Category Theory

Lecture Notes by Andrew Pitts and Marcelo Fiore

What is category theory?

What we are probably seeking is a "purer" view of functions: a theory of functions in themselves, not a theory of functions derived from sets. What, then, is a pure theory of functions? Answer: category theory.

Dana Scott, *Relating theories of the* λ *-calculus*, p406

set theory gives an "element-oriented" account of mathematical structure, whereas

category theory takes a 'function-oriented" view – understand structures not via their elements, but by how they transform, i.e. via morphisms.

(Both theories are part of logic, broadly construed.)

Category Theory emerges

1945 Eilenberg[†] and MacLane[†]
General Theory of Natural Equivalences, Trans AMS 58, 231–294

(algebraic topology, abstract algebra)

- 1950s Grothendieck[†] (algebraic geometry)
- 1960s Lawvere[†] (logic and foundations)
- 1970s Johnstone, Joyal and Tierney[†] (elementary topos theory)
- 1980s Dana Scott, Plotkin (semantics of programming languages) Lambek[†] (linguistics)

Category Theory and Computer Science

"Category theory has...become part of the standard "tool-box" in many areas of theoretical informatics, from programming languages to automata, from process calculi to Type Theory."

> Dagstuhl Perpectives Workshop on Categorical Methods at the Crossroads April 2014

See http://www.appliedcategorytheory.org/events for recent examples of category theory being applied (not just in computer science).

This course

basic concepts of category theory

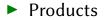
adjunction — natural transformation

applied to { typed lambda-calculus functional programming

Sets

Examples

Empty set: $0 = \emptyset = \{ \}$ Singleton sets: $1 = \{0\}, \{*\}$ Natural numbers: \mathbb{N}



The **cartesian product** of sets *X* and *Y* is the set of all *ordered pairings* (x, y) for $x \in X$ and $y \in Y$:

 $X \times Y = \{p \mid \exists ! x \in X, \exists ! y \in Y, p = (x, y)\}$ $= \{(x, y) \mid x \in X \land y \in Y\}$

The equality for ordered pairs is pointwise: for all $x, x' \in X$ and $y, y' \in Y$,

$$(x, y) = (x', y') \Leftrightarrow x = x' \land y = y'$$

The cartesian product comes equipped with first and second projection operations π_1 and π_2 satifying:

1. for all $x \in X$ and $y \in Y$,

 $\pi_1(x,y) = x$, $\pi_2(x,y) = y$

2. for all $p \in X \times Y$,

 $p=(\pi_1 p, \pi_2 p)$

NB: For all $p, p' \in X \times Y$, p = p' iff $\pi_1(p) = \pi_1(p')$ and $\pi_2(p) = \pi_2(p')$.

Example:

For $n \in \mathbb{N}$, $X^{n} = \begin{cases} \{ () \} &, \text{ if } n = 0 \\ X \times X^{m} &, \text{ if } n = m + 1 \end{cases}$

and, for $x_1, x_2, ..., x_{n-1}, x_n \in \mathbb{N}$, one writes $(x_1, ..., x_n)$ for $(x_1, (x_2, ..., (x_{n-1}, x_n) ...)) \in X^n$.

Functions

The **set of functions** from a set *X* to a set *Y*, for which we write $(X \Rightarrow Y)$ or Y^X , consists of all the single-valued and total relations from *X* to *Y*:

 $(X \Rightarrow Y) = \{ f \subseteq X \times Y \mid f \text{ is single-valued and total} \}$

Single-valued:

 $\forall x \in X, \forall y, y' \in Y, (x, y) \in f \land (x, y') \in f \Rightarrow y = y'$

Total:

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

Notation: We write $f : X \to Y$ or $X \xrightarrow{f} Y$ or $Y \xleftarrow{f} X$ for $f \in (X \Rightarrow Y)$ and, for $x \in X$, we write f x or f(x) or f_x for the unique element y of Y such that $(x, y) \in f$.

The equality for functions is extensional:

 $f = g : X \to Y \iff \forall x \in X, \ f x = g x$

This is because

- 1. Assuming f = g, we have, for all $x \in X$, $(x, f x) \in g$ and so f x = g x.
- 2. Assuming $\forall x \in X$, f x = g x, we have

 $f = \{(x, y) \mid (x, y) \in f\} = \{(x, f x) \mid x \in X\} = \{(x, g x) \mid x \in X\}$ $= \{(x, y) \mid (x, y) \in g\} = g$

In other words, function extensionality reduces to the extensionality property of sets: two sets are equal iff they have the same elements.

Convention: We typically define functions $f : X \to Y$ by a *well-defined rule* that to each element $x \in X$ assigns a unique element $f(x) \in Y$.

Examples:

1. We define $id_X : X \to X$ by: $\operatorname{id}_X(x) = x$ for all $x \in X$ 2. For $f: X \to Y$ and $q: Y \to Z$, we define $q \circ f : X \to Z$ by: $(q \circ f)(x) = q(f(x))$ for all $x \in X$ 3. We define app : $(X \Rightarrow Y) \times X \to Y$ by: app(f, x) = f(x) for all $f \in (X \Rightarrow Y)$ and $x \in X$ 4. For $f: Z \times X \to Y$, we define $\operatorname{cur}(f): Z \to (X \Longrightarrow Y)$ by: $(\operatorname{cur}(f) z)(x) = f(z, x)$ for all $z \in Z$ and $x \in X$

Sums

The **sum** of sets *X* and *Y* is their *disjoint union*:

 $X + Y = \{\iota_1(x) \mid x \in X\} \cup \{\iota_2(y) \mid y \in Y\}$

The sum comes equipped with first and second tagging operations $\iota_1 : X \to X + Y$ and $\iota_2 : Y \to X + Y$.

The equality for *tagged elements* is:

 $\iota_i(x) = \iota_j(y) \iff (i = j) \land (x = y)$

Sets in Grothendieck universes

A Grothendieck universe 21 is a class of sets satisfying

- $\blacktriangleright X \in Y \in \mathcal{U} \Longrightarrow X \in \mathcal{U}$
- $\blacktriangleright X, Y \in \mathcal{U} \Longrightarrow \{X, Y\} \in \mathcal{U}$
- $\blacktriangleright X \in \mathcal{U} \Longrightarrow \mathscr{P}X \triangleq \{Y \mid Y \subseteq X\} \in \mathcal{U}$
- ► $X \in \mathcal{U} \land F : X \to \mathcal{U}$ $\Rightarrow \bigcup_{x \in X} F(x) \triangleq \{y \mid \exists x \in X, y \in F(x)\} \in \mathcal{U}$ (hence also

 $X, Y \in \mathcal{U} \implies (X \times Y), (X \Longrightarrow Y), (X + Y) \in \mathcal{U})$

The above properties are satisfied by $\mathcal{U} = \emptyset$, but we will always assume

▶ $\mathbb{N} \in \mathcal{U}$

Algebras

<u>Monoids</u>

A **monoid** is a structure $\underline{M} = (M, \bullet, \iota)$ consisting of a set M equipped with a binary operation $_ \bullet _ : M \times M \to M$ and an element $\iota \in M$ that satisfy:

- ► the associativity law: $\forall x, y, z \in M, (x \bullet y) \bullet z = x \bullet (y \bullet z)$
- the unit laws:

 $\forall x \in M, \ \iota \bullet x = x = x \bullet \iota$

Examples:

1. Lists (List *X*, @, nil):

List X = set of finite lists of elements of X @ = append $\begin{pmatrix} \text{nil} @ \ell = \ell \\ (x :: \ell) @ \ell' = x :: (\ell @ \ell') \end{pmatrix}$ nil = empty list

2. Sequences $(X^{\star}, \cdot, \varepsilon)$:

 $X^{\star} = \bigcup_{n \in \mathbb{N}} X^{n}$ $\cdot = \text{concatenation}$ $((x_{1}, \dots, x_{m}) \cdot (y_{1}, \dots, y_{n}) = (x_{1}, \dots, x_{m}, y_{1}, \dots, y_{n}))$ $\varepsilon = ()$ 3. The set of **endomorphisms** on a set: $End(X) = (X \Rightarrow X, \circ, id_X)$ is a monoid.

In particular, the monoid End(1) is trivial.

A **monoid homomorphism** $h : \underline{M}_1 \to \underline{M}_2$ from a monoid $\underline{M}_1 = (M_1, \bullet_1, \iota_1)$ to a monoid $\underline{M}_2 = (M_2, \bullet_2, \iota_2)$ is a function $h : M_1 \to M_2$ such that

for all x, y ∈ M₁, h(x •₁ y) = h(x) •₂ h(y)
h(ι₁) = ι₂

Example: For all $f : X \to Y$,

 $\operatorname{map} f:\operatorname{List} X\to\operatorname{List} Y$

is a monoid homomorphism. (Check it.)

The **product** $\underline{M}_1 \times \underline{M}_2$ of monoids $\underline{M}_1 = (M_1, \bullet_1, \iota_1)$ and $\underline{M}_2 = (M_2, \bullet_2, \iota_2)$ is the structure

 $(M_1 \times M_2, \bullet, \iota)$

with

 $(x_1, x_2) \bullet (y_1, y_2) = (x_1 \bullet_1 y_1, x_2 \bullet_2 y_2)$

 $\iota = (\iota_1, \iota_2)$

and projections homomorphisms

$$\underline{M}_1 \xleftarrow{\pi_1} \underline{M}_1 \times \underline{M}_2 \xrightarrow{\pi_2} \underline{M}_2$$

given by $\pi_1(x_1, x_2) = x_1$ and $\pi_2(x_1, x_2) = x_2$.

Explore

- ? Is there a monoid of homomorphisms between monoids?
- ? What is the sum $\underline{M}_1 +_{Mon} \underline{M}_2$ of monoids \underline{M}_1 and \underline{M}_2 ?

Can you make sense of the following? $List(X) +_{Mon} List(Y) = List(X +_{Set} Y)$

Groups

A **group** is a structure $\underline{G} = (G, \neg, \bullet, \iota)$ consisting of a monoid (G, \bullet, ι) and a unary operation $\overline{-}: M \to M$ that satisfies:

• the inverse laws: $\forall x \in G, x \bullet \overline{x} = \iota = \overline{x} \bullet x$

A **group homomorphism** $h : \underline{G}_1 \to \underline{G}_2$ from a group $\underline{G}_1 = (G_1, \stackrel{-1}{}, \bullet_1, \iota_1)$ to a group $\underline{G}_2 = (G_2, \stackrel{-2}{}, \bullet_2, \iota_2)$ is a monoid homomorphism $h : (G_1, \bullet_1, \iota_1) \to (G_2, \bullet_2, \iota_2)$ such that

• for all
$$x \in G_1$$
, $h(\overline{x}^1) = \overline{h(x)}^2$

Examples:

The set of integers modulo a prime p

has a group structure.

► The set of automorphisms on a set Aut(X) = { f : X → X | f is a bijection } has a group structure.

 \mathbb{Z}_{p}

Definitions:

- A function $f : X \to Y$ is a **bijection** whenever $\forall y \in Y, \exists ! x \in X, f(x) = y$
- A function $f : X \to Y$ is an **isomorphism** whenever there exists a (necessarily unique) function $g : Y \to X$ (typically denoted f^{-1}) such that $g \circ f = id_X$ and $f \circ g = id_Y$.

Proposition. A function is a *bijection* if, and only if, it is an *isomorphism*.

Explore the above for homomorphisms between monoids and between groups.

Universal problems

 Vague problem: To manufacture a monoid out of a set in the most general or least constrained possible way.

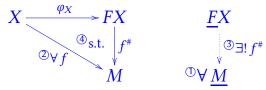
This is typically referred to as *freely generating* a monoid from a set.

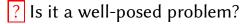
• Mathematical problem: For a set *X*, consider the *data* of interest to be given by a monoid $\underline{M} = (M, \bullet, \iota)$ together with a function $f : X \to M$. Given a set *X*,

1. construct data $\underline{FX} = (FX, \bullet_X, \iota_X)$ and $\varphi_X : X \to FX$ such that

1. for all data $\underline{M} = (M, \bullet, \iota)$ and $f : X \to M$, there exists a unique monoid homomorphism $f^{\#} : FX \to \underline{M}$ such that $f^{\#} \circ \varphi_X = f$.

In diagrammatic form:





- ? Does it have a solution?
- ? If so, how can we construct it?

Categories

A category C is an algebraic structure specified by

- ► a set obj C whose elements are called C-objects
- ► for each $X, Y \in obj C$, a set C(X, Y) whose elements are called C-morphisms from X to Y
- ► a function assigning to each $X \in obj C$ an element $id_X \in C(X, X)$ called the identity morphism for the C-object X
- ► a function assigning to each $f \in C(X, Y)$ and $g \in C(Y, Z)$ (where $X, Y, Z \in obj C$) an element $g \circ f \in C(X, Z)$ called the composition of C-morphisms f and g

satisfying

► associativity: for all $X, Y, Z, W \in obj C$, $f \in C(X, Y), g \in C(Y, Z)$ and $h \in C(Z, W)$

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

• unity: for all $X, Y \in obj C$ and $f \in C(X, Y)$

$$\operatorname{id}_Y\circ f=f=f\circ\operatorname{id}_X$$

Category of sets: Set

- obj Set = a fixed universe of sets
- Set-morphisms are functions: Set $(X, Y) = (X \Rightarrow Y)$
- Identities:

id_X

- Composition of $f \in \text{Set}(X, Y)$ and $g \in \text{Set}(Y, Z)$ is: $g \circ f$
- NB: Associativity and unity laws hold. (Check it.)

Category of monoids: Mon

- objects are monoids,
- morphisms are monoid homomorphisms,
- identities and composition are as for sets and functions.

Q: why is this well-defined?

A: because the set of functions that are monoid homomorphisms contains identity functions and is closed under composition.

Category of groups: Grp

- objects are groups,
- morphisms are group homomorphisms,
- identities and composition are as for sets and functions.

Q: why is this well-defined?

A: because the set of functions that are group homomorphisms contains identity functions and is closed under composition.

Conventions:

• Given a category C, one writes $f: X \to Y$ or $X \xrightarrow{f} Y$ or $Y \xleftarrow{f} X$ for

 $f \in \mathbf{C}(X, Y)$

in which case one says

object X is the domain (or source) of fobject Y is the codomain (or target) of f

and writes

 $X = \operatorname{dom}_{\mathbb{C}} f = \operatorname{dom} f$, $Y = \operatorname{cod}_{\mathbb{C}} f = \operatorname{cod} f$ NB: Which category C one is referring to is often left implicit.

- The sets C(X, Y) are typically referred to as hom-sets and sometimes also denoted hom_C(X, Y) or simply hom(X, Y) when C is clear from the context.
- One often abbreviates $g \circ f$ as g f.
- Because of the associativity law, one unambigously writes

 $h \circ g \circ f$ or hgf

for either of the equal composites $h \circ (g \circ f) = h (g f)$ and $(h \circ g) \circ f = (h g) f$.

Alternative notations

Some people write |C| or C for obj C id for id_X $Hom_C(X, Y)$ for C(X, Y)X or 1_X for id_X

Most people use "applicative order" for morphism composition; some people use "diagrammatic order" and write

 $f; g \text{ for } g \circ f$

Commutative diagrams

In a category C:

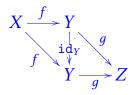
a diagram is

a directed graph whose vertices are C-objects and whose edges are C-morphisms

and the diagram is commutative (or commutes) if any two finite paths in the graph between any two vertices determine equal morphisms in the category under composition

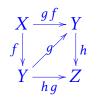


The diagram



commutes by the unity laws.

► The diagram



commutes by the associativity law.

One-object categories

Problem: Give an equivalent elementary description of categories with a singleton set of objects.

Each monoid determines a category

Given a monoid $\underline{M} = (M, \bullet, \iota)$, we get a category

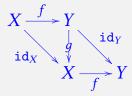
С<u>М</u>

by taking

- objects: obj $C_{\underline{M}} = \{*\}$ (a singleton set)
- morphisms: $C_{\underline{M}}(*,*) = M$
- identity morphism: $id_* = \iota \in M = C_{\underline{M}}(*, *)$
- ► composition $g \circ f \in \mathbf{C}_{\underline{M}}$ of $f \in \mathbf{C}_{\underline{M}}(*,*)$ and $g \in \mathbf{C}_{\underline{M}}(*,*)$ is $g \bullet f \in M = \mathbf{C}_{\underline{M}}(*,*)$

Isomorphism

Let C be a category. A C-morphism $f : X \to Y$ is an isomorphism if there is some $g : Y \to X$ for which



is a commutative diagram.

Isomorphism

Let C be a category. A C-morphism $f : X \to Y$ is an isomorphism if there is some $g : Y \to X$ with $g \circ f = id_X$ and $f \circ g = id_Y$.

- Such a g is uniquely determined by f (why?) and we write f⁻¹ for it.
- Given X, Y ∈ C, if such an f exists, we say the objects X and Y are isomorphic in C and write X ≅ Y

NB: There may be many different morphisms witnessing the fact that two objects are isomorphic.

Proposition. A function $f \in \text{Set}(X, Y)$ is an isomorphism in the category Set iff f is a bijection, equivalently:

• injective: $\forall x, x' \in X, f x = f x' \Rightarrow x = x'$

and

• surjective: $\forall y \in Y, \exists x \in X, f x = y$

Proposition. A function $f \in \text{Set}(X, Y)$ is an isomorphism in the category Set iff f is a bijection, equivalently:

• injective: $\forall x, x' \in X, f x = f x' \Rightarrow x = x'$

and

• surjective: $\forall y \in Y, \exists x \in X, f x = y$

Proposition. A monoid morphism $f \in Mon((M_1, \bullet_1, \iota_1), (M_2, \bullet_2, \iota_2))$ is an isomorphism in the category **Mon** iff $f \in Set(M_1, M_2)$ is a bijection.

Categories with trivial hom-sets

Problem: Give an equivalent elementary description of categories with trivial hom-sets in the sense of being either empty or a singleton set.

Preorders and posets

A **preorder** $\underline{P} = (P, \sqsubseteq)$ consists of a set *P* to equipped with a binary relation on it $_ \sqsubseteq _ \subseteq P \times P$ that is

reflexive: $\forall x \in P, x \sqsubseteq x$

transitive: $\forall x, y, z \in P, x \sqsubseteq y \land y \sqsubseteq z \Rightarrow x \sqsubseteq z$

A **poset** or (**partial order**) is a preorder that is also anti-symmetric: $\forall x, y \in P, x \sqsubseteq y \land y \sqsubseteq x \Rightarrow x = y$

Examples:

- $\blacktriangleright \quad (\mathbb{N}, \leq) \ , \ (\mathbb{N}, \geq)$
- $\blacktriangleright (\mathscr{P}(X), \subseteq), (\mathscr{P}(X), \supseteq)$
- $(\mathbb{Z}, |)$ where $n | m \stackrel{\triangle}{\Leftrightarrow} n$ divides m

Proposition.

 If <u>P</u> = (P, ⊑) is a preorder, then so is <u>P</u>^{op} ≜ (P, ⊒) where x ⊒ y ⇔ y ⊑ x.
 (<u>P</u>^{op})^{op} = <u>P</u>

Each preorder determines a category

Given a preorder $\underline{P} = (P, \sqsubseteq)$, we get a category

by taking

 \mathbf{C}_{P}

- objects: obj $C_P = P$
- ► morphisms: $C_{\underline{P}}(x, y) \triangleq \begin{cases} \{ (x, y) \} & \text{, if } x \sqsubseteq y \\ \emptyset & \text{, if } x \not\sqsubseteq y \end{cases}$
- identity morphisms and composition are uniquely determined (why?)

Each preorder determines a category

Given a preorder $\underline{P} = (P, \sqsubseteq)$, we get a category

by taking

 \mathbf{C}_{P}

- objects: obj $C_P = P$
- ► morphisms: $C_{\underline{P}}(x, y) \triangleq \begin{cases} \{(x, y)\} & \text{, if } x \sqsubseteq y \\ \emptyset & \text{, if } x \not\sqsubseteq y \end{cases}$
- identity morphisms and composition are uniquely determined (why?)

Each preorder determines a category

Given a preorder $\underline{P} = (P, \sqsubseteq)$, we get a category

by taking

 \mathbf{C}_{P}

- objects: obj $C_P = P$
- ► morphisms: $C_{\underline{P}}(x, y) \triangleq \begin{cases} \{(x, y)\} & \text{, if } x \sqsubseteq y \\ \emptyset & \text{, if } x \not\sqsubseteq y \end{cases}$
- identity morphisms and composition are uniquely determined (why?)

E.g. when
$$\underline{P}$$
 has just two elements $0 \sqsubseteq 1$

$$C_{\underline{P}} = \begin{bmatrix} (0,0) = id_0 & 0 & 0 & 0 \\ (0,0) = id_0 & 0 & 0 & 0 & 0 \\ (0,$$

Category of preorders: Preord

- objects: preorders
- morphisms:

Preord((P_1 , \sqsubseteq_1), (P_2 , \sqsubseteq_2)) ≜ { $f \in$ **Set**(P_1 , P_2) | f is monotone}

monotonicity: $\forall x, x' \in P_1, x \sqsubseteq_1 x' \Rightarrow f x \sqsubseteq_2 f x'$

Category of preorders: Preord

- objects: preorders
- morphisms:

Preord((P_1 , \sqsubseteq_1), (P_2 , \sqsubseteq_2)) ≜ { $f \in$ **Set**(P_1 , P_2) | f is monotone}

monotonicity: $\forall x, x' \in P_1, x \sqsubseteq_1 x' \Rightarrow f x \sqsubseteq_2 f x'$

identities and composition: as for Set

Q: why is this well-defined?

A: because the set of monotone functions contains identity functions and is closed under composition.

Subcategory of posets: Poset

Define **Poset** to be the category whose objects are posets, and is otherwise defined like the category **Preord** of preorders.

NB: Pre and partial orders are relevant to the denotational semantics of programming languages (among other things).

Proposition. A morphism $f \in \text{Poset}((P_1, \sqsubseteq_1), (P_2, \sqsubseteq_2))$ is an isomorphism in the category **Poset** iff the function $f \in \text{Set}(P_1, P_2)$ is

surjective: $\forall y \in P_2, \exists x \in P_1, f(x) = y$ and

reflective: $\forall x, x' \in P_1, f x \sqsubseteq_2 f x' \Rightarrow x \sqsubseteq_1 x'$

(Why does this characterisation not work for Preord?)

Problem: Recalling that categories generalise both monoids and preorders, find a notion of morphism between categories that generalises both monoid homomorphisms and monotone functions.

Category-theoretic properties

Any two isomorphic objects in a category should have the same category-theoretic properties – statements that are provable in a formal logic for category theory, whatever that is.

Instead of trying to formalize such a logic, we will just look at examples of category-theoretic properties.

Here is our first one...

Terminal object

An object *T* of a category C is terminal if for all $X \in C$, there is a unique C-morphism from *X* to *T*, which we write as $\langle \rangle_X : X \to T$.

So we have
$$\begin{cases} \forall X \in \mathbf{C}, \ \langle \rangle_X \in \mathbf{C}(X, T) \\ \forall X \in \mathbf{C}, \forall f \in \mathbf{C}(X, T), \ f = \langle \rangle_X \end{cases}$$

(In particular, $id_T = \langle \rangle_T$.)

Convention: Sometimes we write $!_X$ or X for $\langle \rangle_X$ —there is no commonly accepted notation— and also just write $\langle \rangle$ or !.

Examples of terminal objects

- In <u>Set</u> a set is terminal iff it is a singleton.
- Any one-element set has a unique preorder and this makes it terminal in <u>Preord</u> and <u>Poset</u>.
- Any one-element set has a unique monoid (group) structure and this makes it terminal in <u>Mon</u> (Grp).

Examples of terminal objects

- In <u>Set</u> a set is terminal iff it is a singleton.
- Any one-element set has a unique preorder and this makes it terminal in <u>Preord</u> and <u>Poset</u>.
- Any one-element set has a unique monoid (group) structure and this makes it terminal in <u>Mon</u> (Grp).
- ► A preorder $\underline{P} = (P, \sqsubseteq)$, regarded as a category $\underline{C}_{\underline{P}}$, has a terminal object iff it has a greatest element \top , that is: $\forall x \in P, x \sqsubseteq \top$.

Terminal object

terminal objects are unique up to unique isomorphism

Terminal object

terminal objects are unique up to unique isomorphism

Proposition. In a category,

(a) If *T* and *T'* are both terminal, then *T* ≅ *T'* (and there is only one isomorphism between *T* and *T'*).
(b) If *T* is terminal and *T* ≅ *T'*, then *T'* is terminal.

Notation: If a category C has a terminal object we will write that object as 1_{C} or 1_{C} .

Global elements

Given a category **C** with a terminal object **1**.

A global element of an object $X \in obj C$ is by definition a morphism $1 \rightarrow X$ in C.

E.g. Set $(1_{Set}, X) \cong X$; in Mon $(1_{Mon}, \underline{M}) \cong 1_{Set}$.

Global elements

Given a category C with a terminal object 1.

A global element of an object $X \in obj C$ is by definition a morphism $1 \rightarrow X$ in C.

E.g. $\operatorname{Set}(1_{\operatorname{Set}}, X) \cong X$; in $\operatorname{Mon}(1_{\operatorname{Mon}}, \underline{M}) \cong 1_{\operatorname{Set}}$.

Say that **C** is well-pointed if for all $f, g : X \rightarrow Y$ in **C** we have:

$$\left(\forall 1 \xrightarrow{x} X \text{ in } \mathbf{C}, \ f \circ x = g \circ x \right) \implies f = g$$

E.g. Set is well-pointed (by function extensionality); Mon is not.

Generalising elements

• **Proposition**. For all $f, g : M \to M'$ in **Mon**, if

 $\forall e : \mathbb{N} \to M$ in Mon, $f \circ e = g \circ e : \mathbb{N} \to M'$ then

f = g

Directed graphs

Let **DiGph** be the category with

- ▶ objects: $(E, N \in obj Set, s, t : E \rightarrow N in Set)$;
- morphisms:

$$h = (h_{\rm e}, h_{\rm n}) : (E \xrightarrow{s}_{t} N) \longrightarrow (E' \xrightarrow{s'}_{t'} N')$$

given by functions $h_{\rm e}: E \to E'$ and $h_{\rm n}: N \to N'$ such that

$$\forall a \in E. \ s'(h_e a) = h_n(s a)$$

and

$$\forall a \in E. t'(h_e a) = h_n(t a)$$

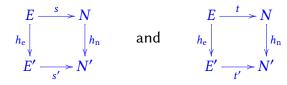
 identities and composition: given pointwise as in Set.

Directed graphs in a category

- For a category C, let DiGph(C) be the category with
 - objects: diagrams $E \xrightarrow{s} N$ in C;
 - morphisms:

$$h = (h_{\rm e}, h_{\rm n}) : (E \xrightarrow[t]{s} N) \longrightarrow (E' \xrightarrow[t]{s'} N')$$

given by $h_e: E \to E'$ and $h_n: N \to N'$ in **C** such that



commute;

identities and composition: given pointwise as in C.

NB. DiGph(Set) = DiGph.

Proposition. DiGph(Set) is not well-pointed. (Why?)

? Is there a single $C \in \text{DiGph}(\text{Set})$ such that, for all $f, g: G \to G'$ in DiGph(Set),

 $\forall c : C \rightarrow G \text{ in DiGph(Set)}, f \circ c = g \circ c : C \rightarrow G'$ implies

$$f = g \quad ?$$

Proposition. There exist $A, B \in \text{DiGph}(\text{Set})$ such that for all $f, g : G \rightarrow G'$ in DiGph(Set), if

 $\forall a : A \rightarrow G \text{ in DiGph(Set)}, f \circ a = g \circ a : A \rightarrow G'$ and

 $\forall b : B \rightarrow G \text{ in DiGph(Set)}, f \circ b = g \circ b : B \rightarrow G'$ then

f = g

Generalised elements

Idea:

Replace global elements $1 \xrightarrow{x} X$ of X by morphisms $C \xrightarrow{x} X$ for $C \in obj C$

Generalised elements

Idea:

Replace global elements $1 \xrightarrow{x} X$ of X

by morphisms $C \xrightarrow{x} X$ for $C \in obj C$

Some people say that x is a generalised element of X at stage C and use the notation $x \in_C X$. For instance, $\langle \rangle_C \in_C 1$ is the unique generalised element of 1 at stage C.

One may also think that *x* inhabits *X* in context *C* and use the notation $C \vdash x : X$; for instance, $C \vdash \langle \rangle_C : 1$.

NB: One has to take into account "change of stage or context": for $\sigma : D \rightarrow C$,

 $x \in_C X \implies x \sigma \in_D X$

 $C \vdash x : X \implies D \vdash x \, \sigma : X$

(cf. Kripke's "possible world" semantics of intuitionistic and modal logics)

Opposite of a category

Given a category C, its opposite category C^{op} is defined by interchanging the operations of dom and cod in C:

- ▶ obj C^{op} ≜ obj C
- $\mathbf{C}^{\mathrm{op}}(X, Y) \triangleq \mathbf{C}(Y, X)$, for all objects X and Y
- identity morphism on X ∈ obj C^{op} is
 id_X ∈ C(X, X) = C^{op}(X, X)
- composition in C^{op} of f ∈ C^{op}(X, Y) and g ∈ C^{op}(Y, Z) is given by the composition f ∘ g ∈ C(Z, X) = C^{op}(X, Z) in C (associativity and unity properties hold for this operation because they do in C)

The Principle of Duality

Whenever one defines a concept / proves a theorem in terms of commutative diagrams in a category C, one obtains another concept / theorem, called its dual, by reversing the direction or morphisms throughout, that is, by replacing C by its opposite category C^{op} .

For example, "isomorphism" is a self-dual concept.

Initial object

(the dual notion to "terminal object")

An object 0 of a category C is initial if for all $X \in C$, there is a unique C-morphism $0 \to X$, which we write as $[]_X : 0 \to X]$.

So we have
$$\begin{cases} \forall X \in \mathbf{C}, \ []_X \in \mathbf{C}(0, X) \\ \forall X \in \mathbf{C}, \forall f \in \mathbf{C}(0, X), \ f = []_X \end{cases}$$
(In particular, id₀ = []₀.)

NB: By duality, we have that initial objects are unique up to unique isomorphism and that any object isomorphic to an initial object is itself initial.

Examples of initial objects

- The empty set is initial in **Set**.
- Any singleton set has a uniquely determined monoid structure and is initial in Mon. (why?)

So initial and terminal objects coincide in Mon An object that is both initial and terminal in a category is called a zero object.

► A preorder $\underline{P} = (P, \sqsubseteq)$, regarded as a category $C_{\underline{P}}$, has an initial object iff it has a least element \bot , that is: $\forall x \in P, \bot \sqsubseteq x$.

The free monoid on a set X is List X = (List X, @, nil) where

List X = set of finite lists of elements of X

(a) = list concatenation

nil = empty list

The free monoid on a set X is List X = (List X, @, nil) where

List X = set of finite lists of elements of X
 @ = list concatenation
 nil = empty list

The singleton-list function

 $s_X : X \rightarrow \text{List} X$ $x \mapsto [x] = x :: \text{nil}$

has the following (initial) universal property ...

Theorem. For any monoid $\underline{M} = (M, \bullet, \iota)$ and function $f: X \to M$, there is a unique monoid morphism $f^{\#} \in \operatorname{Mon}(\operatorname{List} X, \underline{M})$ making



commute in Set.

Theorem. $\forall \underline{M} \in \text{Mon}, \forall f \in \text{Set}(X, M), \exists ! f^{\#} \in \text{Mon}(\underline{\text{List}} X, \underline{M}), f^{\#} \circ s_X = f$

The theorem just says that $s_X : X \rightarrow \text{List } X$ is an initial object in the category X/Mon:

- objects: (\underline{M}, f) where $\underline{M} \in obj$ Mon and $f \in Set(X, M)$
- morphisms in $X/Mon((\underline{M}_1, f_1), (\underline{M}_2, f_2))$ are $h \in Mon(\underline{M}_1, \underline{M}_2)$ such that $h \circ f_1 = f_2$
- identities and composition as in Mon

Theorem. $\forall \underline{M} \in \mathbf{Mon}, \forall f \in \mathbf{Set}(X, M), \exists ! f^{\#} \in \mathbf{Mon}(\underline{\text{List}} X, \underline{M}), f^{\#} \circ \mathbf{s}_{X} = f$

The theorem just says that $s_X : X \rightarrow \text{List } X$ is an initial object in the category X/Mon:

So this "universal property" determines the monoid List X uniquely up to isomorphism in **Mon**.

We will see later that $X \mapsto \texttt{List} X$ is part of a functor (= morphism of categories) which is left adjoint to the "forgetful functor" **Mon** \rightarrow **Set** : $\underline{M} \mapsto M$.

Products

Problem: In a category, find a universal construction specifying a product object $X \times Y$ that internalises pairs of generalised elements of objects X and Y.

Products

Problem: In a category, find a universal construction specifying a product object $X \times Y$ that internalises pairs of generalised elements of objects X and Y.

That is,

$$\frac{C \longrightarrow X \times Y}{C \longrightarrow X \quad C \longrightarrow Y}$$

where the passage from top to bottom is given by projecting on the first and second components.

More precisely,

 $X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y$

such that

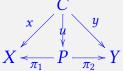
 $\hom(C, X \times Y) \xrightarrow{\langle \pi_1 \circ _, \pi_2 \circ _} \hom(C, X) \times \hom(C, Y)$

is an isomorphism.

Binary products

In a category C, a product for objects $X, Y \in C$ is a diagram $X \xleftarrow{\pi_1} P \xrightarrow{\pi_2} Y$ with the universal property:

For all $X \xleftarrow{x} C \xrightarrow{y} Y$ in C, there is a unique C-morphism $u: C \rightarrow P$ such that the following diagram commutes in C: C



Binary products

In a category C, a product for objects $X, Y \in C$ is a diagram $X \xleftarrow{\pi_1} P \xrightarrow{\pi_2} Y$ with the universal property:

For all $X \xleftarrow{x} C \xrightarrow{y} Y$ in C, there is a unique C-morphism $u: C \to P$ such that $x = \pi_1 \circ u$ and $y = \pi_2 \circ u$

So (P, π_1, π_2) is a terminal object in the category with

- objects: (C, x, y) where $X \xleftarrow{x} C \xrightarrow{y} Y$ in C
- ▶ morphisms $f : (C_1, x_1, y_1) \rightarrow (C_2, x_2, y_2)$ are $f \in C(C_1, C_2)$ such that $x_1 = x_2 \circ f$ and $y_1 = y_2 \circ f$
- composition and identities as in C

So if it exists, the binary product of two objects in a category is unique up to (unique) isomophism.

Binary products

In a category C, a product for objects $X, Y \in C$ is a diagram $X \xleftarrow{\pi_1} P \xrightarrow{\pi_2} Y$ with the universal property:

For all $X \xleftarrow{x} C \xrightarrow{y} Y$ in **C**, there is a unique **C**-morphism $u: C \to P$ such that $x = \pi_1 \circ u$ and $y = \pi_2 \circ u$

N.B. products of objects in a category do not always exist. For example in the category



the objects 0 and 1 do not have a product, because there is no diagram of the form $0 \leftarrow ? \rightarrow 1$ in this category.

Notation for binary products

Assuming C has binary products of objects, the product of $X, Y \in C$ is written

$$X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y$$

and given $X \xleftarrow{x} C \xrightarrow{y} Y$, the unique $u : C \to X \times Y$ with $\pi_1 \circ u = x$ and $\pi_2 \circ u = y$ is written

$$\langle x, y \rangle : C \to X \times Y$$

Examples:

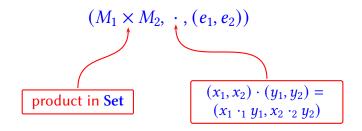
In Set, category-theoretic products are given by the usual cartesian product of sets (set of all ordered pairs) and their projections:

$$X \times Y = \{(x, y) \mid x \in X \land y \in Y\}$$
$$\pi_1(x, y) = x$$
$$\pi_2(x, y) = y$$

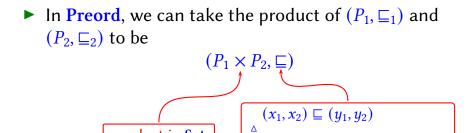
In Mon, can take product of (M₁, ·₁, e₁) and (M₂, ·₂, e₂) to be

 $(M_1 \times M_2, \cdot, (e_1, e_2))$ $(x_1, x_2) \cdot (y_1, y_2) = (x_1 \cdot_1 y_1, x_2 \cdot_2 y_2)$ product in Set

In Mon, can take product of (M₁, ·₁, e₁) and (M₂, ·₂, e₂) to be



The projection functions $M_1 \stackrel{\pi_1}{\leftarrow} M_1 \times M_2 \stackrel{\pi_2}{\rightarrow} M_2$ are monoid morphisms for this monoid structure on $M_1 \times M_2$ and have the universal property needed for a product in **Mon** (check).

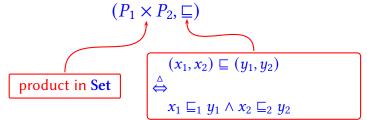


product in Set

 $\stackrel{\wedge}{\Leftrightarrow}$

 $x_1 \sqsubseteq_1 y_1 \wedge x_2 \sqsubseteq_2 y_2$

In Preord, we can take the product of (P₁, ⊑₁) and (P₂, ⊑₂) to be



The projection functions $P_1 \xleftarrow{\pi_1} P_1 \times P_2 \xrightarrow{\pi_2} P_2$ are monotone for this preorder on $P_1 \times P_2$ and have the universal property needed for a product in **Preord** (check).

Recall that each preorder <u>P</u> = (P, ⊑) determines a category C<u>P</u>.
 Given p, q ∈ P = obj C_P, the product p × q (if it exists) is a greatest lower bound (or glb, or meet) for p and q in <u>P</u>:
 lower bound:

 $p \times q \sqsubseteq p \land p \times q \sqsubseteq q$ greatest among all lower bounds: $\forall \ell \in P, \ \ell \sqsubseteq p \land \ell \sqsubseteq q \implies \ell \sqsubseteq p \times q$

Notation: glbs are often written $p \land q$ or $p \sqcap q$

Binary product of morphisms

Suppose a category C has binary products; that is, for every pair of C-objects X and Y there is a product diagram $X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y$.

Given $f \in \mathbf{C}(A, X)$ and $g \in \mathbf{C}(B, Y)$, then

$$f \times g : A \times B \to X \times Y$$

stands for $\langle f \circ \pi_1, g \circ \pi_2 \rangle$; that is, the unique morphism $u \in \mathbb{C}(A \times B, X \times Y)$ satisfying $\pi_1 \circ u = f \circ \pi_1$ and $\pi_2 \circ u = g \circ \pi_2$.

Binary coproducts

A binary coproduct of two objects in a category C is their product in the category C^{op} .

Binary coproducts

A binary coproduct of two objects in a category C is their product in the category C^{op} .

Thus the coproduct of $X, Y \in \mathbb{C}$ if it exists, is a diagram $X \xrightarrow{\iota_1} X + Y \xleftarrow{\iota_2} Y$ with the universal property: $\forall (X \xrightarrow{f} Z \xleftarrow{g} Y),$ $\exists ! (X + Y \xrightarrow{[f,g]} Z),$ $f = [f,g] \circ \iota_1 \land g = [f,g] \circ \iota_2$

Binary coproducts

A binary coproduct of two objects in a category C is their product in the category C^{op} .

Thus the coproduct of $X, Y \in \mathbb{C}$ if it exists, is a diagram $X \xrightarrow{\iota_1} X + Y \xleftarrow{\iota_2} Y$ with the universal property: $\langle _\circ \iota_1, _\circ \iota_2 \rangle : \mathbb{C}(X+Y, Z) \xrightarrow{\cong} \mathbb{C}(X, Z) \times \mathbb{C}(Y, Z)$

Examples:

► In Set, the coproduct of *X* and *Y*

$$X \xrightarrow{\iota_1} X + Y \xleftarrow{\iota_2} Y$$

is given by their disjoint union (tagged sum)

 $X + Y = \{(1, x) \mid x \in X\} \cup \{(2, y) \mid y \in Y\}$ $\iota_1(x) = (1, x)$ $\iota_2(y) = (2, y)$

(prove this)

Recall that each preorder <u>P</u> = (P, ⊑) determines a category C<u>P</u>.
Given p, q ∈ P = obj C_P, the coproduct p + q (if it exists) is a least upper bound (or lub, or join) for p and q in <u>P</u>:
upper bound:
p ⊑ p + q ∧ q ⊑ p + q

least among all upper bounds:

 $\forall u \in P, \ p \sqsubseteq u \land q \sqsubseteq u \implies p + q \sqsubseteq u$ Notation: lubs are often written $p \lor q$ or $p \sqcup q$

77

Binary coproduct of morphisms

Suppose a category C has binary coproducts; that is, for every pair of C-objects X and Y there is a coproduct diagram $X \xrightarrow{\iota_1} X + Y \xleftarrow{\iota_2} Y$.

Given $f \in \mathbf{C}(A, X)$ and $g \in \mathbf{C}(B, Y)$, then

$$f + g : A + B \to X + Y$$

stands for $[\iota_1 \circ f, \iota_2 \circ g]$; that is, the unique morphism $u \in \mathbf{C}(A + B, X + Y)$ satisfying $u \circ \iota_1 = \iota_1 \circ f$ and $u \circ \iota_2 = \iota_2 \circ g$.

Exponentials

Problem: In a category with binary products, find a universal construction specifying an exponential object (or internal hom) $X \Rightarrow Y$ with generalised elements corresponding to parameterised morphisms from X to Y.

Exponentials

Problem: In a category with binary products, find a universal construction specifying an exponential object (or internal hom) $X \Rightarrow Y$ with generalised elements corresponding to parameterised morphisms from X to Y. That is,

 $\frac{C \longrightarrow X \Rightarrow Y}{C \times X \longrightarrow Y}$

where the passage from top to bottom is given by application.

More precisely,

$$\operatorname{app}: (X \Longrightarrow Y) \times X \to Y$$

such that

$$\hom(C, X \Longrightarrow Y) \xrightarrow{\operatorname{app}\circ(_\times \operatorname{id}_X)} \hom(C \times X, Y)$$

is an isomorphism.

Exponential objects

Suppose a category C has binary products An exponential for C-objects X and Y is specified by a C-object $X \Rightarrow Y$ a C-morphism app : $(X \Rightarrow Y) \times X \rightarrow Y$ satisfying the universal property for all $C \in \mathbf{C}$ and $f \in \mathbf{C}(C \times X, Y)$, there is a unique $u \in C(C, X \Rightarrow Y)$ such that $(X \Rightarrow Y) \times X$ $u \times \mathrm{id}_X$ $C \times X$ commutes in C.

Notation: we write $\operatorname{cur} f$ for the unique u such that $\operatorname{app} \circ (u \times \operatorname{id}_X) = f$.

Exponential objects

The universal property of app : $(X \Rightarrow Y) \times X \rightarrow Y$ says that there is a bijection

 $hom(C, X \Rightarrow Y) \cong hom(C \times X, Y)$ $g \mapsto app \circ (g \times id_X)$ $cur f \leftarrow f$ $app \circ (cur f \times id_X) = f$ $g = cur(app \circ (g \times id_X))$

Exponential objects

The universal property of app : $(X \Rightarrow Y) \times X \rightarrow Y$ says that there is a bijection...

It also says that $(X \Rightarrow Y, app)$ is a terminal object in the following category:

- objects: (C, f) where $f \in \mathbf{C}(C \times X, Y)$
- morphisms $g: (C, f) \to (C', f')$ are $g \in \mathbb{C}(Z, Z')$ such that $f' \circ (g \times id_X) = f$
- composition and identities as in C.

So when they exist, exponential objects are unique up to (unique) isomorphism.

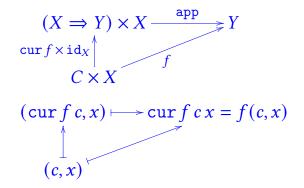
Example: Exponential objects in Set.

Given $X, Y \in$ **Set**, let $(X \Rightarrow Y) \in$ **Set** denote the set of all functions from *X* to *Y*.

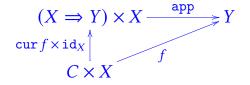
Function application gives a morphism app : $(X \Rightarrow Y) \times X \rightarrow Y$ in **Set**

app(f, x) = f x $(f \in (X \Rightarrow Y), x \in X)$

The Currying operation transforms morphisms $f: C \times X \to Y$ in Set to morphisms $\operatorname{cur} f: C \to X \Rightarrow Y$ in Set $\operatorname{cur} f c x = f(c, x)$ $(f \in (X \Rightarrow Y), c \in C, x \in X)$ For each function $f : C \times X \rightarrow Y$ we get a commutative diagram in Set:



For each function $f : C \times X \rightarrow Y$ we get a commutative diagram in Set:



Furthermore, if any function $g : C \rightarrow X \Rightarrow Y$ also satisfies

$$(X \Longrightarrow Y) \times X \xrightarrow{\text{app}} Y$$

$$g \times \operatorname{id}_X \uparrow f$$

$$C \times X$$

then $g = \operatorname{cur} f$, because of function extensionality.

Indeed,

 $\begin{aligned} \operatorname{app} \circ (g \times \operatorname{id}_X) &= f \\ \Rightarrow \forall (c, x) \in C \times X, \operatorname{app}(g c, x) = f(c, x) \\ \Rightarrow \forall x \in X, \forall c \in C, g c x = \operatorname{cur} f c x \\ \Rightarrow \forall c \in C, g c = \operatorname{cur} f c \\ \Rightarrow g = \operatorname{cur} f \end{aligned}$

Cartesian closed category

Definition. C is a cartesian closed category (ccc) if it is a category with a terminal object, binary products, and exponentials of any pair of objects.

This is a key concept for the semantics of lambda calculus and for the foundations of functional programming languages.

Examples:

- ▶ Set is a ccc as we have seen.
- ▶ **Preord** is a ccc: we already saw that it has a terminal object and binary products; the exponential of (P_1, \sqsubseteq_1) and (P_2, \sqsubseteq_2) is $(P_1 \Rightarrow P_2, \sqsubseteq)$ where

 $P_1 \Rightarrow P_2 \triangleq \mathbf{Preord}((P_1, \sqsubseteq_1), (P_2, \sqsubseteq_2))$

$$f \sqsubseteq g \stackrel{\scriptscriptstyle \Delta}{\Leftrightarrow} \forall x \in P_1, \ f \ x \sqsubseteq_2 g \ x$$

(check that this is a pre-order and does give an exponential in Preord)

DiGph(Set) is a ccc.

Bicartesian closed category

Definition. C is a bicartesian category if it is a category with a terminal and initial object, and binary products and coproducts of any pair of objects.

Definition. C is a bicartesian closed category (biccc) if it is a bicartesian category with exponentials of any pair of objects.

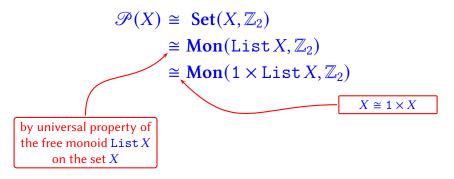
This is a key concept for the semantics of lambda calculus and for the foundations of functional programming languages.

Examples: Set, Preord, DiGph(Set) are bicccs.

Non-example of a ccc

The category Mon of monoids has a terminal object and binary products, but is <u>not</u> a ccc

because of the following bijections between sets, where 1 denotes a one-element set and the corresponding one-element monoid:



Non-example of a ccc

The category **Mon** of monoids has a terminal object and binary products, but is <u>not</u> a ccc

because of the following bijections between sets, where 1 denotes a one-element set and the corresponding one-element monoid:

 $\mathcal{P}(X) \cong \operatorname{Set}(X, \mathbb{Z}_2)$ $\cong \operatorname{Mon}(\operatorname{List} X, \mathbb{Z}_2)$ $\cong \operatorname{Mon}(1 \times \operatorname{List} X, \mathbb{Z}_2)$

Since the one-element monoid is initial in Mon, for any $M \in Mon$, we have $Mon(1, M) \cong 1$ and hence

List $X \Rightarrow \mathbb{Z}_2$ exists in Mon iff $\mathscr{P}(X) \cong 1$ iff X = 0

Btw, a ccc has a zero object if, and only if, it is trivial (check).

Cartesian closed pre-order

Recall that each preorder $\underline{P} = (P, \sqsubseteq)$ gives a category $\underline{C}_{\underline{P}}$. It is a biccc iff \underline{P} has

- ▶ a greatest element \top : $\forall p \in P, p \sqsubseteq \top$
- ▶ a least element \bot : $\forall p \in P, \bot \sqsubseteq p$
- binary meets $p \land q$: $\forall r \in P, r \sqsubseteq p \land q \Leftrightarrow r \sqsubseteq p \land r \sqsubseteq q$
- ► binary joins $p \lor q$: $\forall r \in P, \ p \lor q \sqsubseteq r \iff p \sqsubseteq r \land q \sqsubseteq r$
- Heyting implications $p \rightarrow q$: $\forall r \in P, \ r \sqsubseteq p \rightarrow q \iff r \land p \sqsubseteq q$

Examples:

- Any Boolean algebra (with $p \rightarrow q = \neg p \lor q$).
- ([0,1], \leq) with $\top = 1, \perp = 0, p \land q = \min\{p,q\},$ $p \lor q = \max\{p,q\}, \text{ and } p \to q = \begin{cases} 1 & \text{if } p \leq q \\ q & \text{if } q$

Intuitionistic Propositional Logic (IPL)

We present it in "natural deduction" style and only consider the fragment with conjunction and implication, with the following syntax:

Formulas of IPL: $\varphi, \psi, \theta, \ldots :=$

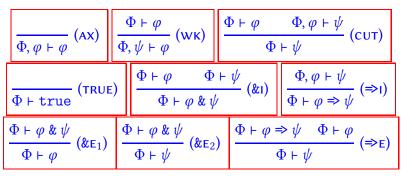
<i>p</i> , <i>q</i> , <i>r</i> ,	propositional identifiers
true	truth
$\varphi \& \psi$	conjunction
$\varphi \Rightarrow \psi$	implication

Sequents of IPL: $\Phi ::= \diamond$ empty Φ, φ non-empty

(so sequents are finite lists of formulas)

IPL entailment $\Phi \vdash \varphi$

The intended meaning of $\Phi \vdash \varphi$ is "the conjunction of the formulas in Φ implies the formula φ ". The relation $_\vdash$ _ is inductively generated by the following rules:



For example, if $\Phi = \diamond, \varphi \Rightarrow \psi, \psi \Rightarrow \theta$, then $\Phi \vdash \varphi \Rightarrow \theta$ is provable in IPL, because:

$$\frac{\overline{(\varphi,\varphi\Rightarrow\psi\vdash\varphi\Rightarrow\psi)}(AX)}{\Phi,\varphi\vdash\psi\Rightarrow\theta}(WK) = \frac{\overline{(\varphi,\varphi\Rightarrow\psi\vdash\varphi\Rightarrow\psi)}(WK)}{\Phi,\varphi\vdash\varphi\neq\psi}(WK) = \frac{\Phi,\varphi\vdash\varphi}{\Phi,\varphi\vdash\psi}(AX) = \frac{\Phi,\varphi\vdash\varphi}{\Phi,\varphi\vdash\psi}(AX) = \frac{\Phi,\varphi\vdash\varphi}{\Phi,\varphi\vdash\psi}(AX) = \frac{\Phi,\varphi\vdash\varphi}{\Phi,\varphi\neq\psi}(AX) = \frac{\Phi,\varphi\downarrow\varphi}{\Phi,\varphi\neq\psi}(AX) = \frac{\Phi,\varphi\varphi\varphi}{\Phi,\varphi\neq\psi}(AX) = \frac{\Phi,\varphi\varphi\varphi}{\Phi,\varphi\neq\psi}(AX) = \frac{\Phi,\varphi\varphi\varphi}{\Phi,\varphi\neq\psi}(AX) = \frac{\Phi,\varphi\varphi}{\Phi,\varphi\neq\psi}(AX) = \frac{\Phi,\varphi\varphi\varphi}{\Phi,\varphi\varphi}(AX) = \frac{\Phi,\varphi\varphi\varphi}{\Phi,\varphi\varphi}(AX) = \frac{\Phi,\varphi\varphi\varphi}{\Phi,\varphi\varphi}(AX) = \frac{\Phi,\varphi\varphi\varphi}{\Phi,\varphi\varphi}(AX) = \frac{\Phi,\varphi\varphi}{\Phi,\varphi\varphi}(AX) = \frac{\Phi,\varphi\varphi}{\Phi,\varphi}(AX) = \frac{\Phi$$

Semantics of IPL in a cartesian closed pre-order (P, \sqsubseteq)

Given a function *M* assigning a meaning to each propositional identifier *p* as an element $M(p) \in P$, we can assign meanings to IPL formula φ and sequents Φ as elements $M[\![\varphi]\!], M[\![\Phi]\!] \in P$ by recursion on their structure:

$$\begin{split} M[\![p]\!] &= M(p) \\ M[\![\texttt{true}]\!] &= \top & \text{greatest element} \\ M[\![\varphi \& \psi]\!] &= M[\![\varphi]\!] \land M[\![\psi]\!] & \text{binary meet} \\ M[\![\varphi \Rightarrow \psi]\!] &= M[\![\varphi]\!] \to M[\![\psi]\!] & \text{Heyting implication} \\ M[\![\diamond]\!] &= \top & \text{greatest element} \\ M[\![\Phi, \varphi]\!] &= M[\![\Phi]\!] \land M[\![\varphi]\!] & \text{binary meet} \end{split}$$

Semantics of IPL in a cartesian closed pre-order (P, \sqsubseteq)

Soundness Theorem. If $\Phi \vdash \varphi$ is provable from the rules of IPL, then $M[\![\Phi]\!] \sqsubseteq M[\![\varphi]\!]$ holds in any cartesian closed pre-order.

Proof. exercise (show that $\{(\Phi, \varphi) \mid M[\![\Phi]\!] \sqsubseteq M[\![\varphi]\!]\}$ is closed under the rules defining IPL entailment and hence contains $\{(\Phi, \varphi) \mid \Phi \vdash \varphi\}$)

Example

Peirce's Law $\diamond \vdash ((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi$ is <u>not</u> provable in IPL.

(whereas the formula $((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi$ is a classical tautology)

Example

Peirce's Law $\diamond \vdash ((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi$ is not provable in IPL.

(whereas the formula $((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi$ is a classical tautology)

For if $\diamond \vdash ((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi$ were provable in IPL, then by the Soundness Theorem we would have $\top = M[\![\diamond]\!] \sqsubseteq M[\![((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi]\!].$

But in the cartesian closed poset ([0, 1], \leq), taking M(p) = 1/2 and M(q) = 0, we get

$$M\llbracket((p \Rightarrow q) \Rightarrow p) \Rightarrow p\rrbracket = ((1/2 \to 0) \to 1/2) \to 1/2$$
$$= (0 \to 1/2) \to 1/2$$
$$= 1 \to 1/2$$
$$= 1/2$$
$$\neq 1$$

Semantics of IPL in a cartesian closed preorder (P, \sqsubseteq)

Completeness Theorem. Given Φ, φ , if for all cartesian closed preorders (P, \sqsubseteq) and all interpretations M of the propositional identifiers as elements of P, it is the case that $M[\![\Phi]\!] \sqsubseteq M[\![\varphi]\!]$ holds in P, then $\Phi \vdash \varphi$ is provable in IPL.

Semantics of IPL in a cartesian closed preorder (P, \sqsubseteq)

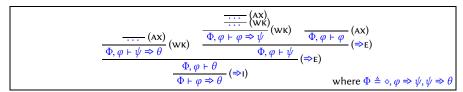
Completeness Theorem. Given Φ, φ , if for all cartesian closed preorders (P, \sqsubseteq) and all interpretations M of the propositional identifiers as elements of P, it is the case that $M[\![\Phi]\!] \sqsubseteq M[\![\varphi]\!]$ holds in P, then $\Phi \vdash \varphi$ is provable in IPL.

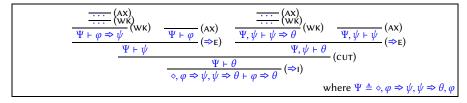
Proof. Define

 $P \triangleq \{\text{formulas of IPL}\}$ $\varphi \sqsubseteq \psi \triangleq \diamond, \varphi \vdash \psi \text{ is provable in IPL}$

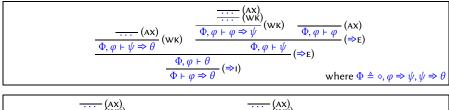
Then one can show that (P, \sqsubseteq) is a cartesian closed preorder. For this preorder, taking *M* to be M(p) = p, one can show that $M[\![\Phi]\!] \sqsubseteq M[\![\varphi]\!]$ holds in *P* iff $\Phi \vdash \varphi$ is provable in IPL.

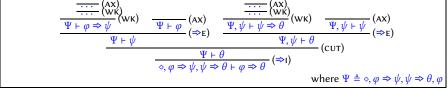
Two IPL proofs of $\diamond, \varphi \Rightarrow \psi, \psi \Rightarrow \theta \vdash \varphi \Rightarrow \theta$



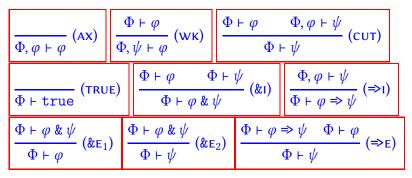


Two IPL proofs of $\diamond, \varphi \Rightarrow \psi, \psi \Rightarrow \theta \vdash \varphi \Rightarrow \theta$

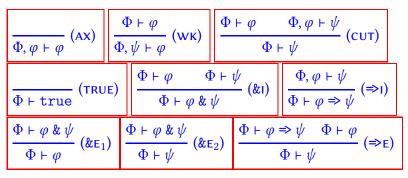




Why is the first proof simpler than the second one?



FACT: if an IPL sequent $\Phi \vdash \phi$ is provable from the rules, it is provable without using the (CUT) rule.



FACT: if an IPL sequent $\Phi \vdash \phi$ is provable from the rules, it is provable without using the (CUT) rule.

Simply-Typed Lambda Calculus provides a language for describing proofs in IPL and their properties.

Simply-Typed Lambda Calculus (STLC)

Types: *A*, *B*, *C*, . . . ::=

$G, G', G'' \dots$	"ground" types
unit	unit type
$A \times B$	product type
A ightarrow B	function type

Simply-Typed Lambda Calculus (STLC)

Types: *A*, *B*, *C*, . . . ::=

$G, G', G'' \dots$	"ground" types
unit	unit type
$A \times B$	product type
A ightarrow B	function type

Terms: *s*, *t*, *r*, . . . ::=

c ^A x () (s,t)	constants (of given type A) variable (countably many) unit value pair
fst t snd t	projections
$\lambda x : A. t$	function abstraction
s t	function application

STLC

Some examples of terms:

- ► $\lambda z : (A \to B) \times (A \to C)$. $\lambda x : A$. ((fst z) x, (snd z) x))(has type $((A \to B) \times (A \to C)) \to (A \to (B \times C)))$
- ► $\lambda z : A \to (B \times C). (\lambda x : A. fst(z x), \lambda y : A. snd(z y))$ (has type $(A \to (B \times C)) \to ((A \to B) \times (A \to C)))$
- ► $\lambda z : A \to (B \times C)$. $\lambda x : A$. ((fst z) x, (snd z) x)(has no type)

 Γ ranges over typing environments

 $\Gamma ::= \diamond \mid \Gamma, x : A$

(so typing environments are comma-separated lists of (variable,type)-pairs — in fact only the lists whose variables are mutually distinct get used)

The typing relation $\Gamma \vdash t : A$ is inductively defined by the following rules, which make use of the notation below Γ ok means: no variable occurs more than once in Γ dom Γ = finite set of variables occurring in Γ

Typing rules for variables

 $\frac{\Gamma \text{ ok } x \notin \text{dom } \Gamma}{\Gamma, x : A \vdash x : A} \text{ (VAR)}$ $\frac{\Gamma \vdash x : A \quad x' \notin \text{dom } \Gamma}{\Gamma, x' : A' \vdash x : A} \text{ (VAR')}$

Typing rules for constants and unit value

$$\frac{\Gamma \text{ ok}}{\Gamma \vdash c^A : A} \text{ (cons)}$$

$$\frac{\Gamma \text{ ok}}{\Gamma \vdash () : \text{ unit }} \text{ (unit)}$$

Typing rules for pairs and projections

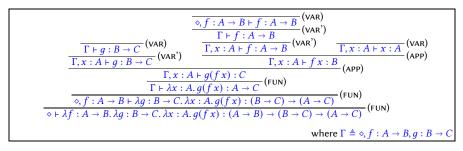
 $\frac{\Gamma \vdash s : A \qquad \Gamma \vdash t : B}{\Gamma \vdash (s, t) : A \times B} (PAIR)$ $\frac{\Gamma \vdash t : A \times B}{\Gamma \vdash fst \ t : A} (FST)$ $\frac{\Gamma \vdash t : A \times B}{\Gamma \vdash snd \ t : B} (SND)$

Typing rules for function abstraction & application

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A \cdot t : A \to B} (\text{fun})$$

$$\frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t : A}{\Gamma \vdash s t : B} (\text{app})$$

Example typing derivation:



NB: The STLC typing rules are "syntax-directed", by the structure of terms *t* and then in the case of variables *x*, by the structure of typing environments Γ .

Given a cartesian closed category **C**, any function *M* mapping ground types *G* to objects $M(G) \in \mathbf{C}$ extends to a function $A \mapsto M[\![A]\!] \in \mathbf{C}$ and $\Gamma \mapsto M[\![\Gamma]\!] \in \mathbf{C}$ from STLC types and typing environments to **C**-objects, by recursion on their structure:

$$\begin{split} M\llbracket G\rrbracket &= M(G) & \text{an object in } \mathbf{C} \\ M\llbracket \text{unit}\rrbracket &= 1 & \text{terminal object in } \mathbf{C} \\ M\llbracket A \times B\rrbracket &= M\llbracket A\rrbracket \times M\llbracket B\rrbracket & \text{product in } \mathbf{C} \\ M\llbracket A \to B\rrbracket &= M\llbracket A\rrbracket \Rightarrow M\llbracket B\rrbracket & \text{exponential in } \mathbf{C} \\ M\llbracket \diamond\rrbracket &= 1 & \text{terminal object in } \mathbf{C} \\ M\llbracket \Gamma, x : A\rrbracket &= M\llbracket \Gamma\rrbracket \times M\llbracket A\rrbracket & \text{product in } \mathbf{C} \end{split}$$

Given a cartesian closed category C, and

given any function M mapping

■ ground types G to C-objects M(G) (which extends to a function mapping all types to objects, A → M[A], as we have seen)

Given a cartesian closed category C, and given any function M mapping

- ► ground types *G* to C-objects *M*(*G*)
- constants c^A to C-morphisms $M(c^A) : 1 \to M[A]$ (In a category with a terminal object 1, given an object $X \in C$, morphisms $1 \to X$ are typically called global elements of X.)

Given a cartesian closed category C, and given any function M mapping

- ► ground types *G* to **C**-objects *M*(*G*)
- constants c^A to C-morphisms $M(c^A) : 1 \to M[A]$

we get a function mapping provable instances of the typing relation $\Gamma \vdash t : A$ to C-morphisms

$$M[\![\Gamma \vdash t : A]\!] : M[\![\Gamma]\!] \to M[\![A]\!]$$

defined by recursing over the proof of $\Gamma \vdash t : A$ from the typing rules (which follows the structure of *t*):

Semantics of STLC terms in a ccc Variables:

 $M\llbracket\Gamma, x : A \vdash x : A\rrbracket = M\llbracket\Gamma\rrbracket \times M\llbracketA\rrbracket \xrightarrow{\pi_2} M\llbracketA\rrbracket$ $M\llbracket\Gamma, x' : A' \vdash x : A\rrbracket$ $= M\llbracket\Gamma\rrbracket \times M\llbracketA'\rrbracket \xrightarrow{\pi_1} M\llbracket\Gamma\rrbracket \xrightarrow{M\llbracket\Gamma \vdash x : A\rrbracket} M\llbracketA\rrbracket$

Constants:

 $M\llbracket\Gamma \vdash c^{A} : A\rrbracket = M\llbracket\Gamma\rrbracket \xrightarrow{\langle\rangle} 1 \xrightarrow{M(c^{A})} M\llbracketA\rrbracket$ Unit value:

 $M[\![\Gamma \vdash (): \texttt{unit}]\!] = M[\![\Gamma]\!] \xrightarrow{()} 1$

Pairing:

$M\llbracket\Gamma \vdash (s, t) : A \times B\rrbracket$ $= M\llbracket\Gamma\rrbracket \xrightarrow{\langle M\llbracket\Gamma \vdash s:A\rrbracket, M\llbracket\Gamma \vdash t:B\rrbracket\rangle} M\llbracketA\rrbracket \times M\llbracketB\rrbracket$

Projections:

$$\begin{split} M[\![\Gamma \vdash \texttt{fst}\,t:A]\!] \\ &= M[\![\Gamma]\!] \xrightarrow{M[\![\Gamma \vdash t:A \times B]\!]} M[\![A]\!] \times M[\![B]\!] \xrightarrow{\pi_1} M[\![A]\!] \end{split}$$

Semantics of STLC terms in a ccc **Pairing**: $M \| \Gamma \vdash (s, t) : A \times B \|$ $= M[\Gamma] \xrightarrow{\langle M[\Gamma \vdash s:A], M[\Gamma \vdash t:B] \rangle} M[A] \times M[B]$ Given that $\Gamma \vdash fst t : A$ holds, there is a unique type B**Projections:** such that $\Gamma \vdash t : A \times B$ already $M[\Gamma \vdash \texttt{fst} t : A]$ holds. $= M[[\Gamma]] \xrightarrow{M[[\Gamma \vdash t:A \times B]]} M[[A]] \times M[[B]] \xrightarrow{\pi_1} M[[A]]$

Lemma. If $\Gamma \vdash t : A$ and $\Gamma \vdash t : B$ are provable, then A = B.

Pairing:

$M\llbracket\Gamma \vdash (s, t) : A \times B\rrbracket$ $= M\llbracket\Gamma\rrbracket \xrightarrow{\langle M\llbracket\Gamma \vdash s:A\rrbracket, M\llbracket\Gamma \vdash t:B\rrbracket\rangle} M\llbracketA\rrbracket \times M\llbracketB\rrbracket$

Projections:

$$\begin{split} M[\![\Gamma \vdash \operatorname{snd} t : B]\!] &= \\ M[\![\Gamma]\!] \xrightarrow{M[\![\Gamma \vdash t: A \times B]\!]} M[\![A]\!] \times M[\![B]\!] \xrightarrow{\pi_2} M[\![B]\!] \end{split}$$

(As for the case of fst, if $\Gamma \vdash \operatorname{snd} t : B$, then $\Gamma \vdash t : A \times B$ already holds for a unique type A.)

Function abstraction:

$$M\llbracket\Gamma \vdash \lambda x : A.t : A \to B\rrbracket$$

= cur f : M[[\Gamma]] $\to (M\llbracketA\rrbracket \Rightarrow M\llbracketB\rrbracket)$

where

 $f = M[\![\Gamma, x : A \vdash t : B]\!] : M[\![\Gamma]\!] \times M[\![A]\!] \to M[\![B]\!]$

Semantics of STLC terms in a ccc

Function application:

$$M\llbracket\Gamma \vdash s \ t : B\rrbracket$$
$$= M\llbracket\Gamma\rrbracket \xrightarrow{\langle f,g \rangle} (M\llbracketA\rrbracket \Longrightarrow M\llbracketB\rrbracket) \times M\llbracketA\rrbracket \xrightarrow{\text{app}} M\llbracketB\rrbracket$$

where

 $\begin{array}{lll} A &=& \text{unique type such that } \Gamma \vdash s : A \to B \text{ and } \Gamma \vdash t : A \\ & \text{already holds (exists because } \Gamma \vdash s t : B \text{ holds}) \end{array}$ $f &=& M[\![\Gamma \vdash s : A \to B]\!] : M[\![\Gamma]\!] \to (M[\![A]\!] \Longrightarrow M[\![B]\!]) \\ g &=& M[\![\Gamma \vdash t : A]\!] : M[\![\Gamma]\!] \to M[\![A]\!] \end{array}$

Example

Consider $t \triangleq \lambda x : A. g(f x)$ so that $\Gamma \vdash t : A \to C$ for $\Gamma \triangleq \diamond, f: \overline{A \to B, q: B \to C}.$ Suppose M[A] = X, M[B] = Y and M[C] = Z in C. Then $M[\![\Gamma]\!] = (1 \times Y^X) \times Z^Y$ $M[\Gamma, x : A] = ((1 \times Y^X) \times Z^Y) \times X$ $M[\Gamma, x : A \vdash x : A] = \pi_2$ $M\llbracket\Gamma, x : A \vdash q : B \to C\rrbracket = \pi_2 \circ \pi_1$ $M\llbracket\Gamma, x : A \vdash f : A \to B\rrbracket = \pi_2 \circ \pi_1 \circ \pi_1$ $M[\Gamma, x : A \vdash f x : B] = \operatorname{app} \circ \langle \pi_2 \circ \pi_1 \circ \pi_1, \pi_2 \rangle$ $M\llbracket\Gamma, x : A \vdash q(f x) : C\rrbracket = \operatorname{app} \circ \langle \pi_2 \circ \pi_1, \operatorname{app} \circ \langle \pi_2 \circ \pi_1 \circ \pi_1, \pi_2 \rangle \rangle$ $M\llbracket\Gamma \vdash t : A \to C\rrbracket = \operatorname{cur}(\operatorname{app} \circ \langle \pi_2 \circ \pi_1, \operatorname{app} \circ \langle \pi_2 \circ \pi_1 \circ \pi_1, \pi_2 \rangle \rangle)$

STLC equations

take the form $\Gamma \vdash s = t : A$ where $\Gamma \vdash s : A$ and $\Gamma \vdash t : A$ are provable.

Such an equation is satisfied by the semantics in a ccc if $M[\Gamma \vdash s : A]$ and $M[\Gamma \vdash t : A]$ are equal C-morphisms $M[\Gamma] \rightarrow M[A]$.

Qu: which equations are always satisfied in any ccc?

STLC equations

take the form $\Gamma \vdash s = t : A$ where $\Gamma \vdash s : A$ and $\Gamma \vdash t : A$ are provable.

Such an equation is satisfied by the semantics in a ccc if $M[\Gamma \vdash s : A]$ and $M[\Gamma \vdash t : A]$ are equal C-morphisms $M[\Gamma] \rightarrow M[A]$.

Qu: which equations are always satisfied in any ccc? Ans: $(\alpha)\beta\eta$ -equivalence — to define this, first have to define alpha-equivalence, substitution and its semantics.

The names of λ -bound variables should not affect meaning.

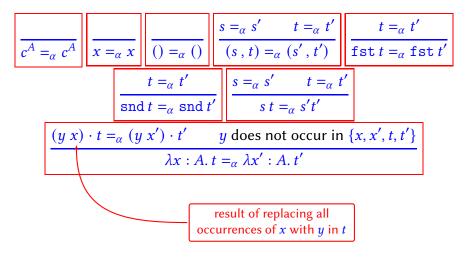
E.g. $\lambda f : A \to B$. $\lambda x : A$. f x should have the same meaning as $\lambda x : A \to B$. $\lambda f : A$. x f.

The names of λ -bound variables should not affect meaning.

E.g. $\lambda f : A \to B$. $\lambda x : A$. f x should have the same meaning as $\lambda x : A \to B$. $\lambda f : A$. x f.

This issue is best dealt with at the level of syntax rather than semantics: from now on we re-define "STLC term" to mean not an abstract syntax tree (generated as described before), but rather an equivalence class of such trees with respect to alpha-equivalence $s =_{\alpha} t$, defined as follows ...

(Alternatively, one can use a "nameless" (de Bruijn) representation of terms.)



$$\frac{1}{c^{A} =_{\alpha} c^{A}} \begin{bmatrix} \frac{1}{x =_{\alpha} x} & \frac{1}{() =_{\alpha} ()} & \frac{s =_{\alpha} s' \quad t =_{\alpha} t'}{(s, t) =_{\alpha} (s', t')} & \frac{t =_{\alpha} t'}{fst t =_{\alpha} fst t'} \\
\frac{1}{snd t =_{\alpha} snd t'} & \frac{s =_{\alpha} s' \quad t =_{\alpha} t'}{s t =_{\alpha} s't'} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not occur in } \{x, x', t, t'\} \\
\frac{1}{st =_{\alpha} (y x') \cdot t'} & y \text{ does not }$$

E.g.

$$\begin{split} \lambda x &: A. x \, x =_{\alpha} \lambda y : A. y \, y \neq_{\alpha} \lambda x : A. x \, y \\ (\lambda y : A. y) \, x =_{\alpha} (\lambda x : A. x) \, x \neq_{\alpha} (\lambda x : A. x) \, y \end{split}$$

Substitution

t[s/x]

= result of replacing all free occurrences of variable xin term t (i.e. those not occurring within the scope of a $\lambda x : A_{-}$ binder) by the term s, alpha-converting λ -bound variables in t to avoid them "capturing" any free variables of t.

E.g. $(\lambda y : A. (y, x))[y/x]$ is $\lambda z : A. (z, y)$ and is not $\lambda y : A. (y, y)$

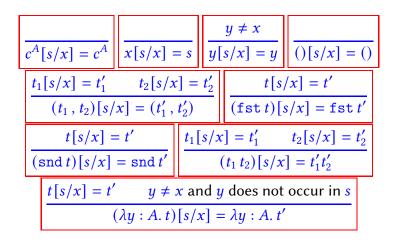
Substitution

t[s/x]

- = result of replacing all free occurrences of variable xin term t (i.e. those not occurring within the scope of a $\lambda x : A_{-}$ binder) by the term s, alpha-converting λ -bound variables in t to avoid them "capturing" any free variables of t.
- E.g. $(\lambda y : A. (y, x))[y/x]$ is $\lambda z : A. (z, y)$ and is not $\lambda y : A. (y, y)$

The relation t[s/x] = t' can be inductively defined by the following rules ...

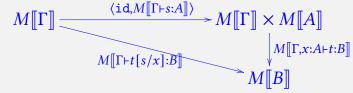
Substitution



Semantics of substitution in a ccc

Substitution Lemma If $\Gamma \vdash s : A$ and $\Gamma, x : A \vdash t : B$ are provable, then so is $\Gamma \vdash t[s/x] : B$.

Substitution Theorem If $\Gamma \vdash s : A$ and $\Gamma, x : A \vdash t : B$ are provable, then in any ccc the following diagram commutes:



STLC equations

take the form $\Gamma \vdash s = t : A$ where $\Gamma \vdash s : A$ and $\Gamma \vdash t : A$ are provable.

Such an equation is satisfied by the semantics in a ccc if $M[\Gamma \vdash s : A]$ and $M[\Gamma \vdash t : A]$ are equal C-morphisms $M[\Gamma] \rightarrow M[A]$.

Qu: which equations are always satisfied in any ccc? Ans: $\beta\eta$ -equivalence...

The relation $\Gamma \vdash s =_{\beta\eta} t : A$ (where Γ ranges over typing environments, *s* and *t* over terms, and *A* over types) is inductively defined by the following rules:

The relation $\Gamma \vdash s =_{\beta\eta} t : A$ (where Γ ranges over typing environments, *s* and *t* over terms, and *A* over types) is inductively defined by the following rules:

• β -conversions

 $\frac{\Gamma, x : A \vdash t : B \qquad \Gamma \vdash s : A}{\Gamma \vdash (\lambda x : A, t)s =_{\beta\eta} t[s/x] : B} \qquad \frac{\Gamma \vdash s : A \qquad \Gamma \vdash t : B}{\Gamma \vdash fst(s, t) =_{\beta\eta} s : A}$ $\frac{\Gamma \vdash s : A \qquad \Gamma \vdash t : B}{\Gamma \vdash snd(s, t) =_{\beta\eta} t : B}$

The relation $\Gamma \vdash s =_{\beta\eta} t : A$ (where Γ ranges over typing environments, *s* and *t* over terms, and *A* over types) is inductively defined by the following rules:

- β -conversions
- \blacktriangleright η -conversions

$$\begin{array}{c|c} \hline \Gamma \vdash t : A \to B & x \text{ does not occur in } t \\ \hline \Gamma \vdash t =_{\beta\eta} (\lambda x : A. t x) : A \to B \\ \hline \hline \Gamma \vdash t : A \times B & \\ \hline \Gamma \vdash t =_{\beta\eta} (\texttt{fst } t, \texttt{snd } t) : A \times B & \\ \hline \Gamma \vdash t =_{\beta\eta} () : \texttt{unit} \end{array}$$

The relation $\Gamma \vdash s =_{\beta\eta} t : A$ (where Γ ranges over typing environments, *s* and *t* over terms, and *A* over types) is inductively defined by the following rules:

- β -conversions
- η -conversions
- congruence rules

$$\frac{\Gamma, x : A \vdash t =_{\beta\eta} t' : B}{\Gamma \vdash \lambda x : A. t =_{\beta\eta} \lambda x : A. t' : A \to B}$$

$$\frac{\Gamma \vdash s =_{\beta\eta} s' : A \to B \qquad \Gamma \vdash t =_{\beta\eta} t' : A}{\Gamma \vdash s t =_{\beta\eta} s' t' : B} \text{ etc}$$

The relation $\Gamma \vdash s =_{\beta\eta} t : A$ (where Γ ranges over typing environments, *s* and *t* over terms, and *A* over types) is inductively defined by the following rules:

- β -conversions
- \blacktriangleright η -conversions
- congruence rules

 $=_{\beta\eta} \text{ is reflexive, symmetric and transitive}$ $\frac{\Gamma + t : A}{\Gamma + t =_{\beta\eta} t : A} \frac{\Gamma + s =_{\beta\eta} t : A}{\Gamma + t =_{\beta\eta} s : A}$ $\frac{\Gamma + r =_{\beta\eta} s : A \qquad \Gamma + s =_{\beta\eta} t : A}{\Gamma + r =_{\beta\eta} t : A}$

Soundness Theorem for semantics of STLC in a ccc. If $\Gamma \vdash s =_{\beta\eta} t : A$ is provable, then in any ccc

 $M\llbracket\Gamma \vdash s:A\rrbracket = M\llbracket\Gamma \vdash t:A\rrbracket$

are equal C-morphisms $M\llbracket \Gamma \rrbracket \to M\llbracket A \rrbracket$.

Proof is by induction on the structure of the proof of $\Gamma \vdash s =_{\beta\eta} t : A$. Here we just check the case of β -conversion for functions.

So suppose we have Γ , $x : A \vdash t : B$ and $\Gamma \vdash s : A$. We have to see that

 $M\llbracket\Gamma \vdash (\lambda x : A, t) s : B\rrbracket = M\llbracket\Gamma \vdash t[s/x] : B\rrbracket$

Suppose $M[[\Gamma]] = X$ M[[A]] = Y M[[B]] = Z $M[[\Gamma, x : A \vdash t : B]] = f : X \times Y \rightarrow Z$ $M[[\Gamma \vdash s : A]] = g : X \rightarrow Z$

Then

$$M[\![\Gamma \vdash \lambda x : A. t : A \to B]\!] = \operatorname{cur} f : X \to Z^Y$$

and hence

$$M[[\Gamma \vdash (\lambda x : A, t) s : B]]$$

= app $\circ \langle \operatorname{cur} f, g \rangle$
= app $\circ (\operatorname{cur} f \times \operatorname{id}_Y) \circ \langle \operatorname{id}_X, g \rangle$
= $f \circ \langle \operatorname{id}_X, g \rangle$
= $M[[\Gamma \vdash t[s/x]] : B]]$

since $(a \times b) \circ \langle c, d \rangle = \langle a \circ c, b \circ d \rangle$ by definition of cur fby the <u>Substitution Theorem</u>

as required.

The internal language of a ccc, C

- one ground type for each C-object *X*
- For each X ∈ C, one constant f^X for each
 C-morphism f : 1 → X ("global element" of the object X)

The types and terms of STLC over this language usefully describe constructions on the objects and morphisms of C using its cartesian closed structure, but in an "element-theoretic" way.

For example, ...

Example

In any ccc C, for any $X, Y, Z \in C$ there is an isomorphism $Z^{(X \times Y)} \cong (Z^Y)^X$

Example

In any ccc C, for any $X, Y, Z \in C$ there is an isomorphism $Z^{(X \times Y)} \cong (Z^Y)^X$

which in the internal language of C is described by the terms

 $\diamond \vdash s : ((X \times Y) \to Z) \to (X \to (Y \to Z))$ $\diamond \vdash t : (X \to (Y \to Z)) \to ((X \times Y) \to Z)$

where
$$\begin{cases} s &\triangleq \lambda f : (X \times Y) \to Z. \ \lambda x : X. \ \lambda y : Y. \ f(x, y) \\ t &\triangleq \lambda g : X \to (Y \to Z). \ \lambda z : X \times Y. \ g \ (\texttt{fst } z) \ (\texttt{snd } z) \end{cases} \text{ and}$$

which satisfy
$$\begin{cases} \diamond, f : (X \times Y) \to Z \vdash t(s \ f) =_{\beta\eta} f \\ \diamond, g : X \to (Y \to Z) \vdash s(t \ g) =_{\beta\eta} g \end{cases}$$

Free cartesian closed categories

The Soundness Theorem has a converse-completeness.

In fact for a given set of ground types and typed constants there is a single ccc **F** (the free ccc for that language) with an interpretation function *M* so that $\Gamma \vdash s =_{\beta_n} t : A$ is provable iff $M[[\Gamma \vdash s : A]] = M[[\Gamma \vdash t : A]]$ in **F**.

Free cartesian closed categories

The Soundness Theorem has a converse-completeness.

In fact for a given set of ground types and typed constants there is a single ccc **F** (the free ccc for that language) with an interpretation function *M* so that $\Gamma \vdash s = \beta_{\eta} t : A$ is provable iff $M[\Gamma \vdash s : A] = M[\Gamma \vdash t : A]$ in **F**.

- F-objects are the STLC types over the given set of ground types
- ► **F**-morphisms $A \to B$ are equivalence classes of STLC terms *t* satisfying $\diamond \vdash t : A \to B$ (so *t* is a *closed* term—it has no free variables) with respect to the equivalence relation equating *s* and *t* if $\diamond \vdash s =_{\beta\eta} t : A \to B$ is provable.
- identity morphism on A is the equivalence class of $\diamond \vdash \lambda x : A \cdot x : A \rightarrow A$.
- ► composition of a morphism $A \to B$ represented by $\diamond \vdash s : A \to B$ and a morphism $B \to C$ represented by $\diamond \vdash t : B \to C$ is represented by $\diamond \vdash \lambda x : A \cdot t(s x) : A \to C$.

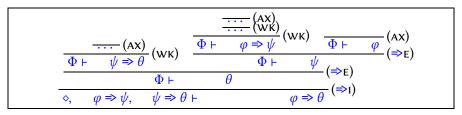
Curry-Howard correspondence

		Туре
Logic		Theory
propositions	\leftrightarrow	types
proofs	\leftrightarrow	terms

E.g. IPL versus STLC.

Curry-Howard for IPL vs STLC

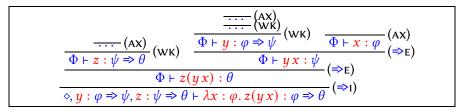
Proof of $\diamond, \varphi \Rightarrow \psi, \psi \Rightarrow \theta \vdash \varphi \Rightarrow \theta$ in IPL



where $\Phi = \diamond, \quad \varphi \Rightarrow \psi, \quad \psi \Rightarrow \theta, \quad \varphi$

Curry-Howard for IPL vs STLC

and a corresponding STLC term



where $\Phi = \diamond, y : \varphi \Rightarrow \psi, z : \psi \Rightarrow \theta, x : \varphi$

Curry-Howard-Lawvere/Lambek correspondence

		Туре		Category
Logic		Theory		Theory
propositions	\leftrightarrow	types	\leftrightarrow	objects
proofs	\leftrightarrow	terms	\leftrightarrow	morphisms

E.g. IPL versus STLC versus CCCs

Curry-Howard-Lawvere/Lambek correspondence

	Туре		Category	
Logic		Theory		Theory
propositions	\leftrightarrow	types	\leftrightarrow	objects
proofs	\leftrightarrow	terms	\leftrightarrow	morphisms

E.g. IPL versus STLC versus CCCs

These correspondences can be made into category-theoretic equivalences—we first need to define the notions of functor and natural transformation in order to define the notion of equivalence of categories.

Functors

morphisms of categories

Given categories C and D, a functor $F : C \rightarrow D$ is specified by:

- a function obj $\mathbf{C} \to \text{obj } \mathbf{D}$ whose value at *X* is written FX
- ► for all $X, Y \in \mathbf{C}$, a function $\mathbf{C}(X, Y) \to \mathbf{D}(FX, FY)$ whose value at $f : X \to Y$ is written $Ff : FX \to FY$

and which is required to preserve composition and identity morphisms:

 $\begin{array}{rcl} F(g \circ f) &=& Fg \circ Ff \\ F(\operatorname{id}_X) &=& \operatorname{id}_{FX} \end{array}$

"Forgetful" functors from categories of set-with-structure back to Set.

E.g. $U : Mon \rightarrow Set$

$$\begin{cases} U(M, \bullet, \iota) &= M \\ U((M_1, \bullet_1, \iota_1) \xrightarrow{f} (M_2, \bullet_2, \iota_2)) &= M_1 \xrightarrow{f} M_2 \end{cases}$$

"Forgetful" functors from categories of set-with-structure back to Set.

E.g. $U : Mon \rightarrow Set$

$$\begin{cases} U(M, \bullet, \iota) &= M \\ U((M_1, \bullet_1, \iota_1) \xrightarrow{f} (M_2, \bullet_2, \iota_2)) &= M_1 \xrightarrow{f} M_2 \end{cases}$$

Similarly U : **Preord** \rightarrow **Set**.

Free monoid functor $F: \mathbf{Set} \to \mathbf{Mon}$

Given $A \in$ Set,

FA = (List A, @, nil), the free monoid on A

Free monoid functor $F : \text{Set} \rightarrow \text{Mon}$ Given $A \in \text{Set}$,

FA = (List A, @, nil), the free monoid on A

Given a function $f : A \rightarrow B$, we get a function $F f : \text{List} A \rightarrow \text{List} B$ by mapping f over finite lists:

$$F f [a_1,\ldots,a_n] = [f a_1,\ldots,f a_n]$$

This gives a monoid morphism $FA \rightarrow FB$; and mapping over lists preserves composition ($F(g \circ f) = Fg \circ Ff$) and identities ($Fid_A = id_{FA}$). So we do get a functor from Set to Mon.

Examples of functors

If **C** is a category with binary products and $X \in \mathbf{C}$, then the function (_) × X : obj **C** \rightarrow obj **C** extends to a functor (_) × X : **C** \rightarrow **C** mapping morphisms $f : Y \rightarrow Y'$ to

$f \times \mathrm{id}_X : Y \times X \to Y' \times X$

 $\left(\text{recall that } f \times g \text{ is the unique morphism with } \begin{cases} \pi_1 \circ (f \times g) &= f \circ \pi_1 \\ \pi_2 \circ (f \times g) &= g \circ \pi_2 \end{cases} \right)$

since it is the case that

$$\begin{cases} \operatorname{id}_X \times \operatorname{id}_Y &= \operatorname{id}_{X \times Y} \\ (f' \circ f) \times \operatorname{id}_X &= (f' \times \operatorname{id}_X) \circ (f \times \operatorname{id}_X) \end{cases}$$

Examples of functors

If **C** is a cartesian closed category and $X \in \mathbf{C}$, then the function $(_)^X : \operatorname{obj} \mathbf{C} \to \operatorname{obj} \mathbf{C}$ extends to a functor $(_)^X : \mathbf{C} \to \mathbf{C}$ mapping morphisms $f : Y \to Y'$ to

$$f^X \triangleq \operatorname{cur}(f \circ \operatorname{app}) : Y^X \to Y'^X$$

since it is the case that

$$\begin{cases} (\operatorname{id}_Y)^X &= \operatorname{id}_{Y^X} \\ (g \circ f)^X &= g^X \circ f^X \end{cases}$$

Contravariance

Given categories C and D, a functor $F : C^{op} \rightarrow D$ is called a contravariant functor from C to D.

Note that if $X \xrightarrow{f} Y \xrightarrow{g} Z$ in C, then $X \xleftarrow{f} Y \xleftarrow{g} Z$ in C^{op}

so $FX \xleftarrow{Ff} FY \xleftarrow{Fg} FZ$ in **D** and hence

$$F(g \circ_{\mathbf{C}} f) = Ff \circ_{\mathbf{D}} Fg$$

(contravariant functors reverse the order of composition)

A functor $\mathbf{C} \rightarrow \mathbf{D}$ is sometimes called a covariant functor from C to D.

Example of a contravariant functor

If **C** is a cartesian closed category and $X \in \mathbf{C}$, then the function $X^{(-)} : \operatorname{obj} \mathbf{C} \to \operatorname{obj} \mathbf{C}$ extends to a functor $X^{(-)} : \mathbf{C}^{\operatorname{op}} \to \mathbf{C}$ mapping morphisms $f : Y \to Y'$ to

$$X^f \triangleq \operatorname{cur}(\operatorname{app} \circ (\operatorname{id}_{X^{Y'}} \times f)) : X^{Y'} \to X^Y$$

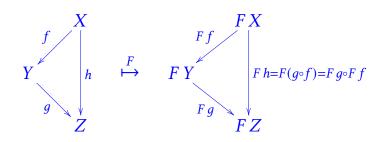
since it is the case that

$$\begin{cases} X^{\operatorname{id}_Y} &= \operatorname{id}_{X^Y} \\ X^{g \circ f} &= X^f \circ X^g \end{cases}$$

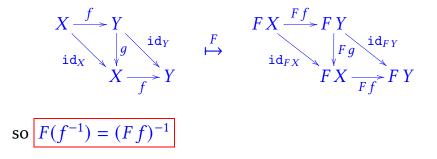
Note that since a functor $F : \mathbb{C} \to \mathbb{D}$ preserves domains, codomains, composition, and identity morphisms

it sends commutative diagrams in **C** to commutative diagrams in **D**

E.g.



Note that since a functor $F : \mathbb{C} \to \mathbb{D}$ preserves domains, codomains, composition, and identity morphisms it sends isomorphisms in \mathbb{C} to isomorphisms in \mathbb{D} , because



Composing functors

Given functors $F : \mathbb{C} \to \mathbb{D}$ and $G : \mathbb{D} \to \mathbb{E}$, we get a functor $G \circ F : \mathbb{C} \to \mathbb{E}$ with

$$G \circ F\begin{pmatrix} X\\ \downarrow f\\ Y \end{pmatrix} = \begin{array}{c} G(FX)\\ \downarrow G(Ff)\\ G(FY) \end{array}$$

(this preserves composition and identity morphisms, because F and G do)

Identity functor

on a category C is $\ensuremath{\operatorname{id}}_C:C\to C$ where

$$\operatorname{id}_{\mathbf{C}}\begin{pmatrix}X\\ \begin{array}{c} \\ \\ \\ Y \end{pmatrix} = \begin{array}{c} \\ \\ \\ \\ Y \end{pmatrix} = \begin{array}{c} \\ \\ \\ \\ Y \end{array}$$

Functor composition and identity functors satisfy

associativity $H \circ (G \circ F) = (H \circ G) \circ F$ unity $\mathrm{id}_{\mathbf{D}} \circ F = F = F \circ \mathrm{id}_{\mathbf{C}}$

So we can get categories whose objects are categories and whose morphisms are functors

but we have to be a bit careful about size...

Size

One of the axioms of set theory is

set membership is a well-founded relation, that is, there is no infinite sequence of sets X_0, X_1, X_2, \ldots with

 $\cdots \in X_{n+1} \in X_n \in \cdots \in X_2 \in X_1 \in X_0$

So in particular there is no set *X* with $X \in X$.

So we cannot form the "set of all sets" or the "category of all categories".

Size

One of the axioms of set theory is

set membership is a well-founded relation, that is, there is no infinite sequence of sets X_0, X_1, X_2, \ldots with

 $\cdots \in X_{n+1} \in X_n \in \cdots \in X_2 \in X_1 \in X_0$

So in particular there is no set *X* with $X \in X$.

So we cannot form the "set of all sets" or the "category of all categories".

But we do assume there are (lots of) big sets

 $\mathcal{U}_0 \in \mathcal{U}_1 \in \mathcal{U}_2 \in \cdots$

where "big" means each \mathcal{U}_n is a Grothendieck universe...

Grothendieck universes

A Grothendieck universe \mathcal{U} is a set of sets satisfying

- $\blacktriangleright X \in Y \in \mathcal{U} \Longrightarrow X \in \mathcal{U}$
- $\blacktriangleright X, Y \in \mathcal{U} \Longrightarrow \{X, Y\} \in \mathcal{U}$
- $\blacktriangleright X \in \mathscr{U} \Longrightarrow \mathscr{P}X \triangleq \{Y \mid Y \subseteq X\} \in \mathscr{U}$
- $I \in \mathcal{U} \land F \in \mathcal{U}^I \Rightarrow$ $\{x \mid \exists i \in I, \ x \in F i\} \in \mathcal{U}$

The above properties are satisfied by $\mathcal{U} = \emptyset$, but we will always assume

▶ $\mathbb{N} \in \mathcal{U}$

Size

We assume

there is an infinite sequence $\mathcal{U}_0 \in \mathcal{U}_1 \in \mathcal{U}_2 \in \cdots$ of bigger and bigger Grothendieck universes

and revise the previous definition of "the" category of sets and functions:

Set_n = category whose objects are all the sets in \mathcal{U}_n and with Set_n(X, Y) = Y^X = all functions from X to Y.

Notation: $Set \triangleq Set_0$ — its objects are called small sets (and other sets we call large).

Size

Set is the category of small sets.

Definition. A category C is locally small if for all $X, Y \in C$, the set of C-morphisms $X \rightarrow Y$ is small; that is, $C(X, Y) \in$ Set.

C is a small category if it is both locally small and obj $C \in Set$.

E.g. Set, Preord, and Mon are all locally small (but not small).

Given $\underline{P} \in \mathbf{Preord}$, the category $C_{\underline{P}}$ it determines is small; similarly, the category C_M determined by $\underline{M} \in \mathbf{Mon}$ is small.

The category of small categories, Cat

- objects are all small categories
- morphisms in Cat(C, D) are all functors $C \rightarrow D$
- composition and identity morphisms as for functors

Cat is a locally small category

Problem: Is Cat a bicartesian closed category?

Cat has an initial object

The empty category (with no objects and no morphisms) is initial in Cat.

Cat has binary coproducts

Given small categories $C, D \in Cat$, their coproduct $C \xrightarrow{\iota_1} C + D \xleftarrow{\iota_2} D$ is:

Cat has binary coproducts

Given small categories $C, D \in Cat$, their coproduct $C \xrightarrow{\iota_1} C + D \xleftarrow{\iota_2} D$ is:

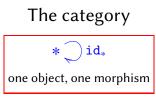
- objects: $obj(C + D) \triangleq obj(C) + obj(D)$
- ► morphisms:

 $(\mathbf{C} + \mathbf{D}) (\iota_1(C), \iota_2(C')) \triangleq \mathbf{C}(C, C')$ $(\mathbf{C} + \mathbf{D}) (\iota_2(D), \iota_2(D')) \triangleq \mathbf{D}(D, D')$ $(\mathbf{C} + \mathbf{D}) (\iota_1(C), \iota_2(D)) \triangleq \emptyset$ $(\mathbf{C} + \mathbf{D}) (\iota_2(D), \iota_1(C)) \triangleq \emptyset$

 composition and identity morphisms are given by those of C (between objects tagged by *i*₁) or D (between objects tagged by *i*₂)

$\begin{cases} \iota_1(C \xrightarrow{f} C') \triangleq \iota_1(C) \xrightarrow{\iota_1(f)} \iota_1(C') \\ \iota_2(D \xrightarrow{g} D') \triangleq \iota_2(D) \xrightarrow{\iota_2(g)} \iota_1(D') \end{cases}$

Cat has a terminal object



is terminal in Cat

Cat has binary products

Given small categories $C, D \in Cat$, their product $C \xleftarrow{\pi_1} C \times D \xrightarrow{\pi_2} D$ is:

Cat has binary products

Given small categories $C, D \in Cat$, their product $C \xleftarrow{\pi_1} C \times D \xrightarrow{\pi_2} D$ is:

• objects: $obj(C \times D) \triangleq obj(C) \times obj(D)$

► morphisms:

 $(\mathbf{C} \times \mathbf{D})((C, D), (C', D')) \triangleq \mathbf{C}(C, C') \times \mathbf{D}(D, D')$

 composition and identity morphisms are given by those of C (in the first component) and D (in the second component)

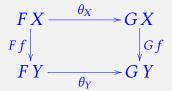
$$\begin{cases} \pi_1\left((C,D) \xrightarrow{(f,g)} (C',D')\right) = C \xrightarrow{f} C' \\ \pi_2\left((C,D) \xrightarrow{(f,g)} (C',D')\right) = D \xrightarrow{g} D' \end{cases}$$

Cat has exponentials

Exponentials in Cat are called functor categories. To define them we need to consider natural transformations, which are the appropriate notion of morphism between functors.

Natural transformations

Definition. Given categories and functors $F, G : \mathbb{C} \to \mathbb{D}$, a natural transformation $\theta : F \to G$ is a family of D-morphisms $\theta_X \in \mathbb{D}(FX, GX)$, one for each $X \in \mathbb{C}$, such that for all \mathbb{C} -morphisms $f : X \to Y$, the diagram



commutes in **D**, that is, $\theta_Y \circ F f = G f \circ \theta_X$.

Composing natural transformations

Given functors $F, G, H : \mathbb{C} \to \mathbb{D}$ and natural transformations $\theta : F \to G$ and $\varphi : G \to H$,

we get $\varphi \circ \theta : F \to H$ with

$$(\varphi \circ \theta)_X = \left(FX \xrightarrow{\theta_X} GX \xrightarrow{\varphi_X} HX \right)$$

Composing natural transformations

Given functors $F, G, H : \mathbb{C} \to \mathbb{D}$ and natural transformations $\theta : F \to G$ and $\varphi : G \to H$,

we get $\varphi \circ \theta$: $F \to H$ with

$$(\varphi \circ \theta)_X = \left(F X \xrightarrow{\theta_X} G X \xrightarrow{\varphi_X} H X \right)$$

Check naturality:

$$\begin{split} Hf \circ (\varphi \circ \theta)_X &\triangleq Hf \circ \varphi_X \circ \theta_X \\ &= \varphi_Y \circ Gf \circ \theta_X \\ &= \varphi_Y \circ \theta_Y \circ Ff \\ &\triangleq (\varphi \circ \theta)_Y \circ Ff \end{split} \qquad \text{naturality of } \theta \end{split}$$

Identity natural transformation

Given a functor $F : \mathbb{C} \to \mathbb{D}$, we get a natural transformation $id_F : F \to F$ with

$$(\operatorname{id}_F)_X = F X \xrightarrow{\operatorname{id}_{FX}} F X$$

Identity natural transformation

Given a functor $F : \mathbb{C} \to \mathbb{D}$, we get a natural transformation $id_F : F \to F$ with

$$(\mathrm{id}_F)_X = F X \xrightarrow{\mathrm{id}_{FX}} F X$$

Check naturality:

$$Ff \circ (id_F)_X \triangleq Ff \circ id_{FX} = Ff = id_{FY} \circ Ff \triangleq (id_F)_Y \circ Ff$$

Functor categories

It is easy to see that composition and identities for natural transformations satisfy

 $(\psi \circ \varphi) \circ \theta = \psi \circ (\varphi \circ \theta)$ $\operatorname{id}_G \circ \theta = \theta \circ \operatorname{id}_F$

so that we get a category:

Definition. Given categories C and D, the functor category D^{C} has

- objects are all functors $C \rightarrow D$
- given $F, G : \mathbb{C} \to \mathbb{D}$, morphism from F to G in $\mathbb{D}^{\mathbb{C}}$ are the natural transformations $F \to G$

composition and identity morphisms as above

If \mathcal{U} is a Grothendieck universe, then for each $I \in \mathcal{U}$ and $F \in \mathcal{U}^I$ we have that their dependent product and dependent function sets

$$\sum_{i \in I} F i \triangleq \{(i, x) \mid i \in I \land x \in F i\}$$
$$\prod_{i \in I} F i \triangleq \{f \subseteq \sum_{i \in I} F i \mid f \text{ is single-valued and total}\}$$

are also in \mathcal{U} ; and, as a special case (of \prod , when *F* is a constant function with value *X*) we also have that $I, X \in \mathcal{U}$ implies $X^I \in \mathcal{U}$.

If \mathcal{U} is a Grothendieck universe, then for each $I \in \mathcal{U}$ and $F \in \mathcal{U}^I$ we have that their dependent product and dependent function sets

$$\sum_{i \in I} F i \triangleq \{(i, x) \mid i \in I \land x \in F i\}$$
$$\prod_{i \in I} F i \triangleq \{f \subseteq \sum_{i \in I} F i \mid f \text{ is single-valued and total}\}$$

are also in \mathcal{U} ; and, as a special case (of \prod , when *F* is a constant function with value *X*) we also have that $I, X \in \mathcal{U}$ implies $X^I \in \mathcal{U}$. Hence

If C and D are small categories, then so is D^C.

because

 $\begin{aligned} \mathsf{obj}(\mathbf{D}^{\mathbf{C}}) &\subseteq \sum_{F \in (\mathsf{obj}\,D)^{\mathsf{obj}\,\mathsf{C}}} \prod_{X,Y \in \mathsf{obj}\,\mathsf{C}} \mathbf{D}(F\,X,F\,Y)^{\mathbf{C}(X,Y)} \\ \mathbf{D}^{\mathbf{C}}(F,G) &\subseteq \prod_{X \in \mathsf{obj}\,\mathsf{C}} \mathbf{D}(F\,X,G\,X) \end{aligned}$

If \mathcal{U} is a Grothendieck universe, then for each $I \in \mathcal{U}$ and $F \in \mathcal{U}^I$ we have that their dependent product and dependent function sets

$$\sum_{i \in I} F i \triangleq \{(i, x) \mid i \in I \land x \in F i\}$$
$$\prod_{i \in I} F i \triangleq \{f \subseteq \sum_{i \in I} F i \mid f \text{ is single-valued and total}\}$$

are also in \mathcal{U} ; and, as a special case (of \prod , when *F* is a constant function with value *X*) we also have that $I, X \in \mathcal{U}$ implies $X^I \in \mathcal{U}$. Hence

If C and D are small categories, then so is D^C.

because

 $\begin{aligned} \mathsf{obj}(\mathbf{D}^{\mathbf{C}}) &\subseteq \sum_{F \in (\mathsf{obj}\,D)^{\mathsf{obj}\,\mathsf{C}}} \prod_{X,Y \in \mathsf{obj}\,\mathsf{C}} \mathbf{D}(FX,FY)^{\mathbf{C}(X,Y)} \\ \mathbf{D}^{\mathbf{C}}(F,G) &\subseteq \prod_{X \in \mathsf{obj}\,\mathsf{C}} \mathbf{D}(FX,GX) \end{aligned}$

Aim to show that functor category D^C is the exponential of C and D in Cat ...

Theorem. There is an application functor $app: D^C \times C \rightarrow D$ that makes D^C the exponential for C and D in Cat.

Given $(F, X) \in \mathbf{D}^{\mathbf{C}} \times \mathbf{C}$, we define

 $app(F,X) \triangleq FX$

and given $(\theta, f) : (F, X) \to (G, Y)$ in $\mathbf{D}^{\mathbf{C}} \times \mathbf{C}$, we define

$$\operatorname{app}\left((F,X) \xrightarrow{(\theta,f)} (G,Y)\right) \triangleq F X \xrightarrow{Ff} F Y \xrightarrow{\theta_Y} G Y$$
$$= F X \xrightarrow{\theta_X} G X \xrightarrow{Gf} G Y$$

Check: $\begin{cases} \operatorname{app}(\operatorname{id}_F, \operatorname{id}_X) &= \operatorname{id}_F X \\ \operatorname{app}(\varphi \circ \theta, g \circ f) &= \operatorname{app}(\varphi, g) \circ \operatorname{app}(\theta, f) \end{cases}$

Theorem. There is an application functor $app: D^C \times C \rightarrow D$ that makes D^C the exponential for C and D in Cat.

Definition of currying: given functor $F : \mathbf{E} \times \mathbf{C} \to \mathbf{D}$, we get a functor $\operatorname{cur} F : \mathbf{E} \to \mathbf{D}^{\mathbf{C}}$ as follows. For each $Z \in \mathbf{E}$, $\operatorname{cur} F Z \in \mathbf{D}^{\mathbf{C}}$ is the functor

$$\operatorname{cur} FZ\begin{pmatrix} X\\ & f\\ \\ & f\\ \\ & X' \end{pmatrix} \triangleq \begin{array}{c} F(Z,X)\\ & f\\ & f \\ & f \\ & F(Z,X') \end{array}$$

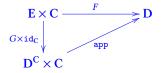
For each $g : Z \to Z'$ in **E**, cur Fg : cur $FZ \to$ cur FZ' is the natural transformation whose component at each $X \in \mathbf{C}$ is

 $(\operatorname{cur} F g)_X \triangleq F(g, \operatorname{id}_X) : F(Z, X) \to F(Z', X)$

(Check that this is natural in X; and that cur F preserves composition and identities in E.)

Theorem. There is an application functor $app: D^C \times C \rightarrow D$ that makes D^C the exponential for C and D in Cat.

Have to check that cur *F* is the unique functor $G : \mathbf{E} \to \mathbf{D}^{\mathbf{C}}$ that makes



commute in Cat (exercise).

Example of natural transformation (I)

Fix a set $S \in$ Set and consider the two functors F, G : Set \rightarrow Set given by

$$F\left(X \xrightarrow{f} Y\right) = S \times X \xrightarrow{\operatorname{id}_S \times f} S \times Y$$
$$G\left(X \xrightarrow{f} Y\right) = X \times S \xrightarrow{f \times \operatorname{id}_S} Y \times S$$

Example of natural transformation (I)

Fix a set $S \in$ **Set** and consider the two functors F, G : **Set** \rightarrow **Set** given by

$$F\left(X \xrightarrow{f} Y\right) = S \times X \xrightarrow{id_S \times f} S \times Y$$
$$G\left(X \xrightarrow{f} Y\right) = X \times S \xrightarrow{f \times id_S} Y \times S$$

For each $X \in$ Set there is an isomorphism (bijection) $\theta_X : FX \cong GX$ in Set given by $\langle \pi_2, \pi_1 \rangle : S \times X \to X \times S$.

These isomorphisms do not depend on the particular nature of each set X (they are "polymorphic in X"). One way to make this precise is ...

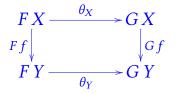
... if we change from *X* to *Y* along a function $f : X \rightarrow Y$, then we get a commutative diagram in **Set**:

The square commutes because for all $s \in S$ and $x \in X$

$$\langle \pi_2, \pi_1 \rangle ((\operatorname{id} \times f)(s, x)) = \langle \pi_2, \pi_1 \rangle (s, f x)$$

= $(f x, s)$
= $(f \times \operatorname{id})(x, s)$
= $(f \times \operatorname{id})(\langle \pi_2, \pi_1 \rangle (s, x))$

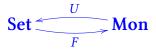
... if we change from *X* to *Y* along a function $f : X \rightarrow Y$, then we get a commutative diagram in **Set**:



We say that the family $(\theta_X \mid X \in \mathbf{Set})$ is natural in *X*.

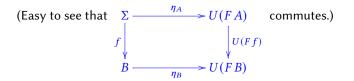
Example of natural transformation (II)

Recall forgetful (U) and free (F) functors:



There is a natural transformation $\eta : id_{Set} \rightarrow U \circ F$, where for each $A \in Set$

 $\eta_A : A \to U(FA) = \text{List}A$ $a \in A \mapsto [a] \in \text{List}A \text{ (one-element list)}$



Example of natural transformation (III)

The covariant powerset functor $\mathscr{P} : \mathbf{Set} \to \mathbf{Set}$ is

$$\mathcal{P}X \triangleq \{S \mid S \subseteq X\}$$
$$\mathcal{P}\left(X \xrightarrow{f} Y\right) \triangleq \mathcal{P}X \xrightarrow{\mathcal{P}f} \mathcal{P}Y$$
$$S \mapsto \mathcal{P}fS \triangleq \{f x \in Y \mid x \in S\}$$

Example of natural transformation (III)

The covariant powerset functor $\mathscr{P} : \mathbf{Set} \to \mathbf{Set}$ is

$$\mathcal{P}X \triangleq \{S \mid S \subseteq X\}$$
$$\mathcal{P}\left(X \xrightarrow{f} Y\right) \triangleq \mathcal{P}X \xrightarrow{\mathcal{P}f} \mathcal{P}Y$$
$$S \mapsto \mathcal{P}fS \triangleq \{f x \in Y \mid x \in S\}$$

There is a natural transformation $\bigcup : \mathscr{P} \circ \mathscr{P} \to \mathscr{P}$ whose component at $X \in \mathbf{Set}$ sends $\mathscr{F} \in \mathscr{P}(\mathscr{P}X)$ to

$$\bigcup_{X} \mathscr{F} \triangleq \{ x \in X \mid \exists S \in \mathscr{F}, \ x \in S \} \in \mathscr{P}X$$

(check that \bigcup_X is natural in X)

Non-example of natural transformation

The classic example of an "un-natural transformation" (the one that caused Eilenberg and MacLane to invent the concept of naturality) is the linear isomorphism between a finite dimensional real vectorspace V and its dual V^* , the vector space of linear functions $V \to \mathbb{R}$.

Both V and V^* have the same finite dimension, so are isomorphic by choosing bases; but there is no choice of basis for each V that makes the family of isomorphisms natural in V.

Adjoint functors

- The concepts of "category", "functor" and "natural transformation" were invented by Eilenberg and MacLane in order to formalise "adjoint situations".
- They appear everywhere in mathematics, logic and (hence) computer science.
- Examples of adjoint situations that we have already seen ...

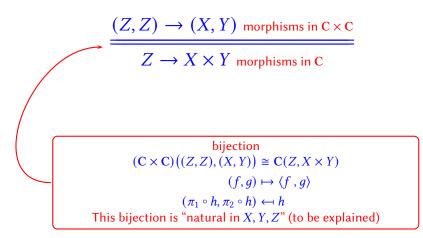
Free monoids

 $\xrightarrow{A \to U(M, \bullet, \iota) \text{ morphisms in Set}} \overline{FA \to (M, \bullet, \iota) \text{ morphisms in Mon}}$

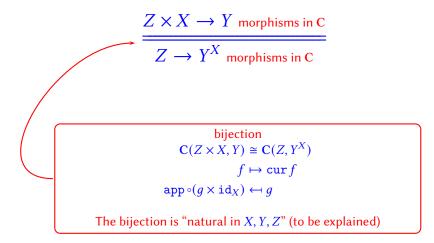
bijection Set(A, U(M, •, i)) \cong Mon(FA, (M, •, i)) $f \mapsto \hat{f}$ $h \circ \eta_A \leftrightarrow h$ (where $\eta_A : A \to UFA = \text{List}A \text{ is } a \mapsto [a]$)

The bijection is "natural in *A* and (M, \bullet, ι) " (to be explained)

Binary product in a category C



Exponentials in a category **C** with binary products



Adjunction

Definition. An adjunction between two categories **C** and **D** is specified by:

• functors
$$C \xrightarrow{f} D$$

► for each $X \in \mathbf{C}$ and $Y \in \mathbf{D}$ a bijection $\theta_{X,Y} : \mathbf{D}(FX, Y) \cong \mathbf{C}(X, GY)$ which is natural in X and Y.

T

for all
$$\begin{cases} u: X' \to X \text{ in } C\\ v: Y \to Y' \text{ in } D \end{cases}$$
 and all $g: FX \to Y \text{ in } D$
 $X' \xrightarrow{u} X \xrightarrow{\theta_{X,Y}(g)} GY \xrightarrow{Gv} GY' = \theta_{X',Y'} \left(FX' \xrightarrow{Fu} FX \xrightarrow{g} Y \xrightarrow{v} Y' \right)$

Adjunction

Definition. An adjunction between two categories **C** and **D** is specified by:

- functors $C \xrightarrow{F} D$
- ► for each $X \in \mathbf{C}$ and $Y \in \mathbf{D}$ a bijection $\theta_{X,Y} : \mathbf{D}(FX, Y) \cong \mathbf{C}(X, GY)$ which is natural in X and Y.

what has this to do with the concept of natural transformation between functors?

Hom functors

If C is a locally small category, then we get a functor

 $\texttt{Hom}_C: C^{op} \times C \to Set$

with $\operatorname{Hom}_{C}(X, Y) \triangleq C(X, Y)$ and

$$\operatorname{Hom}_{\mathbb{C}}\left((X,Y) \xrightarrow{(f,g)} (X',Y')\right) \triangleq \mathbb{C}(X,Y) \xrightarrow{\operatorname{Hom}_{C}(f,g)} \mathbb{C}(X',Y')$$
$$\operatorname{Hom}_{C}(f,g) h \triangleq g \circ h \circ f$$

Hom functors

If C is a locally small category, then we get a functor

 $\texttt{Hom}_C: C^{\texttt{op}} \times C \to Set$

with $\operatorname{Hom}_{C}(X, Y) \triangleq C(X, Y)$ and

$$\operatorname{Hom}_{\mathbb{C}}\left((X,Y) \xrightarrow{(f,g)} (X',Y')\right) \triangleq \mathbb{C}(X,Y) \xrightarrow{\operatorname{Hom}_{\mathbb{C}}(f,g)} \mathbb{C}(X',Y')$$

$$\operatorname{Hom}_{\mathbb{C}}(f,g) h \triangleq g \circ h \circ f$$

$$\operatorname{If}(f,g) : (X,Y) \to (X',Y') \text{ in } \mathbb{C}^{\operatorname{op}} \times \mathbb{C} \text{ and } h : X \to Y \text{ in } \mathbb{C},$$

$$\operatorname{then} \text{ in } \mathbb{C} \text{ we have } f : X' \to X, g : Y \to Y' \text{ and so } g \circ h \circ f : X' \to Y'$$

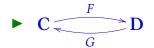
Natural isomorphisms

Given functors $F, G : \mathbb{C} \to \mathbb{D}$, a natural isomorphism $\theta : F \cong G$ is simply an isomorphism between F and G in the functor category $\mathbb{D}^{\mathbb{C}}$.

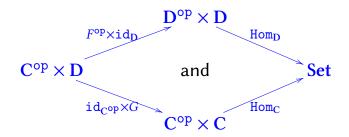
Natural isomorphisms

Given functors $F, G : \mathbb{C} \to \mathbb{D}$, a natural isomorphism $\theta : F \cong G$ is simply an isomorphism between F and G in the functor category $\mathbb{D}^{\mathbb{C}}$.

Lemma. If $\theta : F \to G$ is a natural transformation and for each $X \in \mathbb{C}$, $\theta_X : FX \to GX$ is an isomorphism in \mathbb{D} , then the family of morphisms $(\theta_X^{-1} : GX \to FX \mid X \in \mathbb{C})$ gives a natural transformation $\theta^{-1} : G \to F$ which is inverse to θ in $\mathbb{D}^{\mathbb{C}}$ and hence θ is a natural isomorphism. \Box An adjunction between locally small categories C and D is simply a triple (F, G, θ) where



• θ is a natural isomorphism between the functors



Terminology:

Given C G

if there is some natural isomorphism

 θ : Hom_D \circ ($F^{op} \times id_{D}$) \cong Hom_C \circ ($id_{C^{op}} \times G$)

one says

F is a left adjoint for GG is a right adjoint for F

and writes

$$F \dashv G$$

Notation associated with an adjunction(F, G, θ)

Given
$$\begin{cases} g: FX \to Y \\ f: X \to GY \end{cases}$$

we write
$$\begin{cases} \overline{g} &\triangleq \theta_{X,Y}(g): X \to GY \\ \overline{f} &\triangleq \theta_{X,Y}^{-1}(f): FX \to Y \end{cases}$$

Thus $\overline{\overline{g}} = g, \overline{\overline{f}} = f$ and naturality of $\theta_{X,Y}$ in X and Y means that

 $v \circ g \circ F u = G v \circ \overline{g} \circ u$

Notation associated with an adjunction(F, G, θ)

The existence of θ is sometimes indicated by writing

$$\frac{FX \xrightarrow{g} Y}{X \xrightarrow{\overline{g}} GY}$$

Using this notation, one can split the naturality condition for θ into two:

$$\frac{FX' \xrightarrow{Fu} FX \xrightarrow{g} Y}{X' \xrightarrow{u} X \xrightarrow{\overline{g}} GY} \qquad \frac{FX \xrightarrow{g} Y \xrightarrow{v} Y'}{X \xrightarrow{\overline{g}} GY \xrightarrow{Gv} GY'}$$

Theorem. A category **C** has binary products iff the diagonal functor $\Delta = \langle id_C, id_C \rangle : C \rightarrow C \times C$ has a right adjoint.

Theorem. A category **C** with binary products also has all exponentials of pairs of objects iff for all $X \in \mathbf{C}$, the functor (_) $\times X : \mathbf{C} \to \mathbf{C}$ has a right adjoint.

Common situation: we are given a functor $F : \mathbb{C} \to \mathbb{D}$ and want to know whether it has a right adjoint $G : \mathbb{D} \to \mathbb{C}$ (and dually for left adjoints).

Q: what is the least info we need to specify the existence of a right adjoint?

Theorem. A category **C** has binary products iff the diagonal functor $\Delta = \langle id_C, id_C \rangle : C \rightarrow C \times C$ has a right adjoint.

Theorem. A category **C** with binary products also has all exponentials of pairs of objects iff for all $X \in \mathbf{C}$, the functor (_) $\times X : \mathbf{C} \to \mathbf{C}$ has a right adjoint.

Common situation: we are given a functor $F : \mathbb{C} \to \mathbb{D}$ and want to know whether it has a right adjoint $G : \mathbb{D} \to \mathbb{C}$ (and dually for left adjoints).

Q: what is the least info we need to specify the existence of a right adjoint?

Both the above theorems are instances of the following theorem, which is a very useful characterisation of when a functor has a right adjoint (or dually, a left adjoint).

Theorem. A functor $F : \mathbb{C} \to \mathbb{D}$ has a right adjoint iff for all D-objects $Y \in \mathbb{D}$, there is a C-object $G Y \in \mathbb{C}$ and a D-morphism $\varepsilon_Y : F(G Y) \to Y$ with the following universal property:

Theorem. A functor $F : \mathbb{C} \to \mathbb{D}$ has a right adjoint iff for all D-objects $Y \in \mathbb{D}$, there is a C-object $G Y \in \mathbb{C}$ and a D-morphism $\varepsilon_Y : F(G Y) \to Y$ with the following universal property:

Theorem. A functor $F : \mathbb{C} \to \mathbb{D}$ has a right adjoint iff for all D-objects $Y \in \mathbb{D}$, there is a C-object $G Y \in \mathbb{C}$ and a D-morphism $\varepsilon_Y : F(G Y) \to Y$ with the following universal property:



Theorem. A functor $F : \mathbb{C} \to \mathbb{D}$ has a right adjoint iff for all D-objects $Y \in \mathbb{D}$, there is a C-object $G Y \in \mathbb{C}$ and a D-morphism $\varepsilon_Y : F(G Y) \to Y$ with the following universal property:



Proof of the <u>Theorem</u>—"only if" part:

Given an adjunction (F, G, θ) , for each $Y \in \mathbf{D}$ we produce $\varepsilon_Y : F(GY) \to Y$ in \mathbf{D} satisfying (UP).

Proof of the Theorem—"only if" part:

Given an adjunction (F, G, θ) , for each $Y \in \mathbf{D}$ we produce $\varepsilon_Y : F(GY) \to Y$ in \mathbf{D} satisfying (UP).

We are given $\theta_{X,Y}$: $\mathbf{D}(FX,Y) \cong \mathbf{C}(X,GY)$, natural in X and Y. Define

$$\varepsilon_Y \triangleq \theta_{G\,Y,Y}^{-1}(\operatorname{id}_{G\,Y}): F(G\,Y) \to Y$$

In other words $\varepsilon_Y = \overline{id_G Y}$.

Proof of the <u>**Theorem</u></u>—**"only if" part:</u>

Given an adjunction (F, G, θ) , for each $Y \in \mathbf{D}$ we produce $\varepsilon_Y : F(GY) \to Y$ in \mathbf{D} satisfying (UP).

We are given $\theta_{X,Y}$: $\mathbf{D}(FX,Y) \cong \mathbf{C}(X,GY)$, natural in X and Y. Define

 $\varepsilon_Y \triangleq \theta_{GY,Y}^{-1}(\operatorname{id}_{GY}): F(GY) \to Y$

In other words $\varepsilon_Y = \overline{id_{GY}}$.

Given any
$$\begin{cases} g: FX \to Y & \text{in } \mathbf{D} \\ f: X \to GY & \text{in } \mathbf{C} \end{cases}$$
, by naturality of θ we have
$$\frac{FX \xrightarrow{g} Y}{X \xrightarrow{\overline{g}} GY} \text{ and } \underbrace{\frac{\varepsilon_Y \circ Ff: FX \xrightarrow{Ff} F(GY) \xrightarrow{\overline{\operatorname{id}_{GY}} Y}}{f: X \xrightarrow{f} GY \xrightarrow{\operatorname{id}_{GY}} GY} \end{cases}$$

Hence $g = \varepsilon_Y \circ F \overline{g}$ and $g = \varepsilon_Y \circ F f \implies \overline{g} = f$.

Thus we do indeed have (UP).

Proof of the Theorem—"if" part:

We are given $F : \mathbb{C} \to \mathbb{D}$ and for each $Y \in \mathbb{D}$ a \mathbb{C} -object G Y and \mathbb{C} -morphism $\varepsilon_Y : F(GY) \to Y$ satisfying (UP). We have to

- 1. extend $Y \mapsto G Y$ to a functor $G : \mathbf{D} \to \mathbf{C}$
- 2. construct a natural isomorphism θ : Hom_D \circ ($F^{op} \times id_D$) \cong Hom_C \circ ($id_{C^{op}} \times G$)

Proof of the Theorem—"if" part:

We are given $F : \mathbb{C} \to \mathbb{D}$ and for each $Y \in \mathbb{D}$ a \mathbb{C} -object G Y and \mathbb{C} -morphism $\varepsilon_Y : F(GY) \to Y$ satisfying (UP). We have to

1. extend $Y \mapsto G Y$ to a functor $G : \mathbf{D} \to \mathbf{C}$

For each D-morphism $g: Y' \to Y$ we get $F(GY') \xrightarrow{\varepsilon_{Y'}} Y' \xrightarrow{g} Y$ and can apply (UP) to get

$$Gg \triangleq \overline{g \circ \varepsilon_{Y'}} : GY' \to GY$$

The uniqueness part of (UP) implies

$$G$$
 id = id and $G(g' \circ g) = Gg' \circ Gg$

so that we get a functor $G : \mathbf{D} \to \mathbf{C}$. \Box

Proof of the Theorem-"if" part:

We are given $F : \mathbb{C} \to \mathbb{D}$ and for each $Y \in \mathbb{D}$ a \mathbb{C} -object G Y and \mathbb{C} -morphism $\varepsilon_Y : F(GY) \to Y$ satisfying (UP). We have to

2. construct a natural isomorphism θ : Hom_D \circ ($F^{op} \times id_D$) \cong Hom_C \circ ($id_{C^{op}} \times G$)

Since for all $g : F X \to Y$ there is a unique $f : X \to G Y$ with $g = \varepsilon_Y \circ F f$,

$$f \mapsto \overline{f} \triangleq \varepsilon_Y \circ F f$$

determines a bijection $C(X, GY) \cong C(FX, Y)$; and it is natural in X & Y because

$$G v \circ f \circ u \triangleq \varepsilon_{Y'} \circ F(G v \circ f \circ u)$$

= $(\varepsilon_{Y'} \circ F(G v)) \circ F f \circ F u$ since F is a functor
= $(v \circ \varepsilon_Y) \circ F f \circ F u$ by definition of $G v$
= $v \circ \overline{f} \circ F u$ by definition of \overline{f}

So we can take θ to be the inverse of this natural isomorphism. \Box

Dual of the **Theorem**:

 $G : \mathbb{C} \leftarrow \mathbb{D}$ has a left adjoint iff for all $X \in \mathbb{C}$ there are $FX \in \mathbb{D}$ and $\eta_X \in \mathbb{C}(X, G(FX))$ with the universal property:

for all $Y \in \mathbf{D}$ and $\underline{f} \in \mathbf{C}(X, GY)$ there is a unique $\overline{f} \in \mathbf{D}(FX, Y)$ satisfying $G\overline{f} \circ \eta_X = f$

Dual of the **Theorem**:

 $G : \mathbb{C} \leftarrow \mathbb{D}$ has a left adjoint iff for all $X \in \mathbb{C}$ there are $FX \in \mathbb{D}$ and $\eta_X \in \mathbb{C}(X, G(FX))$ with the universal property:

for all
$$Y \in \mathbf{D}$$
 and $\underline{f} \in \mathbf{C}(X, GY)$
there is a unique $\overline{f} \in \mathbf{D}(FX, Y)$
satisfying $G\overline{f} \circ \eta_X = f$

E.g. we can conclude that the forgetful functor $U : Mon \rightarrow Set$ has a left adjoint $F : Set \rightarrow Mon$, because of the universal property of

 $FA \triangleq (\texttt{List}A, @, \texttt{nil}) \text{ and } \eta_A : A \rightarrow \texttt{List}A$

Why are adjoint functors important/useful?

Their universal property (UP) usually embodies some useful mathematical construction

(e.g. "freely generated structures are left adjoints for forgetting-stucture")

and pins it down uniquely up to isomorphism.

Dependent Types

A brief look at some category theory for modelling type theories with dependent types.

Will restrict attention to the case of Set, rather than in full generality.

Simple types

$$\diamond, x_1: T_1, \ldots, x_n: T_n \vdash t(x_1, \ldots, x_n): T$$

Dependent types

 $\diamond, x_1: T_1, \ldots, x_n: T_n \vdash t(x_1, \ldots, x_n): T(x_1, \ldots, x_n)$

and more generally

 \diamond , $x_1 : T_1$, $x_2 : T_2(x_1)$, $x_3 : T_3(x_1, x_2)$, ... ⊢ $t(x_1, x_2, x_3, ...) : T(x_1, x_2, x_3, ...)$ If type expressions denote sets, then

a type $T_1(x)$ dependent upon x : T

should denote

an indexed family of sets $(E i | i \in I)$ (where *I* is the set denoted by type *T*)

i.e. $E : I \rightarrow Set$ is a set-valued function on a set *I*.

For each $I \in$ **Set**, let **Set**^I be the category with

- $obj(Set^{I}) \triangleq (obj Set)^{I}$, so objects are *I*-indexed families of sets, $X = (X_i \mid i \in I)$
- morphisms $f : X \to Y$ in Set^{*I*} are *I*-indexed families of functions $f = (f_i \in \text{Set}(X_i, Y_i) | i \in I)$
- composition: $(g \circ f) \triangleq (g_i \circ f_i \mid i \in I)$

(i.e. use composition of functions in Set at each index $i \in I$)

• identity: $id_X \triangleq (id_{X_i} \mid i \in I)$

(i.e. use identity functions in Set at each index $i \in I$)

For each $p : I \to J$ in Set, let $p^* : \operatorname{Set}^J \to \operatorname{Set}^I$ be the functor defined by:

$$p^* \begin{pmatrix} Y_j \\ f_j \\ Y'_j \end{pmatrix} j \in J \triangleq \begin{pmatrix} Y_{p i} \\ f_{p i} \\ Y'_{p i} \end{pmatrix} i \in I$$

i.e. p^* takes J-indexed families of sets/functions to I-indexed ones by precomposing with p

Dependent products of families of sets

For $I, J \in \text{Set}$, consider the functor $\pi_1^* : \text{Set}^I \to \text{Set}^{I \times J}$ induced by precomposition with the first projection function $\pi_1 : I \times J \to I$.

Theorem. π_1^* has a left adjoint $\Sigma : \mathbf{Set}^{I \times J} \to \mathbf{Set}^I$.

Proof. We apply the characterisation Theorem. For each $E \in \mathbf{Set}^{I \times J}$ we define $\Sigma E \in \mathbf{Set}^{I}$ and $\eta_E : E \to \pi_1^*(\Sigma E)$ in $\mathbf{Set}^{I \times J}$ with the required universal property ...

For each $E \in \mathbf{Set}^{I \times J}$, define $\Sigma E \in \mathbf{Set}^{I}$ to be the function mapping each $i \in I$ to the set

 $(\Sigma E)_i \triangleq \sum_{j \in J} E_{(i,j)} = \{(j,e) \mid j \in J \land e \in E_{(i,j)}\}$

For each $E \in \mathbf{Set}^{I \times J}$, define $\Sigma E \in \mathbf{Set}^{I}$ to be the function mapping each $i \in I$ to the set

$$(\Sigma E)_i \triangleq \sum_{j \in J} E_{(i,j)} = \{(j,e) \mid j \in J \land e \in E_{(i,j)}\}$$

and define $\eta_E : E \to \pi_1^*(\Sigma E)$ in Set^{*I*×*J*} to be the function mapping each $(i, j) \in I \times J$ to the function $(\eta_E)_{(i,j)} : E_{(i,j)} \to (\Sigma E)_i$ given by $e \mapsto (j, e)$.

Universal property-

For each $E \in \mathbf{Set}^{I \times J}$, define $\Sigma E \in \mathbf{Set}^{I}$ to be the function mapping each $i \in I$ to the set

$$(\Sigma E)_i \triangleq \sum_{j \in J} E_{(i,j)} = \{(j,e) \mid j \in J \land e \in E_{(i,j)}\}$$

and define $\eta_E : E \to \pi_1^*(\Sigma E)$ in Set^{*I*×*J*} to be the function mapping each $(i, j) \in I \times J$ to the function $(\eta_E)_{(i,j)} : E_{(i,j)} \to (\Sigma E)_i$ given by $e \mapsto (j, e)$.

Universal property–*existence part*: given any $X \in \mathbf{Set}^I$ and $f : E \to \pi_1^*(X)$ in

where for all $i \in I$, $j \in J$ and $e \in E_{(i,j)}$ $\overline{f}_i(j,e) \triangleq f_{(i,j)}(e)$

For each $E \in \mathbf{Set}^{I \times J}$, define $\Sigma E \in \mathbf{Set}^{I}$ to be the function mapping each $i \in I$ to the set

$$(\Sigma E)_i \triangleq \sum_{j \in J} E_{(i,j)} = \{(j,e) \mid j \in J \land e \in E_{(i,j)}\}$$

and define $\eta_E : E \to \pi_1^*(\Sigma E)$ in Set^{*I*×*J*} to be the function mapping each $(i, j) \in I \times J$ to the function $(\eta_E)_{(i,j)} : E_{(i,j)} \to (\Sigma E)_i$ given by $e \mapsto (j, e)$.

Universal property-uniqueness part: given $g: \Sigma E \to X$ in Set^I making $E \xrightarrow{\eta_E} \pi_1^*(\Sigma E)$ commute in Set^{I×J}, $f \xrightarrow{\downarrow} \pi_1^*(g)$ $\pi_1^*(X)$

then for all $i \in I$, and $(j, e) \in (\Sigma E)_i$ we have

 $\overline{f}_{i}(j,e) \triangleq f_{(i,j)}(e) = (\pi_{1}^{*}g \circ \eta_{E})_{(i,j)} e = (\pi_{1}^{*}g)_{(i,j)}((\eta_{E})_{(i,j)} e) \triangleq g_{i}(j,e)$

so $g = \overline{f}$. \Box

Dependent functions

of families of sets

We have seen that the left adjoint to $\pi_1^* : \mathbf{Set}^I \to \mathbf{Set}^{I \times J}$ is given by dependent products of sets.

Dually, dependent function sets give:

Theorem. π_1^* has a right adjoint $\Pi : \mathbf{Set}^{I \times J} \to \mathbf{Set}^{I}$.

Proof. We apply the characterisation Theorem. For each $E \in \mathbf{Set}^{I \times J}$ we define $\Pi E \in \mathbf{Set}^{I}$ and $\varepsilon_{E} : \pi_{1}^{*}(\Pi E) \to E$ in $\mathbf{Set}^{I \times J}$ with the required universal property ...

For each $E \in \text{Set}^{I \times J}$, define $\Pi E \in \text{Set}^{I}$ to be the function mapping each $i \in I$ to the set

 $(\Pi E)_i \triangleq \prod_{j \in J} E_{(i,j)} = \{ f \subseteq (\Sigma E)_i \mid f \text{ is single-value and total} \}$

where $f \subseteq (\Sigma E)_i$ is

single-valued if $\forall j \in J, \forall e, e' \in E_{(i,j)}, (j,e) \in f \land (j,e') \in f \Rightarrow e = e'$

total if $\forall j \in J, \exists e \in E_{(i,j)} (j, e) \in f$

Thus each $f \in (\Pi E)_i$ is a dependently typed function mapping elements $j \in J$ to elements of $E_{(i,j)}$ (result set depends on the argument *j*).

For each $E \in \text{Set}^{I \times J}$, define $\Pi E \in \text{Set}^{I}$ to be the function mapping each $i \in I$ to the set

 $(\Pi E)_i \triangleq \prod_{j \in J} E_{(i,j)} = \{ f \subseteq (\Sigma E)_i \mid f \text{ is single-value and total} \}$

and define $\varepsilon_E : \pi_1^*(\Pi E) \to E$ in Set^{$I \times J$} to be the function mapping each $(i, j) \in I \times J$ to the function $(\varepsilon_E)_{(i,j)} : (\Pi E)_i \to E_{(i,j)}$ given by $f \mapsto f j$ = unique $e \in E_{(i,j)}$ such that $(j, e) \in f$.

Universal property-

For each $E \in \text{Set}^{I \times J}$, define $\Pi E \in \text{Set}^{I}$ to be the function mapping each $i \in I$ to the set

 $(\Pi E)_i \triangleq \prod_{j \in J} E_{(i,j)} = \{ f \subseteq (\Sigma E)_i \mid f \text{ is single-value and total} \}$

and define $\varepsilon_E : \pi_1^*(\Pi E) \to E$ in Set^{$I \times J$} to be the function mapping each $(i, j) \in I \times J$ to the function $(\varepsilon_E)_{(i,j)} : (\Pi E)_i \to E_{(i,j)}$ given by $f \mapsto f j$ = unique $e \in E_{(i,j)}$ such that $(j, e) \in f$.

Universal property–*existence part*: given any $X \in \mathbf{Set}^I$ and $f : \pi_1^*(X) \to E$ in



where for all $i \in I$ and $x \in X_i$ $\overline{f}_i x \triangleq \{(j, f_{(i,j)} x) \mid j \in J\}$

For each $E \in \text{Set}^{I \times J}$, define $\Pi E \in \text{Set}^{I}$ to be the function mapping each $i \in I$ to the set

 $(\Pi E)_i \triangleq \prod_{j \in J} E_{(i,j)} = \{ f \subseteq (\Sigma E)_i \mid f \text{ is single-value and total} \}$

and define $\varepsilon_E : \pi_1^*(\Pi E) \to E$ in Set^{$I \times J$} to be the function mapping each $(i, j) \in I \times J$ to the function $(\varepsilon_E)_{(i,j)} : (\Pi E)_i \to E_{(i,j)}$ given by $f \mapsto f j$ = unique $e \in E_{(i,j)}$ such that $(j, e) \in f$.

Universal property–*uniqueness part*: given $g : X \rightarrow \prod E$ in Set^{*I*} making

 $\pi_{1}^{*}(\Pi E) \xrightarrow{\epsilon_{E}} E \text{ commute in Set}^{I \times J},$ $\pi_{1}^{*}(g) \uparrow f$ $\pi_{1}^{*}(X)$

then for all $i \in I$, $j \in J$ and $x \in X_i$ we have

$$\overline{f}_i x j \triangleq f_{(i,j)} x = (\varepsilon_E \circ \pi_1^* g)_{(i,j)} x = (\varepsilon_E)_{(i,j)} (g_i x) \triangleq g_i x j$$

so $q = \overline{f}$. \Box

Isomorphism of categories

Two categories **C** and **D** are isomorphic if they are isomorphic objects in the category of all categories of some given size; that is, if there are functors

 $\mathbf{C} \underbrace{\frown}_{G}^{F} \mathbf{D} \text{ with } \mathrm{id}_{\mathbf{C}} = G \circ F \text{ and } F \circ G = \mathrm{id}_{\mathbf{D}}.$

In which case, as usual, we write $\mathbf{C} \cong \mathbf{D}$.

Equivalence of categories

Two categories **C** and **D** are equivalent if there are functors $C \xrightarrow{F} D$ and natural isomorphisms $\eta : id_C \xrightarrow{G} F$ and $\varepsilon : F \circ G \xrightarrow{\simeq} id_D$. In which case, one writes $C \simeq D$.

Equivalence of categories

Two categories **C** and **D** are equivalent if there are functors $C \xrightarrow{F} D$ and natural isomorphisms $\eta : id_C \xrightarrow{G} F$ and $\varepsilon : F \circ G \xrightarrow{\simeq} id_D$. In which case, one writes $C \simeq D$.

Some deep results in mathematics take the form of equivalences of categories. E.g.

Stone duality: $\begin{pmatrix} \text{category of} \\ \text{Boolean algebras} \end{pmatrix}^{\text{op}} \simeq \begin{pmatrix} \text{category of compact} \\ \text{totally disconnected} \\ \text{Hausdorff spaces} \end{pmatrix}$ Gelfand duality: $\begin{pmatrix} \text{category of} \\ \text{abelian } C^* \text{ algebras} \end{pmatrix}^{\text{op}} \simeq \begin{pmatrix} \text{category of compact} \\ \text{Hausdorff spaces} \end{pmatrix}$

Example: $\mathbf{Set}^I \simeq \mathbf{Set}/I$

Set/*I* is a slice category:

- ▶ objects are pairs (E, p) where E ∈ obj Set and p ∈ Set(E, I)
- morphisms $g: (E, p) \rightarrow (E', p')$ are $f \in \text{Set}(E, E')$ satisfying $p' \circ f = p$ in Set
- composition and identities as for Set

Example: $\mathbf{Set}^I \simeq \mathbf{Set}/I$

There are functors $F : \mathbf{Set}^I \to \mathbf{Set}/I$ and $G : \mathbf{Set}/I \to \mathbf{Set}^I$, given on objects and morphisms by:

 $FX \triangleq (\{(i,x) \mid i \in I \land x \in X_i\}, \texttt{fst})$ $Ff(i,x) \triangleq (i, f_i x)$ $G(E,p) \triangleq (\{e \in E \mid p \ e = i\} \mid i \in I)$ $(Gf)_i \ e \triangleq f \ e$

Example: $\mathbf{Set}^I \simeq \mathbf{Set}/I$

There are functors $F : \mathbf{Set}^I \to \mathbf{Set}/I$ and $G : \mathbf{Set}/I \to \mathbf{Set}^I$, given on objects and morphisms by:

$$FX \triangleq (\{(i, x) \mid i \in I \land x \in X_i\}, \texttt{fst})$$

$$Ff(i, x) \triangleq (i, f_i x)$$

$$G(E, p) \triangleq (\{e \in E \mid p \ e = i\} \mid i \in I)$$

$$(Gf)_i e \triangleq f e$$

There are natural isomorphisms

 $\eta : \mathrm{id}_{\mathrm{Set}^{I}} \cong G \circ F \text{ and } \varepsilon : F \circ G \cong \mathrm{id}_{\mathrm{Set}^{I}}$

defined by ... [exercise]

FACT Given $p : I \rightarrow J$ in **Set**, the composition

$$\operatorname{Set}/J \simeq \operatorname{Set}^J \xrightarrow{p^*} \operatorname{Set}^I \simeq \operatorname{Set}/I$$

is the functor "pullback along p".

One can generalize from Set to any category C with pullbacks and model Σ/Π types by left/right adjoints to pullback functors – see locally cartesian closed categories in the literature.

Presheaf categories

Let C be a small category. The functor category $Set^{C^{op}}$ is called the category of presheaves on C.

- objects are contravariant functors from C to Set
- morphisms are natural transformations

Much used in the semantics of various dependently-typed languages and logics.

Given a category C with a terminal object 1

A global element of an object $X \in obj C$ is by definition a morphism $1 \rightarrow X$ in C

E.g. in Set ...

E.g. in **Mon** ...

Given a category C with a terminal object 1

A global element of an object $X \in obj C$ is by definition a morphism $1 \rightarrow X$ in C

We say that **C** is well-pointed if for all $f, g : X \rightarrow Y$ in **C** we have:

$$\left(\forall 1 \xrightarrow{x} X, f \circ x = g \circ x \right) \implies f = g$$

(Set is, Mon isn't.)

Idea:

replace global elements of X, $1 \xrightarrow{x} X$

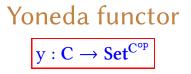
by arbitrary morphisms $C \xrightarrow{x} X$ (for any $C \in obj C$)

Idea:

replace global elements of X, $1 \xrightarrow{x} X$

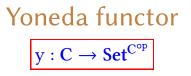
by arbitrary morphisms $C \xrightarrow{x} X$ (for any $C \in obj C$) Some people use the notation $x \in_C X$ and say "x is a generalised element of X at stage C" Have to take into account "change of stage": $x \in_C X \land D \xrightarrow{f} C \implies x \circ f \in_D X$

(cf. Kripke's "possible world" semantics of intuitionistic and modal logics)



is the Curried version of the hom functor

 $C \times C^{op} \xrightarrow{} C^{op} \times C \xrightarrow{Hom_C} Set$

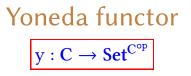


is the Curried version of the hom functor

$$\mathbf{C} \times \mathbf{C}^{\mathrm{op}} \xrightarrow{} \mathbf{C}^{\mathrm{op}} \times \mathbf{C} \xrightarrow{\mathrm{Hom}_{\mathbf{C}}} \mathbf{Set}$$

► For each C-object *X*, the object $yX \in \text{Set}^{C^{op}}$ is the functor $C(_, X) : C^{op} \rightarrow \text{Set}$ given by

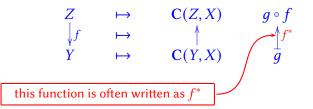
$$\begin{array}{ccccc} Z & \mapsto & \mathbf{C}(Z,X) & g \circ f \\ & \downarrow f & \mapsto & \uparrow & & \uparrow \\ Y & \mapsto & \mathbf{C}(Y,X) & g \end{array}$$



is the Curried version of the hom functor

$$\mathbf{C} \times \mathbf{C}^{\mathrm{op}} \xrightarrow{} \mathbf{C}^{\mathrm{op}} \times \mathbf{C} \xrightarrow{\mathrm{Hom}_{\mathbf{C}}} \mathbf{Set}$$

► For each C-object *X*, the object $yX \in \text{Set}^{C^{op}}$ is the functor $C(_, X) : C^{op} \rightarrow \text{Set}$ given by



Yoneda functor $y: C \rightarrow Set^{C^{op}}$

is the Curried version of the hom functor

 $C \times C^{op} \xrightarrow{} C^{op} \times C \xrightarrow{Hom_C} Set$

► For each C-morphism $Y \xrightarrow{f} X$, the morphism $yY \xrightarrow{yf} yX$ in Set^{C^{op}} is the natural transformation whose component at any given $Z \in C^{op}$ is the function

$$yY(Z) \xrightarrow{(yf)_Z} yX(Z)$$

$$C(Z,Y) \xrightarrow{(II)} C(Z,X)$$

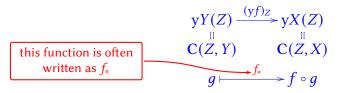
$$q \longmapsto f \circ q$$

Yoneda functor $y: \mathbf{C} \to \mathbf{Set}^{\mathbf{C}^{\mathrm{op}}}$

is the Curried version of the hom functor

 $C \times C^{op} \xrightarrow{} C^{op} \times C \xrightarrow{Hom_C} Set$

► For each C-morphism $Y \xrightarrow{f} X$, the morphism $yY \xrightarrow{yf} yX$ in Set^{C^{op}} is the natural transformation whose component at any given $Z \in C^{op}$ is the function



The Yoneda Lemma

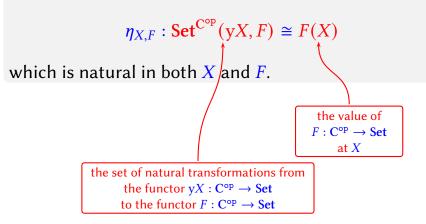
For each small category C, each object $X \in C$ and each presheaf $F \in Set^{C^{op}}$, there is a bijection of sets

 $\eta_{X,F}$: Set^{C^{op}}(yX, F) \cong F(X)

which is natural in both X and F.

The Yoneda Lemma

For each small category C, each object $X \in C$ and each presheaf $F \in Set^{C^{op}}$, there is a bijection of sets



For each small category C, each object $X \in C$ and each presheaf $F \in Set^{C^{op}}$, there is a bijection of sets

 $\eta_{X,F}$: Set^{C^{op}} $(yX,F) \cong F(X)$

which is natural in both X and F.

Definition of the function $\eta_{X,F}$: **Set**^{C^{op}} $(yX, F) \rightarrow F(X)$: for each θ : $yX \rightarrow F$ in **Set**^{C^{op}} we have the function $C(X,X) = yX(X) \xrightarrow{\theta_X} F(X)$ and define

 $\eta_{X,F}(\theta) \triangleq \theta_X(\texttt{id}_X)$

For each small category C, each object $X \in C$ and each presheaf $F \in Set^{C^{op}}$, there is a bijection of sets

 $\eta_{X,F}$: Set^{C^{op}} $(yX,F) \cong F(X)$

which is natural in both X and F.

Definition of the function $\eta_{X,F}^{-1} : F(X) \to \operatorname{Set}^{\operatorname{C^{op}}}(yX,F)$: for each $x \in F(X), Y \in \mathbb{C}$ and $f \in yX(Y) = \mathbb{C}(Y,X)$, we get a $F(X) \xrightarrow{F(f)} F(Y)$ in Set and hence $F(f)(x) \in F(Y)$;

For each small category C, each object $X \in C$ and each presheaf $F \in Set^{C^{op}}$, there is a bijection of sets

 $\eta_{X,F}$: **Set**^{C^{op}}(yX, F) \cong F(X)

which is natural in both X and F.

Definition of the function $\eta_{X,F}^{-1} : F(X) \to \operatorname{Set}^{\operatorname{C^{op}}}(yX,F)$: for each $x \in F(X), Y \in \mathbb{C}$ and $f \in yX(Y) = \mathbb{C}(Y,X)$, we get a $F(X) \xrightarrow{F(f)} F(Y)$ in **Set** and hence $F(f)(x) \in F(Y)$; Define $\left(\eta_{X,F}^{-1}(x)\right)_{Y} : yX(Y) \to F(Y)$ by

$$\left(\eta_{X,F}^{-1}(x)\right)_Y(f) \triangleq F(f)(x)$$

check this gives a natural transformation $\eta_{X,F}^{-1}(x) : yX \to F$ **Proof of** $\eta_{X,F} \circ \eta_{X,F}^{-1} = id_{F(X)}$

For any $x \in F(X)$ we have

$$\eta_{X,F}\left(\eta_{X,F}^{-1}(x)\right) \triangleq \left(\eta_{X,F}^{-1}(x)\right)_{X} (\mathrm{id}_{X})$$
$$\triangleq F(\mathrm{id}_{X})(x)$$
$$= \mathrm{id}_{F(X)}(x)$$
$$= x$$

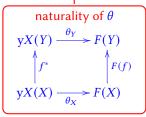
by definition of $\eta_{X,F}$ by definition of $\eta_{X,F}^{-1}$ since *F* is a functor

Proof of
$$\eta_{X,F}^{-1} \circ \eta_{X,F} = \operatorname{id}_{\operatorname{Set}^{\operatorname{Cop}}(yX,F)}$$

For any $yX \xrightarrow{\theta} F$ in Set^{C^{op}} and $Y \xrightarrow{f} X$ in C, we have

$$\begin{pmatrix} \eta_{X,F}^{-1}(\eta_{X,F}(\theta)) \end{pmatrix}_{Y} f \triangleq \begin{pmatrix} \eta_{X,F}^{-1}(\theta_{X}(\operatorname{id}_{X}))) \end{pmatrix}_{Y} \\ \triangleq F(f)(\theta_{X}(\operatorname{id}_{X})) \\ = \theta_{Y}(f^{*}(\operatorname{id}_{X})) \\ \triangleq \theta_{Y}(\operatorname{id}_{X} \circ f) \\ = \theta_{Y}(f)$$

by definition of $\eta_{X,F}$ by definition of $\eta_{X,F}^{-1}$ by naturality of θ by definition of f^*



Proof of
$$\eta_{X,F}^{-1} \circ \eta_{X,F} = \operatorname{id}_{\operatorname{Set}^{\operatorname{Cop}}(yX,F)}$$

For any $yX \xrightarrow{\theta} F$ in Set^{C^{op}} and $Y \xrightarrow{f} X$ in C, we have

$$\begin{pmatrix} \eta_{X,F}^{-1}(\eta_{X,F}(\theta)) \end{pmatrix}_{Y} f \triangleq \begin{pmatrix} \eta_{X,F}^{-1}(\theta_{X}(\mathrm{id}_{X}))) \end{pmatrix}_{Y} f \\ \triangleq F(f)(\theta_{X}(\mathrm{id}_{X})) \\ = \theta_{Y}(f^{*}(\mathrm{id}_{X})) \\ \triangleq \theta_{Y}(\mathrm{id}_{X} \circ f) \\ = \theta_{Y}(f)$$

by definition of $\eta_{X,F}$ by definition of $\eta_{X,F}^{-1}$ by naturality of θ by definition of f^*

so
$$\forall \theta, Y, \left(\eta_{X,F}^{-1}\left(\eta_{X,F}(\theta)\right)\right)_{Y} = \theta_{Y}$$

so $\forall \theta, \eta_{X,F}^{-1}\left(\eta_{X,F}(\theta)\right) = \theta$
so $\eta_{X,F}^{-1} \circ \eta_{X,F} = \text{id.}$

For each small category **C**, each object $X \in \mathbf{C}$ and each presheaf $F \in \mathbf{Set}^{\mathbf{C}^{op}}$, there is a bijection of sets

 $\eta_{X,F}$: Set^{C^{op}}(yX, F) \cong F(X)

which is natural in both X and F.

Proof that $\eta_{X,F}$ is natural in *F*:

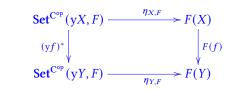
Given $F \xrightarrow{\varphi} G$ in **Set**^{C°P}, have to show that

commutes in Set. For all $yX \xrightarrow{\theta} F$ we have

$$\varphi_X \left(\eta_{X,F}(\theta) \right) \triangleq \varphi_X \left(\theta_X(\mathrm{id}_X) \right) \\ \triangleq \left(\varphi \circ \theta \right)_X(\mathrm{id}_X) \\ \triangleq \eta_{X,G}(\varphi \circ \theta) \\ \triangleq \eta_{X,G}(\varphi_*(\theta))$$

Proof that $\eta_{X,F}$ is natural in X:

Given $Y \xrightarrow{f} X$ in **C**, have to show that



commutes in Set. For all $yX \xrightarrow{\theta} F$ we have

$$F(f)((\eta_{X,F}(\theta)) \triangleq F(f)(\theta_X(id_X))$$

= $\theta_Y(f^*(id_X))$ by naturality of θ
= $\theta_Y(f)$
= $\theta_Y(f_*(id_Y))$
 $\triangleq (\theta \circ yf)_Y(id_Y)$
 $\triangleq \eta_{Y,F}(\theta \circ yf)$
 $\triangleq \eta_{Y,F}((yf)^*(\theta))$

Corollary of the Yoneda Lemma:

the functor $y : C \rightarrow Set^{C^{op}}$ is full and faithful.

In general, a functor $F : \mathbf{C} \to \mathbf{D}$ is

• faithful if for all $X, Y \in \mathbf{C}$ the function

$\mathbf{C}(X,Y)$	\rightarrow	$\mathbf{D}(F(X), F(Y))$
f	\mapsto	F(f)

is injective:

 $\forall f, f' \in \mathbf{C}(X, Y), \ F(f) = F(f') \Longrightarrow f = f'$

► full if the above functions are all surjective: $\forall g \in \mathbf{D}(F(X), F(Y)), \exists f \in \mathbf{C}(X, Y), F(f) = g$ **Corollary** of the Yoneda Lemma:

the functor $y : C \rightarrow Set^{C^{op}}$ is full and faithful.

Proof. From the proof of the Yoneda Lemma, for each $F \in \mathbf{Set}^{\mathbb{C}^{op}}$ we have a bijection

$$F(X) \xrightarrow{(\eta_{X,F})^{-1}} \operatorname{Set}^{\operatorname{C^{op}}}(\mathrm{y}X,F)$$

By definition of $(\eta_{X,F})^{-1}$, when F = yY the above function is equal to

$$yY(X) = C(X, Y) \rightarrow Set^{C^{p}}(yX, yY)$$

 $f \mapsto f_* = yf$

So, being a bijection, $f \mapsto yf$ is both injective and surjective; so y is both faithful and full.

Recall (for a small category C):

<u>Yoneda functor</u> $y : C \rightarrow Set^{C^{op}}$

Yoneda Lemma: there is a bijection $\operatorname{Set}^{\operatorname{C^{op}}}(yX, F) \cong F(X)$ which is natural both in $F \in \operatorname{Set}^{\operatorname{C^{op}}}$ and $X \in \mathbb{C}$.

An application of the Yoneda Lemma:

Theorem. For each small category **C**, the category **Set**^{C^{op}} of presheaves is cartesian closed.

Theorem. For each small category C, the category **Set**^{C^{op}} of presheaves is cartesian closed.

Theorem. For each small category C, the category **Set**^{Cop} of presheaves is cartesian closed.

Proof sketch.

Terminal object in Set^{C^{op}} is the functor $1 : C^{op} \rightarrow Set$ given by

 $\begin{cases} 1(X) \triangleq \{0\} & \text{terminal object in Set} \\ 1(f) \triangleq \text{id}_{\{0\}} \end{cases}$

Theorem. For each small category C, the category **Set**^{C^{op}} of presheaves is cartesian closed.

Proof sketch.

Product of $F, G \in \mathbf{Set}^{\mathbb{C}^{op}}$ is the functor $F \times G : \mathbb{C}^{op} \to \mathbf{Set}$ given by

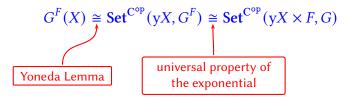
 $\begin{cases} (F \times G)(X) \triangleq F(X) \times G(X) & \text{cartesian product of sets} \\ (F \times G)(f) \triangleq F(f) \times G(f) \end{cases}$

with projection morphisms $F \xleftarrow{\pi_1} F \times G \xrightarrow{\pi_2} G$ given by the natural transformations whose components at $X \in \mathbb{C}$ are the projection functions $F(X) \xleftarrow{\pi_1} F(X) \times G(X) \xrightarrow{\pi_2} G(X)$.

Theorem. For each small category **C**, the category **Set**^{Cop} of presheaves is cartesian closed.

Proof sketch.

We can work out what the value of the exponential $G^F \in \mathbf{Set}^{\mathbf{C}^{\mathrm{op}}}$ at $X \in \mathbf{C}$ has to be using the Yoneda Lemma:



Theorem. For each small category **C**, the category **Set**^{Cop} of presheaves is cartesian closed.

Proof sketch.

We can work out what the value of the exponential $G^F \in \mathbf{Set}^{\mathbb{C}^{op}}$ at $X \in \mathbb{C}$ has to be using the Yoneda Lemma:

 $G^{F}(X) \cong \mathbf{Set}^{\mathbf{C}^{\mathrm{op}}}(\mathbf{y}X, G^{F}) \cong \mathbf{Set}^{\mathbf{C}^{\mathrm{op}}}(\mathbf{y}X \times F, G)$

We take the set $\operatorname{Set}^{C^{\operatorname{op}}}(yX \times F, G)$ to be the definition of the value of G^F at X...

Exponential objects in Set^{C°P}:

$$G^F(X) \triangleq \mathbf{Set}^{\mathbf{C}^{\mathsf{op}}}(\mathbf{y}X \times F, G)$$

Given
$$Y \xrightarrow{f} X$$
 in C, we have $yY \xrightarrow{yf} yX$ in Set^{C^{op}} and hence
 $G^F(X) \triangleq \operatorname{Set}^{\operatorname{C^{op}}}(yX \times F, G) \xrightarrow{} \operatorname{Set}^{\operatorname{C^{op}}}(yY \times F, G) \triangleq G^F(Y)$
 $\theta \mapsto \theta \circ (yf \times \operatorname{id}_F)$

We define

$$G^F(f) \triangleq (\mathbf{y} f \times \mathrm{id}_F)^*$$

Have to check that these definitions make G^F into a functor $C^{op} \rightarrow Set$.

Application morphisms in Set^{C°P}:

Given $F, G \in \mathbf{Set}^{\mathbb{C}^{op}}$, the application morphism

 $\operatorname{app}: G^F \times F \to G$

is the natural transformation whose component at $X \in \mathbb{C}$ is the function

 $(G^F \times F)(X) \triangleq G^F(X) \times F(X) \triangleq \mathbf{Set}^{\mathbf{C}^{\mathrm{op}}}(\mathbf{y}X \times F, G) \times F(X) \xrightarrow{\mathtt{app}_X} G(X)$

defined by

 $\operatorname{app}_X(\theta, x) \triangleq \theta_X(\operatorname{id}_X, x)$

Have to check that this is natural in X.

Currying operation in Set^{C^{op}}:

$$\left(H \times F \xrightarrow{\theta} G\right) \mapsto \left(H \xrightarrow{\operatorname{cur} \theta} G^F\right)$$

Given $H \times F \xrightarrow{\theta} G$ in $\operatorname{Set}^{\operatorname{Cop}}$, the component of $\operatorname{cur} \theta$ at $X \in \mathbb{C}$ $H(X) \xrightarrow{(\operatorname{cur} \theta)_X} G^F(X) \triangleq \operatorname{Set}^{\operatorname{Cop}}(yX \times F, G)$

is the function mapping each $z \in H(X)$ to the natural transformation $yX \times F \to G$ whose component at $Y \in \mathbb{C}$ is the function

 $(\mathbf{y}X \times F)(Y) \triangleq \mathbf{C}(Y,X) \times F(Y) \to G(Y)$

defined by

$$((\operatorname{cur} \theta)_X(z))_Y(f,y) \triangleq \theta_Y(H(f)(z),y)$$

Currying operation in Set^{C^{op}}:

$$\left(H \times F \xrightarrow{\theta} G\right) \mapsto \left(H \xrightarrow{\operatorname{cur} \theta} G^F\right)$$

$$((\operatorname{cur} \theta)_X(z))_Y(f,y) \triangleq \theta_Y(H(f)(z),y)$$

Have to check that this is natural in Y,

then that $(\operatorname{cur} \theta)_X$ is natural in *X*,

then that $\operatorname{cur} \theta$ is the unique morphism $H \xrightarrow{\varphi} G^F$ in $\operatorname{Set}^{C^{\operatorname{op}}}$ satisfying $\operatorname{app} \circ (\varphi \times \operatorname{id}_F) = \theta$.

Theorem. For each small category **C**, the category **Set**^{Cop} of presheaves is cartesian closed.

So we can interpret simply typed lambda calculus in any presheaf category.

More than that, presheaf categories (usefully) model dependently-typed languages.

Appendix

Used in Haskell to abstract generic aspects of computation (return a value, sequencing) and to encapsulate effectful code.

Concept imported into functional programming from category theory, first for its denotational semantics by Moggi and then for its practice by <u>Wadler</u>.

Used in Haskell to abstract generic aspects of computation (return a value, sequencing) and to encapsulate effectful code.

Concept imported into functional programming from category theory, first for its denotational semantics by Moggi and then for its practice by Wadler.

Here, a quick overview of:

- Moggi's computational λ-calculus and its categorical semantics using (strong) monads
- monads and adjunctions

Computational Lambda Calculus (CLC)

CLC extends STLC with new types, terms and equations ...

Types: *A*, *B*, . . . ::= STLC types, plus

T(A) type of "computations" of values of type A

Terms: *s*, *t*, . . . ::= STLC terms, plus

 $\begin{array}{ll} \textbf{return } t & \textbf{trivial computation} \\ \textbf{do} \{ x \leftarrow s; t \} & \textbf{sequenced computation (binds free } x \text{ in } t) \\ \end{array}$ $\begin{array}{l} \textbf{As for STLC, we identify CLC syntax trees up to α-equivalence, where =α is extended by the rules} \\ \hline & t = α t' & (y x) \cdot t = α (y x') \cdot t'$ \\ \hline & t = α t' & y \text{ does not occur in } \{s, s', x, x', t, t'\} \\ \hline & \textbf{do} \{x \leftarrow s; t\} = α do} \{x' \leftarrow s'; t'\} \end{array}$

Computational Lambda Calculus (CLC)

CLC extends STLC with new types, terms and equations ...

Types: *A*, *B*, . . . ::= STLC types, plus

T(A) type of "computations" of values of type A

Terms: *s*, *t*, . . . ::= STLC terms, plus

return ttrivial computation $do{x \leftarrow s; t}$ sequenced computation (binds free x in t)

Typing rules:

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \texttt{return } t : \texttt{T}(A)} (\texttt{VAL}) \quad \frac{\Gamma \vdash s : \texttt{T}(A) \qquad \Gamma, x : A \vdash t : \texttt{T}(B)}{\Gamma \vdash \texttt{do}\{x \leftarrow s; t\} : \texttt{T}(B)} (\texttt{seq})$$

Equations ...

CLC equations

Extend STLC $\beta\eta$ -equality ($\Gamma \vdash s =_{\beta\eta} t : A$) to a relation $\Gamma \vdash s = t : A$ by adding the following rules:

$$\frac{\Gamma \vdash s : A \qquad \Gamma, x : A \vdash t : \mathsf{T}(B)}{\Gamma \vdash \mathsf{do}\{x \leftarrow \texttt{return} \, s; t\} = t[s/x] : \mathsf{T}(B)}$$

$$\Gamma \vdash t : T(A)$$

$$\Gamma \vdash t = \mathrm{do}\{x \leftarrow t; \mathtt{return}\, x\} : \mathtt{T}(A)$$

$$\frac{\Gamma \vdash s: \mathsf{T}(A) \qquad \Gamma, x: A \vdash t: \mathsf{T}(B) \qquad \Gamma, y: B \vdash u: \mathsf{T}(C)}{\Gamma \vdash \mathsf{do}\{y \leftarrow \mathsf{do}\{x \leftarrow s; t\}; u\} = \mathsf{do}\{x \leftarrow s; \mathsf{do}\{y \leftarrow t; u\}\}}$$

CLC equations

Extend STLC $\beta\eta$ -equality ($\Gamma \vdash s =_{\beta\eta} t : A$) to a relation $\Gamma \vdash s = t : A$ by adding the following rules:

$$\begin{split} & \frac{\Gamma \vdash s : A \qquad \Gamma, x : A \vdash t : T(B)}{\Gamma \vdash do\{x \leftarrow \texttt{return} s; t\} = t[s/x] : T(B)} \\ & \frac{\Gamma \vdash t : T(A)}{\Gamma \vdash t = do\{x \leftarrow t; \texttt{return} x\} : T(A)} \\ & \frac{\Gamma \vdash s : T(A) \qquad \Gamma, x : A \vdash t : T(B) \qquad \Gamma, y : B \vdash u : T(C)}{\Gamma \vdash do\{y \leftarrow do\{x \leftarrow s; t\}; u\} = do\{x \leftarrow s; do\{y \leftarrow t; u\}\}} \end{split}$$

(To describe a particular notion of computation (I/O, mutable state, exceptions, concurrent processes, ...) one can consider extensions of vanilla CLC, e.g. with extra ground types, constants and equations.)

Parameterised Kleisli triple

is the following extra structure on a category **C** with binary products:

- a function mapping each X ∈ obj C to an object
 T(X) ∈ obj C
- ► for each $X \in obj \mathbb{C}$, a \mathbb{C} -morphism $X \xrightarrow{\eta_X} T(X)$
- ► for each C-morphism $X \times Y \xrightarrow{f} T(Z)$ a C-morphism $X \times T(Y) \xrightarrow{f^*} T(Z)$

satisfying ...

Parameterised Kleisli triple[cont.]

• if $X \xrightarrow{f} X'$ and $X' \times Y \xrightarrow{g} T(Z)$, then

 $(g \circ (f \times id_Y))^* = g^* \circ (f \times id_{T(Y)})$

• if $X \times Y \xrightarrow{f} T(Z)$, then $f^* \circ (\operatorname{id}_X \times \eta_Y) = f$

• if $X \times Y \xrightarrow{f} T(Z)$ and $X \times Z \xrightarrow{g} T(W)$, then $(g^* \circ \langle \pi_1, f \rangle)^* = g^* \circ \langle \pi_1, f^* \rangle$

State: fix a set *S* (of "states") and define $T(X) \triangleq (X \times S)^S$ $\eta_X x s \triangleq (x, s)$ $f^*(x, t) s \triangleq f(x, y) s'$ where t s = (y, s')

State: fix a set *S* (of "states") and define

computations are functions $S \rightarrow X \times S$ taking states to values in X paired with a next state

 $f^*(x, t) s \triangleq f(x, y) s'$ where t s = (y, s')

 $T(X) \triangleq (X \times S)^S \leftarrow$

 $\eta_X x s \triangleq (x, s)$

 $f^*(x, _)$ first "runs" $t \in T(Y)$ in state *s* to get (y, s'), then runs $f(x, y) \in T(Z)$ in the new state *s*'

Error:

 $T(X) \triangleq X + 1 = \{(0, x) \mid x \in X\} \cup \{(1, 0)\}$ $\eta_X x \triangleq (0, x)$ $f^*(x, t) \triangleq \begin{cases} f(x, y) & \text{if } t = (0, y) \\ (1, 0) & \text{if } t = (1, 0) \end{cases}$

Error:

 $T(X) \triangleq X + 1 = \{(0, x) \mid x \in X\} \cup \{(1, 0)\} \leftarrow$ computations are either $\eta_X x \triangleq (0, x)$ copies (0, x) of values in $f^*(x,t) \triangleq \begin{cases} f(x,y) & \text{if } t = (0,y) \\ (1,0) & \text{if } t = (1,0) \end{cases}$ $x \in X$ or an error (1, 0)if $t \in T(Y)$ is the error, then $f^*(x, _)$ propagates it, otherwise it acts like f

Continuations: fix a set *R* (of "results") and define $T(X) \triangleq R^{(R^X)}$ $\eta_X x \triangleq \lambda c \in R^X. c x$ $f^*(x, r) \triangleq \lambda c \in R^Z. r(\lambda y \in Y. f(x, y) c)$

Continuations: fix a set *R* (of "results") and define

computations are functions $r : \mathbb{R}^X \to \mathbb{R}$ mapping continuations $c \in \mathbb{R}^X$ of the computation to results $r c \in \mathbb{R}$

 $\eta_X x \triangleq \lambda c \in R^X. c x$

 $T(X) \triangleq R^{(R^X)}$

 $f^*(x,r) \triangleq \lambda c \in R^Z. r(\lambda y \in Y. f(x,y) c)$

 f^* maps a computation $r \in R^{(R^Y)}$ to the function taking a continuation $c \in R^Z$ to the result of applying r to the continuation $\lambda y \in Y$. f(x, y) c in R^Y

Semantics of CLC

Given a ccc C equipped with a parameterised Kleisli triple $(T, \eta, (_)^*)$, we can extend the semantics of STLC to one for CLC.

Computation types: [T(A)] = T([A])Trivial computations:

$$\begin{bmatrix} \Gamma \vdash \operatorname{return} t : \operatorname{T}(A) \end{bmatrix} = \llbracket \Gamma \rrbracket \xrightarrow{\llbracket \Gamma \vdash t:A \rrbracket} \llbracket A \rrbracket \xrightarrow{\eta \llbracket A \rrbracket} T(\llbracket A \rrbracket)$$

Sequencing: $\llbracket \Gamma \vdash \operatorname{do} \{x \leftarrow s; t\} : \operatorname{T}(B) \rrbracket = f^* \circ \langle \operatorname{id}_{\llbracket \Gamma} \rrbracket, g \rangle$
where
$$\begin{cases} f = \llbracket \Gamma \rrbracket \times \llbracket A \rrbracket \xrightarrow{\llbracket \Gamma, x: A \vdash t: \operatorname{T}(B) \rrbracket} T(\llbracket B \rrbracket) \\ g = \llbracket \Gamma \rrbracket \xrightarrow{\llbracket \Gamma \vdash s: \operatorname{T}(A) \rrbracket} T(\llbracket A \rrbracket) \end{cases}$$

(and where A is uniquely determined from the proof of $\Gamma \vdash \operatorname{do} \{x \leftarrow s; t\} : \operatorname{T}(B)$)

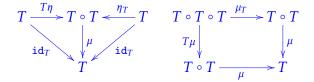
Semantics of CLC

Given a ccc C equipped with a parameterised Kleisli triple $(T, \eta, (_)^*)$, we can extend the semantics of STLC to one for CLC.

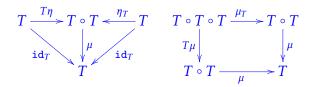
As for STLC versus cccs,

- the semantics of CLC in cc+Kleisli categories is equationally sound and complete
- one can use CLC as an internal language for describing constructs in cc+Kleisli categories
- there is a correspondence between equational theories in CLC and cc+Kleisli categories

A monad on a category C is given by a functor $T : C \to C$ and natural transformations $\eta : id \to T$ and $\mu : T \circ T \to T$ satisfying

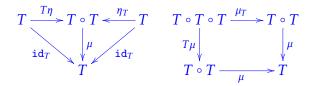


A monad on a category C is given by a functor $T : C \to C$ and natural transformations $\eta : id \to T$ and $\mu : T \circ T \to T$ satisfying



If **C** has binary products, then the monad is **strong** if there is a family of **C**-morphisms $(X \times T(Y) \xrightarrow{s_{X,Y}} T(X \times Y) | X, Y \in \text{obj } \mathbf{C})$ satisfying a number (7, in fact) of commutative diagrams (details omitted, see Moggi).

A monad on a category C is given by a functor $T : C \to C$ and natural transformations $\eta : id \to T$ and $\mu : T \circ T \to T$ satisfying



If **C** has binary products, then the monad is **strong** if there is a family of **C**-morphisms $(X \times T(Y) \xrightarrow{s_{X,Y}} T(X \times Y) | X, Y \in \text{obj } \mathbf{C})$ satisfying a number (7, in fact) of commutative diagrams (details omitted, see Moggi).

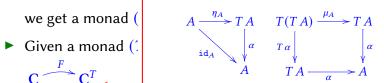
FACT: for a given category with binary products, "parameterised Kleisli triple" and "strong monad" are equivalent notions – each gives rise to the other in a bijective fashion.

• Given an adjunction
$$C \bigoplus_{G}^{F} D$$
 $F \dashv G$
we get a monad $(G \circ F, \eta, \mu)$ on C
where $\begin{cases} \eta_X = \overline{id_{FX}} \\ \mu_X = G(\overline{id_{G(FX)}}) \end{cases}$
E.g. for Set \bigoplus_{U}^{F} Mon where U is the forgetful functor, $T = U \circ F$ is
the list monad on Set $(T(X) = \text{List } X, \eta$ given by singleton lists, μ by
flattening lists of lists). It's a strong monad (all monads of Set have a
strength), but in general the monad associated with an adjunction may
not be strong.

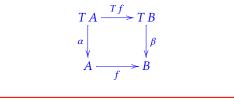
- Given an adjunction $C \bigoplus_{G}^{F} D$ we get a monad $(G \circ F, \eta, \mu)$ on C
- Given a monad (T, η, μ) on **C** we get an adjunction
 - $C \underbrace{\overset{F}{\underset{G}{\longrightarrow}}}_{G} C^{T} \underbrace{F + G}$

 $(\mathbf{C}^T$ is the category of Eilenberg-Moore algebras for the monad *T*, which has objects (A, α) with $\alpha: T(A) \rightarrow A$ satisfying

Given an adjunct



and morphisms $f(A, \alpha) \rightarrow (B, \beta)$ with $f : A \rightarrow B$ satisfying



- Given an adjunction $C \bigoplus_{G}^{F} D$ we get a monad $(G \circ F, \eta, \mu)$ on C
- Given a monad (T, η, μ) on **C** we get an adjunction

$$C \underbrace{\overbrace{G}}^{F} C^{T} \qquad \underline{F + G}$$

Starting from $C \bigoplus_{G}^{F} D F \dashv G$ and forming the monad

 $T = G \circ F$, there's an obvious functor $K : \mathbf{D} \to \mathbf{C}^T$.

Monadicity Theorems impose conditions on $G : \mathbf{D} \to \mathbf{C}$ which ensure that *K* is an equivalence of categories. E.g. Mon is equivalent to the category of Eilenberg-Moore algebras for the list monad on Set (and similarly for any algebraic theory).

Some current themes involving category theory in computer science

 semantics of effects & co-effects in programming languages

(monads and comonads)

- homotopy type theory (higher-dimensional category theory)
- structural aspects of networks, quantum computation/protocols, ...

(string diagrams for monoidal categories)