Notes on Type Theory

[DRAFT: January 23, 2025]

Steve Awodey

with contributions from Andrej Bauer

Contents

Chapter 2

Simple Type Theory

2.1 The λ-calculus

The λ-calculus is an abstract theory of functions, much like group theory is an 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, which takes an argument x to the value t. But we could have expressed the same function without giving it a name; this is sometimes written as

$$x \mapsto t\ ,$$

and it means "x is mapped to t". In λ-calculus we use a different notation, which is more convenient when such abstractions are nested within more complex expressions, namely

$$\lambda x.\, t\ .$$

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

It may seem strange that in discussing 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 λ-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 this chapter (and, indeed, the entire subject of logic!).

There are two kinds of λ-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 allowed, whatever it may mean. We will concentrate here on the typed λ-calculus (but see Example 2.1.7 below). In typed λ-calculus every expression has a *type*, and there are rules for forming valid expressions and assigning types. For example, we can only form an application fa when f has, say, type $A \to B$ and a has type A, and then fa will necessarily have type B. The basic judgement that an expression t has a type T is written as

$$t : T$$

and it is one of the primitive notions of type theory (meaning that it is not defined). 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 $t \in A$ (at least to get started).

Simply-typed λ-calculus. We now give a more formal definition of what constitutes a *simply-typed λ-calculus*. First, we are given a collection of *simple types*, which are generated from some *basic types* by formation of product and function types:

$$\begin{aligned} \text{Basic types} \quad & \mathtt{B} ::= \mathtt{B}_0 \mid \mathtt{B}_1 \mid \mathtt{B}_2 \cdots \\ \text{Simple types} \quad & A ::= \mathtt{B} \mid A_1 \times A_2 \mid A_1 \to A_2. \end{aligned}$$

When convenient, we may adopt the convention that function types associate to the right,

$$A \to B \to C \;=\; A \to (B \to C)\ .$$

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

$$\begin{aligned} \text{Variables} \quad & v ::= x \mid y \mid z \mid \cdots \\ \text{Constants} \quad & c ::= \mathtt{c}_1 \mid \mathtt{c}_2 \mid \cdots \\ \text{Terms} \quad & t ::= v \mid c \mid * \mid \langle t_1, t_2 \rangle \mid \mathtt{fst}\, t \mid \mathtt{snd}\, t \mid t_1\, t_2 \mid \lambda x : A\,.\, t \end{aligned}$$

In words, this means:

1. any variable is a term,
2. each basic constant is a term,
3. the constant $*$ is a term, called the *unit*,
4. if s and t are terms then $\langle s, t \rangle$ is a term, called a *pair*,

5. if t is a term then $\mathtt{fst}\, t$ and $\mathtt{snd}\, t$ are terms,

6. if s and t are terms then $s\, t$ 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 *λ-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* $\mathsf{FV}(t)$ of a term t is determined as follows:

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

A term t is *closed* if all of its variables are bound, so that $\mathsf{FV}(t) = \emptyset$. If $x_1, \ldots, x_n$ are *distinct* variables and $A_1, \ldots, A_n$ are types then the sequence

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

is a *typing context*, or just *context*. The empty sequence is sometimes denoted by a dot $\cdot$, and it is a valid context. We may identify contexts under reordering, regarding them as sets rather than sequences. Contexts may be denoted by capital Greek letters Γ, Δ, ...

A *typing judgment* is a judgment of the form

$$\Gamma \mid t : A$$

where Γ is a context, t is a term, and A is a type. In addition, the free variables of t must occur in Γ, but Γ may contain other variables as well. We read the above judgment as "in context Γ 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):

$$\frac{}{\Gamma \mid \mathsf{c}_i : C_i}$$

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

$$\frac{}{\Gamma, x_n : A_n \mid x_n : A_n}$$

- The constant $*$ has type $\mathtt{1}$:

$$\frac{}{\Gamma \mid * : \mathtt{1}}$$

- The typing rules for pairs and projections are:

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

- The typing rules for application and λ-abstraction are:

$$\frac{\Gamma \mid t : A \to B \qquad \Gamma \mid a : A}{\Gamma \mid t\, a : B} \qquad \frac{\Gamma, x : A \mid t : B}{\Gamma \mid (\lambda x : A\,.\, t) : A \to B}$$

Lastly, we have *equations* between terms: for terms of type A in context Γ,

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

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:

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

- The substitution rule:

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

- The weakening rule:

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

- Unit type:

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

- Equations for product types:

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

$$\frac{\Gamma \mid s = t : A \times B}{\Gamma \mid \mathtt{fst}\, s = \mathtt{fst}\, t : A} \qquad \frac{\Gamma \mid s = t : A \times B}{\Gamma \mid \mathtt{snd}\, s = \mathtt{snd}\, t : A}$$

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

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

- Equations for function types:

$$\frac{\Gamma \mid s = t : A \to B \qquad \Gamma \mid u = v : A}{\Gamma \mid s\, u = t\, v : B}$$

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

$$\frac{\Gamma \mid t : A \to B}{\Gamma \mid \lambda x : A\,.\,(t\, x) = t : A \to B} \qquad (\eta\text{-rule})$$

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

  where the substitution $t[u/x]$ is defined as usual (see the Appendix).

This completes the description of a simply-typed λ-calculus.

Simply-typed λ-theories. Apart from the above rules for equality, which are part of the λ-calculus, we might want to impose additional equations between terms. In this case we speak of a λ-*theory*. Thus, a λ-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

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

Note that we can always state the equations equivalently in *closed* form simply by λ-abstracting all the variables in the context Γ.

We summarize the preceding definitions.

Definition 2.1.1. A *(simply-typed) signature* S is given by a set of *basic types* $(\mathtt{B}_i)_{i\in I}$ together with a set of *basic (typed) constants* $(\mathtt{c}_j : C_j)_{j\in J}$,

$$S \;=\; \big((\mathtt{B}_i)_{i\in I},\, (\mathtt{c}_j : C_j)_{j\in J}\big)\,.$$

A simply-typed λ-theory $\mathbb{T} = (S, E)$ is a simply-typed signature S together with a set of equations between closed terms,

$$E = \big(u_k = v_k : A_k\big)_{k\in K}\,.$$

Example 2.1.2. The theory of a group is a simply-typed λ-theory. It has one basic type $\mathtt{G}$ and three basic constants, the unit $\mathtt{e}$, the inverse $\mathtt{i}$, and the group operation $\mathtt{m}$,

$$\mathtt{e} : \mathtt{G}\,, \qquad \mathtt{i} : \mathtt{G} \to \mathtt{G}\,, \qquad \mathtt{m} : \mathtt{G} \times \mathtt{G} \to \mathtt{G}\,,$$

with the following familiar equations (which we need not give in closed form):

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

Example 2.1.3. More generally, any (Lawvere) algebraic theory $\mathbb{A}$ (as in Chapter **??**) determines a λ-theory $\mathbb{A}^\lambda$. There is one basic type $\mathtt{A}$ and for each operation f of arity k there is a basic constant $\mathtt{f} : \mathtt{A}^k \to \mathtt{A}$, where $\mathtt{A}^k$ is the k-fold product $\mathtt{A} \times \cdots \times \mathtt{A}$. It is understood that $\mathtt{A}^0 = \mathtt{1}$. The terms of $\mathbb{A}$ are translated to corresponding terms of $\mathbb{A}^\lambda$ in a straightforward manner. For every axiom $u = v$ of $\mathbb{A}$ there is a corresponding one in $\mathbb{A}^\lambda$,

$$x_1 : \mathtt{A}, \ldots, x_n : \mathtt{A} \mid u = v : \mathtt{A}$$

where $x_1, \ldots, x_n$ are the variables occurring in u and v.

Example 2.1.4. The theory of a directed graph is a simply-typed theory with two basic types, $\mathtt{V}$ for vertices and $\mathtt{E}$ for edges, and two basic constants, source $\mathtt{src}$ and target $\mathtt{trg}$,

$$\mathtt{src} : \mathtt{E} \to \mathtt{V}\,, \qquad \mathtt{trg} : \mathtt{E} \to \mathtt{V}\,.$$

There are no equations.

Example 2.1.5. The theory of a simplicial set is a simply-typed theory with one basic type $\mathtt{X}_n$ for each natural number n, and the following basic constants, also for each n, and each $0 \leq i \leq n$:

$$\mathtt{d}_i : \mathtt{X}_{n+1} \to \mathtt{X}_n\,, \qquad \mathtt{s}_i : \mathtt{X}_n \to \mathtt{X}_{n+1}\,.$$

The equations are the usual simplicial identities, which are as follows, for all natural numbers i, j:

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

Example 2.1.6. An example of a λ-theory found in the theory of programming languages is the mini-programming language *PCF*. It is a theory in simply-typed λ-calculus with a basic type `nat` for natural numbers, and a basic type `bool` of Boolean values,

$$\text{Basic types} \quad \mathtt{B} ::= \mathtt{nat\ type} \mid \mathtt{bool\ type}.$$

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

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

The equational axioms of PCF are:

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

Example 2.1.7 (D.S. Scott)**.** Another example of a λ-theory is the *theory of a reflexive type*. This theory has one basic type `D` and two constants

$$\mathtt{r} : \mathtt{D} \to \mathtt{D} \to \mathtt{D} \qquad\qquad \mathtt{s} : (\mathtt{D} \to \mathtt{D}) \to \mathtt{D}$$

satisfying the equation

$$f : \mathtt{D} \to \mathtt{D} \mid \mathtt{r}\,(\mathtt{s}\,f) = f : \mathtt{D} \to \mathtt{D} \tag{2.1}$$

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

$$x : \mathtt{D} \mid \mathtt{s}\,(\mathtt{r}\,x) = x : \mathtt{D} \tag{2.2}$$

which implies that `D` is isomorphic to $\mathtt{D} \to \mathtt{D}$.

A reflexive type can be used to interpret the untyped λ-calculus into the typed λ-calculus.

Untyped λ-calculus

We briefly describe the *untyped λ-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 $t\,t'$ is a term, for any terms t and t', and a λ-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,

$$x_1, \ldots, x_n\ .$$

We say that a term t is valid in context Γ if the free variables of t are listed in Γ. 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 Γ. The context Γ 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:

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

2. The weakening rule:

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

3. Equations for application and λ-abstraction:

$$\frac{\Gamma \mid s = t \qquad \Gamma \mid u = v}{\Gamma \mid s\,u = t\,v} \qquad \frac{\Gamma, x \mid t = u}{\Gamma \mid \lambda x.\, t = \lambda x.\, u}$$

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

$$\frac{}{\Gamma \mid \lambda x.\, (t\,x) = t} \text{ if } x \notin \mathsf{FV}(t) \qquad (\eta\text{-rule})$$

The untyped λ-calculus can be translated into the theory of a reflexive type from Example 2.1.7. An untyped context Γ is translated to a typed context Γ^* by typing each variable in Γ with the reflexive type $\mathtt{D}$, i.e., a context $x_1, \ldots, x_k$ is translated to $x_1 : \mathtt{D}, \ldots, x_k : \mathtt{D}$. An untyped term t is translated to a typed term t^* as follows:

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

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

$$\Gamma \mid u = t \tag{2.3}$$

is translated to the judgment

$$\Gamma^* \mid u^* = t^* : \mathtt{D}\,. \tag{2.4}$$

Exercise* 2.1.8. Prove that if equation (2.3) is provable then equation (2.4) is provable as well. Identify precisely at which point in your proof you need to use equations (2.1) and (2.2). Does provability of (2.4) imply provability of (2.3)?

2.2 Cartesian closed categories

We next review of the theory of cartesian closed categories, which will form the basis for the semantics of simple type theory.

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 $\mathsf{Hom}(P, Q)$ of all monotone functions between them is again a poset, with the pointwise order:

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

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

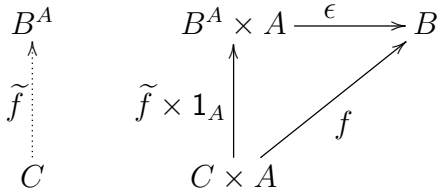
Similarly, given monoids $K, M \in \mathsf{Mon}$, there is a natural monoid structure on the set $\mathsf{Hom}(K, M)$, defined pointwise by

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

Thus the category Mon also admits such "internal Homs". The same thing works in the category Group of groups and group homomophisms, where the set $\mathsf{Hom}(G, H)$ 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 $A \to B$" which corresponds to the hom-set $\mathsf{Hom}(A, B)$. The other ingredient needed is an "evaluation" operation $\mathsf{eval} : B^A \times A \to B$ which evaluates a morphism $f \in B^A$ at an argument $a \in A$ to give a value $\mathsf{eval} \circ \langle f, a\rangle = f(a) \in B$. This is always going to be present as an operation on underlying sets, if we're starting from a set of functions $\mathsf{Hom}(A, B)$ 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 $f : C \times A \to B$ to one $\widetilde{f} : C \to A^B$. We shall see that this in fact separates the previous two examples.

Definition 2.2.1. In a category $\mathcal{C}$ with binary products, an *exponential* (B^A, ϵ) of objects A and B is an object B^A together with a morphism $\epsilon : B^A \times A \to B$, called the *evaluation* morphism, such that for every $f : C \times A \to B$ there exists a *unique* morphism $\widetilde{f} : C \to B^A$, called the *transpose*[1] of f, for which the following diagram commutes.



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

Definition 2.2.1 is called the *universal property of the exponential*. It is just the category-theoretic way of saying that a function $f : C \times A \to B$ of two variables can be viewed as a function $\widetilde{f} : C \to B^A$ of one variable that maps $z \in C$ to a function $\widetilde{f}z = f\langle z, - \rangle : A \to B$ that maps $x \in A$ to $f\langle z, x \rangle$. The relationship between f and $\widetilde{f}$ is then the expected one:

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

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

Definition 2.2.2. 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 $\epsilon : B^A \times A \to B$). 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 2.2.3. Consider again the example of the set $\mathsf{Hom}(M, N)$ of homomorphisms between two monoids M, N, equipped with the pointwise monoid structure. To be a monoid homomorphism, the transpose $\widetilde{h} : 1 \to \mathsf{Hom}(M, N)$ of a homomorphism $h : 1 \times M \to N$ would have to take the unit element $u \in 1$ to the unit homomorphism $u : M \to N$, which is the constant function at the unit $u \in N$. Since $1 \times M \cong M$, that would mean that *all* homomorphisms $h : M \to N$ would have the same transpose, namely $\widetilde{h} = u : 1 \to \mathsf{Hom}(M, N)$. So Mon cannot be cartesian closed. The same argument works in the category Group, and in many related ones.

[1] Also, f is called the transpose of $\widetilde{f}$, so that f and $\widetilde{f}$ are each other's transpose.

Exercise 2.2.4. Recall that monoids and groups can be regarded as (1-object) categories, and then their homomorphisms are just functors. So we have full subcategories,

$$\mathsf{Mon} \hookrightarrow \mathsf{Group} \hookrightarrow \mathsf{Cat}\,.$$

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

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

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

Two characterizations of CCCs

Proposition 2.2.5. *In a category $\mathcal{C}$ with binary products an object A is exponentiable if, and only if, the functor*

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

has a right adjoint

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

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

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

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

as indicated below.

$$\begin{array}{ccc} B^A \times A & \xrightarrow{\epsilon_B} & B \\ {\scriptstyle g^A \times \mathsf{1}_A}\downarrow & & \downarrow{\scriptstyle g} \\ C^A \times A & \xrightarrow[\epsilon_C]{} & C \end{array} \tag{2.5}$$

The counit $\epsilon : -^A \times A \to \mathsf{1}_{\mathcal{C}}$ at B is then ϵ_B itself, and the naturality square for ϵ is then exactly (2.5), i.e. the defining property of $(f \circ \epsilon_B)^\sim$:

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

The universal property of the counit ϵ is precisely the universal property of the exponential (B^A, ϵ_B) □

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

$$\begin{aligned} !_{\mathcal{C}} &: \mathcal{C} \to \mathsf{1} \ , \\ \Delta &: \mathcal{C} \to \mathcal{C} \times \mathcal{C} \ , \\ (- \times A) &: \mathcal{C} \to \mathcal{C} \ . \end{aligned} \qquad (A \in \mathcal{C})$$

Here $!_{\mathcal{C}}$ is the unique functor from $\mathcal{C}$ to the terminal category $\mathsf{1}$ and Δ 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 2.2.7 (Equational version of CCC)**.** *A category $\mathcal{C}$ is cartesian closed if, and only if, it has the following structure:*

1. *An object $\mathsf{1} \in \mathcal{C}$ and a morphism $!_A : A \to \mathsf{1}$ for every $A \in \mathcal{C}$.*

2. *An object $A \times B$ for all $A, B \in \mathcal{C}$ together with morphisms $\pi_0 : A \times B \to A$ and $\pi_1 : A \times B \to B$, and for every pair of morphisms $f : C \to A$, $g : C \to B$ a morphism $\langle f, g \rangle : C \to A \times B$.*

3. *An object B^A for all $A, B \in \mathcal{C}$ together with a morphism $\epsilon : B^A \times A \to B$, and a morphism $\widetilde{f} : C \to B^A$ for every morphism $f : C \times A \to B$.*

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

1. *For every $f : A \to \mathsf{1}$,*
$$f = !_A \ .$$

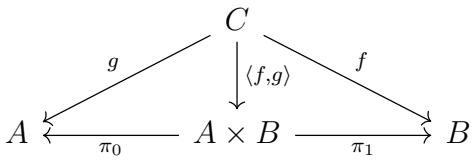
2. *For all $f : C \to A$, $g : C \to B$, $h : C \to A \times B$,*
$$\pi_0 \circ \langle f, g \rangle = f \ , \qquad \pi_1 \circ \langle f, g \rangle = g \ , \qquad \langle \pi_0 \circ h, \pi_1 \circ h \rangle = h \ .$$

3. *For all $f : C \times A \to B$, $g : C \to B^A$,*
$$\epsilon \circ (\widetilde{f} \times \mathsf{1}_A) = f \ , \qquad (\epsilon \circ (g \times \mathsf{1}_A))^{\sim} = g \ .$$
where for $e : E \to E'$ and $f : F \to F'$ we define
$$e \times f := \langle e\pi_0, f\pi_1 \rangle : E \times F \to E' \times F'.$$

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 $f : C \to A$ and $g : C \to B$ we have the morphism $\langle f, g\rangle : C \to A \times B$. The equations

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

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



Suppose $h : C \to A \times B$ 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 2.2.8. Use the equational characterization of CCCs, Proposition 2.2.7, to show that the category Pos of posets and monotone functions *is* cartesian closed, as claimed. Also verify that that Mon is not. Which parts of the definition fail in Mon?

Exercise 2.2.9. Use the equational characterization of CCCs, Proposition 2.2.7, to show that the product category $\Pi_{i\in I}\,\mathcal{C}_i$ 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 2.2.10. 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,

$$Y^X = \left\{ f \subseteq X \times Y \;\middle|\; \forall\, x : X \,.\, \exists!\, y : Y \,.\, \langle x, y\rangle \in f \right\} \ .$$

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

Example 2.2.11. The category 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 Cat. Since limits of groupoids in Cat are also groupoids, the inclusion of the full subcategory $\mathsf{Grpd} \hookrightarrow \mathsf{Cat}$ preserves limits. It also preserves the CCC structure.

Example 2.2.12. The same reasoning as in the previous example shows that the full subcategory $\mathsf{Pos} \hookrightarrow \mathsf{Cat}$ of all small posets and monotone maps is also cartesian closed, and the (limit preserving) inclusion $\mathsf{Pos} \hookrightarrow \mathsf{Cat}$ also preserves exponentials. Note that the (non-full) forgetful functor $U : \mathsf{Pos} \to \mathsf{Set}$ does not, and that $U(Q^P) \subseteq (UQ)^{UP}$ is in general a *proper* subset.

Exercise 2.2.13. There is a full and faithful functor $I : \mathsf{Set} \to \mathsf{Poset}$ that preserves finite limits as well as exponentials. Note the similarity to the example $\mathsf{Grpd} \hookrightarrow \mathsf{Cat}$.

The foregoing examples are instances of the following general situation.

Proposition 2.2.14. *Let $\mathcal{E}$ be a CCC and $i : \mathcal{S} \hookrightarrow \mathcal{E}$ a full subcategory with finite products and a left adjoint reflection $L : \mathcal{E} \to \mathcal{S}$ 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:

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

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 $L(iB^{iA}) = LiE \cong E$. □

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

Proposition 2.2.15. *Let $\mathcal{E}$ be a CCC and $i : \mathcal{S} \hookrightarrow \mathcal{E}$ a full subcategory with finite products and a right adjoint reflection $R : \mathcal{E} \to \mathcal{S}$. 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:

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

□

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 $|X| \to X$ of locales (or in this case, spaces) from the discrete space on the set of points of X.

Exercise 2.2.16. Which of the examples follows from which proposition?

Example 2.2.17. For any set X, the slice category $\mathsf{Set}/_X$ is cartesian closed. The product of $f : A \to X$ and $g : B \to X$ is the pullback $A \times_X B \to X$, which can be constructed as the set of pairs

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

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

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

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

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

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

Exercise 2.2.18. Prove that $\mathsf{Set}/_X$ is always cartesian closed.

Example 2.2.19. 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 $A \in \mathbb{C}$,

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

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

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

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

$$E_A : \mathsf{Nat}(\mathsf{y}A \times P, Q) \times PA \to QA \ ,$$
$$E_A : \langle \eta, x \rangle \mapsto \eta_A \langle \mathsf{1}_A, x \rangle \ .$$

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

$$(\widetilde{\phi}_A z)_B : \mathcal{C}(B, A) \times PB \to QB \ ,$$
$$(\widetilde{\phi}_A z)_B : \langle f, y \rangle \mapsto \phi_B \langle (Rf)z, y \rangle \ .$$

Exercise 2.2.20. 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 $\mathsf{Set}^{\cdot \rightrightarrows \cdot}$. The same is of course true for the "category of functions", i.e. the arrow category $\mathsf{Set}^{\to}$, as well as the category of simplicial sets $\mathsf{Set}^{\Delta^{\mathsf{op}}}$ from topology.

Exercise 2.2.21. 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 $\mathcal{L}(X, -) : \mathsf{Vec} \to \mathsf{Vec}$ to be the functor which maps a vector space Y to the vector space $\mathcal{L}(X, Y)$, and it maps a linear map $f : Y \to Z$ to the linear map $\mathcal{L}(X, f) : \mathcal{L}(X, Y) \to \mathcal{L}(X, Z)$ 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.

We will meet a couple of further examples later in this chapter:

- Etale spaces over a base space X. This category can be described as consisting of *local homeomorphisms* $f : Y \to X$ and commutative triangles over X between such maps. It is equivalent to the category $\mathsf{Sh}(X)$ of *sheaves* on X (Section **??**).
- Dana Scott's category Equ of equilogical spaces (Section **??**).

2.3 Interpretation of the λ-calculus in a CCC

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

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

$$[\![\mathsf{1}]\!] = 1 \ , \qquad [\![A \times B]\!] = [\![A]\!] \times [\![B]\!] \ , \qquad [\![A \to B]\!] = [\![B]\!]^{[\![A]\!]} \ .$$

- For every basic constant $\mathtt{c}$ of type C, a morphism $[\![\mathtt{c}]\!] : \mathbf{1} \to [\![\mathtt{C}]\!]$.

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

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

$$[\![A_1]\!] \times \cdots \times [\![A_n]\!] ,$$

  and the empty context is interpreted as the terminal object,

$$[\![\cdot]\!] = \mathbf{1} .$$

- A typing judgment

$$\Gamma \mid t : A$$

  will be interpreted as a morphism

$$[\![\Gamma \mid t : A]\!] : [\![\Gamma]\!] \to [\![A]\!] .$$

The interpretation is defined inductively by the following rules:

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

$$[\![x_0 : A_0, \ldots, x_n : A_n \mid x_i : A_i]\!] = \pi_i : [\![\Gamma]\!] \to [\![A_i]\!] .$$

- A basic constant $\mathtt{c} : \mathtt{C}$ in context Γ is interpreted as the composition

$$[\![\Gamma]\!] \xrightarrow{!_{[\![\Gamma]\!]}} \mathbf{1} \xrightarrow{[\![\mathtt{c}]\!]} [\![A]\!]$$

- The interpretation of projections and pairs is

$$\begin{aligned} [\![\Gamma \mid \langle t, u\rangle : A \times B]\!] &= \langle [\![\Gamma \mid t : A]\!], [\![\Gamma \mid u : B]\!]\rangle : [\![\Gamma]\!] \to [\![A]\!] \times [\![B]\!] \\ [\![\Gamma \mid \mathtt{fst}\, t : A]\!] &= \pi_0 \circ [\![\Gamma \mid t : A \times B]\!] : [\![\Gamma]\!] \to [\![A]\!] \\ [\![\Gamma \mid \mathtt{snd}\, t : A]\!] &= \pi_1 \circ [\![\Gamma \mid t : A \times B]\!] : [\![\Gamma]\!] \to [\![B]\!] . \end{aligned}$$

- The interpretation of application and λ-abstraction is

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

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

$$[\![\Gamma, x : A \mid t : B]\!] : [\![\Gamma]\!] \times [\![A]\!] \to [\![B]\!] .$$

Definition 2.3.1. An interpretation of a λ-theory $\mathbb{T}$ is a *model* of $\mathbb{T}$ if it *satisfies* all the axioms of $\mathbb{T}$, in the sense that for every axiom $\Gamma \mid u = v : A$ of $\mathbb{T}$, the interpretations of u and v coincide as arrows in $\mathcal{C}$,

$$[\![\Gamma \mid u : A]\!] = [\![\Gamma \mid v : A]\!] : [\![\Gamma]\!] \longrightarrow [\![A]\!].$$

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

Proposition 2.3.2 (Soundness)**.** *If $\mathbb{T}$ is a λ-theory and $[\![-]\!]$ a model of $\mathbb{T}$ in a cartesian closed category $\mathcal{C}$, then for every equation in context $\Gamma \mid s = t : C$ that is provable from the axioms of $\mathbb{T}$, we have*

$$[\![\Gamma \mid s : C]\!] = [\![\Gamma \mid t : C]\!] : [\![\Gamma]\!] \longrightarrow [\![C]\!]\,.$$

Briefly, for all $\mathbb{T}$-models $[\![-]\!]$,

$$\mathbb{T} \vdash (\Gamma \mid s = t : C) \quad \textit{implies} \quad [\![-]\!] \models (\Gamma \mid s = t : C)\,.$$

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 following notion of *inhabitation*:

Remark 2.3.3 (Inhabitation)**.** There is another notion of provability for the λ-calculus, related to the Curry-Howard correspondence of section **??**, 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 $s = t : A$. Of course, this only makes sense when A is considered in the empty context $\cdot \vdash A$, rather than $\Gamma \vdash A$ for non-empty Γ (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 as one that is *provable*. There is then a different notion of soundness related to this notion of provability.

Proposition 2.3.4 (Inhabitation soundness)**.** *If $\mathbb{T}$ is a λ-theory and $[\![-]\!]$ a model of $\mathbb{T}$ in a cartesian closed category $\mathcal{C}$, then for every type A that is inhabited in $\mathbb{T}$, there is a point $1 \to [\![A]\!]$ in $\mathcal{C}$. Thus for all $\mathbb{T}$-models $[\![-]\!]$,*

$$\vdash a : A \quad \textit{implies there is a point} \quad 1 \to [\![A]\!]\,.$$

This follows immediately from the fact that $[\![\cdot]\!] = 1$ for the empty context; for then the interpretation of any $\vdash a : A$ is a point

$$[\![a]\!] : 1 \to [\![A]\!]\,.$$

Example 2.3.5. 1. A model of an algebraic theory $\mathbb{A}$, extended to a λ-theory $\mathbb{A}^\lambda$ as in Example 2.1.3, taken in a CCC $\mathcal{C}$, is just a model of the algebraic theory $\mathbb{A}$ in the

underlying finite product category $|\mathcal{C}|_\times$ of $\mathcal{C}$. An important difference, however, is that in defining the *category of models*

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

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

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

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

2. A model of the theory of a reflexive type, Example 2.1.7, in Set must be the one-element set $1 = \{\star\}$ (prove this!). Fortunately, the exponentials in categories of presheaves are *not* computed pointwise; otherwise it would follow that this theory has no non-trivial models at all! (And then, by Theorem **??**, that the theory itself is degenerate, in the sense that all equations are provable.) That there are 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 λ-theory, and a model in a cartesian closed poset P is then the same thing as before: an interpretation of the atomic propositions $p_1, p_2, ...$ of $\mathbb{T}$ as elements $[\![p_1]\!], [\![p_2]\!], ... \in P$, such that the axioms $\phi_1, \phi_2, ...$ of $\mathbb{T}$ are all sent to $1 \in P$ by the extension of $[\![-]\!]$ to all formulas,

$$1 = [\![\phi_1]\!] = [\![\phi_2]\!] = \cdots \in P\,.$$

Exercise 2.3.6. How are models of a (not necessarily propositional) λ-theory $\mathbb{T}$ in Cartesian closed *posets* related to models in arbitrary Cartesian closed categories? (*Hint:* Consider the inclusion $\mathsf{CCPos} \hookrightarrow \mathsf{CCC}$. Does it have any adjoints?)

2.4 Functorial semantics

In Chapter **??** we saw how algebraic theories can be viewed as categories (with finite products), and their algebras, or models, as functors (preserving finite products), and we arranged this analysis of the traditional relationship between syntax and sematics into a 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 λ-theories and their models in CCCs. The first step is to build a *classifying category* $\mathcal{C}_\mathbb{T}$ from a λ-theory $\mathbb{T}$, which again is constructed from the theory itself as a syntactic category. This is done as follows:

Definition 2.4.1. For any λ-theory $\mathbb{T}$, the *syntactic category* $\mathcal{C}_\mathbb{T}$ is determined as follows.

- The objects of $\mathcal{C}_{\mathbb{T}}$ are the types of $\mathbb{T}$.
- Arrows $A \to B$ are terms in context
$$[x : A \mid t : B] \,,$$
where two such terms $x : A \mid s : B$ and $x : A \mid t : B$ represent the same morphism when $\mathbb{T}$ proves $x : A \mid s = t : B$.
- Composition of the terms
$$[x : A \mid t : B] : A \longrightarrow B \qquad \text{and} \qquad [y : B \mid u : C] : B \longrightarrow C$$
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 2.4.2. *The syntactic category $\mathcal{C}_{\mathbb{T}}$ built from a λ-theory is cartesian closed.*

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

- The terminal object is the unit type $\mathtt{1}$. For any type A the unique morphism $!_A : A \to \mathtt{1}$ is the term
$$x : A \mid * : \mathtt{1} \,.$$
This morphism is indeed unique, because we have the equation
$$\Gamma \mid t = * : \mathtt{1}$$
is an axiom for the terms of unit type $\mathtt{1}$.
- The product of objects A and B is the type $A \times B$. The first and the second projections are the terms
$$c : A \times B \mid \mathtt{fst}\, c : A \,, \qquad\qquad c : A \times B \mid \mathtt{snd}\, c : B \,.$$
Given morphisms
$$z : C \mid a : A \,, \qquad\qquad z : C \mid b : B \,,$$
the term
$$z : C \mid \langle a, b \rangle : A \times B$$
represents the unique morphism satisfying
$$z : C \mid \mathtt{fst}\, \langle a, b \rangle = a : A \,, \qquad\qquad z : C \mid \mathtt{snd}\, \langle a, b \rangle = b : B \,.$$
Indeed, if $\mathtt{fst}\, t = a$ and $\mathtt{snd}\, t = b$ for some t, then
$$t = \langle \mathtt{fst}\, t, \mathtt{snd}\, t \rangle = \langle a, b \rangle \,.$$

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

$$e : (A \to B) \times A \mid (\mathtt{fst}\, e)(\mathtt{snd}\, e) : B\ .$$

The transpose of a morphism $w : C \times A \mid t : B$ is the term

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

Showing that this is the transpose of t amounts to showing, in context $w : C \times A$,

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

Indeed, we have:

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

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

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

is equal to $\lambda x : A\,.\,(t[\langle z, x\rangle/w])$, because then

$$t[\langle z, x\rangle/w] = (s[\mathtt{fst}\, w/z])(\mathtt{snd}\, w)[\langle z, x\rangle/w] = (s[\mathtt{fst}\, \langle z, x\rangle/z])(\mathtt{snd}\, \langle z, x\rangle) = (s[z/z])\, x = s\, x\ .$$

Therefore,

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

as claimed.

□

Now as before, the syntactic category allows us to replace a model $[\![-]\!]$ in a CCC $\mathcal{C}$ with a functor $M : \mathcal{C}_{\mathbb{T}} \to \mathcal{C}$. More precisely, we have the following.

Lemma 2.4.3. *A* model $[\![-]\!]$ *of a* λ*-theory* $\mathbb{T}$ *in a cartesian closed category* $\mathcal{C}$ *determines a cartesian closed functor* $M : \mathcal{C}_{\mathbb{T}} \to \mathcal{C}$ *with*

$$M(\mathtt{B}) = [\![\mathtt{B}]\!], \quad M(\mathtt{c}) = [\![\mathtt{c}]\!] : 1 \to [\![C]\!] = M(C)\,, \tag{2.6}$$

for all basic types $\mathtt{B}$ *and basic constants* $\mathtt{c} : C$*. Moreover,* M *is unique up to a unique isomorphism of CCC functors, in the sense that given another model* N *satisfying* (2.6)*, there is a unique natural iso* $M \cong N$*, determined inductively by the comparison maps* $M(1) \cong N(1)$*,*

$$M(A \times B) \;\cong\; MA \times MB \;\cong\; NA \times NB \;\cong\; N(A \times B)\,,$$

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

Proof. Straightforward. □

We then also have the usual functorial semantics theorem:

Theorem 2.4.4. *For any λ-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*

$$\mathsf{Mod}_\lambda\big(\mathbb{T}, \mathcal{C}\big) \;\simeq\; \mathsf{CCC}\big(\mathcal{C}_\mathbb{T}, \mathcal{C}\big), \tag{2.7}$$

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 $[\![-]\!]$ in a CCC $\mathcal{C}$ and a CCC functor $f : \mathcal{C} \to \mathcal{D}$, there is an induced model $[\![-]\!]^f$ in $\mathcal{D}$, given by the interpretation $[\![A]\!]^f = f[\![A]\!]$. This is straightforward, just as for algebraic theories. □

Remark 2.4.5. As mentioned in Example 2.3.5(1) the categories involved in the equivalence (2.7) 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 $A \to B$ prevents us from specifying a non-iso homomorphism of models $h : M \to N$ by the obvious recursion on the type structure.

In more detail, given $h_A : [\![A]\!]^M \to [\![A]\!]^N$ and $h_B : [\![B]\!]^M \to [\![B]\!]^N$, there is no obvious candidate for a map

$$h_{A\to B} : [\![A \to B]\!]^M \longrightarrow [\![A \to B]\!]^N,$$

when all we have are the following induced maps:

$$\begin{array}{ccccc}
[\![A \to B]\!]^M & \xrightarrow{=} & ([\![B]\!]^M)^{[\![A]\!]^M} & \xrightarrow{(h_B)^{[\![A]\!]^M}} & ([\![B]\!]^N)^{[\![A]\!]^M} \\
 & & \uparrow{\scriptstyle ([\![B]\!]^M)^{h_A}} & & \uparrow{\scriptstyle ([\![B]\!]^N)^{h_A}} \\
 & & ([\![B]\!]^M)^{[\![A]\!]^N} & \xrightarrow[(h_B)^{[\![A]\!]^N}]{} & ([\![B]\!]^N)^{[\![A]\!]^N} \xrightarrow{=} [\![A \to B]\!]^N
\end{array}$$

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

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

This suffices to at get a *category* of models $\mathsf{Mod}_\lambda\big(\mathbb{T}, \mathcal{C}\big)$, rather than just as set, which is enough structure to determine the equivalence (2.7). Note that for an algebraic theory $\mathbb{A}$, this category of λ-models in Set, say, $\mathsf{Mod}_\lambda(\mathbb{A}^\lambda)$ is still the (wide but non-full) subcategory of isomorphisms of conventional (algebraic) $\mathbb{A}$-models

$$\mathsf{Mod}_\lambda(\mathbb{A}^\lambda) \rightarrowtail \mathsf{Mod}(\mathbb{A})\,.$$

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 λ-theories in cartesian closed categories is *complete*, in virtue of the syntactic construction of the classifying category $\mathcal{C}_\mathbb{T}$. Specifically, a λ-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 $[x : \mathtt{1} \mid c : A]$. 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 now shown:

Proposition 2.4.6. *For any λ-theory $\mathbb{T}$,*

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

Of course, the syntactic model $[-]$ is the one associated under (2.7) to the identity functor $\mathcal{C}_\mathbb{T} \to \mathcal{C}_\mathbb{T}$, *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 2.3.2).

Corollary 2.4.7. *For any λ-theory $\mathbb{T}$,*

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

Moreover, a closed type A is inhabited $\vdash a : A$ if, and only if, there is a point $1 \to [\![A]\!]^M$ in every model M.

2.5 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 then 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 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 λ-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 λ-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* $\tau : \mathbb{S} \to \mathbb{T}$ of a λ-theory $\mathbb{S}$ into a λ-theory $\mathbb{T}$ is given by the following data:

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

$$\tau 1 = 1 \,, \qquad \tau(A \times B) = \tau A \times \tau B \,, \qquad \tau(A \to B) = \tau A \to \tau B \,.$$

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

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

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

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

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

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

Definition 2.5.1. Let $\lambda\mathsf{Thr}$ be the category whose objects are λ-theories and morphisms are translations between them.

We now have an *isomorphism of sets*,

$$\mathsf{Hom}_{\lambda\mathsf{Thr}}(\mathbb{S}, \mathbb{T}) \cong \mathsf{Mod}_\lambda(\mathbb{S}, \mathcal{C}_\mathbb{T}) \,, \tag{2.8}$$

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

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

1. For every object $A \in \mathcal{C}$ there is a basic type $\ulcorner A \urcorner$.
2. For every morphism $f : A \to B$ there is a basic constant $\ulcorner f \urcorner$ whose type is $\ulcorner A \urcorner \to \ulcorner B \urcorner$.
3. For every $A \in \mathcal{C}$ there is an axiom
$$x : \ulcorner A \urcorner \mid \ulcorner \mathtt{1}_A \urcorner x = x : \ulcorner A \urcorner \,.$$
4. For all morphisms $f : A \to B$, $g : B \to C$, and $h : A \to C$ 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
$$\mathtt{T} : \mathtt{1} \to \ulcorner \mathtt{1} \urcorner \,,$$
and for all $A, B \in \mathcal{C}$ there are constants
$$\mathtt{P}_{A,B} : \ulcorner A \urcorner \times \ulcorner B \urcorner \to \ulcorner A \times B \urcorner \,, \qquad \mathtt{E}_{A,B} : (\ulcorner A \urcorner \to \ulcorner B \urcorner) \to \ulcorner B^A \urcorner \,.$$
They satisfy the following axioms:
$$\begin{aligned}
u : \ulcorner \mathtt{1} \urcorner \mid \mathtt{T} * &= u : \ulcorner \mathtt{1} \urcorner \\
z : \ulcorner A \times B \urcorner \mid \mathtt{P}_{A,B} \langle \ulcorner \pi_0 \urcorner z, \ulcorner \pi_1 \urcorner z \rangle &= z : \ulcorner A \times B \urcorner \\
w : \ulcorner A \urcorner \times \ulcorner B \urcorner \mid \langle \ulcorner \pi_0 \urcorner (\mathtt{P}_{A,B} w), \ulcorner \pi_1 \urcorner (\mathtt{P}_{A,B} w) \rangle &= w : \ulcorner A \urcorner \times \ulcorner B \urcorner \\
f : \ulcorner B^A \urcorner \mid \mathtt{E}_{A,B} (\lambda x : \ulcorner A \urcorner . (\ulcorner \mathsf{ev}_{A,B} \urcorner (\mathtt{P}_{A,B} \langle f, x \rangle))) &= f : \ulcorner B^A \urcorner \\
f : \ulcorner A \urcorner \to \ulcorner B \urcorner \mid \lambda x : \ulcorner A \urcorner . (\ulcorner \mathsf{ev}_{A,B} \urcorner (\mathtt{P}_{A,B} \langle (\mathtt{E}_{A,B} f), x \rangle)) &= f : \ulcorner A \urcorner \to \ulcorner B \urcorner
\end{aligned}$$

The purpose of the constants $\mathtt{T}$, $\mathtt{P}_{A,B}$, $\mathtt{E}_{A,B}$, and the axioms for them is to ensure the isomorphisms $\ulcorner \mathtt{1} \urcorner \cong \mathtt{1}$, $\ulcorner A \times B \urcorner \cong \ulcorner A \urcorner \times \ulcorner B \urcorner$, and $\ulcorner B^A \urcorner \cong \ulcorner A \urcorner \to \ulcorner B \urcorner$. Types A and B are said to be *isomorphic* if there are terms
$$x : A \mid t : B \,, \qquad\qquad y : B \mid u : A \,,$$
such that $\mathbb{S}$ proves
$$x : A \mid u[t/y] = x : A \,, \qquad\qquad y : B \mid t[u/x] = y : B \,.$$

Furthermore, an *equivalence of theories* $\mathbb{S}$ and $\mathbb{T}$ is a pair of translations
$$\mathbb{S} \underset{\sigma}{\overset{\tau}{\rightleftarrows}} \mathbb{T}$$

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

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

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

$$\mathbb{L} : \mathsf{CCC} \to \lambda\mathsf{Thr} \,,$$

where 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 $F : \mathcal{C} \to \mathcal{D}$ is a cartesian closed functor then $\mathbb{L}(F) : \mathbb{L}(\mathcal{C}) \to \mathbb{L}(\mathcal{D})$ 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 $\mathtt{T}$, $\mathtt{P}_{A,B}$ and $\mathtt{E}_{A,B}$ are translated to $\mathtt{T}$, $\mathtt{P}_{FA,BA}$ and $\mathtt{E}_{FA,FB}$, respectively.

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

$$\mathcal{C} : \lambda\mathsf{Thr} \to \mathsf{CCC} \,,$$

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

Theorem 2.5.2. *The functors $\mathbb{L} : \mathsf{CCC} \to \lambda\mathsf{Thr}$ and $\mathcal{C} : \lambda\mathsf{Thr} \to \mathsf{CCC}$ constitute an equivalence of categories "up to equivalence" (a* biequivalence *of* 2-categories*). This means that for any $\mathcal{C} \in \mathsf{CCC}$ there is an equivalence of categories*

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

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

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

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

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

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

$$x : \ulcorner A \urcorner \mid \ulcorner \mathtt{1}_A \urcorner x = x : \ulcorner A \urcorner$$

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

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

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

$$\theta_{\mathcal{C}} \mathtt{1} = \mathtt{1} \, , \qquad \theta_{\mathcal{C}} \ulcorner A \urcorner = A \, ,$$
$$\theta_{\mathcal{C}}(Y \times Z) = \theta_{\mathcal{C}} X \times \theta_{\mathcal{C}} Y \, , \qquad \theta_{\mathcal{C}}(Y \to Z) = (\theta_{\mathcal{C}} Z)^{\theta_{\mathcal{C}} Y} \, .$$

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 $\theta_{\mathcal{C}} : \mathcal{C}_{\mathbb{L}(\mathcal{C})} \to \mathcal{C}$ so that $\theta_{\mathcal{C}} \circ \eta_{\mathcal{C}} \cong \mathtt{1}_{\mathcal{C}}$ and $\eta_{\mathcal{C}} \circ \theta_{\mathcal{C}} \cong \mathtt{1}_{\mathcal{C}_{\mathbb{L}(\mathcal{C})}}$.

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

$$\tau_{\mathbb{S}} A = \ulcorner A \urcorner \, .$$

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

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

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

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

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

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

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

$$\sigma_{\mathbb{S}} \, \mathtt{T} = \lambda x : \mathtt{1} \, . \, x \, ,$$
$$\sigma_{\mathbb{S}} \, \mathtt{P}_{A,B} = \lambda p : A \times B \, . \, p \, ,$$
$$\sigma_{\mathbb{S}} \, \mathtt{E}_{A,B} = \lambda f : A \to B \, . \, f \, .$$

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

1. If $X = \mathtt{1}$ then $\tau_{\mathbb{S}}(\sigma_{\mathbb{S}} \mathtt{1}) = \mathtt{1}$.

2. If $X = \ulcorner A \urcorner$ is a basic type then A is a type in $\mathbb{S}$. We proceed by induction on the structure of A:

   (a) If $A = \mathtt{1}$ then $\tau_{\mathbb{S}}(\sigma_{\mathbb{S}} \ulcorner \mathtt{1} \urcorner) = \mathtt{1}$. The types $\mathtt{1}$ and $\ulcorner \mathtt{1} \urcorner$ are isomorphic via the constant $\mathtt{T} : \mathtt{1} \to \ulcorner \mathtt{1} \urcorner$.

(b) If A is a basic type then $\tau_{\mathbb{S}}(\sigma_{\mathbb{S}}\ulcorner A\urcorner) = \ulcorner A\urcorner$.

(c) If $A = B \times C$ then $\tau_{\mathbb{S}}(\sigma_{\mathbb{S}}\ulcorner B \times C\urcorner) = \ulcorner B\urcorner \times \ulcorner C\urcorner$. But we know $\ulcorner B\urcorner \times \ulcorner C\urcorner \cong \ulcorner B \times C\urcorner$ via the constant $\mathtt{P}_{A,B}$.

(d) The case $A = B \to C$ is similar.

3. If $X = Y \times Z$ then $\tau_{\mathbb{S}}(\sigma_{\mathbb{S}}(Y \times Z)) = \tau_{\mathbb{S}}(\sigma_{\mathbb{S}}Y) \times \tau_{\mathbb{S}}(\sigma_{\mathbb{S}}Z)$. By induction hypothesis, $\tau_{\mathbb{S}}(\sigma_{\mathbb{S}}Y) \cong Y$ and $\tau_{\mathbb{S}}(\sigma_{\mathbb{S}}Z) \cong Z$, from which we easily obtain

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

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

□

Composing the isomorphism 2.8 with the equivalence 2.7 we can formulate the foregoing Theorem 2.5.2 as an adjoint equivalence.

Corollary 2.5.3. *There is a biequivalence between the categories* $\lambda\mathsf{Thr}$ *of* λ*-theories and translations between them (and isos thereof), and the category* CCC *of cartesian closed categories and CCC functors (and natural isos),*

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

This is mediated by an adjunction,

$$\mathsf{CCC} \underset{\mathcal{C}}{\overset{\mathbb{L}}{\rightleftarrows}} \lambda\mathsf{Thr}$$

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

Exercise 2.5.4. In the proof of Theorem 2.5.2 we defined, for each $\mathcal{C} \in \mathsf{CCC}$, a functor $\eta_C : \mathcal{C} \to \mathcal{C}_{\mathbb{L}(\mathcal{C})}$. Verify that this determines a natural transformation $\eta : \mathbf{1}_{\mathsf{CCC}} \Longrightarrow \mathcal{C} \circ \mathbb{L}$ which is an equivalence of categories. What about the translation $\epsilon_{\mathbb{T}} : \mathbb{T} \to \mathbb{L}(\mathcal{C}_{\mathbb{T}})$—is that an isomorphism?

See the book [LS88] for another approach to the biequivalence of Corollary 2.5.3, which turns it into an equivalence of categories by fixing the CCC structure and requiring it to be preserved *strictly*.

Lawvere's fixed point theorem

As an application of the internal language of a CCC, we can use the λ-calculus to give a neat proof of a fixed point theorem for CCCs due to Lawvere [Law69]. Andrej Bauer has called Lawvere's theorem the "quintessential diagonal argument" [Baub].

Theorem 2.5.5 (Lawvere)**.** *In any cartesian closed category, if $e : A \to B^A$ is a pointwise surjection, then every map $f : B \to B$ has a fixed point.*

By "pointwise surjection" we mean a map that induces a surjection from points $1 \to A$ to points $1 \to B^A$ by composition.

Proof. Given $f : B \to B$, consider the map $\lambda x : A.f(ex)x : 1 \to B^A$. Since e is pointwise surjective, there is a point $a : 1 \to A$ such that $ea = \lambda x : A.f(ex)x$. Thus

$$(ea)a = (\lambda x : A.f(ex)x)a = f(ea)a\,,$$

so $(ea)a : 1 \to B$ is a fixed point of $f : B \to B$. □

Among the consequences of this theorem stated in [Law69]: Cantor's theorem (Corollary 1.2); Gödel's incompleteness theorem (Theorem 3.3); and Tarski's indefinability of truth (Theorem 3.2). These are all derived from the contrapositive form: if a certain object has an endomap without fixed points, then some pointwise surjection is not possible. For instance, in Set the contrapositive form of Theorem 2.5.5 implies that there is no pointwise surjection from A to its powerset $\mathcal{P}A \cong 2^A$, because the "negation" map $\neg : 2 \to 2$ has no fixed points. (The same argument of course works in any topos, [Baua].)

Lawvere's original version is a bit more general, but even in the present form it is clear that Lawvere's fixed point theorem is the essence of many familiar diagonal arguments.

Bibliography

[AGH21] S. Awodey, N. Gambino, and S. Hazratpour. Kripke-Joyal forcing for type theory and uniform fibrations, October 2021. Preprint available as `https://arxiv.org/abs/2110.14576`.

[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.

[Awo] Steve Awodey. Introduction to categorical logic. Fall 2024, `https://awodey.github.io/catlog/notes/catlogdraft.pdf`.

[Awo00] Steve Awodey. Topological representation of the λ-calculus. *Mathematical Structures in Computer Science*, 10:81–96, 2000.

[Awo21] Steve Awodey. Sheaf representations and duality in logic. In C. Casadio and P.J. Scott, editors, *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics*. Springer, 2021. arXiv:2001.09195.

[Baua] A. Bauer. On a proof of cantor's theorem. Blogpost at `https://math.andrej.com/2007/04/08/on-a-proof-of-cantors-theorem`.

[Baub] A. Bauer. On fixed-point theorems in synthetic computability. Blogpost at `https://math.andrej.com/2019/11/07/on-fixed-point-theorems-in-synthetic-computability`.

[Coq] Thierry Coquand. Type theory. The Stanford Encyclopedia of Philosophy, Fall 2022 Edition, `https://plato.stanford.edu/archives/fall2022/entries/type-theory`.

[Fri75] H. Friedman. Equality between functionals. In R. Parikh, editor, *Logic Colloquium*. Springer-Verlag, New York, 1975.

[FS99] Marcelo Fiore and Alex Simpson. Lambda definability with sums via Grothendieck logical relations. In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications*, pages 147–161, Berlin, Heidelberg, 1999. Springer.

[Joh82] P.T. Johnstone. *Stone Spaces*. Number 3 in Cambridge studies in advanced mathematics. Cambridge University Press, 1982.

[Joh03] P.T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium, 2 vol.s*. Number 43 in Oxford Logic Guides. Oxford University Press, 2003.

[JT84] A. Joyal and M. Tierney. *An extension of the Galois theory of Grothendieck*. Memoirs of the AMS. American Mathematical Society, 1984.

[Law69] F.W. Lawvere. Diagonal arguments and cartesian closed categories. In *Category Theory, Homology Theory and their Applications II*, volume 92 of *Lecture Notes in Mathematics*. Springer, Berlin, 1969. Reprinted with author commentary in *Theory and Applications of Categories* (15): 1–13, (2006).

[LS88] J. Lambek and P.J. Scott. *Introduction to Higher-Order Categorical Logic*. Cambridge, 1988.

[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.

[MR95] Michael Makkai and Gonzalo Reyes. Completeness results for intuitionistic and modal logic in a categorical setting. *Annals of Pure and Applied Logic*, 72:25–101, 1995.

[Plo73] G. D. Plotkin. Lambda-definability in the full type hierarchy. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, New York, 1973.

[Sco70] Dana S. 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.

[Sco80a] Dana S. Scott. The lambda calculus: Some models, some philosophy. In *The Kleene Symposium*, pages 223–265. North-Holland, 1980.

[Sco80b] Dana S. Scott. Relating theories of the lambda calculus. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 403–450. Academic Press, 1980.

[Sim95] A. Simpson. Categorical completeness results for the simply-typed lambda-calculus. In M. Dezani-Ciancaglini and G. Plotkin, editors, *Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science, pages 414–427. Springer, 1995.

[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.