Polynomial functors

Models of type theories

Adrien Mathieu

2025-09-05

Hanabi

Fibrational Models of Linear Dependent Type Theories

Adrien Mathieu

2025-09-05

Dependent type theory

Type theory

Type theory is a family of foundational theories for mathematics, based on the abstract notion of *types*. A *type* is a general form of collection.

Dependent type theory

Type theory

Type theory is a family of foundational theories for mathematics, based on the abstract notion of *types*. A *type* is a general form of collection.

Set theory	Type theory

Dependent type theory

Type theory

Type theory is a family of foundational theories for mathematics, based on the abstract notion of *types*. A *type* is a general form of collection.

Set theory	Type theory
$n \in \mathbb{N}$	$n : \mathbb{N}$

Dependent type theory

Type theory

Type theory is a family of foundational theories for mathematics, based on the abstract notion of *types*. A *type* is a general form of collection.

Set theory	Type theory
$n \in \mathbb{N}$	$n : \mathbb{N}$
π is a proof of P	$\pi : P$

Dependent type theory

Type theory

Type theory is a family of foundational theories for mathematics, based on the abstract notion of *types*. A *type* is a general form of collection.

Set theory	Type theory
$n \in \mathbb{N}$	$n : \mathbb{N}$
π is a proof of P	$\pi : P$
$\mathbb{N}$ is a set	$\mathbb{N} : \mathrm{Type}$

Dependent type theory

Type theory

Type theory is a family of foundational theories for mathematics, based on the abstract notion of *types*. A *type* is a general form of collection.

Set theory	Type theory
$n \in \mathbb{N}$	$n : \mathbb{N}$
π is a proof of P	$\pi : P$
$\mathbb{N}$ is a set	$\mathbb{N} : \mathrm{Type}$
bijection	equivalence

Dependent type theory

Type theory

Type theory is a family of foundational theories for mathematics, based on the abstract notion of *types*. A *type* is a general form of collection.

Set theory	Type theory
$n \in \mathbb{N}$	$n : \mathbb{N}$
π is a proof of P	$\pi : P$
$\mathbb{N}$ is a set	$\mathbb{N} : \mathrm{Type}$
bijection	equivalence

Type theories, and in particular dependent type theories, are much more amenable to proof mechanization.

Dependent type theory

Homotopy Type Theory

Problem:

mathematicians mostly work with partially informal mathematics

Computers do not understand handwaving.

Dependent type theory

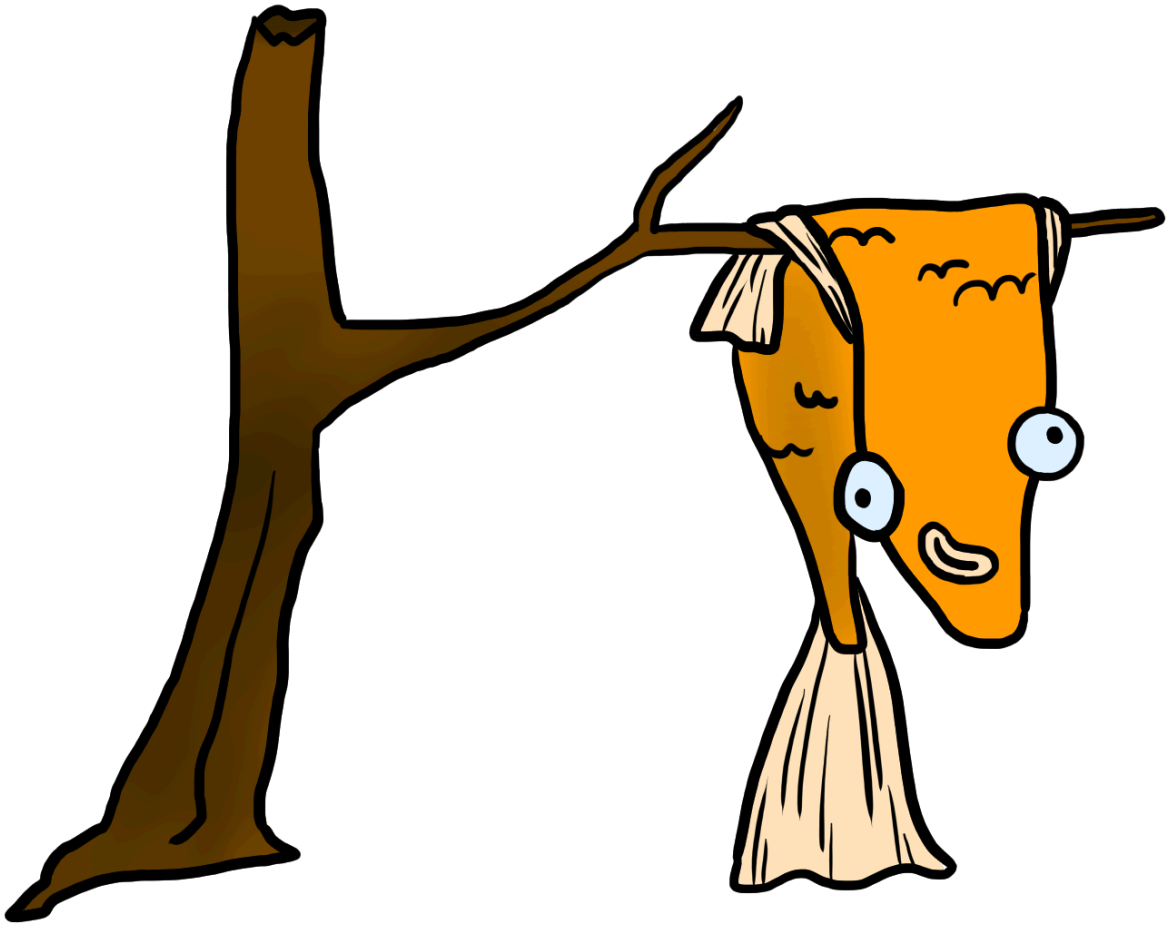
Homotopy Type Theory

Problem:

mathematicians mostly work with partially informal mathematics

Computers do not understand handwaving.

Solution:



Linear logic

Double negation elimination

Unfortunately, type theories build up *intuitionistic* mathematics, in which it is not necessarily true that, for any proposition P, we have

$$P \vee \neg P$$

Linear logic

Double negation elimination

Unfortunately, type theories build up *intuitionistic* mathematics, in which it is not necessarily true that, for any proposition P, we have

$$P \vee \neg P$$

This is a radical change with respect to classical mathematics!

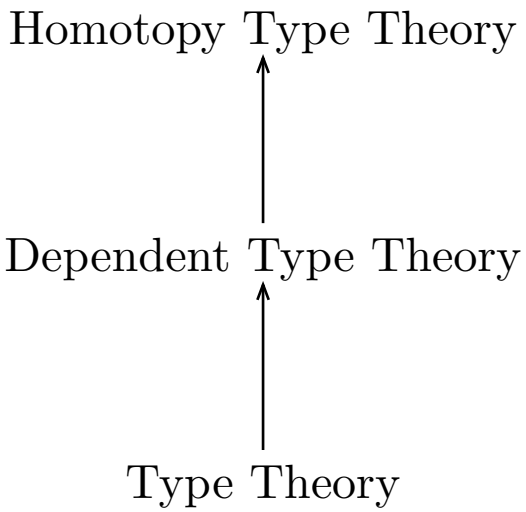
Linear logic

Double negation elimination

Unfortunately, type theories build up *intuitionistic* mathematics, in which it is not necessarily true that, for any proposition P, we have

$$P \vee \neg P$$

This is a radical change with respect to classical mathematics!



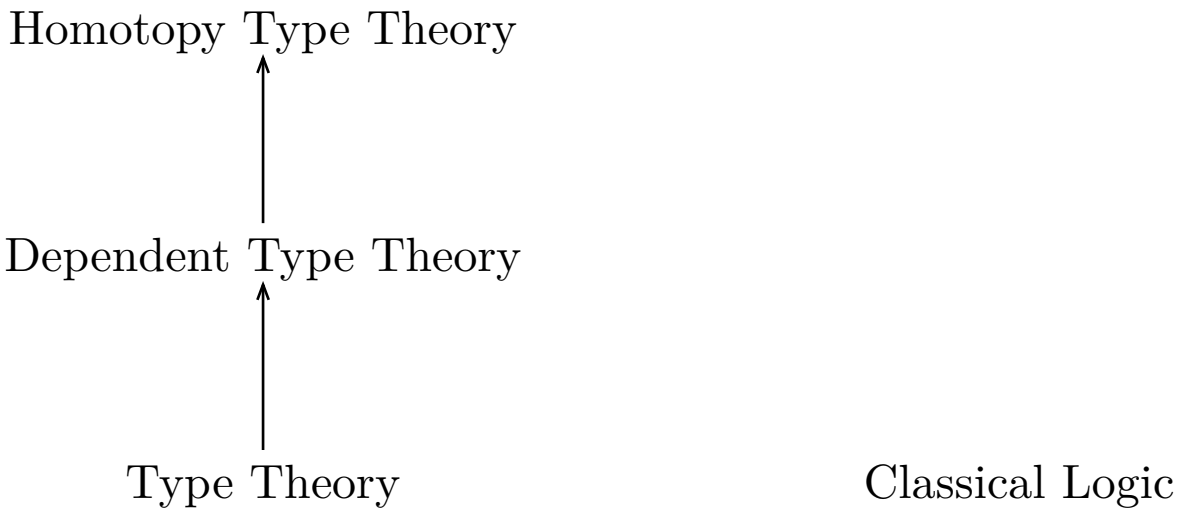
Linear logic

Double negation elimination

Unfortunately, type theories build up *intuitionistic* mathematics, in which it is not necessarily true that, for any proposition P, we have

$$P \vee \neg P$$

This is a radical change with respect to classical mathematics!



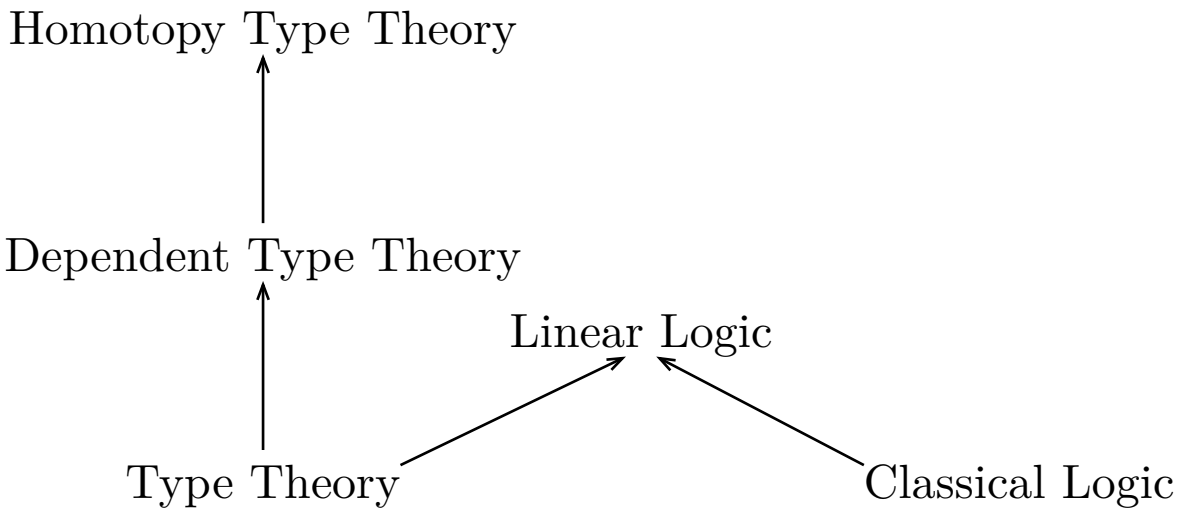
Linear logic

Double negation elimination

Unfortunately, type theories build up *intuitionistic* mathematics, in which it is not necessarily true that, for any proposition P, we have

$$P \vee \neg P$$

This is a radical change with respect to classical mathematics!



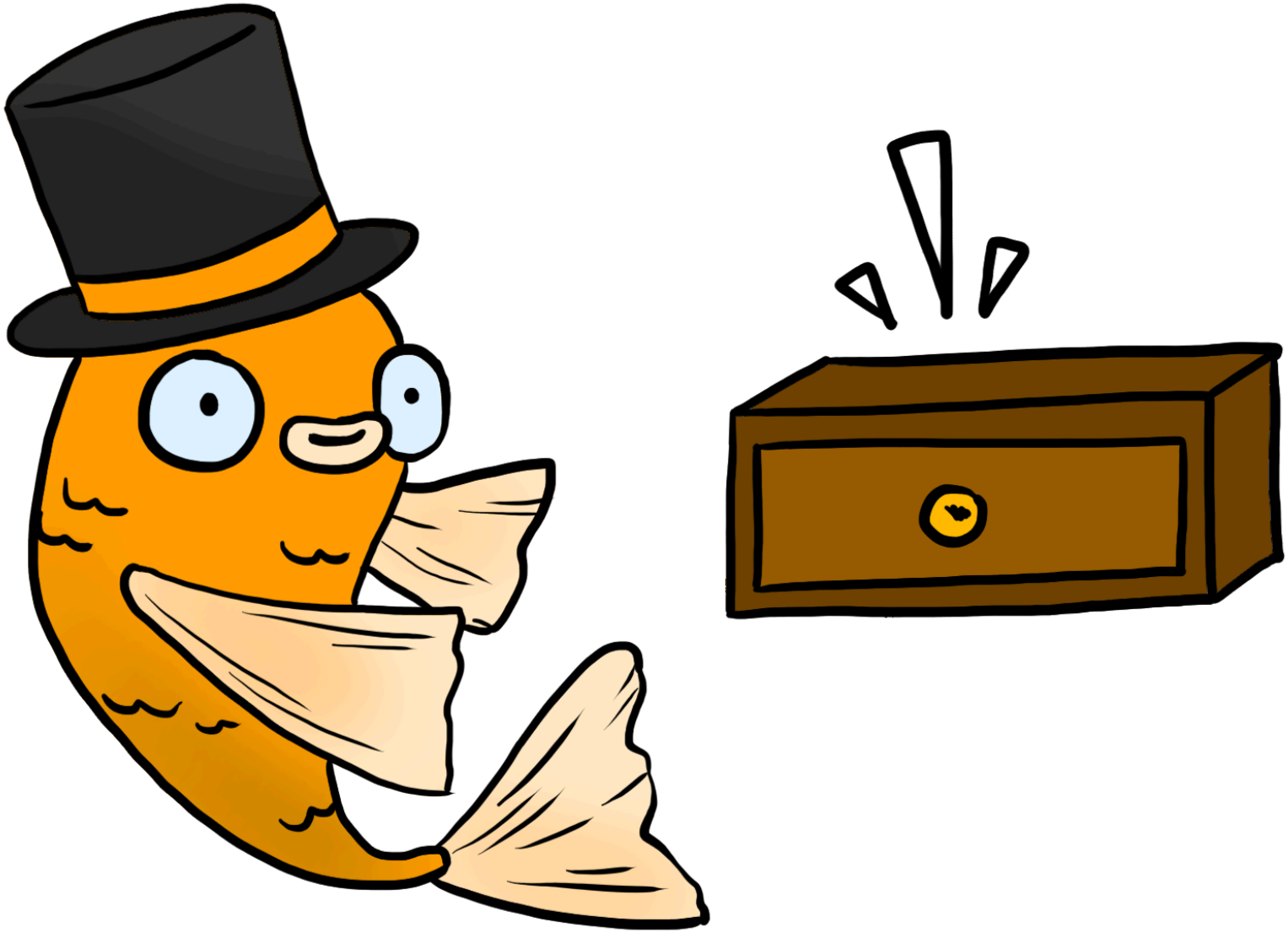
Linear logic

Linear Dependent Type Theory

What we need is a Dependent Linear Type Theory. There are, *a priori*, lots of possibilities to unify DTT and LL. However,

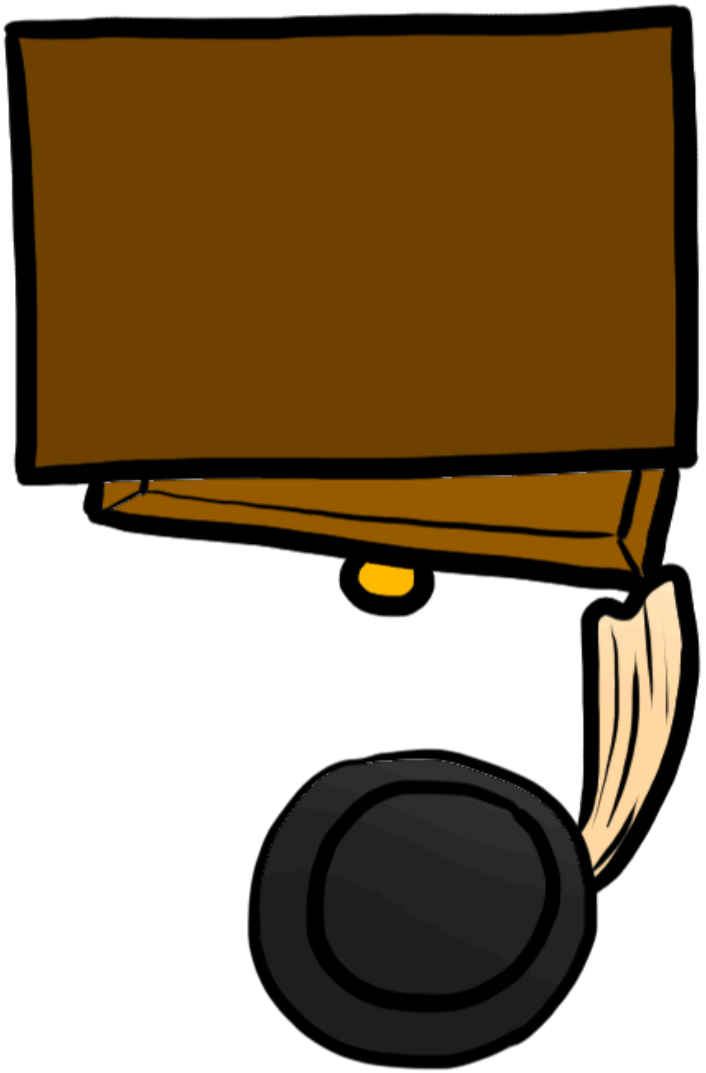
Work during the internship

We have a problem to solve.



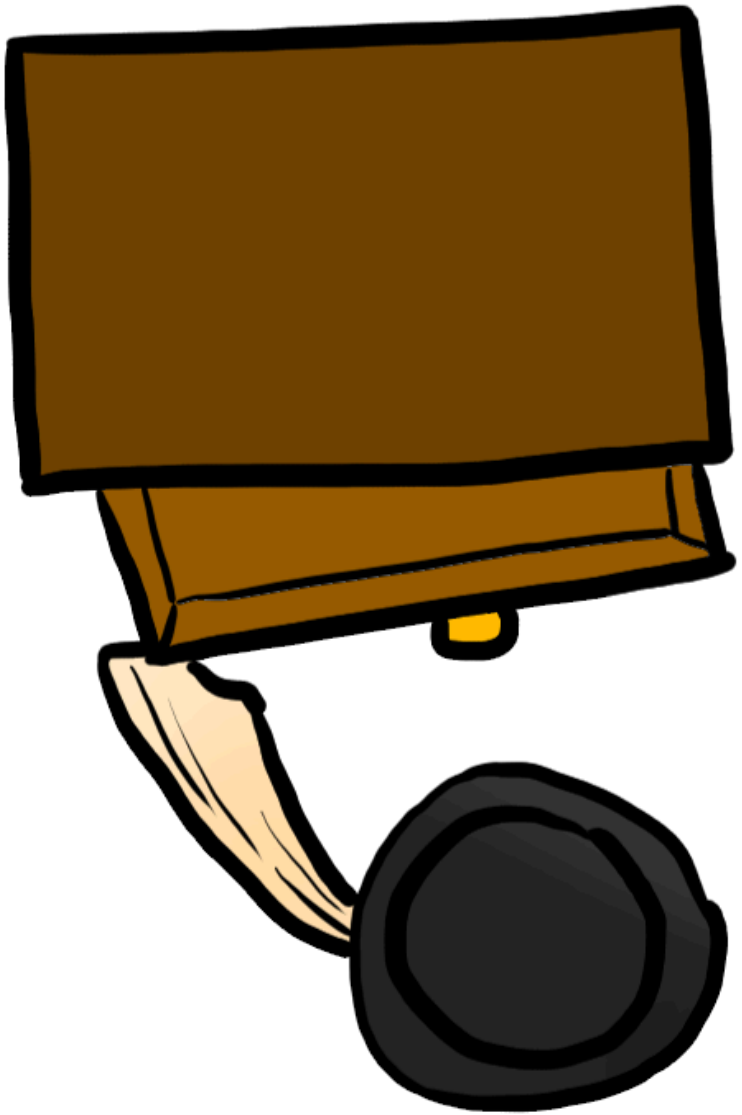
Work during the internship

We try the syntaxic part. It doesn't fully work.



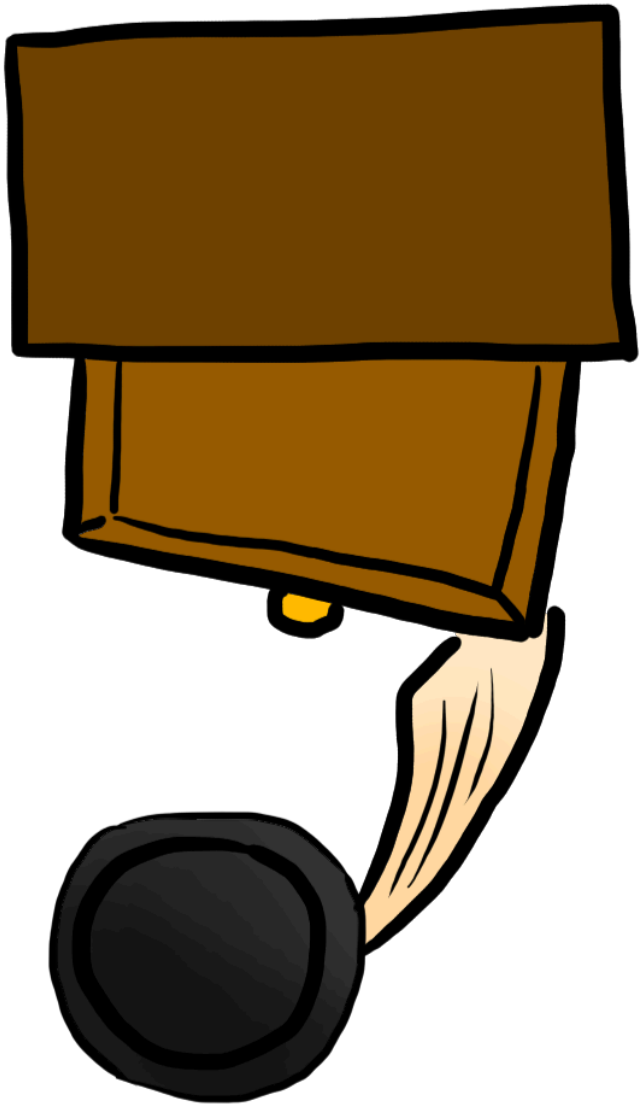
Work during the internship

So we pull from the semantic part. We make some progress.



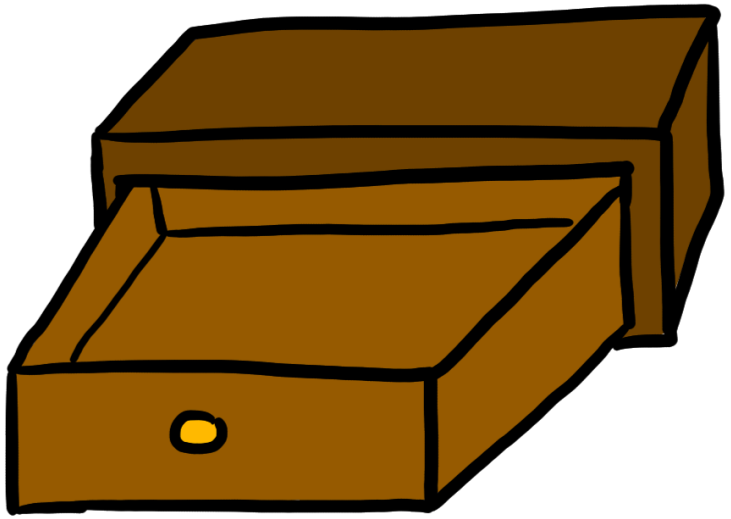
Work during the internship

We go back to the syntactic point of view, and keep pulling.



Work during the internship

The problem is solved.



Category theory reminders

Categories and functors

Definition (Category) A category $\mathcal{C}$ is

- a collection of objects $\mathcal{C}$
- for $X, Y : \mathcal{C}$, a collection of morphisms $\mathcal{C}(X, Y)$
- for any $X : \mathcal{C}$, an *identity at* X: $\mathrm{id}_X : \mathcal{C}(X, X)$
- a composition operation $\circ : \mathcal{C}(Y, Z) \times \mathcal{C}(X, Y) \to \mathcal{C}(X, Z)$
- that is associative: $(f \circ g) \circ h = f \circ (g \circ h)$
- such that $\mathrm{id} \circ f = f \circ \mathrm{id} = f$

Category theory reminders

Categories and functors

Definition (Category) A category $\mathcal{C}$ is

- a collection of objects $\mathcal{C}$
- for $X, Y : \mathcal{C}$, a collection of morphisms $\mathcal{C}(X, Y)$
- for any $X : \mathcal{C}$, an *identity at* X: $\mathrm{id}_X : \mathcal{C}(X, X)$
- a composition operation $\circ : \mathcal{C}(Y, Z) \times \mathcal{C}(X, Y) \to \mathcal{C}(X, Z)$
- that is associative: $(f \circ g) \circ h = f \circ (g \circ h)$
- such that $\mathrm{id} \circ f = f \circ \mathrm{id} = f$

Example. Sets and functions form the category **Set**.

Category theory reminders

Categories and functors

Definition (Category) A category $\mathcal{C}$ is

- a collection of objects $\mathcal{C}$
- for $X, Y : \mathcal{C}$, a collection of morphisms $\mathcal{C}(X, Y)$
- for any $X : \mathcal{C}$, an *identity at* X: $\mathrm{id}_X : \mathcal{C}(X, X)$
- a composition operation $\circ : \mathcal{C}(Y, Z) \times \mathcal{C}(X, Y) \to \mathcal{C}(X, Z)$
- that is associative: $(f \circ g) \circ h = f \circ (g \circ h)$
- such that $\mathrm{id} \circ f = f \circ \mathrm{id} = f$

Example. Sets and functions form the category **Set**.

Definition (Functor) Given two categories $\mathcal{C}$ and $\mathcal{D}$, a functor $F : \mathcal{C} \to \mathcal{D}$ is:

- a map from objets of $\mathcal{C}$ into objects of $\mathcal{D}$
- for $X, Y : \mathcal{C}$, a map $\mathcal{C}(X, Y) \to \mathcal{D}(F(X), F(Y))$
- such that $F(\mathrm{id}_X) = \mathrm{id}_{F(X)}$
- and $F(f \circ g) = F(f) \circ F(g)$

Category theory reminders

Categories and functors

Definition (Category) A category $\mathcal{C}$ is
- a collection of objects $\mathcal{C}$
- for $X, Y : \mathcal{C}$, a collection of morphisms $\mathcal{C}(X, Y)$
- for any $X : \mathcal{C}$, an *identity at* X: $\mathrm{id}_X : \mathcal{C}(X, X)$
- a composition operation $\circ : \mathcal{C}(Y, Z) \times \mathcal{C}(X, Y) \to \mathcal{C}(X, Z)$
- that is associative: $(f \circ g) \circ h = f \circ (g \circ h)$
- such that $\mathrm{id} \circ f = f \circ \mathrm{id} = f$

Example. Sets and functions form the category **Set**.

Definition (Functor) Given two categories $\mathcal{C}$ and $\mathcal{D}$, a functor $F : \mathcal{C} \to \mathcal{D}$ is:
- a map from objets of $\mathcal{C}$ into objects of $\mathcal{D}$
- for $X, Y : \mathcal{C}$, a map $\mathcal{C}(X, Y) \to \mathcal{D}(F(X), F(Y))$
- such that $F(\mathrm{id}_X) = \mathrm{id}_{F(X)}$
- and $F(f \circ g) = F(f) \circ F(g)$

Example. Categories and functors form the category **Cat**.

Category theory reminders

Pullbacks

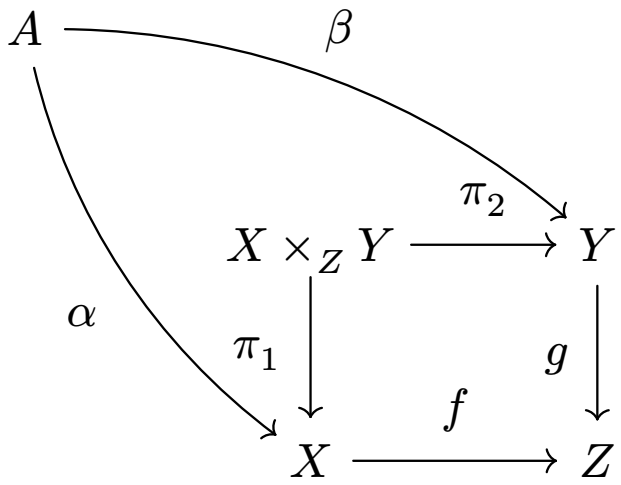
A square as follows

$$\begin{array}{ccc} X \times_Z Y & \xrightarrow{\pi_2} & Y \\ \downarrow{\scriptstyle \pi_1} & & \downarrow{\scriptstyle g} \\ X & \xrightarrow{f} & Z \end{array}$$

Category theory reminders

Pullbacks

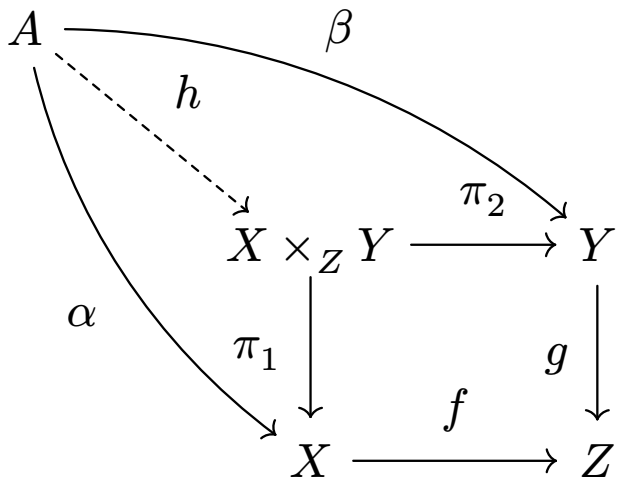
A square as follows



Category theory reminders

Pullbacks

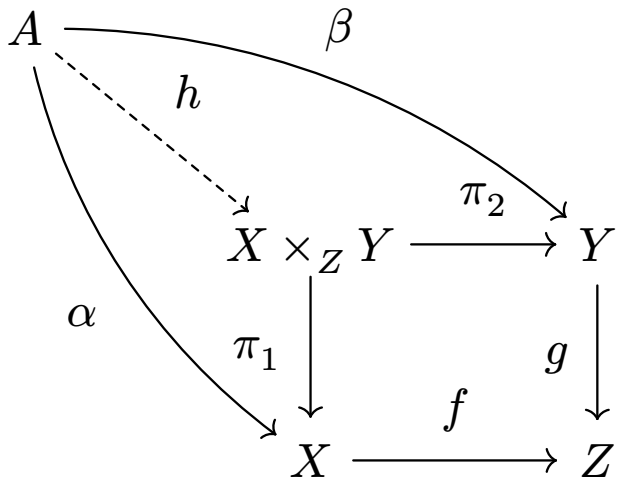
A square as follows



Category theory reminders

Pullbacks

A square as follows



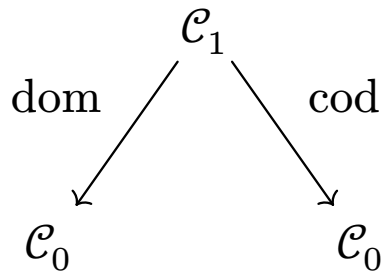
Example. If we are given two functions $f : X \to Z$ and $g : Y \to Z$, their pullback in **Set** is

$$\begin{array}{ccc} \{(x,y); x \in X, y \in Y \mid f(x) = g(y)\} & \xrightarrow{\pi_2} & Y \\ \downarrow{\scriptstyle \pi_1} & & \downarrow{\scriptstyle g} \\ X & \xrightarrow{f} & Z \end{array}$$

Double categories

Category theory, again

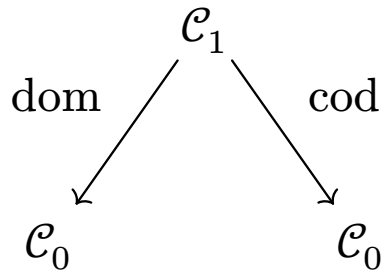
Let's repeat what a category is. A category $\mathcal{C}$ is a collection of objects $\mathcal{C}_0$ and a collection of morphisms $\mathcal{C}_1$, with two functions



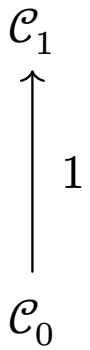
Double categories

Category theory, again

Let's repeat what a category is. A category $\mathcal{C}$ is a collection of objects $\mathcal{C}_0$ and a collection of morphisms $\mathcal{C}_1$, with two functions



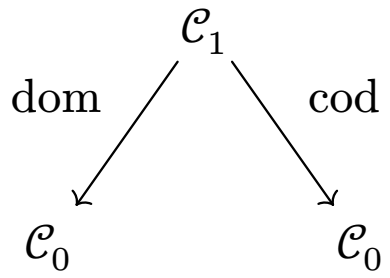
as well as an identity



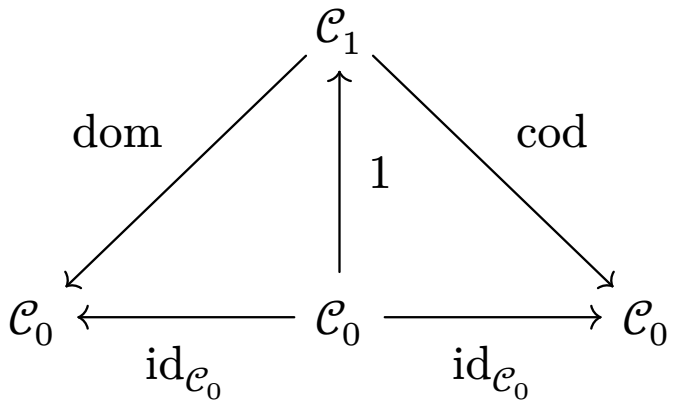
Double categories

Category theory, again

Let's repeat what a category is. A category $\mathcal{C}$ is a collection of objects $\mathcal{C}_0$ and a collection of morphisms $\mathcal{C}_1$, with two functions



as well as an identity



Double categories

Category theory, again

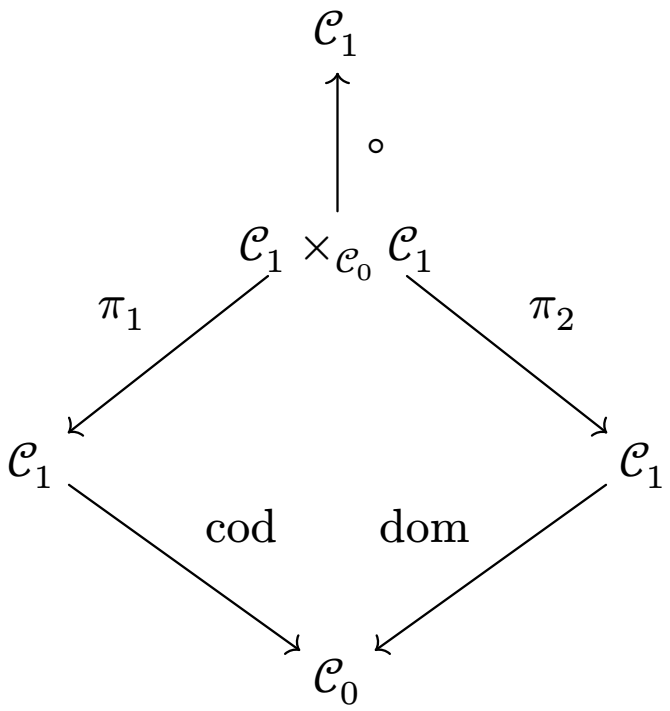
we also need a composition

$$\begin{array}{c} \mathcal{C}_1 \\ \uparrow \\ \{(f,g) \mid \mathrm{cod}(f) = \mathrm{dom}(g)\} \end{array} \circ$$

Double categories

Category theory, again

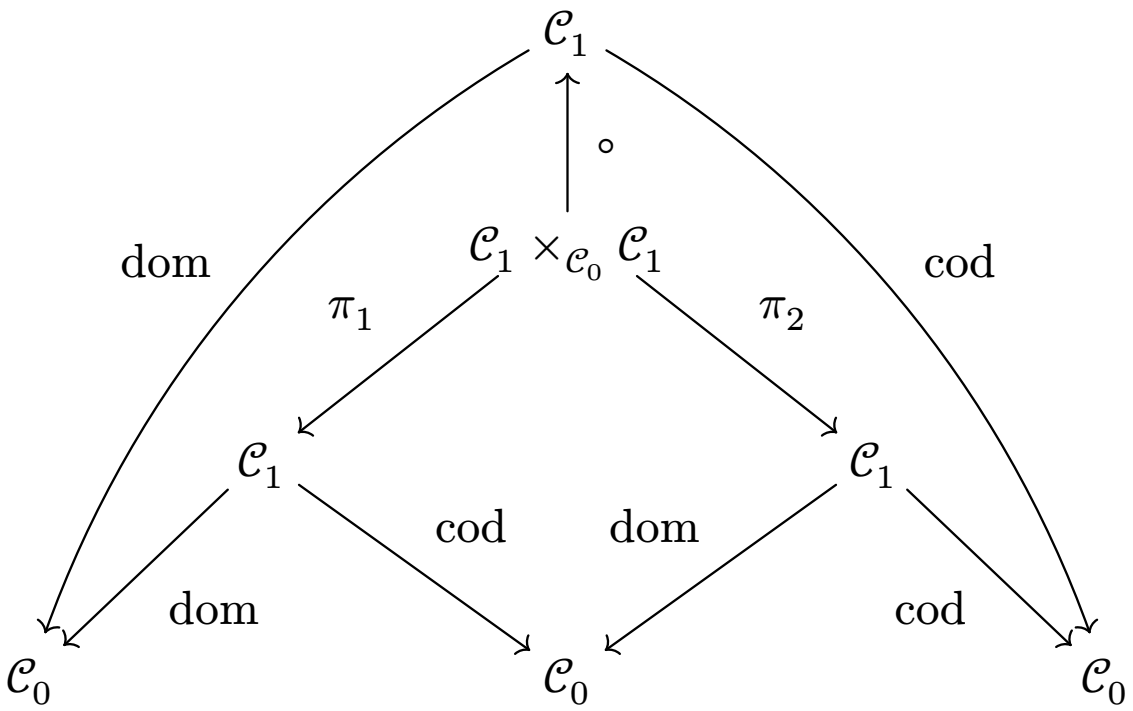
we also need a composition



Double categories

Category theory, again

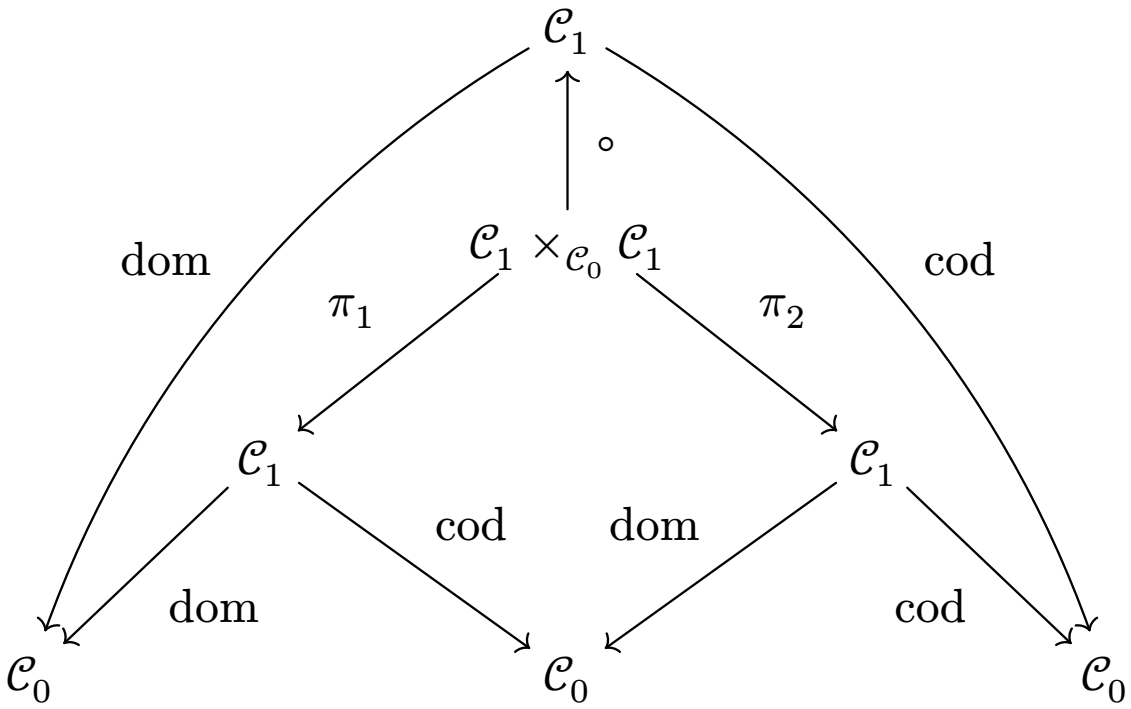
we also need a composition



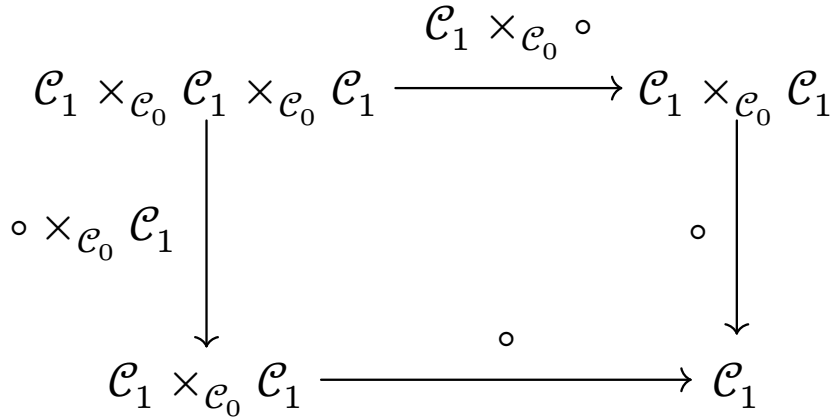
Double categories

Category theory, again

we also need a composition



that is associative



Double categories

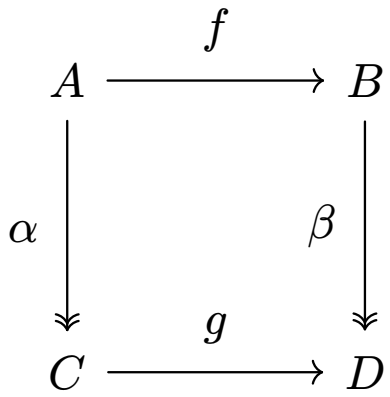
Categories in categories

So… a category is a diagram in **Set**, making additional diagrams commute. But these diagrams make sense in any category with pullbacks. For instance, we can consider internal categories in **Cat**

Definition (Double category) A *double category* is an internal category in **Cat**.

> ⚠ **Warning**
>
> A double category is not a 2-category, or a bicategory!

A double category is like a category, but with two kinds of morphisms: vertical and horizontal morphisms. Each class of morphism behaves like a category, and additionally, a double category tells when the following diagrams "commute"



Double categories

Mental break

Fibrations

Refinement system

Suppose we have a functor $p : \mathcal{E} \to \mathcal{B}$. For $R : \mathcal{E}$ and $X : \mathcal{B}$, we note

$$\begin{array}{c} R \\ p \,\big|\, \\ X \end{array}$$

if $X = p(R)$.

Refinement system

Suppose we have a functor $p : \mathcal{E} \to \mathcal{B}$. For $R : \mathcal{E}$ and $X : \mathcal{B}$, we note

$$\begin{array}{c} R \\ p \,\big|\, \\ X \end{array}$$

if $X = p(R)$.

Similarly, we say that the following diagram commutes

$$\begin{array}{ccc} R & \xrightarrow{\alpha} & S \\ p \,\big|\, & & \big|\, p \\ X & \xrightarrow{f} & Y \end{array}$$

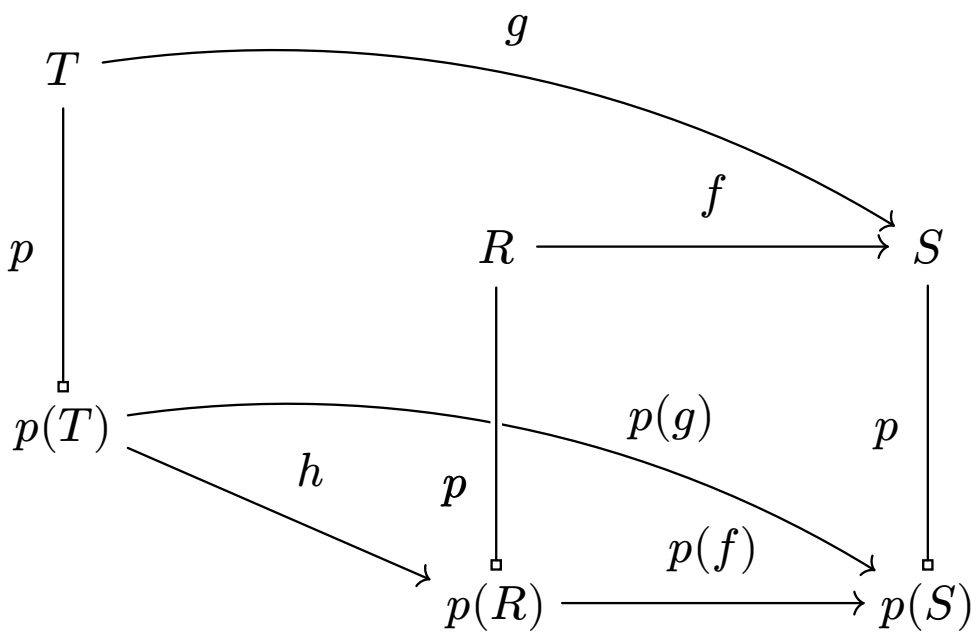
when

$$p(\alpha) = f$$

Fibrations

Cartesian morphism

A morphism $f : R \to S$ is *cartesian* if



Fibrations

Cartesian morphism

A morphism $f : R \to S$ is *cartesian* if