Introduction to Categorical Logic

Lawvere theories and cartesian closed categories

Thibaut Benjamin

31st August – 4th September 2026

Proof Assistants and Applications Summer School

Why categorical logic in a proof-asssistant shool?

Categorical logic provides a way to study formal languages, which are helpful to:

Why categorical logic in a proof-asssistant shool?

Categorical logic provides a way to study formal languages, which are helpful to:

- Reason about proof-assistants
  It unerpins many design choices of proof-assistants and provides theoretical foundations.

Why categorical logic in a proof-asssistant shool?

Categorical logic provides a way to study formal languages, which are helpful to:

- Reason about proof-assistants
  It unerpins many design choices of proof-assistants and provides theoretical foundations.

- Work inside a proof-assistant
  Define your own mini DSL to work with your favourite algebraic structures.

Logic, categories

Category theory and logic have a lot to tell each other:

Logic	**Categories**
Algebraic Theories	Lawvere Theories
Essentially Algebraic Theories	Left Exact Categories
λ-calculus	Cartesian Closed categories
Generalied algebraic theory	Category with families
$\vdots$	

Syntax/Semantics dualities

In all these instances, there exists a duality between syntax and semantics, that always work the same way:

1. Each logical theory has a notion of a *model*.
   an algebraic structure equipped with the structure prescribed by the theory.

2. The initial model is the collection of all closed terms.

3. The corresponding categorical description is the collection of freely finitely generated objects, which is also the collection of all open terms.

Disclaimer

- This course aims at connecting category theory, logic and type theory, so we will be doing a lot of setups and definitions before seeing the first results, but buckle up!
  I will try to limit every of the topic to the bare minimum, but still give you intuition of what we are talking about.

- These slides are also meant to serve as reference material.
  Some may dense or heavy, don't be scared

A primer on category theory

A primer on category theory

Categories and functors

Categories

For pedagogical purposes, size issues are ignored

Definition
A category C consists in the following pieces of data:

- a set $\mathrm{Ob}(C)$ of objects,

Satisfying the following equations:

Categories

For pedagogical purposes, size issues are ignored

Definition
A category C consists in the following pieces of data:

- a set $\mathrm{Ob}(C)$ of objects,
- for every $x, y \in \mathrm{Ob}(C)$, a set $C(x, y)$ of morphisms from x to y, we may write $f : x \to y$ for $f \in C(x, y)$

Satisfying the following equations:

Categories

For pedagogical purposes, size issues are ignored

Definition
A category C consists in the following pieces of data:

- a set $\mathrm{Ob}(C)$ of objects,
- for every $x, y \in \mathrm{Ob}(C)$, a set $C(x, y)$ of morphisms from x to y, we may write $f : x \to y$ for $f \in C(x, y)$
- for every $x \in \mathrm{Ob}(C)$, a morphism $\mathrm{id}_x \in C(x, x)$,

Satisfying the following equations:

Categories

For pedagogical purposes, size issues are ignored

Definition
A category C consists in the following pieces of data:

- a set $\mathrm{Ob}(C)$ of objects,
- for every $x, y \in \mathrm{Ob}(C)$, a set $C(x, y)$ of morphisms from x to y,
  we may write $f : x \to y$ for $f \in C(x, y)$
- for every $x \in \mathrm{Ob}(C)$, a morphism $\mathrm{id}_x \in C(x, x)$,
- for every $x, y, z \in \mathrm{Ob}(C)$, a composition operation $\circ : C(y, z) \times C(x, y) \to C(x, z)$

Satisfying the following equations:

Categories

For pedagogical purposes, size issues are ignored

Definition
A category C consists in the following pieces of data:

- a set $\mathrm{Ob}(C)$ of objects,
- for every $x, y \in \mathrm{Ob}(C)$, a set $C(x, y)$ of morphisms from x to y,
  we may write $f : x \to y$ for $f \in C(x, y)$
- for every $x \in \mathrm{Ob}(C)$, a morphism $\mathrm{id}_x \in C(x, x)$,
- for every $x, y, z \in \mathrm{Ob}(C)$, a composition operation $\circ : C(y, z) \times C(x, y) \to C(x, z)$

Satisfying the following equations:

- for every $x, y \in \mathrm{Ob}(C)$, and $f \in C(x, y)$, $\mathrm{id}_y \circ f = f = f \circ \mathrm{id}_x$

Categories

For pedagogical purposes, size issues are ignored

Definition
A category C consists in the following pieces of data:

- a set $\mathrm{Ob}(C)$ of objects,
- for every $x, y \in \mathrm{Ob}(C)$, a set $C(x, y)$ of morphisms from x to y, we may write $f : x \to y$ for $f \in C(x, y)$
- for every $x \in \mathrm{Ob}(C)$, a morphism $\mathrm{id}_x \in C(x, x)$,
- for every $x, y, z \in \mathrm{Ob}(C)$, a composition operation $\circ : C(y, z) \times C(x, y) \to C(x, z)$

Satisfying the following equations:

- for every $x, y \in \mathrm{Ob}(C)$, and $f \in C(x, y)$, $\mathrm{id}_y \circ f = f = f \circ \mathrm{id}_x$
- for every $x, y, z, w \in \mathrm{Ob}(C)$, and $f \in C(x, y), g \in C(y, z), h \in C(z, w)$, $h \circ (g \circ f) = (h \circ g) \circ f$

Common categories

There are plenty of categories in the wild, here are some examples

It is a good exercise to take time to understand why each of them is indeed a category

- Set : objects are sets, and morphisms are functions

Common categories

There are plenty of categories in the wild, here are some examples

It is a good exercise to take time to understand why each of them is indeed a category

- Set : objects are sets, and morphisms are functions
- Grp : objects are groups, morphisms are group homomorphisms

Common categories

There are plenty of categories in the wild, here are some examples

It is a good exercise to take time to understand why each of them is indeed a category

- Set : objects are sets, and morphisms are functions
- Grp : objects are groups, morphisms are group homomorphisms
- $\mathrm{Vect}_{\mathbb{R}}$: objects are $\mathbb{R}$-vector spaces, morphisms are linear applications

Common categories

There are plenty of categories in the wild, here are some examples

It is a good exercise to take time to understand why each of them is indeed a category

- Set : objects are sets, and morphisms are functions
- Grp : objects are groups, morphisms are group homomorphisms
- $\mathrm{Vect}_{\mathbb{R}}$: objects are $\mathbb{R}$-vector spaces, morphisms are linear applications
- Top : objects are topological spaces, morphisms are continuous maps

Common categories

There are plenty of categories in the wild, here are some examples

It is a good exercise to take time to understand why each of them is indeed a category

- Set : objects are sets, and morphisms are functions
- Grp : objects are groups, morphisms are group homomorphisms
- $\mathrm{Vect}_{\mathbb{R}}$: objects are $\mathbb{R}$-vector spaces, morphisms are linear applications
- Top : objects are topological spaces, morphisms are continuous maps
- Mon : objects are monoids, morphisms are monoid morphisms

Common categories

There are plenty of categories in the wild, here are some examples

It is a good exercise to take time to understand why each of them is indeed a category

- Set : objects are sets, and morphisms are functions
- Grp : objects are groups, morphisms are group homomorphisms
- $\mathrm{Vect}_{\mathbb{R}}$: objects are $\mathbb{R}$-vector spaces, morphisms are linear applications
- Top : objects are topological spaces, morphisms are continuous maps
- Mon : objects are monoids, morphisms are monoid morphisms
- Given a graph G, the $\mathrm{Path}(G)$, where objects are vertices of G, and morphisms are paths in G

Functors

Definition
A fucntor $F\colon C \to D$ from a category C to a category D consists in the following pieces of data:

- a function $F_0\colon \mathrm{Ob}(C) \to \mathrm{Ob}(D)$,

satisfying the following equations:

Functors

Definition
A fucntor $F\colon C \to D$ from a category C to a category D consists in the following pieces of data:

- a function $F_0\colon \mathrm{Ob}(C) \to \mathrm{Ob}(D)$,
- for every $x, y \in \mathrm{Ob}(C)$, a function $F_{x,y}\colon C(x,y) \to D(F_0 x, F_0 y)$

satisfying the following equations:

Functors

Definition
A fucntor $F\colon C \to D$ from a category C to a category D consists in the following pieces of data:

- a function $F_0\colon \mathrm{Ob}(C) \to \mathrm{Ob}(D)$,
- for every $x, y \in \mathrm{Ob}(C)$, a function $F_{x,y}\colon C(x,y) \to D(F_0x, F_0y)$

satisfying the following equations:

- for every $x \in \mathrm{Ob}(C), F_{x,x}(\mathrm{id}_x) = \mathrm{id}_{F_0x}$

Functors

Definition
A fucntor $F\colon C \to D$ from a category C to a category D consists in the following pieces of data:

- a function $F_0\colon \mathrm{Ob}(C) \to \mathrm{Ob}(D)$,
- for every $x, y \in \mathrm{Ob}(C)$, a function $F_{x,y}\colon C(x,y) \to D(F_0x, F_0y)$

satisfying the following equations:

- for every $x \in \mathrm{Ob}(C), F_{x,x}(\mathrm{id}_x) = \mathrm{id}_{F_0x}$
- for every $x, y, z \in \mathrm{Ob}(C)$, and $f \in C(x,y), g \in C(y,z)$
  $F_{x,z}(g \circ f) = F_{y,z}(g) \circ F_{x,y}(f)$

Definition
A fucntor $F\colon C \to D$ from a category C to a category D consists in the following pieces of data:

- a function $F_0\colon \mathrm{Ob}(C) \to \mathrm{Ob}(D)$,
- for every $x, y \in \mathrm{Ob}(C)$, a function $F_{x,y}\colon C(x,y) \to D(F_0x, F_0y)$

satisfying the following equations:

- for every $x \in \mathrm{Ob}(C), F_{x,x}(\mathrm{id}_x) = \mathrm{id}_{F_0x}$
- for every $x, y, z \in \mathrm{Ob}(C)$, and $f \in C(x,y), g \in C(y,z)$
  $F_{x,z}(g \circ f) = F_{y,z}(g) \circ F_{x,y}(f)$

In practice, we abuse notations and write $F(x) = F_0(x)$ for objects, and $F(f) = F_{x,y}(f)$ for morphisms.

Examples: forgetful functors

Here are some example of "forgetful functors" between the categories we've seen before

- $\mathsf{U}\colon \mathsf{Grp} \to \mathsf{Set}$ associates to every group its underlying set (its "carrier"), and every group homomorphism its underlying function.

Examples: forgetful functors

Here are some example of "forgetful functors" between the categories we've seen before

- $U\colon \mathsf{Grp} \to \mathsf{Set}$ associates to every group its underlying set (its "carrier"), and every group homomorphism its underlying function.

- $U\colon \mathsf{Grp} \to \mathsf{Mon}$ associates to every group itself seen as a monoid, and every group homorophism the same function seen as a monoid morphism.

Examples: forgetful functors

Here are some example of "forgetful functors" between the categories we've seen before

- $\mathsf{U}\colon \mathsf{Grp} \to \mathsf{Set}$ associates to every group its underlying set (its "carrier"), and every group homomorphism its underlying function.
- $\mathsf{U}\colon \mathsf{Grp} \to \mathsf{Mon}$ associates to every group itself seen as a monoid, and every group homorophism the same function seen as a monoid morphism.
- $\mathsf{U}\colon \mathsf{Top} \to \mathsf{Set}$ associates to every topological space its underlying set of points, and to every continuous map the underlying function.

Example: the free monoid aka list functor

Definition
Given a set X, the free monoid $X^\star$ on X is the set of lists of elements of X. It is a monoid, thanks to list concatenation and the empty list.

Example: the free monoid aka list functor

Definition
Given a set X, the free monoid $X^\star$ on X is the set of lists of elements of X. It is a monoid, thanks to list concatenation and the empty list.

This gives rise to a "free monoid functor" $(-)^\star \colon \mathrm{Set} \to \mathrm{Mon}$, which associates to every set X the set $X^\star$.

Example: the free monoid aka list functor

Definition
Given a set X, the free monoid $X^\star$ on X is the set of lists of elements of X. It is a monoid, thanks to list concatenation and the empty list.

This gives rise to a "free monoid functor" $(-)^\star \colon \mathrm{Set} \to \mathrm{Mon}$, which associates to every set X the set $X^\star$.

It is a good exercise to try and figure what it does on morphisms

The category of category

There is a category Cat whose objects are categories, and morphisms are functors

The category of category

There is a category Cat whose objects are categories, and morphisms are functors

It is a good idea to verify this, it requires:

- defining an identity functor
- defining a composition of functors
- showing unitality
- showing associativity

A primer on category theory

Products

Products in a category

Definition
Given a category C and two objects $x, y \in \mathrm{Ob}(C)$, a product of x, y the data of:

- a triple (z, π_x, π_y), where z is an object and $\pi_x \colon z \to x$ and $\pi_y \colon z \to y$,

satisfying the following property:

Products in a category

Definition
Given a category C and two objects $x, y \in \mathrm{Ob}(C)$, a product of x, y the data of:

- a triple (z, π_x, π_y), where z is an object and $\pi_x \colon z \to x$ and $\pi_y \colon z \to y$,

satisfying the following property:

- for any triple (w, ρ_x, ρ_y) where w is an object, and $\rho_x \colon w \to x$ and $\rho_y \colon w \to y$, there exists a unique morphism $\xi : w \to z$ such that $\rho_x = \xi \circ \pi_x$ and $\rho_y = \xi \circ \pi_y$

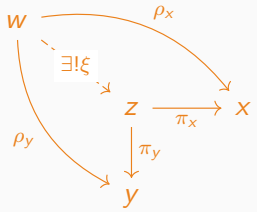
Products in a category

Definition
Given a category C and two objects $x, y \in \mathrm{Ob}(C)$, a product of x, y the data of:

- a triple (z, π_x, π_y), where z is an object and $\pi_x \colon z \to x$ and $\pi_y \colon z \to y$,

satisfying the following property:

- for any triple (w, ρ_x, ρ_y) where w is an object, and $\rho_x \colon w \to x$ and $\rho_y \colon w \to y$, there exists a unique morphism $\xi : w \to z$ such that $\rho_x = \xi \circ \pi_x$ and $\rho_y = \xi \circ \pi_y$



Uniqueness products

If a product of x, y exists, then it is unique up to isomorphism (i.e, morphisms back and forth that compose to an identity both ways)

We abusively then talk about "the" product of x and y, and denote it $x \times y$

Example of products

In each case, it is a very good exercise to verify the claims

- In Set, all products exist, and the product of X and Y is given by the cartesian product $X \times Y$

Example of products

In each case, it is a very good exercise to verify the claims

- In Set, all products exist, and the product of X and Y is given by the cartesian product $X \times Y$
- In Grp, all products exist, and the product of G and H is the direct product $G \times H$

Example of products

In each case, it is a very good exercise to verify the claims

- In Set, all products exist, and the product of X and Y is given by the cartesian product $X \times Y$
- In Grp, all products exist, and the product of G and H is the direct product $G \times H$
- In $\mathrm{Vect}_{\mathbb{R}}$, all products exist and the product of U and V is the direct sum $U \oplus V$

Product preserving functor

Definition
Given two categories C, D with products, a functor $F\colon C \to D$ is said to be product-preserving when for every $x, y \in \mathrm{Ob}(C)$, denoting $(x \times y, \pi_x, \pi_y)$ the product of x, y in C, $(F(x \times y), F(\pi_x), F(\pi_y))$ is the product of $F(x)$ and $F(y)$ in D.

We sometimes abusively write $F(x \times y) = F(x) \times F(y)$, but remember that it must also preserve the projection morphisms π.

A primer on category theory

Natural transformations, covariant presheaves, and the Yoneda lemma

Beyond functors: natural transformations

Definition
Given two categories C, D, and two functors $F, G \colon C \to D$, a natural transformation $\alpha \colon F \Rightarrow G$ consists in the following data:

- For every $x \in \mathrm{Ob}(C)$, a morphism $\alpha_x \colon Fx \to Gx$ in D

Satisfying the following equation:

Beyond functors: natural transformations

Definition
Given two categories C, D, and two functors $F, G \colon C \to D$, a natural transformation $\alpha \colon F \Rightarrow G$ consists in the following data:

- For every $x \in \mathrm{Ob}(C)$, a morphism $\alpha_x \colon Fx \to Gx$ in D

Satisfying the following equation:

- For every $x, y \in \mathrm{Ob}(C)$ and $f \in C(x, y)$, $\alpha_y \circ F(f) = G(f) \circ \alpha_x$

Beyond functors: natural transformations

Definition
Given two categories C, D, and two functors $F, G \colon C \to D$, a natural transformation $\alpha \colon F \Rightarrow G$ consists in the following data:

- For every $x \in \mathrm{Ob}(C)$, a morphism $\alpha_x \colon Fx \to Gx$ in D

Satisfying the following equation:

- For every $x, y \in \mathrm{Ob}(C)$ and $f \in C(x, y)$, $\alpha_y \circ F(f) = G(f) \circ \alpha_x$

$$\begin{array}{ccc} Fx & \xrightarrow{\alpha_x} & Gx \\ {\scriptstyle F(f)}\downarrow & & \downarrow{\scriptstyle G(f)} \\ Fy & \xrightarrow[\alpha_y]{} & Gy \end{array}$$

Beyond functors: natural transformations

Definition
Given two categories C, D, and two functors $F, G \colon C \to D$, a natural transformation $\alpha \colon F \Rightarrow G$ consists in the following data:

- For every $x \in \mathrm{Ob}(C)$, a morphism $\alpha_x \colon Fx \to Gx$ in D

Satisfying the following equation:

- For every $x, y \in \mathrm{Ob}(C)$ and $f \in C(x, y)$, $\alpha_y \circ F(f) = G(f) \circ \alpha_x$

$$\begin{array}{ccc} Fx & \xrightarrow{\alpha_x} & Gx \\ {\scriptstyle F(f)}\downarrow & & \downarrow{\scriptstyle G(f)} \\ Fy & \xrightarrow[\alpha_y]{} & Gy \end{array}$$

Denote $\mathrm{Nat}(F, G)$ the set of natural transformations between F and G.

The category of covariant presheaves

Given a category C, we associate the category $\mathrm{Psh}(C)$ whose objects are functors $C \to \mathrm{Set}$, and morphisms are natural transformations between these functors

The category of covariant presheaves

Given a category C, we associate the category $\mathrm{Psh}(C)$ whose objects are functors $C \to \mathrm{Set}$, and morphisms are natural transformations between these functors

Again, it is a good idea to make sense of this, it requires:

- defining an identity natural transformation
- defining composition of natural transformations
- showing unitality
- showing associativity

Representable presheaves

Definition
Given a category C and an object $x \in \mathrm{Ob}(C)$, we define the functor $\mathrm{Yon}(x) = C(x, -) \colon C \to \mathrm{Set}$, associating :

- to an object y the set $C(x, y)$
- to a function $g \colon y \to z$, the function $C(x, y) \to C(x, z)$ defined by $f \mapsto g \circ f$

It is a good exercise to check that it is a functor

Representable presheaves

Definition
Given a category C and an object $x \in \mathrm{Ob}(C)$, we define the functor $\mathrm{Yon}(x) = C(x, -) \colon C \to \mathrm{Set}$, associating :

- to an object y the set $C(x, y)$
- to a function $g \colon y \to z$, the function $C(x, y) \to C(x, z)$ defined by $f \mapsto g \circ f$

It is a good exercise to check that it is a functor

Theorem
Given a category C and an object $x \in \mathrm{Ob}(C)$, the functor $\mathrm{Yon}(x)$ is product-preserving

Can you see why? You should use the definition of product, and the explicit computation of products in the category of sets.

(covariant) Yoneda lemma

Theorem ((covariant) Yoneda Lemma)
Given a category C, an object x of C, and a functor $F : C \to \mathrm{Set}$, there is a natural equivalence

$$\mathrm{Nat}(\mathrm{Yon}(x), F) \simeq F(x)$$

You will be proving this as part of the exercises

Yoneda embedding

- Given a category C, there is a category C^{op} that has the same objects as C, and with $C^{\mathrm{op}}(x, y) = C(y, x)$
  It is a good exercise to verify that this defines a category

Yoneda embedding

- Given a category C, there is a category C^{op} that has the same objects as C, and with $C^{\mathrm{op}}(x, y) = C(y, x)$
  It is a good exercise to verify that this defines a category

- There is a Yoneda embedding functor $\mathrm{Yon}\colon C^{\mathrm{op}} \to \mathrm{Psh}(C)$
  It is a good exercise to understand the action on morphisms

Yoneda embedding

- Given a category C, there is a category C^{op} that has the same objects as C, and with $C^{\mathrm{op}}(x, y) = C(y, x)$
  It is a good exercise to verify that this defines a category

- There is a Yoneda embedding functor $\mathrm{Yon}\colon C^{\mathrm{op}} \to \mathrm{Psh}(C)$
  It is a good exercise to understand the action on morphisms

- Consequence of Yoneda lemma: this functor is fully faithful, i.e., it induces bijections

$$C^{\mathrm{op}}(x, y) \simeq \mathrm{Psh}(C)(\mathrm{Yon}(x), \mathrm{Yon}(y))$$

Example: the category of (multi-)graphs

Definition
A multigraph X is a pair of sets X_0 (vertices) and X_1 (edges) together with two functions $s, t \colon X_1 \to X_0$ (source and target). A morphism of multigraphs $X \to Y$ is a pair of functions $X_0 \to Y_0$ and $X_1 \to Y_1$ that commute with source and targets.

Example: the category of (multi-)graphs

Definition
A multigraph X is a pair of sets X_0 (vertices) and X_1 (edges) together with two functions $s, t \colon X_1 \to X_0$ (source and target). A morphism of multigraphs $X \to Y$ is a pair of functions $X_0 \to Y_0$ and $X_1 \to Y_1$ that commute with source and targets.

The category whose objects are multigraphs and whose morphisms are morphisms of multigraphs is the category $\mathrm{Psh}(\mathbb{G})$ for

$$\mathbb{G} \quad : \quad 0 \underset{\tau}{\overset{\sigma}{\leftleftarrows}} t$$

It is a good exercise to verify this.

Representable multigraphs

One can check that the multigraphs Yon(0) and Yon(1) are the following

$$\mathrm{Yon}(0) \quad : \quad \bullet \qquad\qquad \mathrm{Yon}(1) \quad : \quad \bullet \to \bullet$$

It is a good exercise to verify this yourself

Representable multigraphs

One can check that the multigraphs Yon(0) and Yon(1) are the following

$$\text{Yon}(0) \quad : \quad \bullet \qquad\qquad \text{Yon}(1) \quad : \quad \bullet \to \bullet$$

It is a good exercise to verify this yourself

In this example, the Yoneda lemma states that a vertices in a multigraph are in bijection with morphisms from Yon(0) to this multigraph, and edges are in bijection with morphisms from Yon(1) to this multigraph

Make sure you take time to understand this.

The Yoneda embedding for multigraphs

Since $\mathrm{Yon}\colon \mathbb{G}^{\mathrm{op}} \to \mathrm{Psh}(\mathbb{G})$ is fully faithful, there are exactly two morphisms:

$$\mathrm{Yon}(0) \underset{\tau}{\overset{\sigma}{\rightrightarrows}} \mathrm{Yon}(1)$$

The Yoneda embedding for multigraphs

Since $\mathrm{Yon}\colon \mathbb{G}^{\mathrm{op}} \to \mathrm{Psh}(\mathbb{G})$ is fully faithful, there are exactly two morphisms:

$$\mathrm{Yon}(0) \underset{\tau}{\overset{\sigma}{\rightrightarrows}} \mathrm{Yon}(1)$$

These correspond to the source and target vertex of the unique edge

Lawvere Theories

Lawvere Theories

Algebraic theories : Definitions and models

Signatures

Definition
A signature $\Sigma = (\mathcal{S}, a)$ consists in a set of symbols $\mathcal{S}$ and an arity function $a \colon \mathcal{S} \to \mathbb{N}$

Elements of $\mathcal{S}$ are function symbols, constants are functions of arity 0.

Signatures

Definition
A signature $\Sigma = (\mathcal{S}, a)$ consists in a set of symbols $\mathcal{S}$ and an arity function $a \colon \mathcal{S} \to \mathbb{N}$

Elements of $\mathcal{S}$ are function symbols, constants are functions of arity 0.

Definition
A context Γ is a list of variables, a term in a context Γ over the signature Σ is defined inductively as

- A variable x that appears in Γ
- An $(n+1)$-tuple $(f, u_1, \ldots, u_n)$ with $f \in \mathcal{S}$, and $a(f) = n$, and $u_1, \ldots, u_n$ are terms in Γ.

We write $\Gamma \vdash u$ to say that u is a term in Γ

Theories

Definition
A (single-sorted) algebraic theory $\mathcal{T} = (\Sigma_{\mathcal{T}}, E_{\mathcal{T}})$ consists of a signature $\Sigma_{\mathcal{T}}$ together with a set $E_{\mathcal{T}}$ of triples (Γ, u, v) called equations, where Γ is a context, and u and v are terms in this context.

We suggestively write $\Gamma \vdash u \equiv v \in E_{\mathcal{T}}$ if $(\Gamma, u, v) \in E_{\mathcal{T}}$.

Definition
A (single-sorted) algebraic theory $\mathcal{T} = (\Sigma_\mathcal{T}, E_\mathcal{T})$ consists of a signature $\Sigma_\mathcal{T}$ together with a set $E_\mathcal{T}$ of triples (Γ, u, v) called equations, where Γ is a context, and u and v are terms in this context.

We suggestively write $\Gamma \vdash u \equiv v \in E_\mathcal{T}$ if $(\Gamma, u, v) \in E_\mathcal{T}$.

Example (The theory of monoids $\mathcal{T}_{\mathsf{Mon}}$)

- Signature: e or arity 0, $- * -$ of arity 2.
- Equations:

$$x \vdash x * \mathsf{e} \equiv x \qquad x \vdash \mathsf{e} * x \equiv x \qquad x, y, z \vdash x * (y * z) \equiv (x * y) * z$$

Models of a theory

Definition
A model of an algebraic theory $\mathcal{T}$ consists in a set X together with, for every symbol f of arity n in $\Sigma_{\mathcal{T}}$, a function

$$[\![f]\!] \colon X^n \to X$$

such that all the equations in $E_{\mathcal{T}}$ are satisfied, where in an equation $\Gamma \vdash u \equiv v$, variables of Γ are universally quantified.

Models of a theory

Definition
A model of an algebraic theory $\mathcal{T}$ consists in a set X together with, for every symbol f of arity n in $\Sigma_{\mathcal{T}}$, a function

$$[\![f]\!] \colon X^n \to X$$

such that all the equations in $E_{\mathcal{T}}$ are satisfied, where in an equation $\Gamma \vdash u \equiv v$, variables of Γ are universally quantified.

Example
A model of the theory $\mathcal{T}_{\mathsf{Mon}}$ of monoids is...

Models of a theory

Definition
A model of an algebraic theory $\mathcal{T}$ consists in a set X together with, for every symbol f of arity n in $\Sigma_{\mathcal{T}}$, a function

$$[\![f]\!] : X^n \to X$$

such that all the equations in $E_{\mathcal{T}}$ are satisfied, where in an equation $\Gamma \vdash u \equiv v$, variables of Γ are universally quantified.

Example
A model of the theory $\mathcal{T}_{\mathsf{Mon}}$ of monoids is... A monoid !

About variable names

- Variable names are annoying to deal with
  Technically, terms should be quotiented up to α-equivalence

About variable names

- Variable names are annoying to deal with
  Technically, terms should be quotiented up to α-equivalence

- From now on, we use De Bruijn indices
  Thus, a context is just a natural number, we denote v_i the i^{th} De Bruijn index

About variable names

- Variable names are annoying to deal with
  Technically, terms should be quotiented up to α-equivalence

- From now on, we use De Bruijn indices
  Thus, a context is just a natural number, we denote v_i the i^{th} De Bruijn index

- For the sake of readability in the examples, we still use variable names
  Unless you want to look at things like

$$3 \vdash v_2 * (v_1 * v_0) \equiv v_2 * (v_1 * v_0)$$

Syntactic category of a theory

- For a theory $\mathcal{T}$, we denote $n \vdash u$ the equivalence class of the term u under the relation $n \vdash u \equiv v$.
- A substitution $m \vdash \sigma : n$ is an n-tuple of terms $\langle u_1, \ldots, u_n \rangle$ such that $m \vdash u_i$ for all i. We denote $v_i[\sigma] = u_i$.

Definition
The syntactic category $\mathrm{Syn}(\mathcal{T})$ of a theory $\mathcal{T}$ is the category whose objects are contexts, and whose morphisms $\sigma \colon \Gamma \to \Delta$ are substitutions $\Gamma \vdash \sigma : \Delta$.

Wait, is that even a category?

It is not immediately clear that the definition of the syntectic category forms indeed a category.

What are identity morphisms? How do morphisms compose?

You will be checking this as an exercise.

Lawvere Theories

Algebraic theories as type theories

Simple type theories with a unique type

- Before delving into the categorical description, we connect things to type theories
  We have already used suggestive notations in this direction

Simple type theories with a unique type

- Before delving into the categorical description, we connect things to type theories
  We have already used suggestive notations in this direction

- A single-sorted algebraic theory is a simple type theory with a unique type
  A very simple one: no λ-terms or function types

Simple type theories with a unique type

- Before delving into the categorical description, we connect things to type theories
  We have already used suggestive notations in this direction
- A single-sorted algebraic theory is a simple type theory with a unique type
  A very simple one: no λ-terms or function types
- $x, y, z \vdash u \qquad \rightsquigarrow \qquad (x : \star), (y : \star), (z : \star) \vdash u : \star$
  Function symbols $\leftrightarrow$ term constructors

The type theory of monoids

$$\frac{\Gamma \vdash}{\Gamma \vdash \star \ \mathsf{Type}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash e : \star} \qquad \frac{\Gamma \vdash u : \star \qquad \Gamma \vdash v : \star}{\Gamma \vdash u * v : \star}$$

$$\frac{\Gamma \vdash u : \star}{\Gamma \vdash u * e \equiv u} \qquad \frac{\Gamma \vdash u : \star}{\Gamma \vdash e * u \equiv u} \qquad \frac{\Gamma \vdash u : \star \qquad \Gamma \vdash v : \star \qquad \Gamma \vdash w : \star}{\Gamma \vdash u * (v * w) \equiv (u * v) * w}$$

Lawvere Theories

Definition, models

Core idea

An algebraic theory describes a bunch of functions $X^n \to X$ satisfying some equations.

Category theory is well suited to describing bunches of functions

Can we use category theory to describe algebraic theories?

This is what Lawvere theories do. **Idea**: axiomatize how syntactic categories are.

Definition of Lawvere theories

Definition
A Lawever theory $\mathbb{T}$ is a category with objects the natural numbers, and products such that $n \times m = n + m$

LHS = product in $\mathbb{T}$, RHS = addition of natural numbers.

Definition
A model of a Lawvere theory $\mathbb{T}$ is a product-preserving functor $F : \mathbb{T} \to \mathrm{Set}$.

Summary: The role on natural numbers

We have seen natural numbers appear at three places:

- The arities of symbols of an algebraic theory
- The contexts of an algebraic theory
- The objects of a Lawvere theory

Summary: The role on natural numbers

We have seen natural numbers appear at three places:

- The arities of symbols of an algebraic theory
- The contexts of an algebraic theory
- The objects of a Lawvere theory

This is not exactly a coincidence!

Lawvere Theories

From algebraic theories to Lawvere theories

Lawvere theory associated to an algebraic theory

Recall: Given an algebraic theory $\mathcal{T}$, we associate the syntactic category $\mathrm{Syn}(\mathcal{T})$, whose objects are contexts (i.e. natural numbers), and whose morphisms are substitutions.

Theorem
The syntactic category $\mathrm{Syn}(\mathcal{T})$ *is a Lawvere theory.*

You will be proving this as part of your exercises.

Relation with the semantics

The association from algebraic theories to Lawvere theories preserves the semantics. More precisely:

Theorem
Given an algebraic theory $\mathcal{T}$, its models are the same as the models of $\mathrm{Syn}(\mathcal{T})$ *seen as a Lawvere theory.*

You will be proving this as part of your exercises.

Gaining intuition

Suppose that $F\colon \mathbb{T} \to \mathrm{Set}$ is a model of a Lawvere theory $\mathbb{T}$

- Since F preserves products, we have $F(n) = F(1)^n$

Gaining intuition

Suppose that $F \colon \mathbb{T} \to \mathrm{Set}$ is a model of a Lawvere theory $\mathbb{T}$

- Since F preserves products, we have $F(n) = F(1)^n$
- Denote $X = F(1)$, and call it the carrier

Gaining intuition

Suppose that $F\colon \mathbb{T} \to \mathrm{Set}$ is a model of a Lawvere theory $\mathbb{T}$

- Since F preserves products, we have $F(n) = F(1)^n$
- Denote $X = F(1)$, and call it the carrier
- A morphism $n \to 1$ in the Lawvere theory provides a function

$$F(n) = X^n \to X = F(1)$$

This is an n-ary function, built by composing the function symbols of the corresponding theory!

Wait, isn't this too general?

- Lawvere theories allow for morphisms $n \to m$ corresponding in a model to functions

$$X^n \to X^m$$

Wait, isn't this too general?

- Lawvere theories allow for morphisms $n \to m$ corresponding in a model to functions

$$X^n \to X^m$$

- In algebraic theories, we only introduced n-ary function symbols:

$$X^n \to X^1$$

In technical terms, they have arity, but no coarity

Wait, isn't this too general?

- Lawvere theories allow for morphisms $n \to m$ corresponding in a model to functions

$$X^n \to X^m$$

- In algebraic theories, we only introduced n-ary function symbols:

$$X^n \to X^1$$

  In technical terms, they have arity, but no coarity

- It turns out that in a Lawvere theory, morphisms $n \to m$ are completely determined from morphisms $n \to 1$
  Can you see why?

Lawvere Theories

The Lawvere theory of monoids

An abstract description

Definition
the Lawvere theory of monoids is defined as $\mathbb{T}_{\mathsf{Mon}} = \mathsf{Syn}(\mathcal{T}_{\mathsf{Mon}})$.

Unfolding the definition

Let us describe a few morphisms in the category $\mathbb{T}_{\mathsf{Mon}}$:

- Examples of morphisms $2 \to 1$, corresponding to the following substitutions:

$$2 \vdash v_1 : 1 \qquad 2 \vdash v_0 : 1 \qquad 2 \vdash v_1 * v_0 : 1 \qquad 2 \vdash v_0 * v_1 : 1$$

$$2 \vdash v_0 * v_0 * v_1 : 1 \qquad 2 \vdash e : 1 \qquad \ldots$$

Unfolding the definition

Let us describe a few morphisms in the category $\mathbb{T}_{\mathsf{Mon}}$:

- Examples of morphisms $2 \to 1$, corresponding to the following substitutions:

$$2 \vdash v_1 : 1 \qquad 2 \vdash v_0 : 1 \qquad 2 \vdash v_1 * v_0 : 1 \qquad 2 \vdash v_0 * v_1 : 1$$
$$2 \vdash v_0 * v_0 * v_1 : 1 \qquad 2 \vdash e : 1 \qquad \ldots$$

- Examples of morphisms $3 \to 1$

$$3 \vdash v_1 : 1 \qquad 3 \vdash v_2 * v_0 : 1 \qquad 3 \vdash v_2 * v_0 * v_1 : 1$$
$$3 \vdash e : 1 \qquad 3 \vdash v_0 * v_0 : 1 \qquad 3 \vdash v_0 * v_1 * v_0 * v_1 : 1$$

Lawvere Theories

Syntax/Semantics duality

Embedding syntax in semantics

Recall the *Yoneda embedding*

- Start with a Lawvere theory $\mathbb{T}$

Embedding syntax in semantics

Recall the *Yoneda embedding*

- Start with a Lawvere theory $\mathbb{T}$
- We have the functor $\mathrm{Yon}\colon \mathbb{T}^{\mathrm{op}} \to \mathrm{Psh}(\mathbb{T})$

Embedding syntax in semantics

Recall the *Yoneda embedding*

- Start with a Lawvere theory $\mathbb{T}$
- We have the functor $\mathrm{Yon}\colon \mathbb{T}^{\mathrm{op}} \to \mathrm{Psh}(\mathbb{T})$
- Since $\mathrm{Yon}(x)$ preserves products, this restricts to a functor

$$\mathrm{Yon}\colon \mathbb{T}^{\mathrm{op}} \to \mathrm{Mod}(\mathbb{T})$$

Free models

We now look at the image of the functor Yon.

Free models

We now look at the image of the functor Yon.

Theorem (Recall, (covariant) Yoneda Lemma)
Given a category C, an object c of C, and a functor $F : C \to \mathrm{Set}$, there is a natural equivalence

$$\mathrm{Nat}(C(c, -), F) \simeq F(c)$$

Free models

We now look at the image of the functor Yon.

Theorem (Recall, (covariant) Yoneda Lemma)
Given a category C, an object c of C, and a functor $F : C \to \mathrm{Set}$, there is a natural equivalence

$$\mathrm{Nat}(C(c, -), F) \simeq F(c)$$

Application: Given a model $F : \mathbb{T} \to \mathrm{Set}$, we have:

$$\mathrm{Nat}(\mathrm{Yon}(n), F) \simeq F(n) \simeq F(1)^n$$

Reformulation

Given a model F with carrier $X = F(1)$:

- Morphisms of models $\mathrm{Yon}(n) \to F$ are in natural equivalence with tuples X^n

Reformulation

Given a model F with carrier $X = F(1)$:

- Morphisms of models $\mathrm{Yon}(n) \to F$ are in natural equivalence with tuples X^n
  This defines free models with finitely many generators

Reformulation

Given a model F with carrier $X = F(1)$:

- Morphisms of models $\mathrm{Yon}(n) \to F$ are in natural equivalence with tuples X^n
  This defines free models with finitely many generators

- If you know adjunctions: free means left adjoint to the forgetful functor $U\colon \mathrm{Mod}(\mathbb{T}) \to \mathrm{Set}$ mapping F onto $F(1)$.
  Even you do not know adjunctions, it is a good exercise to look them up and taking the time to understand why this is just a reformulation of the previous point.

Duality: the full picture

Theorem (recall, consequence of the Yoneda Lemma)
The Yoneda embedding $\mathsf{Yon}\colon \mathbb{T}^{\mathsf{op}} \to \mathsf{Mod}(\mathbb{T})$ *is fully faithful*

Duality: the full picture

Theorem (recall, consequence of the Yoneda Lemma)
The Yoneda embedding $\mathrm{Yon}\colon \mathbb{T}^{\mathrm{op}} \to \mathrm{Mod}(\mathbb{T})$ *is fully faithful*

- **Consequence**: it induces an equivalence of category onto its essential image

Duality: the full picture

Theorem (recall, consequence of the Yoneda Lemma)
The Yoneda embedding $\mathsf{Yon}\colon \mathbb{T}^{\mathrm{op}} \to \mathsf{Mod}(\mathbb{T})$ *is fully faithful*

- **Consequence**: it induces an equivalence of category onto its essential image
- **Summary**:

$$\mathbb{T}^{\mathrm{op}} \simeq \mathsf{FreeMod}(\mathbb{T}) \hookrightarrow \mathsf{Mod}(\mathbb{T})$$

Duality: the full picture

Theorem (recall, consequence of the Yoneda Lemma)
The Yoneda embedding $\mathrm{Yon}\colon \mathbb{T}^{\mathrm{op}} \to \mathrm{Mod}(\mathbb{T})$ *is fully faithful*

- **Consequence**: it induces an equivalence of category onto its essential image
- **Summary**:

$$\mathbb{T}^{\mathrm{op}} \simeq \mathrm{FreeMod}(\mathbb{T}) \hookrightarrow \mathrm{Mod}(\mathbb{T})$$

- Or, starting with an algberaic theory $\mathcal{T}$,

$$\mathrm{Syn}(\mathcal{T})^{\mathrm{op}} \simeq \mathrm{FreeMod}(\mathcal{T}) \hookrightarrow \mathrm{Mod}(\mathcal{T})$$

Lawvere Theories

Syntax/Semantics duality for monoids

Free monoids!

Theorem
Morphisms of monoids $X^\star \to M$ *are in natural bijection with functions between sets* $X \to M$.

You will be proving this as part of the exercices

Free monoids!

Theorem
Morphisms of monoids $X^\star \to M$ are in natural bijection with functions between sets $X \to M$.

You will be proving this as part of the exercices

Consequence: for the theory of monoids,

$$\mathrm{Yon}(n) \simeq [n]^\star$$

Another description of the Lawvere theory of monoids

- Denote FreeMon the full subcategory of Mon, whose objects are $[n]^\star$
  Objects are the monoids freely generated with finitely many generators

Another description of the Lawvere theory of monoids

- Denote FreeMon the full subcategory of Mon, whose objects are $[n]^\star$
  Objects are the monoids freely generated with finitely many generators

- We have the equivalence of category

$$\mathbb{T} \simeq \mathsf{FreeMon}^{\mathrm{op}}$$

Lawvere Theories

Key take-away

Generators and universally quantified variables

- In an algebraic theory $\mathcal{T}$, equations are universally quantified
  For instance, in the theory of monoids,

$$x, y, z \vdash x * (y * z) \equiv (x * y) * z$$

Given a monoid M, $\forall x, y, z \in M, x * (y * z) = (x * y) * z$

Generators and universally quantified variables

- In an algebraic theory $\mathcal{T}$, equations are universally quantified
  For instance, in the theory of monoids,

$$x, y, z \vdash x * (y * z) \equiv (x * y) * z$$

  Given a monoid M, $\forall x, y, z \in M, x * (y * z) = (x * y) * z$

- These equations are completely captured by morphisms $n \to m$ in $\mathrm{Syn}(\mathcal{T})$, which are morphisms between the free models.

Generators and universally quantified variables

- In an algebraic theory $\mathcal{T}$, equations are universally quantified
  For instance, in the theory of monoids,

$$x, y, z \vdash x * (y * z) \equiv (x * y) * z$$

  Given a monoid M, $\forall x, y, z \in M, x * (y * z) = (x * y) * z$

- These equations are completely captured by morphisms $n \to m$ in $\mathrm{Syn}(\mathcal{T})$, which are morphisms between the free models.

- Universally quantified variables = generators of a free gadget
  Free models turn reasoning up to universal quantification into computing.

Multisorted algebraic theories and Lawvere theories

λ-calculi and cartesian closed categories

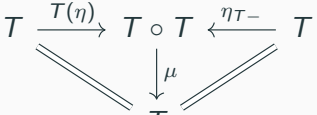
Bonus on Lawvere theories

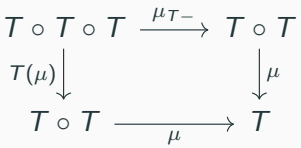
Bonus on Lawvere theories

Monads on sets

Monads are monoids in the category of endofunctors

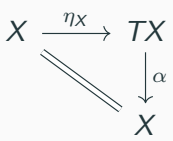
Definition
A monad is a functor $T : \text{Set} \to \text{Set}$, equipped with two natural transformations $\eta : \text{id} \to T$ and $\mu : T \circ T \to T$, such that the following diagrams commute:

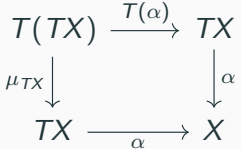




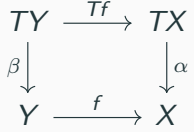
Algebras over a monad

Definition
An algebra for a monad is a set X equipped with a structure map $\alpha : TX \to X$, such that the following diagrams commute:





A morphism of T-algebras $(X, \alpha\colon TX \to X) \to (Y, \beta\colon TY \to Y)$ is a function $f\colon X \to Y$ such that



Define $\mathrm{Alg}(T)$ the category of T-algebras and morphisms of T-algebras

From Lawvere theories to monads

Given a Lawvere theory $\mathbb{T}$, we can define a monad T as follows:

- Given a finite set $[n]$, we take $T[n]$ to be the set of terms in context n

From Lawvere theories to monads

Given a Lawvere theory $\mathbb{T}$, we can define a monad T as follows:

- Given a finite set $[n]$, we take $T[n]$ to be the set of terms in context n
- We extend this construction to arbitrary sets by colimits

From Lawvere theories to monads

Given a Lawvere theory $\mathbb{T}$, we can define a monad T as follows:

- Given a finite set $[n]$, we take $T[n]$ to be the set of terms in context n
- We extend this construction to arbitrary sets by colimits
- The result is a monad
  Good exercise: work out why. The action of substitutions on terms give the μ, and the variables give the η.

Semantics preservation

Theorem
The models of a Lawvere theory are the same as the algebras for the associated monad.

From monads to Lawvere theories

Theorem (Eilenberg-Moore, Street)
Given a monad T, there is a "free-forgetful adjunction"

$$\mathsf{Set} \underset{\mathsf{U}}{\overset{(-)^\star}{\rightleftarrows}} \mathsf{Alg}(T) \qquad \textit{such that } \mathsf{U}((-)^\star) = T$$

Define $\mathsf{FreeAlg}(T)$ the full subcategory of $\mathsf{Alg}(T)$, whose objects are of the form $[n]^\star$

$\mathsf{FreeAlg}(T)^{\mathrm{op}}$ is a Lawvere theory

Semantics preservation

- **Warning**: In general the models of FreeAlg(T) are NOT the algebras of T

Semantics preservation

- **Warning**: In general the models of FreeAlg(T) are NOT the algebras of T
- This is because the monad may define infinitary operation, that are missed by the Lawvere theory

Semantics preservation

- **Warning**: In general the models of FreeAlg(T) are NOT the algebras of T
- This is because the monad may define infinitary operation, that are missed by the Lawvere theory
- For this reason, people introduce finitary monads, those whose endofunctor preserves filtered colimit
  This is a fancy way of saying that no infinitary operation are allowed, by stating that free algebras on infinite sets are completely determined by the free algebras on finite sets

Semantics preservation

- **Warning**: In general the models of FreeAlg(T) are NOT the algebras of T
- This is because the monad may define infinitary operation, that are missed by the Lawvere theory
- For this reason, people introduce finitary monads, those whose endofunctor preserves filtered colimit
  This is a fancy way of saying that no infinitary operation are allowed, by stating that free algebras on infinite sets are completely determined by the free algebras on finite sets

Theorem
Given a finitary monad T, the models of $\mathrm{FreeAlg}(T)^{\mathrm{op}}$ *are exactly the algebras of T.*

Summary

Lawvere theories correspond to finitary monads

Summary

Lawvere theories correspond to finitary monads

- From a Lawvere theory, we get a finitary monad by associating to any set the collection of all terms using elements of the set as variables

Summary

Lawvere theories correspond to finitary monads

- From a Lawvere theory, we get a finitary monad by associating to any set the collection of all terms using elements of the set as variables
- From a finitary monad, we get a Lawvere theory as the opposite of the subcategory of freely finitely generated models.

Summary

Lawvere theories correspond to finitary monads

- From a Lawvere theory, we get a finitary monad by associating to any set the collection of all terms using elements of the set as variables
- From a finitary monad, we get a Lawvere theory as the opposite of the subcategory of freely finitely generated models.

Each time, the algebras of the finitary monad and the models of the Lawvere theory are the same.

Bonus on Lawvere theories

Theories internal to structures

Key idea

- A model of $\mathbb{T}$ is a product-preserving functor
  We chose sets to coincide with model theory, but there is nothing special to the category of sets

Key idea

- A model of $\mathbb{T}$ is a product-preserving functor
  We chose sets to coincide with model theory, but there is nothing special to the category of sets

- Given any category with finite products C, we can define a model of $\mathbb{T}$ internal to C, as a product-preserving functor $\mathbb{T} \to C$.
  This gives a way to interpret theories in other places than the category of sets.

Example

- Take the Lawvere theory of groups $\mathbb{T}_{\mathsf{Grp}}$

Example

- Take the Lawvere theory of groups $\mathbb{T}_{\mathsf{Grp}}$
- Consider the category Top whose objects are topological spaces, and morphisms are continuous functions between them
  Exercise: check that this category has finite products

Example

- Take the Lawvere theory of groups $\mathbb{T}_{\mathsf{Grp}}$
- Consider the category Top whose objects are topological spaces, and morphisms are continuous functions between them
  Exercise: check that this category has finite products
- The models of $\mathbb{T}_{\mathsf{Grp}}$ internal to Top are exactly the topological groups
  Exercise: if you don't know what these are, look it up and convince yourself of this assertion

Bonus on Lawvere theories

Substructural logic