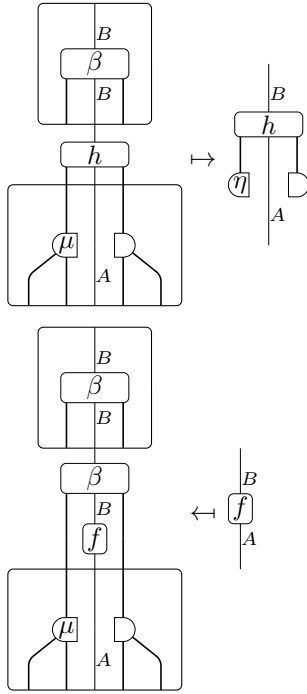


a functor $U : \mathcal{C}^T \rightarrow \mathcal{C}$ as



They constitute the *EM adjunction* $M \dashv U$ whose adjunct is defined by

$$\mathcal{C}^T(MA, (B, \beta)) \cong \mathcal{C}(A, U(B, \beta))$$



This adjunction is T -associated.

References

- [1] Jiri Adamek, Horst Herrlich, and George E Strecker. *Abstract and Concrete Categories: The Joy of Cats (Dover Books on Mathematics)*. Dover Publications, 8 2009.
- [2] Steve Awodey. *Category Theory (Oxford Logic Guides)*. Oxford University Press, 2 edition, 8 2010.
- [3] John Baez. Classical vs quantum computation (week 5). https://golem.ph.utexas.edu/category/2006/11/classical_vs_quantum_computati_5.html, 2006.
- [4] Francis Borceux. *Handbook of Categorical Algebra: Volume 1, Basic Category Theory (Encyclopedia of Mathematics and its Applications)*. Cambridge University Press, 1 edition, 4 2008.
- [5] John Bourke and Micah Blake McCurdy. Frobenius morphisms of bicategories, 2009.
- [6] Ralf Hinze. Kan extensions for program optimisation or: Art and dan explain an old trick. In *Mathematics of Program Construction*, pages 324–362. Springer, 2012.
- [7] Max Kelly. *Basic Concepts of Enriched Category Theory (London Mathematical Society Lecture Note Series)*. Cambridge University Press, 4 1982.
- [8] Aleks Kissinger. Pictures of processes: automated graph rewriting for monoidal categories and applications to quantum computing. *arXiv preprint arXiv:1203.0202*, 2012.
- [9] Saunders Mac Lane. *Categories for the Working Mathematician (Graduate Texts in Mathematics)*. Springer, 2nd ed. 1978. softcover reprint of the original 2nd ed. 1978 edition, 11 2010.
- [10] Dan Marsden. Category theory using string diagrams. *CoRR*, abs/1401.7220, 2014.
- [11] Micah Blake McCurdy. Strings and stripes, graphical calculus for monoidal functors and monads, 2010.
- [12] Paul-André Mellies. Functorial boxes in string diagrams. In *Computer science logic*, pages 1–30. Springer, 2006.
- [13] Peter Selinger. A survey of graphical languages for monoidal categories. In *New structures for physics*, pages 289–355. Springer, 2010.
- [14] Daniele Turi. Category theory lecture notes. *Laboratory for Foundations of Computer Science, University of Edinburgh*, 2001.