

# Introduction to Categorical Logic

[DRAFT: APRIL 6, 2023]

Steve Awodey

Andrej Bauer



# Contents

<b>3</b>	<b>The <math>\lambda</math>-calculus</b>	<b>5</b>
3.1	Curry-Howard as categorification . . . . .	5
3.2	Cartesian closed categories . . . . .	7
3.3	Simple type theory . . . . .	14
3.4	Interpretation of $\lambda$ -calculus in a CCC . . . . .	22
3.5	Functorial semantics . . . . .	25
3.6	The internal language of a CCC . . . . .	29
	<b>Bibliography</b>	<b>35</b>



# Chapter 3

## The $\lambda$ -calculus

### 3.1 Curry-Howard as categorification

Consider the following natural deduction proof in propositional calculus.

$$\frac{\frac{\frac{[(A \wedge B) \wedge (A \Rightarrow B)]^1}{A \wedge B}}{A} \quad \frac{[(A \wedge B) \wedge (A \Rightarrow B)]^1}{A \Rightarrow B}}{B} \quad \frac{B}{(A \wedge B) \wedge (A \Rightarrow B) \Rightarrow B}^{(1)}$$

This deduction shows that

$$\vdash (A \wedge B) \wedge (A \Rightarrow B) \Rightarrow B.$$

But so does the following:

$$\frac{\frac{[(A \wedge B) \wedge (A \Rightarrow B)]^1}{A \Rightarrow B} \quad \frac{\frac{[(A \wedge B) \wedge (A \Rightarrow B)]^1}{A \wedge B}}{A}}{B} \quad \frac{B}{(A \wedge B) \wedge (A \Rightarrow B) \Rightarrow B}^{(1)}$$

As does:

$$\frac{\frac{[(A \wedge B) \wedge (A \Rightarrow B)]^1}{A \wedge B}}{B} \quad \frac{B}{(A \wedge B) \wedge (A \Rightarrow B) \Rightarrow B}^{(1)}$$

There is a sense in which the first two proofs are “equivalent”, but not the first and the third. The relation (or property) of *provability* in propositional calculus  $\vdash A$  discards such differences in the proofs that witness it. According to the “proof-relevant” point of view, sometimes called *propositions as types*, one retains as relevant some information about the way in which a proposition is proved. This can be done by annotating the proofs with *proof-terms* as they are constructed, as follows:

$$\frac{\frac{[x : (A \wedge B) \wedge (A \Rightarrow B)]^1}{\pi_2(x) : A \Rightarrow B} \quad \frac{\frac{[x : (A \wedge B) \wedge (A \Rightarrow B)]^1}{\pi_1(x) : A \wedge B} \quad \frac{\pi_1(x) : A \wedge B}{\pi_1(\pi_1(x)) : A}}{\pi_2(x)(\pi_1(\pi_1(x))) : B} \quad \frac{}{\lambda x. \pi_2(x)(\pi_1(\pi_1(x))) : (A \wedge B) \wedge (A \Rightarrow B) \Rightarrow B}^{(1)}$$

$$\frac{\frac{[x : (A \wedge B) \wedge (A \Rightarrow B)]^1}{\pi_1(x) : A \wedge B} \quad \frac{\pi_1(x) : A \wedge B}{\pi_1(\pi_1(x)) : A} \quad \frac{[x : (A \wedge B) \wedge (A \Rightarrow B)]^1}{\pi_2(x) : A \Rightarrow B}}{\pi_2(x)(\pi_1(\pi_1(x))) : B} \quad \frac{}{\lambda x. \pi_2(x)(\pi_1(\pi_1(x))) : (A \wedge B) \wedge (A \Rightarrow B) \Rightarrow B}^{(1)}$$

$$\frac{\frac{[x : (A \wedge B) \wedge (A \Rightarrow B)]^1}{\pi_1(x) : A \wedge B} \quad \frac{\pi_1(x) : A \wedge B}{\pi_2(\pi_1(x)) : B}}{\lambda x. \pi_2(\pi_1(x)) : (A \wedge B) \wedge (A \Rightarrow B) \Rightarrow B}^{(1)}$$

The proof terms for the first two proofs are the same, namely  $\lambda x. \pi_2(x)(\pi_1(\pi_1(x)))$ , but the term for the third one is  $\lambda x. \pi_2(\pi_1(x))$ , reflecting the difference in the proofs. The assignment works by labelling assumptions as variables, and then associating term-constructors to the different rules of inference: pairing and projection to conjunction introduction and elimination, function application and  $\lambda$ -abstraction to implication elimination (*modus ponens*) and introduction. The use of variable binding to represent cancellation of premisses is a particularly effective device.

From the categorical point of view, the relation of deducibility  $A \vdash B$  is a mere preorder. The addition of proof terms  $x : A \vdash t : B$  results in a *categorification* of this preorder, in the sense that it becomes a “proper” category, the preordered reflection of which is the deducibility preorder. And now a remarkable fact emerges: it is hardly surprising that the deducibility preorder has, say, finite products  $A \wedge B$  or even exponentials  $A \Rightarrow B$ ; but it is *amazing* that the category with proof terms  $x : A \vdash t : B$  as arrows also turns out to be a cartesian closed category, and indeed a proper one, with distinct parallel arrows, such as

$$\begin{aligned} \pi_2(x)(\pi_1(\pi_1(x))) : (A \wedge B) \wedge (A \Rightarrow B) &\longrightarrow B, \\ \pi_2(\pi_1(x)) : (A \wedge B) \wedge (A \Rightarrow B) &\longrightarrow B. \end{aligned}$$

This *category of proofs* contains information about the “proof theory” of the propositional calculus, as opposed to its mere relation of deducibility.

And now another remarkable fact emerges: when the calculus of proof terms is formulated as a system of *simple type theory*, it admits an alternate interpretation as a formal

system of *function abstraction and application*. This dual interpretation of the system of type theory—as the proof theory of propositional logic, and as formal system for manipulating functions—is sometimes called the *Curry-Howard correspondence* [Sco70, ML84, Tai68]. From the categorical point of view, it expresses a structural equivalence between the cartesian closed categories of *proofs* in propositional logic and *terms* in simple type theory, both of which are categorifications of their common preorder reflection, the deducibility preorder of propositional logic (cf. [MH92]).

In the following sections, we shall consider this remarkable correspondence in detail, as well as some extensions of the basic case represented by cartesian closed categories: categories with coproducts, cocomplete categories, and categories equipped with modal operators. In the next chapter, it will be seen that this correspondence even extends to proofs in quantified predicate logic and terms in dependent type theory, and beyond.

## 3.2 Cartesian closed categories

### Exponentials

We begin with the notion of an exponential  $B^A$  of two objects  $A, B$  in a category, motivated by a couple of important examples. Consider first the category **Pos** of posets and monotone functions. For posets  $P$  and  $Q$  the set  $\mathbf{Hom}(P, Q)$  of all monotone functions between them is again a poset, with the pointwise order:

$$f \leq g \iff fx \leq gx \quad \text{for all } x \in P. \quad (f, g : P \rightarrow Q)$$

Thus, when equipped with a suitable order, the set  $\mathbf{Hom}(P, Q)$  becomes an object of **Pos**.

Similarly, given monoids  $K, M \in \mathbf{Mon}$ , there is a natural monoid structure on the set  $\mathbf{Hom}(K, M)$ , defined pointwise by

$$(f \cdot g)x = fx \cdot gx. \quad (f, g : K \rightarrow M, x \in K)$$

Thus the category **Mon** also admits such “internal **Homs**”. The same thing works in the category **Group** of groups and group homomorphisms, where the set  $\mathbf{Hom}(G, H)$  of all homomorphisms between groups  $G$  and  $H$  can be given a pointwise group structure.

These examples suggest a general notion of an “internal **Hom**” in a category: an “object of morphisms  $A \rightarrow B$ ” which corresponds to the hom-set  $\mathbf{Hom}(A, B)$ . The other ingredient needed is an “evaluation” operation  $\mathbf{eval} : B^A \times A \rightarrow B$  which evaluates a morphism  $f \in B^A$  at an argument  $a \in A$  to give a value  $\mathbf{eval} \circ \langle f, a \rangle = f(a) \in B$ . This is always going to be present as an operation on underlying sets, if we’re starting from a set of functions  $\mathbf{Hom}(A, B)$  between structured sets  $A$  and  $B$ , but even in that case it also needs to be an actual morphism in the category. Finally, we need an operation of “transposition”, taking a morphism  $f : C \times A \rightarrow B$  to one  $\tilde{f} : C \rightarrow B^A$ . We shall see that this in fact separates the previous two examples.

**Definition 3.2.1.** In a category  $\mathcal{C}$  with binary products, an *exponential*  $(B^A, \epsilon)$  of objects  $A$  and  $B$  is an object  $B^A$  together with a morphism  $\epsilon : B^A \times A \rightarrow B$ , called the *evaluation* morphism, such that for every  $f : C \times A \rightarrow B$  there exists a *unique* morphism  $\tilde{f} : C \rightarrow B^A$ , called the *transpose*<sup>1</sup> of  $f$ , for which the following diagram commutes.

$$\begin{array}{ccc}
 B^A & & B^A \times A \xrightarrow{\epsilon} B \\
 \tilde{f} \uparrow & & \uparrow \tilde{f} \times 1_A \\
 C & & C \times A \xrightarrow{f} B
 \end{array}$$

Commutativity of the diagram of course means that  $\epsilon \circ (\tilde{f} \times 1_A) = f$ .

Definition 3.2.1 is called the *universal property of the exponential*. It is just the category-theoretic way of saying that a function  $f : C \times A \rightarrow B$  of two variables can be viewed as a function  $\tilde{f} : C \rightarrow B^A$  of one variable that maps  $z \in C$  to a function  $\tilde{f}z = f\langle z, - \rangle : A \rightarrow B$  that maps  $x \in A$  to  $f\langle z, x \rangle$ . The relationship between  $f$  and  $\tilde{f}$  is then the expected one:

$$(\tilde{f}z)x = f\langle z, x \rangle.$$

That is all there is to it, except that by making the evaluation explicit, variables and elements never need to be mentioned! The benefit of this is that the definition makes sense also in categories whose objects are not *sets*, and whose morphisms are not *functions*—even though some of the basic examples are of that sort.

In **Poset** the exponential  $Q^P$  of posets  $P$  and  $Q$  is the set of all monotone maps  $P \rightarrow Q$ , ordered pointwise, as above. The evaluation map  $\epsilon : Q^P \times P \rightarrow Q$  is just the usual evaluation of a function at an argument. The transpose of a monotone map  $f : R \times P \rightarrow Q$  is the map  $\tilde{f} : R \rightarrow Q^P$ , defined by,  $(\tilde{f}z)x = f\langle z, x \rangle$ , i.e. the transposed *function*. We say that the category **Pos** has *all exponentials*.

**Definition 3.2.2.** Suppose  $\mathcal{C}$  has all finite products. An object  $A \in \mathcal{C}$  is *exponentiable* when the exponential  $B^A$  exists for every  $B \in \mathcal{C}$  (along with an associated evaluation map  $\epsilon : B^A \times A \rightarrow B$ ). We say that  $\mathcal{C}$  has *exponentials* if every object is exponentiable. A *cartesian closed category (ccc)* is a category that has all finite products and exponentials.

**Example 3.2.3.** Consider again the example of the set  $\mathbf{Hom}(M, N)$  of homomorphisms between two monoids  $M, N$ , equipped with the pointwise monoid structure. To be a monoid homomorphism, the transpose  $\tilde{h} : 1 \rightarrow \mathbf{Hom}(M, N)$  of a homomorphism  $h : 1 \times M \rightarrow N$  would have to take the unit element  $u \in 1$  to the unit homomorphism  $u : M \rightarrow N$ , which is the constant function at the unit  $u \in N$ . Since  $1 \times M \cong M$ , that would mean that *all* homomorphisms  $h : M \rightarrow N$  would have the same transpose, namely  $\tilde{h} = u : 1 \rightarrow \mathbf{Hom}(M, N)$ . So **Mon** cannot be cartesian closed. The same argument works in the category **Group**, and in many related ones.

<sup>1</sup>Also,  $f$  is called the transpose of  $\tilde{f}$ , so that  $f$  and  $\tilde{f}$  are each other's transpose.



**Exercise 3.2.4.** Recall that monoids and groups can be regarded as (1-object) categories, and then their homomorphisms are just functors. So we have full subcategories,

$$\mathbf{Mon} \hookrightarrow \mathbf{Group} \hookrightarrow \mathbf{Cat}.$$

Is the category  $\mathbf{Cat}$  of all (small) categories and functors cartesian closed? What about the subcategory of all *groupoids*,

$$\mathbf{Grpd} \hookrightarrow \mathbf{Cat},$$

defined as those categories in which every arrow is an iso?

## Two characterizations of CCCs

**Proposition 3.2.5.** *In a category  $\mathcal{C}$  with binary products an object  $A$  is exponentiable if, and only if, the functor*

$$- \times A : \mathcal{C} \rightarrow \mathcal{C}$$

*has a right adjoint*

$$-^A : \mathcal{C} \rightarrow \mathcal{C}.$$

*Proof.* If such a right adjoint exists then the exponential of  $A$  and  $B$  is  $(B^A, \epsilon_B)$ , where  $\epsilon_B : B^A \times A \rightarrow A$  is the counit of the adjunction at  $B$ . Indeed, the universal property of the exponential is just the universal property of the counit  $\epsilon : (-)^A \Rightarrow 1_{\mathcal{C}}$ .

Conversely, suppose for every  $B$  there is an exponential  $(B^A, \epsilon_B)$ . As the object part of the right adjoint we then take  $B^A$ . For the morphism part, given  $g : B \rightarrow C$ , we can define  $g^A : B^A \rightarrow C^A$  to be the transpose of  $g \circ \epsilon_B$ ,

$$g^A = (g \circ \epsilon_B)^\sim$$

as indicated below.

$$\begin{array}{ccc} B^A \times A & \xrightarrow{\epsilon_B} & B \\ g^A \times 1_A \downarrow & & \downarrow g \\ C^A \times A & \xrightarrow{\epsilon_C} & C \end{array} \quad (3.1)$$

The counit  $\epsilon : -^A \times A \Rightarrow 1_{\mathcal{C}}$  at  $B$  is then  $\epsilon_B$  itself, and the naturality square for  $\epsilon$  is then exactly (3.1), i.e. the defining property of  $(f \circ \epsilon_B)^\sim$ :

$$\epsilon_C \circ (g^A \times 1_A) = \epsilon_C \circ ((g \circ \epsilon_B)^\sim \times 1_A) = g \circ \epsilon_B.$$

The universal property of the counit  $\epsilon$  is precisely the universal property of the exponential  $(B^A, \epsilon_B)$   $\square$

Note that because exponentials can be expressed as right adjoints to binary products, they are determined uniquely up to isomorphism. Moreover, the definition of a cartesian closed category can then be phrased entirely in terms of adjoint functors: we just need to require the existence of the terminal object, binary products, and exponentials.

**Proposition 3.2.6.** *A category  $\mathcal{C}$  is cartesian closed if, and only if, the following functors have right adjoints:*

$$\begin{aligned} !_{\mathcal{C}} : \mathcal{C} &\rightarrow \mathbf{1} , \\ \Delta : \mathcal{C} &\rightarrow \mathcal{C} \times \mathcal{C} , \\ (- \times A) : \mathcal{C} &\rightarrow \mathcal{C} . \end{aligned} \quad (A \in \mathcal{C})$$

Here  $!_{\mathcal{C}}$  is the unique functor from  $\mathcal{C}$  to the terminal category  $\mathbf{1}$  and  $\Delta$  is the diagonal functor  $\Delta A = \langle A, A \rangle$ , and the right adjoint of  $- \times A$  is exponentiation by  $A$ .

□

The significance of the adjoint formulation is that it implies the possibility of a purely equational specification (adjoint structure on a category is “algebraic”, in a sense that can be made precise; see [?]). It follows that there is a equational formulation of the definition of a cartesian closed category.

**Proposition 3.2.7** (Equational version of CCC). *A category  $\mathcal{C}$  is cartesian closed if, and only if, it has the following structure:*

1. An object  $\mathbf{1} \in \mathcal{C}$  and a morphism  $!_A : A \rightarrow \mathbf{1}$  for every  $A \in \mathcal{C}$ .
2. An object  $A \times B$  for all  $A, B \in \mathcal{C}$  together with morphisms  $\pi_0 : A \times B \rightarrow A$  and  $\pi_1 : A \times B \rightarrow B$ , and for every pair of morphisms  $f : C \rightarrow A$ ,  $g : C \rightarrow B$  a morphism  $\langle f, g \rangle : C \rightarrow A \times B$ .
3. An object  $B^A$  for all  $A, B \in \mathcal{C}$  together with a morphism  $\epsilon : B^A \times A \rightarrow B$ , and a morphism  $\tilde{f} : C \rightarrow B^A$  for every morphism  $f : C \times A \rightarrow B$ .

These new objects and morphisms are required to satisfy the following equations:

1. For every  $f : A \rightarrow \mathbf{1}$ ,

$$f = !_A .$$

2. For all  $f : C \rightarrow A$ ,  $g : C \rightarrow B$ ,  $h : C \rightarrow A \times B$ ,

$$\pi_0 \circ \langle f, g \rangle = f , \quad \pi_1 \circ \langle f, g \rangle = g , \quad \langle \pi_0 \circ h, \pi_1 \circ h \rangle = h .$$

3. For all  $f : C \times A \rightarrow B$ ,  $g : C \rightarrow B^A$ ,

$$\epsilon \circ (\tilde{f} \times \mathbf{1}_A) = f , \quad (\epsilon \circ (g \times \mathbf{1}_A))^{\sim} = g .$$

where for  $e : E \rightarrow E'$  and  $f : F \rightarrow F'$  we define

$$e \times f := \langle e\pi_0, f\pi_1 \rangle : E \times F \rightarrow E' \times F' .$$

These equations ensure that certain diagrams commute and that the morphisms that are required to exist are unique. For example, let us prove that  $(A \times B, \pi_0, \pi_1)$  is the product of  $A$  and  $B$ . For  $f : C \rightarrow A$  and  $g : C \rightarrow B$  there exists a morphism  $\langle f, g \rangle : C \rightarrow A \times B$ . Equations

$$\pi_0 \circ \langle f, g \rangle = f \quad \text{and} \quad \pi_1 \circ \langle f, g \rangle = g$$

enforce the commutativity of the two triangles in the following diagram:

$$\begin{array}{ccccc} & & C & & \\ & g \swarrow & \downarrow \langle f, g \rangle & \searrow f & \\ A & \xleftarrow{\pi_0} & A \times B & \xrightarrow{\pi_1} & B \end{array}$$

Suppose  $h : C \rightarrow A \times B$  is another morphism such that  $f = \pi_0 \circ h$  and  $g = \pi_1 \circ h$ . Then by the third equation for products we get

$$h = \langle \pi_0 \circ h, \pi_1 \circ h \rangle = \langle f, g \rangle ,$$

and so  $\langle f, g \rangle$  is unique.

**Exercise 3.2.8.** Use the equational characterization of CCCs, Proposition 3.2.7, to show that the category **Pos** of posets and monotone functions is cartesian closed, as claimed. Also verify that **Mon** is not. Which parts of the definition fail in **Mon**?

**Exercise 3.2.9.** Use the equational characterization of CCCs, Proposition 3.2.7, to show that the product category  $\prod_{i \in I} \mathcal{C}_i$  of any (set-indexed) family  $(\mathcal{C}_i)_{i \in I}$  of cartesian closed categories  $\mathcal{C}_i$  is cartesian closed. Is the same true for an arbitrary limit in **Cat**?

## Some proper CCCs

We next review some important examples of (non-poset) cartesian closed categories, most of which have already been discussed.

**Example 3.2.10.** The first example is the category **Set**. We already know that the terminal object is a singleton set and that binary products are cartesian products. The exponential of  $X$  and  $Y$  in **Set** is just the set of all functions from  $X$  to  $Y$ ,

$$Y^X = \{ f \subseteq X \times Y \mid \forall x : X . \exists ! y : Y . \langle x, y \rangle \in f \} .$$

The evaluation morphism  $\text{eval} : Y^X \times X \rightarrow Y$  is the usual evaluation of a function at an argument, i.e.,  $\text{eval}\langle f, x \rangle$  is the unique  $y \in Y$  for which  $\langle x, y \rangle \in f$ .

**Example 3.2.11.** The category  $\mathbf{Cat}$  of all small categories is cartesian closed. The exponential of small categories  $\mathcal{C}$  and  $\mathcal{D}$  is the category  $\mathcal{D}^{\mathcal{C}}$  of functors, with natural transformations as arrows (see ??). Note that if  $\mathcal{D}$  is a groupoid (all arrows are isos), then so is  $\mathcal{D}^{\mathcal{C}}$ . It follows that the category of groupoids is full (even as a 2-category) in  $\mathbf{Cat}$ . Since limits of groupoids in  $\mathbf{Cat}$  are also groupoids, the inclusion of the full subcategory  $\mathbf{Grpd} \hookrightarrow \mathbf{Cat}$  preserves limits. It also preserves the CCC structure.

**Example 3.2.12.** The same reasoning as in the previous example shows that the full subcategory  $\mathbf{Pos} \hookrightarrow \mathbf{Cat}$  of all small posets and monotone maps is also cartesian closed, and the (limit preserving) inclusion  $\mathbf{Pos} \hookrightarrow \mathbf{Cat}$  also preserves exponentials. Note that the (non-full) forgetful functor  $U : \mathbf{Pos} \rightarrow \mathbf{Set}$  does not, and that  $U(Q^P) \subseteq (UQ)^{UP}$  is in general a *proper* subset.

**Exercise 3.2.13.** There is a full and faithful functor  $I : \mathbf{Set} \rightarrow \mathbf{Poset}$  that preserves finite limits as well as exponentials. How is this related to the example  $\mathbf{Grpd} \hookrightarrow \mathbf{Cat}$ ?

The foregoing examples are instances of the following general situation.

**Proposition 3.2.14.** *Let  $\mathcal{E}$  be a CCC and  $i : \mathcal{S} \hookrightarrow \mathcal{E}$  a full subcategory with finite products and a left adjoint reflection  $L : \mathcal{E} \rightarrow \mathcal{S}$  preserving finite products. Suppose moreover that for any two objects  $A, B$  in  $\mathcal{S}$ , the exponential  $iB^{iA}$  is again in  $\mathcal{S}$ . Then  $\mathcal{S}$  has all exponentials, and these are preserved by  $i$ .*

*Proof.* By assumption, we have  $L \dashv i$  with isomorphic counit  $LiS \cong S$  for all  $S \in \mathcal{S}$ . Let us identify  $\mathcal{S}$  with the subcategory of  $\mathcal{E}$  that is its image under  $i : \mathcal{S} \hookrightarrow \mathcal{E}$ . The assumption that  $B^A$  is again in  $\mathcal{S}$  for all  $A, B \in \mathcal{S}$ , along with the fullness of  $\mathcal{S}$  in  $\mathcal{E}$ , gives the exponentials, and the closure of  $\mathcal{S}$  under finite products in  $\mathcal{E}$  ensures that the required transposes will also be in  $\mathcal{S}$ .

Alternately, for any  $A, B \in \mathcal{S}$  set  $B^A = L(iB^{iA})$ . Then for any  $C \in \mathcal{S}$ , we have natural isos:

$$\begin{aligned} \mathcal{S}(C \times A, B) &\cong \mathcal{E}(i(C \times A), iB) \\ &\cong \mathcal{E}(iC \times iA, iB) \\ &\cong \mathcal{E}(iC, iB^{iA}) \\ &\cong \mathcal{E}(iC, iL(iB^{iA})) \\ &\cong \mathcal{S}(C, L(iB^{iA})) \\ &\cong \mathcal{S}(C, B^A) \end{aligned}$$

where in the fifth line we used the assumption that  $iB^{iA}$  is again in  $\mathcal{S}$ , in the form  $iB^{iA} \cong iE$  for some  $E \in \mathcal{S}$ , which is then necessarily  $L(iB^{iA}) = LiE \cong E$ .  $\square$

A related general situation that covers some (but not all) of the above examples is this:

**Proposition 3.2.15.** *Let  $\mathcal{E}$  be a CCC and  $i : \mathcal{S} \hookrightarrow \mathcal{E}$  a full subcategory with finite products and a right adjoint reflection  $R : \mathcal{E} \rightarrow \mathcal{S}$ . If  $i$  preserves finite products, then  $\mathcal{S}$  also has all exponentials, and these are computed first in  $\mathcal{E}$ , and then reflected by  $R$  into  $\mathcal{S}$ .*

*Proof.* For any  $A, B \in \mathcal{S}$  set  $B^A = R(iB^{iA})$  as described. Now for any  $C \in \mathcal{S}$ , we have natural isos:

$$\begin{aligned} \mathcal{S}(C \times A, B) &\cong \mathcal{E}(i(C \times A), iB) \\ &\cong \mathcal{E}(iC \times iA, iB) \\ &\cong \mathcal{E}(iC, iB^{iA}) \\ &\cong \mathcal{S}(C, R(iB^{iA})) \\ &\cong \mathcal{S}(C, B^A). \end{aligned}$$

□

An example of the foregoing is the inclusion of the opens into the powerset of points of a space  $X$ ,

$$\mathcal{O}X \hookrightarrow \mathcal{P}X$$

This frame homomorphism is associated to the map  $|X| \rightarrow X$  of locales (or in this case, spaces) from the discrete space on the set of points of  $X$ .

**Exercise 3.2.16.** Which of the examples follows from which proposition?

**Example 3.2.17.** For any set  $X$ , the slice category  $\mathbf{Set}/_X$  is cartesian closed. The product of  $f : A \rightarrow X$  and  $g : B \rightarrow X$  is the pullback  $A \times_X B \rightarrow X$ , which can be constructed as the set of pairs

$$A \times_X B \rightarrow X = \{\langle a, b \rangle \mid fa = gb\}.$$

The exponential, however, is *not* simply the set

$$\{h : A \rightarrow B \mid f = g \circ h\},$$

(what would the projection to  $X$  be?), but rather the set of all pairs

$$\{\langle x, h : A_x \rightarrow B_x \rangle \mid x \in X, f = g \circ h\},$$

where  $A_x = f^{-1}\{x\}$  and  $B_x = g^{-1}\{x\}$ , with the evident projection to  $X$ .

**Exercise 3.2.18.** Prove that  $\mathbf{Set}/_X$  is always cartesian closed.

**Example 3.2.19.** A presheaf category  $\widehat{\mathbb{C}}$  is cartesian closed, provided the index category  $\mathbb{C}$  is small. To see what the exponential of presheaves  $P$  and  $Q$  ought to be, we use the Yoneda Lemma. If  $Q^P$  exists, then by Yoneda Lemma and the adjunction  $(- \times P) \dashv (-^P)$ , we have for all  $A \in \mathbb{C}$ ,

$$Q^P(A) \cong \mathbf{Nat}(\mathbf{y}A, Q^P) \cong \mathbf{Nat}(\mathbf{y}A \times P, Q).$$

Because  $\mathbb{C}$  is small  $\mathbf{Nat}(\mathbf{y}A \times P, Q)$  is a set, so we can *define*  $Q^P$  to be the presheaf

$$Q^P = \mathbf{Nat}(\mathbf{y}- \times P, Q).$$

The evaluation morphism  $E : Q^P \times P \Longrightarrow Q$  is the natural transformation whose component at  $A$  is

$$\begin{aligned} E_A &: \text{Nat}(\mathbf{y}A \times P, Q) \times PA \rightarrow QA, \\ E_A &: \langle \eta, x \rangle \mapsto \eta_A \langle \mathbf{1}_A, x \rangle. \end{aligned}$$

The transpose of a natural transformation  $\phi : R \times P \Longrightarrow Q$  is the natural transformation  $\tilde{\phi} : R \Longrightarrow Q^P$  whose component at  $A$  is the function that maps  $z \in RA$  to the natural transformation  $\tilde{\phi}_A z : \mathbf{y}A \times P \Longrightarrow Q$ , whose component at  $B \in \mathcal{C}$  is

$$\begin{aligned} (\tilde{\phi}_A z)_B &: \mathcal{C}(B, A) \times PB \rightarrow QB, \\ (\tilde{\phi}_A z)_B &: \langle f, y \rangle \mapsto \phi_B \langle (Rf)z, y \rangle. \end{aligned}$$

**Exercise 3.2.20.** Verify that the above definition of  $Q^P$  really gives an exponential of presheaves  $P$  and  $Q$ .

It follows immediately that the category of graphs **Graph** is cartesian closed because it is the presheaf category  $\mathbf{Set}^{\rightrightarrows}$ . The same is of course true for the “category of functions”, i.e. the arrow category  $\mathbf{Set}^{\rightarrow}$ , as well as the category of simplicial sets  $\mathbf{Set}^{\Delta^{\text{op}}}$  from topology.

**Exercise 3.2.21.** This exercise is for students with some background in linear algebra. Let **Vec** be the category of real vector spaces and linear maps between them. Given vector spaces  $X$  and  $Y$ , the linear maps  $\mathcal{L}(X, Y)$  between them form a vector space. So define  $\mathcal{L}(X, -) : \mathbf{Vec} \rightarrow \mathbf{Vec}$  to be the functor which maps a vector space  $Y$  to the vector space  $\mathcal{L}(X, Y)$ , and it maps a linear map  $f : Y \rightarrow Z$  to the linear map  $\mathcal{L}(X, f) : \mathcal{L}(X, Y) \rightarrow \mathcal{L}(X, Z)$  defined by  $h \mapsto f \circ h$ . Show that  $\mathcal{L}(X, -)$  has a left adjoint  $- \otimes X$ , but also show that this adjoint is *not* the binary product in **Vec**.

A few other instructive examples that can be explored by the interested reader are the following.

- Etale spaces over a base space  $X$ . This category can be described as consisting of *local homeomorphisms*  $f : Y \rightarrow X$  and commutative triangles over  $X$  between such maps. It is equivalent to the category  $\mathbf{Sh}(X)$  of *sheaves* on  $X$ . See [?, ch.n].
- Various subcategories of topological spaces (sequential spaces, compactly-generated spaces). Cf. [?].
- Dana Scott’s category **Equ** of equilogical spaces [?].

### 3.3 Simple type theory

The  $\lambda$ -calculus is the abstract theory of functions, just like group theory is the abstract theory of symmetries. There are two basic operations that can be performed with functions. The first one is the *application* of a function to an argument: if  $f$  is a function and  $a$  is an

argument, then  $fa$  is the application of  $f$  to  $a$ , also called the *value* of  $f$  at  $a$ . The second operation is *abstraction*: if  $x$  is a variable and  $t$  is an expression in which  $x$  may appear, then there is a function  $f$  defined by the equation

$$fx = t .$$

Here we gave the name  $f$  to the newly formed function. But we could have expressed the same function without giving it a name; this is usually written as

$$x \mapsto t ,$$

and it means “ $x$  is mapped to  $t$ ”. In  $\lambda$ -calculus we use a different notation, which is more convenient when abstractions are nested:

$$\lambda x. t .$$

This operation is called  *$\lambda$ -abstraction*. For example,  $\lambda x. \lambda y. (x + y)$  is the function which maps an argument  $a$  to the function  $\lambda y. (a + y)$ , which maps an argument  $b$  ... . The variable  $x$  is said to be *bound in  $t$*  in the expression  $\lambda x. t$ .

It may seem strange that in specifying the abstraction of a function, we switched from talking about objects (functions, arguments, values) to talking about *expressions*: variables, names, equations. This “syntactic” point of view seems to have been part of the notion of a function from the start, in the theory of algebraic equations. It is the reason that the  $\lambda$ -calculus is part of *logic*, unlike the theory of cartesian closed categories, which remains thoroughly semantical (and “variable-free”). The relation between the two different points of view occupies the rest of this chapter—and, indeed, the entire subject of logic!

There are two kinds of  $\lambda$ -calculus: the *typed* and the *untyped*. In the untyped version there are no restrictions on how application is formed, so that an expression such as

$$\lambda x. (xx)$$

is valid, whatever it may mean. We will concentrate here on the typed  $\lambda$ -calculus. In typed  $\lambda$ -calculus every expression has a *type*, and there are rules for forming valid expressions and types. For example, we can only form an application  $fa$  when  $a$  has a type  $A$  and  $f$  has a type  $A \rightarrow B$ , which indicates a function taking arguments of type  $A$  and giving results of type  $B$ . The judgment that expression  $t$  has a type  $A$  is written as

$$t : A .$$

To computer scientists the idea of expressions having types is familiar from programming languages, whereas mathematicians can think of types as sets and read  $t : A$  as  $t \in A$ .

**Simply-typed  $\lambda$ -calculus.** We now give a more formal definition of what constitutes a *simply-typed  $\lambda$ -calculus*. First, we are given a set of *simple types*, which are generated from *basic types* by formation of products and function types:

$$\begin{aligned} \text{Basic type } B &::= B_0 \mid B_1 \mid B_2 \cdots \\ \text{Simple type } A &::= B \mid A_1 \times A_2 \mid A_1 \rightarrow A_2. \end{aligned}$$

We adopt the convention that function types associate to the right:

$$A \rightarrow B \rightarrow C = A \rightarrow (B \rightarrow C) .$$

We assume there is a countable set of *variables*  $x, y, u, \dots$ . We are also given a set of *basic constants*. The set of *terms* is generated from variables and basic constants by the following grammar:

$$\begin{aligned} \text{Variable } v &::= x \mid y \mid z \mid \cdots \\ \text{Constant } c &::= c_1 \mid c_2 \mid \cdots \\ \text{Term } t &::= v \mid c \mid * \mid \langle t_1, t_2 \rangle \mid \mathbf{fst} \, t \mid \mathbf{snd} \, t \mid t_1 \, t_2 \mid \lambda x : A . t \end{aligned}$$

In words, this means:

1. a variable is a term,
2. each basic constant is a term,
3. the constant  $*$  is a term, called the *unit*,
4. if  $u$  and  $t$  are terms then  $\langle u, t \rangle$  is a term, called a *pair*,
5. if  $t$  is a term then  $\mathbf{fst} \, t$  and  $\mathbf{snd} \, t$  are terms,
6. if  $u$  and  $t$  are terms then  $u \, t$  is a term, called an *application*
7. if  $x$  is a variable,  $A$  is a type, and  $t$  is a term, then  $\lambda x : A . t$  is a term, called a  *$\lambda$ -abstraction*.

The variable  $x$  is *bound* in  $\lambda x : A . t$ . Application associates to the left, thus  $s \, t \, u = (s \, t) \, u$ . The set of *free variables*  $\mathbf{FV}(t)$  of a term  $t$  is determined as follows:

$$\begin{aligned} \mathbf{FV}(x) &= \{x\} \quad \text{if } x \text{ is a variable} \\ \mathbf{FV}(a) &= \emptyset \quad \text{if } a \text{ is a basic constant} \\ \mathbf{FV}(\langle u, t \rangle) &= \mathbf{FV}(u) \cup \mathbf{FV}(t) \\ \mathbf{FV}(\mathbf{fst} \, t) &= \mathbf{FV}(t) \\ \mathbf{FV}(\mathbf{snd} \, t) &= \mathbf{FV}(t) \\ \mathbf{FV}(u \, t) &= \mathbf{FV}(u) \cup \mathbf{FV}(t) \\ \mathbf{FV}(\lambda x . t) &= \mathbf{FV}(t) \setminus \{x\} . \end{aligned}$$



If  $x_1, \dots, x_n$  are *distinct* variables and  $A_1, \dots, A_n$  are types then the sequence

$$x_1 : A_1, \dots, x_n : A_n$$

is a *typing context*, or just *context*. The empty sequence is sometimes denoted by a dot  $\cdot$ , and it is a valid context. Contexts are denoted by capital Greek letters  $\Gamma, \Delta, \dots$

A *typing judgment* is a judgment of the form

$$\Gamma \mid t : A$$

where  $\Gamma$  is a context,  $t$  is a term, and  $A$  is a type. In addition the free variables of  $t$  must occur in  $\Gamma$ , but  $\Gamma$  may contain other variables as well. We read the above judgment as “in context  $\Gamma$  the term  $t$  has type  $A$ ”. Next we describe the rules for deriving typing judgments.

- Each basic constant  $c_i$  has a uniquely determined type  $C_i$  (not necessarily basic):

$$\overline{\Gamma \mid c_i : C_i}$$

- The type of a variable is determined by the context:

$$\overline{x_1 : A_1, \dots, x_i : A_i, \dots, x_n : A_n \mid x_i : A_i} \quad (1 \leq i \leq n)$$

- The constant  $*$  has type 1:

$$\overline{\Gamma \mid * : 1}$$

- The typing rules for pairs and projections are:

$$\frac{\Gamma \mid a : A \quad \Gamma \mid b : B}{\Gamma \mid \langle a, b \rangle : A \times B} \quad \frac{\Gamma \mid t : A \times B}{\Gamma \mid \mathbf{fst} \, t : A} \quad \frac{\Gamma \mid c : A \times B}{\Gamma \mid \mathbf{snd} \, t : B}$$

- The typing rules for application and  $\lambda$ -abstraction are:

$$\frac{\Gamma \mid t : A \rightarrow B \quad \Gamma \mid a : A}{\Gamma \mid ta : B} \quad \frac{\Gamma, x : A \mid t : B}{\Gamma \mid (\lambda x : A. t) : A \rightarrow B}$$

Lastly, we have *equations* between terms: for terms of type  $A$  in context  $\Gamma$ ,

$$\Gamma \mid s : A, \quad \Gamma \mid t : A,$$

the judgment that they are equal is written as

$$\Gamma \mid s = t : A.$$

Note that  $s$  and  $t$  necessarily have the same type; it does *not* make sense to compare terms of different types. We have the following rules for equations, the effect of which is to make equality between terms into an equivalence relation at each type, and a congruence with respect to all of the operations, just as for algebraic theories:

- Equality is an equivalence relation:

$$\frac{}{\Gamma \mid t = t : A} \quad \frac{\Gamma \mid s = t : A}{\Gamma \mid t = s : A} \quad \frac{\Gamma \mid s = t : A \quad \Gamma \mid t = u : A}{\Gamma \mid s = u : A}$$

- The substitution rule:

$$\frac{\Gamma \mid s = t : A \quad \Gamma, x : A \mid u = v : B}{\Gamma \mid u[s/x] = v[t/x] : B}$$

- The weakening rule:

$$\frac{\Gamma \mid s = t : A}{\Gamma, x : B \mid s = t : A}$$

- Unit type:

$$\frac{}{\Gamma \mid t = * : 1}$$

- Equations for product types:

$$\frac{\Gamma \mid u = v : A \quad \Gamma \mid s = t : B}{\Gamma \mid \langle u, s \rangle = \langle v, t \rangle : A \times B}$$

$$\frac{\Gamma \mid s = t : A \times B}{\Gamma \mid \mathbf{fst} s = \mathbf{fst} t : A} \quad \frac{\Gamma \mid s = t : A \times B}{\Gamma \mid \mathbf{snd} s = \mathbf{snd} t : A}$$

$$\frac{}{\Gamma \mid t = \langle \mathbf{fst} t, \mathbf{snd} t \rangle : A \times B}$$

$$\frac{}{\Gamma \mid \mathbf{fst} \langle s, t \rangle = s : A} \quad \frac{}{\Gamma \mid \mathbf{snd} \langle s, t \rangle = t : A}$$

- Equations for function types:

$$\frac{\Gamma \mid s = t : A \rightarrow B \quad \Gamma \mid u = v : A}{\Gamma \mid s u = t v : B}$$

$$\frac{\Gamma, x : A \mid t = u : B}{\Gamma \mid (\lambda x : A. t) = (\lambda x : A. u) : A \rightarrow B}$$

$$\frac{}{\Gamma \mid (\lambda x : A. t) u = t[u/x] : A} \quad (\beta\text{-rule})$$

$$\frac{}{\Gamma \mid \lambda x : A. (t x) = t : A \rightarrow B} \text{ if } x \notin \mathbf{FV}(t) \quad (\eta\text{-rule})$$

This completes the description of a simply-typed  $\lambda$ -calculus.

**Simply-typed  $\lambda$ -theories.** Apart from the above rules for equality we might want to impose additional equations. In this case we do not speak of a  $\lambda$ -calculus but rather of a  $\lambda$ -theory. Thus, a  $\lambda$ -theory  $\mathbb{T}$  is given by a set of basic types and a set of basic constants, called the *signature*, and a set of *equations* of the form

$$\Gamma \mid s = t : A .$$

(Observe that we can always state the equations equivalently in *closed form* simply by  $\lambda$ -abstracting all the variables in the context  $\Gamma$ .)

We summarize the preceding definitions.

**Definition 3.3.1.** A (*simply-typed*) *signature*  $S_\lambda$  for the  $\lambda$ -calculus is given by a set of *basic types*  $B_i$  together with a set of *basic constants*  $c_j : C_j$ , together with their types,

$$S^\lambda = (\{B_i \mid i \in I\}, \{c_j : C_j \mid j \in J\}) .$$

A *simply-typed  $\lambda$ -theory*  $\mathbb{T}^\lambda$  is a simply-typed signature  $S^\lambda$  together with a set of equations,

$$\mathbb{T}^\lambda = (S^\lambda(\mathbb{T}), E(\mathbb{T}) = \{s_k = t_k : A_k \mid k \in K\}) .$$

**Example 3.3.2.** The theory of a group is a simply-typed  $\lambda$ -theory. It has one basic type  $\mathbf{G}$  and three basic constants, the unit  $\mathbf{e}$ , the inverse  $\mathbf{i}$ , and the group operation  $\mathbf{m}$ ,

$$\mathbf{e} : \mathbf{G} , \quad \mathbf{i} : \mathbf{G} \rightarrow \mathbf{G} , \quad \mathbf{m} : \mathbf{G} \times \mathbf{G} \rightarrow \mathbf{G} ,$$

with the following equations:

$$\begin{aligned} x : \mathbf{G} \mid \mathbf{m}\langle x, \mathbf{e} \rangle &= x : \mathbf{G} \\ x : \mathbf{G} \mid \mathbf{m}\langle \mathbf{e}, x \rangle &= x : \mathbf{G} \\ x : \mathbf{G} \mid \mathbf{m}\langle x, \mathbf{i} x \rangle &= \mathbf{e} : \mathbf{G} \\ x : \mathbf{G} \mid \mathbf{m}\langle \mathbf{i} x, x \rangle &= \mathbf{e} : \mathbf{G} \\ x : \mathbf{G}, y : \mathbf{G}, z : \mathbf{G} \mid \mathbf{m}\langle x, \mathbf{m}\langle y, z \rangle \rangle &= \mathbf{m}\langle \mathbf{m}\langle x, y \rangle, z \rangle : \mathbf{G} \end{aligned}$$

These are just the familiar axioms for a group.

**Example 3.3.3.** More generally, any (Lawvere) algebraic theory  $\mathbb{A}$  (as in Chapter ??) determines a  $\lambda$ -theory  $\mathbb{A}^\lambda$ . There is one basic type  $\mathbf{A}$  and for each operation  $f$  of arity  $k$  there is a basic constant  $\mathbf{f} : \mathbf{A}^k \rightarrow \mathbf{A}$ , where  $\mathbf{A}^k$  is the  $k$ -fold product  $\mathbf{A} \times \cdots \times \mathbf{A}$ . It is understood that  $\mathbf{A}^0 = \mathbf{1}$ . The terms of  $\mathbb{A}$  are translated to corresponding terms of  $\mathbb{A}^\lambda$  in a straightforward manner. For every axiom  $t = u$  of  $\mathbb{A}$  the corresponding axiom in  $\mathbb{A}^\lambda$  is

$$x_1 : \mathbf{A}, \dots, x_n : \mathbf{A} \mid t = u : \mathbf{A}$$

where  $x_1, \dots, x_n$  are the variables occurring in  $t$  and  $u$ .

**Example 3.3.4.** The theory of a directed graph is a simply-typed theory with two basic types,  $V$  for vertices and  $E$  for edges, and two basic constant, source  $\mathbf{src}$  and target  $\mathbf{trg}$ ,

$$\mathbf{src} : E \rightarrow V, \quad \mathbf{trg} : E \rightarrow V.$$

There are no equations.

**Example 3.3.5.** The theory of a simplicial set is a simply-typed theory with one basic type  $X_n$  for each natural number  $n$ , and the following basic constants, also for each  $n$ , and each  $0 \leq i \leq n$ :

$$\mathbf{d}_i : X_n \rightarrow X_{n-1}, \quad \mathbf{s}_i : X_n \rightarrow X_{n+1}.$$

The equations are as follows, for all natural numbers  $i, j$ :

$$\begin{aligned} \mathbf{d}_i \mathbf{d}_j &= \mathbf{d}_{j-1} \mathbf{d}_i, & \text{if } i < j, \\ \mathbf{s}_i \mathbf{s}_j &= \mathbf{s}_{j+1} \mathbf{s}_i, & \text{if } i \leq j, \\ \mathbf{d}_i \mathbf{s}_j &= \begin{cases} \mathbf{s}_{j-1} \mathbf{d}_i, & \text{if } i < j, \\ \mathbf{id}, & \text{if } i = j \text{ or } i = j + 1, \\ \mathbf{s}_j \mathbf{d}_{i-1}, & \text{if } i > j + 1. \end{cases} \end{aligned}$$

**Example 3.3.6.** An example of a  $\lambda$ -theory is readily found in the theory of programming languages. The mini-programming language *PCF* is a simply-typed  $\lambda$ -calculus with a basic type  $\mathbf{nat}$  for natural numbers, and a basic type  $\mathbf{bool}$  of Boolean values,

$$\text{Basic type } B ::= \mathbf{nat} \text{ type} \mid \mathbf{bool} \text{ type}.$$

There are basic constants zero  $0$ , successor  $\mathbf{succ}$ , the Boolean constants  $\mathbf{true}$  and  $\mathbf{false}$ , comparison with zero  $\mathbf{iszero}$ , and for each type  $A$  the *conditional*  $\mathbf{cond}_A$  and the *fixpoint* operator  $\mathbf{fix}_A$ . They have the following types:

$$\begin{aligned} 0 &: \mathbf{nat} \\ \mathbf{succ} &: \mathbf{nat} \rightarrow \mathbf{nat} \\ \mathbf{true} &: \mathbf{bool} \\ \mathbf{false} &: \mathbf{bool} \\ \mathbf{iszero} &: \mathbf{nat} \rightarrow \mathbf{bool} \\ \mathbf{cond}_A &: \mathbf{bool} \rightarrow A \rightarrow A \\ \mathbf{fix}_A &: (A \rightarrow A) \rightarrow A \end{aligned}$$

The equational axioms of PCF are:

$$\begin{aligned} &\cdot \mid \mathbf{iszero} \, 0 = \mathbf{true} : \mathbf{bool} \\ x : \mathbf{nat} &\mid \mathbf{iszero} (\mathbf{succ} \, x) = \mathbf{false} : \mathbf{bool} \\ u : A, t : A &\mid \mathbf{cond}_A \, \mathbf{true} \, u \, t = u : A \\ u : A, t : A &\mid \mathbf{cond}_A \, \mathbf{false} \, u \, t = t : A \\ t : A \rightarrow A &\mid \mathbf{fix}_A \, t = t (\mathbf{fix}_A \, t) : A \end{aligned}$$

**Example 3.3.7** (D. Scott). Another example of a  $\lambda$ -theory is the *theory of a reflexive type*. This theory has one basic type  $D$  and two constants

$$\mathbf{r} : D \rightarrow D \rightarrow D \qquad \mathbf{s} : (D \rightarrow D) \rightarrow D$$

satisfying the equation

$$f : D \rightarrow D \mid \mathbf{r}(\mathbf{s} f) = f : D \rightarrow D \quad (3.2)$$

which says that  $\mathbf{s}$  is a section and  $\mathbf{r}$  is a retraction, so that the function type  $D \rightarrow D$  is a subspace (even a retract) of  $D$ . A type with this property is said to be *reflexive*. We may additionally stipulate the axiom

$$x : D \mid \mathbf{s}(\mathbf{r} x) = x : D \quad (3.3)$$

which implies that  $D$  is isomorphic to  $D \rightarrow D$ .

### Untyped $\lambda$ -calculus

We briefly describe the *untyped  $\lambda$ -calculus*. It is a theory whose terms are generated by the following grammar:

$$t ::= v \mid t_1 t_2 \mid \lambda x. t .$$

In words, a variable is a term, an application  $tt'$  is a term, for any terms  $t$  and  $t'$ , and a  $\lambda$ -abstraction  $\lambda x. t$  is a term, for any term  $t$ . Variable  $x$  is bound in  $\lambda x. t$ . A *context* is a list of distinct variables,

$$x_1, \dots, x_n .$$

We say that a term  $t$  is valid in context  $\Gamma$  if the free variables of  $t$  are listed in  $\Gamma$ . The judgment that two terms  $u$  and  $t$  are equal is written as

$$\Gamma \mid u = t ,$$

where it is assumed that  $u$  and  $t$  are both valid in  $\Gamma$ . The context  $\Gamma$  is not really necessary but we include it because it is always good practice to list the free variables.

The rules of equality are as follows:

1. Equality is an equivalence relation:

$$\frac{}{\Gamma \mid t = t} \qquad \frac{\Gamma \mid t = u}{\Gamma \mid u = t} \qquad \frac{\Gamma \mid t = u \quad \Gamma \mid u = v}{\Gamma \mid t = v}$$

2. The weakening rule:

$$\frac{\Gamma \mid u = t}{\Gamma, x \mid u = t}$$

3. Equations for application and  $\lambda$ -abstraction:

$$\frac{\Gamma \mid s = t \quad \Gamma \mid u = v}{\Gamma \mid s u = t v} \quad \frac{\Gamma, x \mid t = u}{\Gamma \mid \lambda x. t = \lambda x. u} \quad (\beta\text{-rule})$$

$$\frac{\Gamma \mid (\lambda x. t) u = t[u/x]}{\Gamma \mid \lambda x. (t x) = t} \text{ if } x \notin \text{FV}(t) \quad (\eta\text{-rule})$$

The untyped  $\lambda$ -calculus can be translated into the theory of a reflexive type from Example 3.3.7. An untyped context  $\Gamma$  is translated to a typed context  $\Gamma^*$  by typing each variable in  $\Gamma$  with the reflexive type  $D$ , i.e., a context  $x_1, \dots, x_k$  is translated to  $x_1 : D, \dots, x_k : D$ . An untyped term  $t$  is translated to a typed term  $t^*$  as follows:

$$\begin{aligned} x^* &= x && \text{if } x \text{ is a variable,} \\ (u t)^* &= (\mathbf{r} u^*) t^*, \\ (\lambda x. t)^* &= \mathbf{s} (\lambda x : D. t^*). \end{aligned}$$

For example, the term  $\lambda x. (x x)$  translates to  $\mathbf{s} (\lambda x : D. ((\mathbf{r} x) x))$ . A judgment

$$\Gamma \mid u = t \quad (3.4)$$

is translated to the judgment

$$\Gamma^* \mid u^* = t^* : D. \quad (3.5)$$

**Exercise\* 3.3.8.** Prove that if equation (3.4) is provable then equation (3.5) is provable as well. Identify precisely at which point in your proof you need to use equations (3.2) and (3.3). Does provability of (3.5) imply provability of (3.4)?

### Higher-order logic

This example presumes familiarity with the results of Chapter ??, or at least with the basic categorical approach to first-order logic as presented in [?, ?]. The approach to IHOL presented here is closely tied to *topos theory*, which is to be treated in greater depth in Chapter ??.

## 3.4 Interpretation of $\lambda$ -calculus in a CCC

We now consider semantic aspects of  $\lambda$ -calculus and  $\lambda$ -theories. Suppose  $\mathbb{T}$  is a  $\lambda$ -calculus and  $\mathcal{C}$  is a cartesian closed category. An *interpretation*  $\llbracket - \rrbracket$  of  $\mathbb{T}$  in  $\mathcal{C}$  is given by the following data:

- For every basic type  $B$  in  $\mathbb{T}$  an object  $\llbracket B \rrbracket \in \mathcal{C}$ . The interpretation is extended to all types by

$$\llbracket 1 \rrbracket = 1, \quad \llbracket A \times B \rrbracket = \llbracket A \rrbracket \times \llbracket B \rrbracket, \quad \llbracket A \rightarrow B \rrbracket = \llbracket B \rrbracket^{\llbracket A \rrbracket}.$$

- For every basic constant  $c$  of type  $C$ , a morphism  $\llbracket c \rrbracket : 1 \rightarrow \llbracket C \rrbracket$ .

The interpretation is extended to all terms in context as follows.

- A context  $\Gamma = x_1 : A_1, \dots, x_n : A_n$  is interpreted as the object

$$\llbracket A_1 \rrbracket \times \dots \times \llbracket A_n \rrbracket ,$$

and the empty context is interpreted as the terminal object,

$$\llbracket \cdot \rrbracket = 1 .$$

- A typing judgment

$$\Gamma \mid t : A$$

will be interpreted as a morphism

$$\llbracket \Gamma \mid t : A \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket A \rrbracket .$$

The interpretation is defined inductively by the following rules:

- The  $i$ -th variable is interpreted as the  $i$ -th projection,

$$\llbracket x_0 : A_0, \dots, x_n : A_n \mid x_i : A_i \rrbracket = \pi_i : \llbracket \Gamma \rrbracket \rightarrow \llbracket A_i \rrbracket .$$

- A basic constant  $c : C$  in context  $\Gamma$  is interpreted as the composition

$$\llbracket \Gamma \rrbracket \xrightarrow{!_{\llbracket \Gamma \rrbracket}} 1 \xrightarrow{\llbracket c \rrbracket} \llbracket A \rrbracket$$

- The interpretation of projections and pairs is

$$\begin{aligned} \llbracket \Gamma \mid \langle t, u \rangle : A \times B \rrbracket &= \langle \llbracket \Gamma \mid t : A \rrbracket, \llbracket \Gamma \mid u : B \rrbracket \rangle : \llbracket \Gamma \rrbracket \rightarrow \llbracket A \rrbracket \times \llbracket B \rrbracket \\ \llbracket \Gamma \mid \mathbf{fst} \, t : A \rrbracket &= \pi_0 \circ \llbracket \Gamma \mid t : A \times B \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket A \rrbracket \\ \llbracket \Gamma \mid \mathbf{snd} \, t : A \rrbracket &= \pi_1 \circ \llbracket \Gamma \mid t : A \times B \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket B \rrbracket . \end{aligned}$$

- The interpretation of application and  $\lambda$ -abstraction is

$$\begin{aligned} \llbracket \Gamma \mid t u : B \rrbracket &= \epsilon \circ \langle \llbracket \Gamma \mid t : A \rightarrow B \rrbracket, \llbracket \Gamma \mid u : A \rrbracket \rangle : \llbracket \Gamma \rrbracket \rightarrow \llbracket B \rrbracket \\ \llbracket \Gamma \mid \lambda x : A . t : A \rightarrow B \rrbracket &= (\llbracket \Gamma, x : A \mid t : B \rrbracket)^\sim : \llbracket \Gamma \rrbracket \rightarrow \llbracket B \rrbracket^{[A]} \end{aligned}$$

where  $\epsilon : \llbracket A \rightarrow B \rrbracket \times \llbracket A \rrbracket \rightarrow \llbracket B \rrbracket$  is the evaluation morphism for  $\llbracket B \rrbracket^{[A]}$  and  $(\llbracket \Gamma, x : A \mid t : B \rrbracket)^\sim$  is the transpose of the morphism

$$\llbracket \Gamma, x : A \mid t : B \rrbracket : \llbracket \Gamma \rrbracket \times \llbracket A \rrbracket \rightarrow \llbracket B \rrbracket .$$

**Definition 3.4.1.** An interpretation of a  $\lambda$ -theory  $\mathbb{T}$  is a *model* of  $\mathbb{T}$  if it *satisfies* all the axioms of  $\mathbb{T}$ . This means that, for every axiom  $\Gamma \mid s = t : A$  of  $\mathbb{T}$ , the interpretations of  $s$  and  $t$  coincide as arrows in  $\mathcal{C}$ ,

$$\llbracket \Gamma \mid s : A \rrbracket = \llbracket \Gamma \mid t : A \rrbracket : \llbracket \Gamma \rrbracket \longrightarrow \llbracket A \rrbracket.$$

It now follows that all equations provable in  $\mathbb{T}$  are satisfied in any model, by the following fact.

**Proposition 3.4.2** (Soundness). *If  $\mathbb{T}$  is a  $\lambda$ -theory and  $\llbracket - \rrbracket$  a model of  $\mathbb{T}$  in a cartesian closed category  $\mathcal{C}$ , then for every equation in context  $\Gamma \mid u = v : C$  that is provable from the axioms of  $\mathbb{T}$ , we have*

$$\llbracket \Gamma \mid u : C \rrbracket = \llbracket \Gamma \mid v : C \rrbracket : \llbracket \Gamma \rrbracket \longrightarrow \llbracket C \rrbracket.$$

Briefly, for all  $\mathbb{T}$ -models  $\llbracket - \rrbracket$ ,

$$\mathbb{T} \vdash (\Gamma \mid u = v : C) \quad \text{implies} \quad \llbracket - \rrbracket \models (\Gamma \mid u = v : C).$$

The proof is a straightforward induction, first on the typing judgements for the interpretation, and then on the equational rules for the equations. If we stop after the first step, we can consider just the notion of *inhabitation*:

**Remark 3.4.3** (Inhabitation). There is another notion of provability for the  $\lambda$ -calculus, related to the Curry-Howard correspondence of section 3.1, relating it to propositional logic. If we regard types as “propositions” rather than generalized algebraic structures, and terms as “proofs” rather than operations in such structures, then it is more natural to ask whether there even *is* a term  $a : A$  of some type, than whether two terms of the same type are equal  $s = t : A$ . This only makes sense when  $A$  is considered in the empty context  $\cdot \vdash A$ , rather than  $\Gamma \vdash A$  for non-empty  $\Gamma$  (consider the case where  $\Gamma = x : A, \dots$ ). We say that a type  $A$  is *inhabited* (by a closed term) when there is some  $\vdash a : A$ , and regard an inhabited type  $A$  a *provable*. There is then a different notion of soundness related to this notion of provability.

**Proposition 3.4.4** (Inhabitation soundness). *If  $\mathbb{T}$  is a  $\lambda$ -theory and  $\llbracket - \rrbracket$  a model of  $\mathbb{T}$  in a cartesian closed category  $\mathcal{C}$ , then for every closed type  $A$  that is inhabited in  $\mathbb{T}$ , there is a point  $1 \rightarrow \llbracket A \rrbracket$  in  $\mathcal{C}$ . Thus we have that for all  $\mathbb{T}$ -models  $\llbracket - \rrbracket$ ,*

$$\vdash a : A \quad \text{implies there is a point} \quad 1 \rightarrow \llbracket A \rrbracket.$$

This follows immediately from the fact that  $\llbracket \cdot \rrbracket = 1$  for  $\Gamma = \cdot$ , the empty context. For then given  $\vdash a : A$ , we get a point

$$\llbracket a \rrbracket : 1 \rightarrow \llbracket A \rrbracket.$$



**Example 3.4.5.** 1. A model of an algebraic theory  $\mathbb{A}$ , extended to a  $\lambda$ -theory  $\mathbb{A}^\lambda$  as in Example 3.3.3, taken in a CCC  $\mathcal{C}$ , is just a model of the algebraic theory  $\mathbb{A}$  in the underlying finite product category  $|\mathcal{C}|$  of  $\mathcal{C}$ . A difference, however, is that in defining the *category of models*

$$\mathbf{Mod}_\times(\mathbb{A}, |\mathcal{C}|)$$

we can take all homomorphisms of models of  $\mathbb{A}$  as arrows, while the arrows in the category

$$\mathbf{Mod}_\lambda(\mathbb{A}^\lambda, \mathcal{C})$$

of  $\lambda$ -models are best taken to be isomorphisms, for which one has an obvious way to deal with the contravariance of the function type  $\llbracket A \rightarrow B \rrbracket = \llbracket B \rrbracket^{[A]}$  (this is discussed in more detail in the next section).

2. A model of the theory of a reflexive type, Example 3.3.7, in **Set** must be the one-element  $1$  (prove this!). Fortunately, the exponentials in categories of presheaves are *not* computed pointwise - otherwise it would follow that there would be no non-trivial models at all in small categories! That there are such non-trivial models is an important fact in the semantics of programming languages and the subject called *domain theory*. A fundamental paper in which this is shown is [?].
3. A (positive) propositional theory  $\mathbb{T}$  may be regarded as a  $\lambda$ -theory, and a model in a cartesian closed poset  $P$  is then the same thing as before: an interpretation of the atomic propositions  $p_1, p_2, \dots$  of  $\mathbb{T}$  as elements  $\llbracket p_1 \rrbracket, \llbracket p_2 \rrbracket, \dots \in P$ , such that the axioms  $\phi_1, \phi_2, \dots$  of  $\mathbb{T}$  are all sent to  $1 \in P$  by the extension of  $\llbracket - \rrbracket$  to all formulas,

$$1 = \llbracket \phi_1 \rrbracket = \llbracket \phi_2 \rrbracket = \dots \in P.$$

**Exercise 3.4.6.** How are models of a (not necessarily propositional)  $\lambda$ -theory  $\mathbb{T}^\lambda$  in Cartesian closed *posets* related to models in arbitrary Cartesian closed categories? (*Hint*: Consider the inclusion  $\mathbf{CCPos} \hookrightarrow \mathbf{CCC}$ . Does it have any adjoints?)

## 3.5 Functorial semantics

In Chapter ?? we saw that algebraic theories can be viewed as categories, and models as functors, and we arranged this analysis of the traditional relationship between syntax and semantics into the framework that we called *functorial semantics*. In Chapter ?? We did the same for propositional logic. As a common generalization of both, the same framework of functorial semantics can be applied to  $\lambda$ -theories and their models in CCCs. The first step is to build a *classifying category*  $\mathcal{C}_\mathbb{T}$  from a  $\lambda$ -theory  $\mathbb{T}$ , which again is constructed from the theory itself as a *syntactic category*. This is done as follows:

- The objects of  $\mathcal{C}_\mathbb{T}$  are the types of  $\mathbb{T}$ .

- Morphisms  $A \rightarrow B$  are terms in context

$$[x : A \mid t : B],$$

where two such terms  $x : A \mid t : B$  and  $x : A \mid u : B$  represent the same morphism when  $\mathbb{T}$  proves  $x : A \mid t = u : B$ .

- Composition of the terms

$$[x : A \mid t : B] : A \longrightarrow B \quad \text{and} \quad [y : B \mid u : C] : B \longrightarrow C$$

is the term obtained by substituting  $t$  for  $y$  in  $u$ :

$$[x : A \mid u[t/y] : C] : A \longrightarrow C.$$

- The identity morphism on  $A$  is the term  $[x : A \mid x : A]$ .

**Proposition 3.5.1.** *The syntactic category  $\mathcal{C}_{\mathbb{T}}$  built from a  $\lambda$ -theory is cartesian closed.*

*Proof.* We omit the equivalence classes brackets  $[x : A \mid t : B]$  and treat equivalent terms as equal.

- The terminal object is the unit type  $1$ . For any type  $A$  the unique morphism  $!_A : A \rightarrow 1$  is

$$x : A \mid * : 1.$$

This morphism is unique because

$$\Gamma \mid t = * : 1$$

is an axiom for the terms of unit type  $1$ .

- The product of objects  $A$  and  $B$  is the type  $A \times B$ . The first and the second projections are the terms

$$p : A \times B \mid \mathbf{fst} p : A, \quad p : A \times B \mid \mathbf{snd} p : B.$$

Given morphisms

$$z : C \mid t : A, \quad z : C \mid u : B,$$

the term

$$z : C \mid \langle t, u \rangle : A \times B$$

represents the unique morphism satisfying

$$z : C \mid \mathbf{fst} \langle t, u \rangle = t : A, \quad z : C \mid \mathbf{snd} \langle t, u \rangle = u : B.$$

Indeed, if  $\mathbf{fst} s = t$  and  $\mathbf{snd} s = u$  then

$$s = \langle \mathbf{fst} s, \mathbf{snd} s \rangle = \langle t, u \rangle.$$

- The exponential of objects  $A$  and  $B$  is the type  $A \rightarrow B$  with the evaluation morphism

$$p : (A \rightarrow B) \times A \mid (\mathbf{fst} \, p)(\mathbf{snd} \, p) : B .$$

The transpose of the morphism  $p : C \times A \mid t : B$  is

$$z : C \mid \lambda x : A . (t[\langle z, x \rangle / p]) : A \rightarrow B .$$

Showing that this is the transpose of  $t$  amounts to

$$(\lambda x : A . (t[\langle \mathbf{fst} \, p, x \rangle / p]))(\mathbf{snd} \, p) = t[\langle \mathbf{fst} \, p, \mathbf{snd} \, p \rangle / p] = t[p/p] = t ,$$

which is a valid chain of equations in  $\lambda$ -calculus. The transpose is unique, because any morphism  $z : C \mid s : A \rightarrow B$  that satisfies

$$(s[\mathbf{fst} \, p / z])(\mathbf{snd} \, p) = t$$

is equal to  $\lambda x : A . (t[\langle z, x \rangle / p])$ . First observe that

$$\begin{aligned} t[\langle z, x \rangle / p] &= (s[\mathbf{fst} \, p / z])(\mathbf{snd} \, p)[\langle z, x \rangle / p] = \\ &= (s[\mathbf{fst} \, \langle z, x \rangle / z])(\mathbf{fst} \, \langle z, x \rangle) = (s[z/z]) \, x = s \, x . \end{aligned}$$

Therefore,

$$\lambda x : A . (t[\langle z, x \rangle / p]) = \lambda x : A . (s \, x) = s ,$$

as required. □

The syntactic category allows us to “redefine” models as functors. More precisely, we have the following.

**Lemma 3.5.2.** *A model  $\llbracket - \rrbracket$  of a  $\lambda$ -theory  $\mathbb{T}$  in a cartesian closed category  $\mathcal{C}$  determines a cartesian closed functor  $M : \mathcal{C}_{\mathbb{T}} \rightarrow \mathcal{C}$  with*

$$M(A) = \llbracket A \rrbracket, \quad M(c) = \llbracket c \rrbracket, \tag{3.6}$$

*for all basic types  $A$  and basic constants  $c$ . Moreover,  $M$  is unique up to a unique isomorphism of CCC functors, in the sense that given another model  $N$  satisfying (3.6), there is a unique natural iso  $M \cong N$  determined inductively by the comparison maps  $M(1) \cong N(1)$ ,*

$$M(A \times B) \cong M A \times M B \cong N A \times N B \cong N(A \times B) ,$$

*and similarly for  $M(B^A)$ .*

*Proof.* Straightforward. □

We now have the usual functorial semantics theorem:

**Theorem 3.5.3.** *For any  $\lambda$ -theory  $\mathbb{T}$ , the syntactic category  $\mathcal{C}_{\mathbb{T}}$  classifies  $\mathbb{T}$ -models, in the sense that for any cartesian closed category  $\mathcal{C}$  there is an equivalence of categories*

$$\mathbf{Mod}_{\lambda}(\mathbb{T}, \mathcal{C}) \simeq \mathbf{CCC}(\mathcal{C}_{\mathbb{T}}, \mathcal{C}), \quad (3.7)$$

*naturally in  $\mathcal{C}$ . The morphisms of  $\mathbb{T}$ -models on the left are the isomorphisms of the underlying structures, and on the right we take the natural isomorphisms of CCC functors.*

*Proof.* The only thing remaining to show is that, given a model  $\llbracket - \rrbracket^M$  in a CCC  $\mathcal{C}$  and a CCC functor  $f : \mathcal{C} \rightarrow \mathcal{D}$ , there is an induced model  $\llbracket - \rrbracket^{fM}$  in  $\mathcal{D}$ , given by the interpretation  $\llbracket A \rrbracket^{fM} = f\llbracket A \rrbracket^M$ . This is straightforward, just as for algebraic theories.  $\square$

**Remark 3.5.4.** As mentioned in Example 3.4.5(1), the categories involved in (3.7) are *groupoids*, in which every arrow is iso. The reason we have defined them as such is that the contravariant argument  $A$  in the function type  $A \rightarrow B$  prevents us from specifying a non-iso homomorphism of models  $h : M \rightarrow N$  by the obvious recursion on the type structure. In more detail, given  $h_A : \llbracket A \rrbracket^M \rightarrow \llbracket A \rrbracket^N$  and  $h_B : \llbracket B \rrbracket^M \rightarrow \llbracket B \rrbracket^N$ , there's no obvious map

$$h_{A \rightarrow B} : \llbracket A \rightarrow B \rrbracket^M \longrightarrow \llbracket A \rightarrow B \rrbracket^N,$$

when all we have are the induced maps:

$$\begin{array}{ccccc} \llbracket A \rightarrow B \rrbracket^M & \xrightarrow{=} & (\llbracket B \rrbracket^M)^{\llbracket A \rrbracket^M} & \xrightarrow{(h_B)^{\llbracket A \rrbracket^M}} & (\llbracket B \rrbracket^N)^{\llbracket A \rrbracket^M} \\ & & \uparrow (\llbracket B \rrbracket^M)^{h_A} & & \uparrow (\llbracket B \rrbracket^N)^{h_A} \\ & & (\llbracket B \rrbracket^M)^{\llbracket A \rrbracket^N} & \xrightarrow{(h_B)^{\llbracket A \rrbracket^N}} & (\llbracket B \rrbracket^N)^{\llbracket A \rrbracket^N} \xrightarrow{=} \llbracket A \rightarrow B \rrbracket^N \end{array}$$

One solution is to take *isos*  $h_A : \llbracket A \rrbracket^M \cong \llbracket A \rrbracket^N$  and  $h_B : \llbracket B \rrbracket^M \cong \llbracket B \rrbracket^N$  and then use the inverses  $h_A^{-1} : \llbracket A \rrbracket^N \rightarrow \llbracket A \rrbracket^M$  and  $h_B^{-1} : \llbracket B \rrbracket^N \rightarrow \llbracket B \rrbracket^M$  in the contravariant positions, in order to get things to line up:

$$\begin{array}{ccccc} \llbracket A \rightarrow B \rrbracket^M & \xrightarrow{=} & (\llbracket B \rrbracket^M)^{\llbracket A \rrbracket^M} & \xrightarrow[\sim]{(h_B)^{\llbracket A \rrbracket^M}} & (\llbracket B \rrbracket^N)^{\llbracket A \rrbracket^M} \\ & & \downarrow (\llbracket B \rrbracket^M)^{h_A^{-1}} \sim & & \downarrow \sim (\llbracket B \rrbracket^N)^{h_A^{-1}} \\ & & (\llbracket B \rrbracket^M)^{\llbracket A \rrbracket^N} & \xrightarrow[\sim]{(h_B)^{\llbracket A \rrbracket^N}} & (\llbracket B \rrbracket^N)^{\llbracket A \rrbracket^N} \xrightarrow{=} \llbracket A \rightarrow B \rrbracket^N \end{array}$$

This suffice to at least get a category of models  $\mathbf{Mod}_{\lambda}(\mathbb{T}, \mathcal{C})$ , which is enough structure to determine the equivalence (3.7). Note that for an algebraic theory  $\mathbb{A}$ , this category of models in **Set**, say,  $\mathbf{Mod}_{\lambda}(\mathbb{A}^{\lambda})$  is still the subcategory of isomorphisms of conventional (algebraic)  $\mathbb{A}$ -models  $\mathbf{Mod}(\mathbb{A})$ . We shall consider other solutions to the problem of contravariance below.

We can now proceed just as we did in the case of algebraic theories and prove that the semantics of  $\lambda$ -theories in cartesian closed categories is *complete*, in virtue of the syntactic construction of the classifying category  $\mathcal{C}_{\mathbb{T}}$ . Specifically, a  $\lambda$ -theory  $\mathbb{T}$  has a canonical interpretation  $[-]$  in the syntactic category  $\mathcal{C}_{\mathbb{T}}$ , which interprets a basic type  $A$  as itself, and a basic constant  $c$  of type  $A$  as the morphism  $[x : 1 \mid c : A]$ . The canonical interpretation is a model of  $\mathbb{T}$ , also known as the *syntactic model*, in virtue of the definition of the equivalence relation  $[-]$  on terms. In fact, it is a *logically generic* model of  $\mathbb{T}$ , because by the construction of  $\mathcal{C}_{\mathbb{T}}$ , for any terms  $\Gamma \mid u : A$  and  $\Gamma \mid t : A$ , we have

$$\begin{aligned} \mathbb{T} \vdash (\Gamma \mid u = t : A) &\iff [\Gamma \mid u : A] = [\Gamma \mid t : A] \\ &\iff [-] \models \Gamma \mid u = t : A. \end{aligned}$$

For the record, we therefore have shown:

**Proposition 3.5.5.** *For any  $\lambda$ -theory  $\mathbb{T}$ ,*

$$\mathbb{T} \vdash (\Gamma \mid t = u : A) \quad \text{if, and only if,} \quad [-] \models (\Gamma \mid t = u : A) \text{ for the syntactic model } [-].$$

Of course, the syntactic model  $[-]$  is the one associated under (3.7) to the identity functor  $\mathcal{C}_{\mathbb{T}} \rightarrow \mathcal{C}_{\mathbb{T}}$ , i.e. it is the *universal* one. It therefore satisfies an equation just in case the equation holds in all models, by the classifying property of  $\mathcal{C}_{\mathbb{T}}$ , and the preservation of satisfaction of equations by CCC functors (Proposition 3.4.2).

**Corollary 3.5.6.** *For any  $\lambda$ -theory  $\mathbb{T}$ ,*

$$\mathbb{T} \vdash (\Gamma \mid t = u : A) \quad \text{if, and only if,} \quad M \models (\Gamma \mid t = u : A) \text{ for every CCC model } M.$$

*Moreover, a closed type  $A$  is inhabited  $\vdash a : A$  if, and only if, there is a point  $1 \rightarrow \llbracket A \rrbracket$  in every model  $M$ .*

## 3.6 The internal language of a CCC

In the case of algebraic theories, we were able to recover the syntactic category from the semantics by taking certain **Set**-valued functors on the category of models in **Set**. This extended to a duality between the category of all algebraic theories and that of all “algebraic categories”, which we defined as the categories of **Set**-valued models of some algebraic theory (and also characterized abstractly). In the (classical) propositional case, this syntax-semantics duality was then seen to be exactly the classical Stone duality between the categories of Boolean algebras and of Stone topological spaces. That sort of duality theory seems to be more difficult to formulate for  $\lambda$ -theories, however, now that we have taken the category of models to be just a groupoid (but see Remark ??). Nonetheless, there is still a correspondence between  $\lambda$ -theories and CCCs, which we get by organizing the former into a category, which is then equivalent to that of the latter. But note that this is analogous to the equivalence between algebraic theories, regarded syntactically, and regarded as finite product categories—rather than to the duality between syntax and semantics.

In order to define the equivalence in question, we first need a suitable notion of *morphism of theories*. A *translation*  $\tau : \mathbb{T} \rightarrow \mathbb{U}$  of a  $\lambda$ -theory  $\mathbb{T}$  into a  $\lambda$ -theory  $\mathbb{U}$  is given by the following data:

1. For each basic type  $A$  in  $\mathbb{T}$  a type  $\tau A$  in  $\mathbb{U}$ . The translation is then extended to all types by the rules

$$\tau 1 = 1, \quad \tau(A \times B) = \tau A \times \tau B, \quad \tau(A \rightarrow B) = \tau A \rightarrow \tau B.$$

2. For each basic constant  $c$  of type  $A$  in  $\mathbb{A}$  a term  $\tau c$  of type  $\tau A$  in  $\mathbb{U}$ . The translation of terms is then extended to all terms by the rules

$$\begin{aligned} \tau(\mathbf{fst} \, t) &= \mathbf{fst} \, (\tau t), & \tau(\mathbf{snd} \, t) &= \mathbf{snd} \, (\tau t), \\ \tau\langle t, u \rangle &= \langle \tau t, \tau u \rangle, & \tau(\lambda x : A. t) &= \lambda x : \tau A. \tau t, \\ \tau(t \, u) &= (\tau t)(\tau u), & \tau x &= x \quad (\text{if } x \text{ is a variable}). \end{aligned}$$

A context  $\Gamma = x_1 : A_1, \dots, x_n : A_n$  is translated by  $\tau$  to the context

$$\tau \Gamma = x_1 : \tau A_1, \dots, x_n : \tau A_n.$$

Furthermore, a translation is required to preserve the axioms of  $\mathbb{T}$ : if  $\Gamma \mid t = u : A$  is an axiom of  $\mathbb{T}$  then  $\mathbb{U}$  proves  $\tau \Gamma \mid \tau t = \tau u : \tau A$ . It then follows that all equations proved by  $\mathbb{T}$  are translated to valid equations in  $\mathbb{U}$ .

A moment's consideration shows that a translation  $\tau : \mathbb{T} \rightarrow \mathbb{U}$  is the same thing as a model of  $\mathbb{T}$  in  $\mathcal{C}_{\mathbb{U}}$ , despite being specified entirely syntactically. More precisely,  $\lambda$ -theories and translations between them clearly form a category: translations compose as functions, therefore composition is associative. The identity translation  $\iota_{\mathbb{T}} : \mathbb{T} \rightarrow \mathbb{T}$  translates every type to itself and every constant to itself.

**Definition 3.6.1.** Let  $\lambda\text{Thr}$  be the category whose objects are  $\lambda$ -theories and morphisms are translations between them.

We then have an *isomorphism of sets*,

$$\text{Hom}_{\lambda\text{Thr}}(\mathbb{T}, \mathbb{U}) \cong \text{Mod}_{\lambda}(\mathbb{T}, \mathcal{C}_{\mathbb{U}}), \quad (3.8)$$

which is natural in the theory  $\mathbb{T}$ , as can be seen by considering the canonical interpretation of  $\mathbb{T}$  in  $\mathcal{C}_{\mathbb{T}}$  induced by the identity translation  $\iota_{\mathbb{T}} : \mathbb{T} \rightarrow \mathbb{T}$ .

Let  $\mathcal{C}$  be a small cartesian closed category. There is a  $\lambda$ -theory  $\mathbb{L}(\mathcal{C})$  that corresponds to  $\mathcal{C}$ , called the *internal language of  $\mathcal{C}$* , defined as follows:

1. For every object  $A \in \mathcal{C}$  there is a basic type  $\ulcorner A \urcorner$ .
2. For every morphism  $f : A \rightarrow B$  there is a basic constant  $\ulcorner f \urcorner$  whose type is  $\ulcorner A \urcorner \rightarrow \ulcorner B \urcorner$ .

3. For every  $A \in \mathcal{C}$  there is an axiom

$$x : \ulcorner A \urcorner \mid \ulcorner 1_A \urcorner x = x : \ulcorner A \urcorner .$$

4. For all morphisms  $f : A \rightarrow B$ ,  $g : B \rightarrow C$ , and  $h : A \rightarrow C$  such that  $h = g \circ f$ , there is an axiom

$$x : \ulcorner A \urcorner \mid \ulcorner h \urcorner x = \ulcorner g \urcorner (\ulcorner f \urcorner x) : \ulcorner C \urcorner .$$

5. There is a constant

$$\mathbb{T} : 1 \rightarrow \ulcorner 1 \urcorner ,$$

and for all  $A, B \in \mathcal{C}$  there are constants

$$\mathbb{P}_{A,B} : \ulcorner A \urcorner \times \ulcorner B \urcorner \rightarrow \ulcorner A \times B \urcorner , \quad \mathbb{E}_{A,B} : (\ulcorner A \urcorner \rightarrow \ulcorner B \urcorner) \rightarrow \ulcorner B^A \urcorner .$$

They satisfy the following axioms:

$$\begin{aligned} u : \ulcorner 1 \urcorner \mid \mathbb{T} * &= u : \ulcorner 1 \urcorner \\ z : \ulcorner A \times B \urcorner \mid \mathbb{P}_{A,B} \langle \ulcorner \pi_0 \urcorner z, \ulcorner \pi_1 \urcorner z \rangle &= z : \ulcorner A \times B \urcorner \\ w : \ulcorner A \urcorner \times \ulcorner B \urcorner \mid \langle \ulcorner \pi_0 \urcorner (\mathbb{P}_{A,B} w), \ulcorner \pi_1 \urcorner (\mathbb{P}_{A,B} w) \rangle &= w : \ulcorner A \urcorner \times \ulcorner B \urcorner \\ f : \ulcorner B^A \urcorner \mid \mathbb{E}_{A,B} (\lambda x : \ulcorner A \urcorner . (\ulcorner \text{ev}_{A,B} \urcorner (\mathbb{P}_{A,B} \langle f, x \rangle))) &= f : \ulcorner B^A \urcorner \\ f : \ulcorner A \urcorner \rightarrow \ulcorner B \urcorner \mid \lambda x : \ulcorner A \urcorner . (\ulcorner \text{ev}_{A,B} \urcorner (\mathbb{P}_{A,B} \langle (\mathbb{E}_{A,B} f), x \rangle)) &= f : \ulcorner A \urcorner \rightarrow \ulcorner B \urcorner \end{aligned}$$

The purpose of the constants  $\mathbb{T}$ ,  $\mathbb{P}_{A,B}$ ,  $\mathbb{E}_{A,B}$ , and the axioms for them is to ensure the isomorphisms  $\ulcorner 1 \urcorner \cong 1$ ,  $\ulcorner A \times B \urcorner \cong \ulcorner A \urcorner \times \ulcorner B \urcorner$ , and  $\ulcorner B^A \urcorner \cong \ulcorner A \urcorner \rightarrow \ulcorner B \urcorner$ . Types  $A$  and  $B$  are said to be *isomorphic* if there are terms

$$x : A \mid t : B , \quad y : B \mid u : A ,$$

such that  $\mathbb{T}$  proves

$$x : A \mid u[t/y] = x : A , \quad y : B \mid t[u/x] = y : B .$$

Furthermore, an *equivalence of theories*  $\mathbb{T}$  and  $\mathbb{U}$  is a pair of translations

$$\begin{array}{ccc} & \tau & \\ \mathbb{T} & \xrightleftharpoons{\sigma} & \mathbb{U} \end{array}$$

such that, for any type  $A$  in  $\mathbb{T}$  and any type  $B$  in  $\mathbb{U}$ ,

$$\sigma(\tau A) \cong A , \quad \tau(\sigma B) \cong B .$$

The assignment  $\mathcal{C} \mapsto \mathbb{L}(\mathcal{C})$  extends to a functor

$$\mathbb{L} : \text{CCC} \rightarrow \lambda\text{Thr} ,$$

where  $\text{CCC}$  is the category of small cartesian closed categories and functors between them that preserve finite products and exponentials. Such functors are also called *cartesian closed functors* or *ccc functors*. If  $F : \mathcal{C} \rightarrow \mathcal{D}$  is a cartesian closed functor then  $\mathbb{L}(F) : \mathbb{L}(\mathcal{C}) \rightarrow \mathbb{L}(\mathcal{D})$  is the translation given by:

1. A basic type  $\ulcorner A \urcorner$  is translated to  $\ulcorner FA \urcorner$ .
2. A basic constant  $\ulcorner f \urcorner$  is translated to  $\ulcorner Ff \urcorner$ .
3. The basic constants  $\mathbf{T}$ ,  $\mathbf{P}_{A,B}$  and  $\mathbf{E}_{A,B}$  are translated to  $\mathbf{T}$ ,  $\mathbf{P}_{FA,BA}$  and  $\mathbf{E}_{FA,FB}$ , respectively.

We now have a functor  $\mathbb{L} : \mathbf{CCC} \rightarrow \lambda\mathbf{Thr}$ . How about the other direction? We already have the construction of syntactic category which maps a  $\lambda$ -theory  $\mathbb{T}$  to a small cartesian closed category  $\mathcal{C}_{\mathbb{T}}$ . This extends to a functor

$$\mathcal{C} : \lambda\mathbf{Thr} \rightarrow \mathbf{CCC} ,$$

because a translation  $\tau : \mathbb{T} \rightarrow \mathbb{U}$  induces a functor  $\mathcal{C}_{\tau} : \mathcal{C}_{\mathbb{T}} \rightarrow \mathcal{C}_{\mathbb{U}}$  in an obvious way: a basic type  $A \in \mathcal{C}_{\mathbb{T}}$  is mapped to the object  $\tau A \in \mathcal{C}_{\mathbb{U}}$ , and a basic constant  $x : 1 \mid c : A$  is mapped to the morphism  $x : 1 \mid \tau c : A$ . The rest of  $\mathcal{C}_{\tau}$  is defined inductively on the structure of types and terms.

**Theorem 3.6.2.** *The functors  $\mathbb{L} : \mathbf{CCC} \rightarrow \lambda\mathbf{Thr}$  and  $\mathcal{C} : \lambda\mathbf{Thr} \rightarrow \mathbf{CCC}$  constitute an equivalence of categories “up to equivalence” (a biequivalence of 2-categories). This means that for any  $\mathcal{C} \in \mathbf{CCC}$  there is an equivalence of categories*

$$\mathcal{C} \simeq \mathcal{C}_{\mathbb{L}(\mathcal{C})} ,$$

and for any  $\mathbb{T} \in \lambda\mathbf{Thr}$  there is an equivalence of theories

$$\mathbb{T} \simeq \mathbb{L}(\mathcal{C}_{\mathbb{T}}) .$$

*Proof.* For a small cartesian closed category  $\mathcal{C}$ , consider the functor  $\eta_{\mathcal{C}} : \mathcal{C} \rightarrow \mathcal{C}_{\mathbb{L}(\mathcal{C})}$ , defined for an object  $A \in \mathcal{C}$  and  $f : A \rightarrow B$  in  $\mathcal{C}$  by

$$\eta_{\mathcal{C}} A = \ulcorner A \urcorner , \quad \eta_{\mathcal{C}} f = (x : \ulcorner A \urcorner \mid \ulcorner f \urcorner x : \ulcorner B \urcorner) .$$

To see that  $\eta_{\mathcal{C}}$  is a functor, observe that  $\mathbb{L}(\mathcal{C})$  proves, for all  $A \in \mathcal{C}$ ,

$$x : \ulcorner A \urcorner \mid \ulcorner 1_A \urcorner x = x : \ulcorner A \urcorner$$

and for all  $f : A \rightarrow B$  and  $g : B \rightarrow C$ ,

$$x : \ulcorner A \urcorner \mid \ulcorner g \circ f \urcorner x = \ulcorner g \urcorner (\ulcorner f \urcorner x) : \ulcorner C \urcorner .$$

To see that  $\eta_{\mathcal{C}}$  is an equivalence of categories, it suffices to show that for every object  $X \in \mathcal{C}_{\mathbb{L}(\mathcal{C})}$  there exists an object  $\theta_{\mathcal{C}} X \in \mathcal{C}$  such that  $\eta_{\mathcal{C}}(\theta_{\mathcal{C}} X) \cong X$ . The choice map  $\theta_{\mathcal{C}}$  is defined inductively by

$$\begin{aligned} \theta_{\mathcal{C}} 1 &= 1 , & \theta_{\mathcal{C}} \ulcorner A \urcorner &= A , \\ \theta_{\mathcal{C}}(Y \times Z) &= \theta_{\mathcal{C}} Y \times \theta_{\mathcal{C}} Z , & \theta_{\mathcal{C}}(Y \rightarrow Z) &= (\theta_{\mathcal{C}} Z)^{\theta_{\mathcal{C}} Y} . \end{aligned}$$



We skip the verification that  $\eta_{\mathcal{C}}(\theta_{\mathcal{C}}X) \cong X$ . In fact,  $\theta_{\mathcal{C}}$  can be extended to a functor  $\theta_{\mathcal{C}} : \mathcal{C}_{\mathbb{L}(\mathcal{C})} \rightarrow \mathcal{C}$  so that  $\theta_{\mathcal{C}} \circ \eta_{\mathcal{C}} \cong 1_{\mathcal{C}}$  and  $\eta_{\mathcal{C}} \circ \theta_{\mathcal{C}} \cong 1_{\mathcal{C}_{\mathbb{L}(\mathcal{C})}}$ .

Given a  $\lambda$ -theory  $\mathbb{T}$ , we define a translation  $\tau_{\mathbb{T}} : \mathbb{T} \rightarrow \mathbb{L}(\mathcal{C}_{\mathbb{T}})$ . For a basic type  $A$  let

$$\tau_{\mathbb{T}}A = \ulcorner A \urcorner.$$

The translation  $\tau_{\mathbb{T}}c$  of a basic constant  $c$  of type  $A$  is

$$\tau_{\mathbb{T}}c = \ulcorner x : 1 \mid c : \tau_{\mathbb{T}}A \urcorner.$$

In the other direction we define a translation  $\sigma_{\mathbb{T}} : \mathbb{L}(\mathcal{C}_{\mathbb{T}}) \rightarrow \mathbb{T}$  as follows. If  $\ulcorner A \urcorner$  is a basic type in  $\mathbb{L}(\mathcal{C}_{\mathbb{T}})$  then

$$\sigma_{\mathbb{T}}\ulcorner A \urcorner = A,$$

and if  $\ulcorner x : A \mid t : B \urcorner$  is a basic constant of type  $\ulcorner A \urcorner \rightarrow \ulcorner B \urcorner$  then

$$\sigma_{\mathbb{T}}\ulcorner x : A \mid t : B \urcorner = \lambda x : A . t.$$

The basic constants  $\mathbf{T}$ ,  $\mathbf{P}_{A,B}$  and  $\mathbf{E}_{A,B}$  are translated by  $\sigma_{\mathbb{T}}$  into

$$\begin{aligned} \sigma_{\mathbb{T}}\mathbf{T} &= \lambda x : 1 . x, \\ \sigma_{\mathbb{T}}\mathbf{P}_{A,B} &= \lambda p : A \times B . p, \\ \sigma_{\mathbb{T}}\mathbf{E}_{A,B} &= \lambda f : A \rightarrow B . f. \end{aligned}$$

If  $A$  is a type in  $\mathbb{T}$  then  $\sigma_{\mathbb{T}}(\tau_{\mathbb{T}}A) = A$ . For the other direction, we would like to show, for any type  $X$  in  $\mathbb{L}(\mathcal{C}_{\mathbb{T}})$ , that  $\tau_{\mathbb{T}}(\sigma_{\mathbb{T}}X) \cong X$ . We prove this by induction on the structure of type  $X$ :

1. If  $X = 1$  then  $\tau_{\mathbb{T}}(\sigma_{\mathbb{T}}1) = 1$ .
2. If  $X = \ulcorner A \urcorner$  is a basic type then  $A$  is a type in  $\mathbb{T}$ . We proceed by induction on the structure of  $A$ :
  - (a) If  $A = 1$  then  $\tau_{\mathbb{T}}(\sigma_{\mathbb{T}}\ulcorner 1 \urcorner) = 1$ . The types  $1$  and  $\ulcorner 1 \urcorner$  are isomorphic via the constant  $\mathbf{T} : 1 \rightarrow \ulcorner 1 \urcorner$ .
  - (b) If  $A$  is a basic type then  $\tau_{\mathbb{T}}(\sigma_{\mathbb{T}}\ulcorner A \urcorner) = \ulcorner A \urcorner$ .
  - (c) If  $A = B \times C$  then  $\tau_{\mathbb{T}}(\sigma_{\mathbb{T}}\ulcorner B \times C \urcorner) = \ulcorner B \urcorner \times \ulcorner C \urcorner$ . But we know  $\ulcorner B \urcorner \times \ulcorner C \urcorner \cong \ulcorner B \times C \urcorner$  via the constant  $\mathbf{P}_{A,B}$ .
  - (d) The case  $A = B \rightarrow C$  is similar.
3. If  $X = Y \times Z$  then  $\tau_{\mathbb{T}}(\sigma_{\mathbb{T}}(Y \times Z)) = \tau_{\mathbb{T}}(\sigma_{\mathbb{T}}Y) \times \tau_{\mathbb{T}}(\sigma_{\mathbb{T}}Z)$ . By induction hypothesis,  $\tau_{\mathbb{T}}(\sigma_{\mathbb{T}}Y) \cong Y$  and  $\tau_{\mathbb{T}}(\sigma_{\mathbb{T}}Z) \cong Z$ , from which we easily obtain

$$\tau_{\mathbb{T}}(\sigma_{\mathbb{T}}Y) \times \tau_{\mathbb{T}}(\sigma_{\mathbb{T}}Z) \cong Y \times Z.$$

4. The case  $X = Y \rightarrow Z$  is similar.

□

Composing the isomorphism 3.8 with the equivalence 3.7 we can formulate the foregoing Theorem 3.6.2 as an adjoint equivalence.

**Corollary 3.6.3.** *There is a biequivalence between the categories  $\lambda\text{Thr}$  of  $\lambda$ -theories and translations between them (and isos thereof), and the category  $\text{CCC}$  of cartesian closed categories and CCC functors (and natural isos),*

$$\begin{aligned} \text{Hom}_{\lambda\text{Thr}}(\mathbb{T}, \mathbb{L}\mathcal{C}) &\cong \text{Mod}_{\lambda}(\mathbb{T}, \mathcal{C}) , \\ &\simeq \text{Hom}_{\text{CCC}}(\mathcal{C}_{\mathbb{T}}, \mathcal{C}) . \end{aligned}$$

*This is mediated by an adjunction,*

$$\text{CCC} \begin{array}{c} \xrightarrow{\mathbb{L}} \\ \xleftarrow{\mathcal{C}} \end{array} \lambda\text{Thr}$$

*with  $\mathcal{C} \dashv \mathbb{L}$ , between the syntactic category functor  $\mathcal{C}$  and the internal language functor  $\mathbb{L}$ .*

**Exercise 3.6.4.** In the previous proof we defined, for each  $\mathcal{C} \in \text{CCC}$ , a functor  $\eta_{\mathcal{C}} : \mathcal{C} \rightarrow \mathcal{C}_{\mathbb{L}(\mathcal{C})}$ . Verify that this determines a natural transformation  $\eta : \mathbf{1}_{\text{CCC}} \Longrightarrow \mathcal{C} \circ \mathbb{L}$ . What can we say about the naturality of the translations  $\tau_{\mathbb{T}}$  and  $\sigma_{\mathbb{T}}$ ? What does it even mean for such a translation to be natural?

# Bibliography

- [AK08] Steve Awodey and Kohei Kishida. Topology and modality: Topological semantics for first-order modal logic. *The Review of Symbolic Logic*, 1:146–166, 2008.
- [AR11] S. Awodey and F. Rabe. Kripke semantics for Martin-Löf’s extensional type theory. *Logical Methods in Computer Science*, 7(3):1–25, 2011.
- [Awo00] Steve Awodey. Topological representation of the  $\lambda$ -calculus. *Mathematical Structures in Computer Science*, 10:81–96, 2000.
- [MH92] Michael Makkai and Victor Harnik. Lambek’s categorical proof theory and Läuchli’s abstract realizability. *Journal of Symbolic Logic*, 57(1):200–230, 1992.
- [ML84] Per Martin-Löf. *Intuitionistic type theory*, volume 1 of *Studies in Proof Theory*. Bibliopolis, 1984.
- [Sco70] Dana Scott. Constructive validity. In M. Laudet, D. Lacombe, L. Nolin, and M. Schützenberger, editors, *Symposium on Automatic Demonstration*, volume 125, pages 237–275. Springer-Verlag, 1970.
- [Tai68] William W. Tait. Constructive reasoning. In *Logic, Methodology and Philos. Sci. III (Proc. Third Internat. Congr., Amsterdam, 1967)*, pages 185–199. North-Holland, Amsterdam, 1968.