

# Notes on Type Theory

[DRAFT: FEBRUARY 13, 2025]

Steve Awodey

with contributions from Andrej Bauer



# Contents

<b>2</b>	<b>Simple Type Theory</b>	<b>5</b>
2.1	The $\lambda$ -calculus . . . . .	5
2.2	Cartesian closed categories . . . . .	13
2.3	Interpretation of the $\lambda$ -calculus in a CCC . . . . .	21
2.4	Functorial semantics . . . . .	24
2.5	The internal language of a CCC . . . . .	28
2.6	Embedding theorems and completeness . . . . .	34
2.7	Kripke models . . . . .	37
2.8	Topological models . . . . .	40
	<b>Bibliography</b>	<b>45</b>



# Chapter 2

## Simple Type Theory

### 2.1 The $\lambda$ -calculus

The  $\lambda$ -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  $\lambda$ -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  *$\lambda$ -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  $\lambda$ -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 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 allowed, whatever it may mean. We will concentrate here on the typed  $\lambda$ -calculus (but see Example 2.1.7 below). In typed  $\lambda$ -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  $A \rightarrow B$  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  $\lambda$ -calculus.** We now give a more formal definition of what constitutes a *simply-typed  $\lambda$ -calculus*. First, we are given a collection of *simple types*, which are generated from some *basic types* by formation of product and function types:

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

When convenient, we may 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*  $x, y, z, \dots$  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:

$$\begin{array}{ll} \text{Variables} & v ::= x \mid y \mid z \mid \cdots \\ \text{Constants} & c ::= c_1 \mid c_2 \mid \cdots \\ \text{Terms} & t ::= v \mid c \mid * \mid \langle t_1, t_2 \rangle \mid \mathbf{fst} \, t \mid \mathbf{snd} \, t \mid t_1 \, t_2 \mid \lambda x : A. t \end{array}$$

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  $\mathbf{fst} \, t$  and  $\mathbf{snd} \, t$  are terms,
6. if  $s$  and  $t$  are terms then  $st$  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:

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

A term  $t$  is *closed* if all of its variables are bound, so that  $\mathbf{FV}(t) = \emptyset$ . 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. We may identify contexts under reordering, regarding them as sets rather than sequences. Contexts may be 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:

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

- 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:

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



- Equations for product types:

$$\begin{array}{c}
\frac{\Gamma \mid u = v : A \quad \Gamma \mid s = t : B}{\Gamma \mid \langle u, s \rangle = \langle v, t \rangle : A \times B} \\
\frac{\Gamma \mid s = t : A \times B}{\Gamma \mid \mathbf{fst} s = \mathbf{fst} t : A} \qquad \frac{\Gamma \mid s = t : A \times B}{\Gamma \mid \mathbf{snd} s = \mathbf{snd} t : A} \\
\frac{}{\Gamma \mid t = \langle \mathbf{fst} t, \mathbf{snd} t \rangle : A \times B} \\
\frac{}{\Gamma \mid \mathbf{fst} \langle s, t \rangle = s : A} \qquad \frac{}{\Gamma \mid \mathbf{snd} \langle s, t \rangle = t : A}
\end{array}$$

- Equations for function types:

$$\begin{array}{c}
\frac{\Gamma \mid s = t : A \rightarrow B \quad \Gamma \mid u = v : A}{\Gamma \mid s u = t v : B} \\
\frac{\Gamma, x : A \mid t = u : B}{\Gamma \mid (\lambda x : A. t) = (\lambda x : A. u) : A \rightarrow B} \\
\frac{\Gamma \mid t : A \rightarrow B}{\Gamma \mid \lambda x : A. (t x) = t : A \rightarrow B} \quad (\eta\text{-rule}) \\
\frac{}{\Gamma \mid (\lambda x : A. t) u = t[u/x] : A} \quad (\beta\text{-rule})
\end{array}$$

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

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

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

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

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

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

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

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

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

$$e : G , \quad i : G \rightarrow G , \quad m : G \times G \rightarrow G ,$$

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

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

**Example 2.1.3.** More generally, any (Lawvere) algebraic theory  $\mathbb{A}$  (as in Chapter ??) determines a  $\lambda$ -theory  $\mathbb{A}^\lambda$ . There is one basic type  $A$  and for each operation  $f$  of arity  $k$  there is a basic constant  $f : A^k \rightarrow A$ , where  $A^k$  is the  $k$ -fold product  $A \times \cdots \times A$ . It is understood that  $A^0 = 1$ . 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  $\mathbb{A}^\lambda$ ,

$$x_1 : A, \dots, x_n : A \mid u = v : A$$

where  $x_1, \dots, x_n$  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,  $V$  for vertices and  $E$  for edges, and two basic constants, source  $\text{src}$  and target  $\text{trg}$ ,

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

There are no equations.

**Example 2.1.5.** 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  $0 \leq i \leq n$ :

$$d_i : X_{n+1} \rightarrow X_n , \quad s_i : X_n \rightarrow X_{n+1} .$$

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

$$\begin{aligned} d_i d_j &= d_{j-1} d_i, & \text{if } i < j, \\ s_i s_j &= s_{j+1} s_i, & \text{if } i \leq j, \\ d_i s_j &= \begin{cases} s_{j-1} d_i, & \text{if } i < j, \\ \text{id}, & \text{if } i = j \text{ or } i = j + 1, \\ s_j d_{i-1}, & \text{if } i > j + 1. \end{cases} \end{aligned}$$

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

Basic types    $B ::= \text{nat} \mid \text{bool}$  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*  $\text{cond}_A$  and the *fixpoint* operator  $\text{fix}_A$ . They have the following types:

$0 : \text{nat}$   
 $\text{succ} : \text{nat} \rightarrow \text{nat}$   
 $\text{true} : \text{bool}$   
 $\text{false} : \text{bool}$   
 $\text{iszero} : \text{nat} \rightarrow \text{bool}$   
 $\text{cond}_A : \text{bool} \rightarrow A \rightarrow A$   
 $\text{fix}_A : (A \rightarrow A) \rightarrow A$

The equational axioms of PCF are:

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

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

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

satisfying the equation

$$f : D \rightarrow D \mid \mathbf{r}(\mathbf{s} f) = f : D \rightarrow D \tag{2.1}$$

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

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

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

A reflexive type can be used to interpret the untyped  $\lambda$ -calculus into the typed  $\lambda$ -calculus.

## 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:

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

$$\frac{\Gamma \mid t = t}{\Gamma \mid \lambda x. (tx) = t} \qquad (\eta\text{-rule})$$

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

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

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

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

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

$$\Gamma \mid u = t \quad (2.3)$$

is translated to the judgment

$$\Gamma^* \mid u^* = t^* : \mathbf{D} . \quad (2.4)$$

**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  $\mathbf{Hom}(P, Q)$  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 for cartesian closure 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 2.2.1.** 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.

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

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

Definition 2.2.1 is called the *universal property of the exponential*. It is just the category-theoretic way of saying that a function  $f : C \times A \rightarrow B$  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 **Pos** 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, which is easily seen to be monotone. 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*, which is also easily seen to be monotone. 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  $B \in \mathcal{C}$  (including 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 2.2.3.** Consider again the example of the set  $\mathbf{Hom}(M, N)$  of homomorphisms between two monoids  $M, N$ , equipped with the pointwise monoid structure. Let  $1 = \{u\}$  be the terminal monoid, having only a unit element  $u$ . 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 2.2.4.** Recall that monoids and groups can be regarded as (1-object) categories, and then their homomorphisms are just functors. Thus 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 2.2.5.** *In a category  $\mathcal{C}$  with binary products an object  $A$  is exponentiable if, and only if, the functor*

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

*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.

$$\begin{array}{ccc} B^A \times A & \xrightarrow{\epsilon_B} & B \\ g^A \times 1_A \downarrow & & \downarrow g \\ C^A \times A & \xrightarrow{\epsilon_C} & C \end{array} \quad (2.5)$$

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 (2.5), 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 adjoints, 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 all 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  $A$ .

□

**Exercise 2.2.7.** Show that being cartesian closed is a *categorical property*, in the sense that it respects equivalence of categories: if  $\mathcal{C}$  is cartesian closed and  $\mathcal{C} \simeq \mathcal{D}$  then  $\mathcal{D}$  is also cartesian closed.

Another consequence 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 an equational formulation of the definition of a cartesian closed category.

**Proposition 2.2.8** (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  $\pi_1 : A \times B \rightarrow A$  and  $\pi_2 : A \times B \rightarrow B$ , 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  $f : C \rightarrow A$ ,  $g : C \rightarrow B$ ,  $h : C \rightarrow A \times B$ ,

$$\pi_1 \circ \langle f, g \rangle = f , \quad \pi_2 \circ \langle f, g \rangle = g , \quad \langle \pi_1 \circ h, \pi_2 \circ h \rangle = h .$$

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

$$e \times f := \langle e\pi_1, f\pi_2 \rangle : E \times F \rightarrow E' \times F' .$$



These equations ensure that certain diagrams commute and that the morphisms that are required to exist are unique. For example, let us prove that  $(A \times B, \pi_1, \pi_2)$  is the product of  $A$  and  $B$ . For  $f : C \rightarrow A$  and  $g : C \rightarrow B$  we have the morphism  $\langle f, g \rangle : C \rightarrow A \times B$ . The equations

$$\pi_1 \circ \langle f, g \rangle = f \quad \text{and} \quad \pi_2 \circ \langle f, g \rangle = g$$

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

$$\begin{array}{ccccc} & & C & & \\ & g \swarrow & \downarrow \langle f, g \rangle & \searrow f & \\ A & \xleftarrow{\pi_1} & A \times B & \xrightarrow{\pi_2} & B \end{array}$$

Suppose  $h : C \rightarrow A \times B$  is another morphism such that  $f = \pi_1 \circ h$  and  $g = \pi_2 \circ h$ . Then by the third equation for products we get

$$h = \langle \pi_1 \circ h, \pi_2 \circ h \rangle = \langle f, g \rangle ,$$

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

**Exercise 2.2.9.** Use the equational characterization of CCCs, Proposition 2.2.8, 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 2.2.10.** Use the equational characterization of CCCs, Proposition 2.2.8, 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

As we have seen ??, a cartesian closed poset is a  $\wedge$ -semilattice with exponentials  $p \Rightarrow q$ , such as a Heyting algebra, or a syntactic category arising from a positive propositional calculus. We next review some important examples of non-poset cartesian closed categories, most of which should be familiar.

**Example 2.2.11.** 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 2.2.12.** 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 2.2.13.** 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 2.2.14.** Show that there is a full and faithful functor  $D : \mathbf{Set} \rightarrow \mathbf{Poset}$  that preserves finite limits as well as exponentials. Note the similarity to the example  $\mathbf{Grpd} \hookrightarrow \mathbf{Cat}$ .

The foregoing examples are instances of the following general situation.

**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 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  $iB^A$  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  $B^A = L(iB^A)$ . 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 2.2.16.** *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 the inverse image of the one 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 2.2.17.** Which of the foregoing examples follows from which of the previous two propositions?

**Example 2.2.18.** 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 2.2.19.** Prove that  $\mathbf{Set}/_X$  is always cartesian closed. (Hint: Use the fact that  $\mathbf{Set}/_X \simeq \mathbf{Set}^X$ , and the category of CCCs is closed under products of the underlying categories.)

Lest it be thought that the foregoing example is typical, and every slice of a CCC is again a CCC, one can consider the counterexample of **Pos**. By an argument like that in [Pal03] for the cartesian closed category **Grpd** of groupoids, the slice categories of **Pos** need not be cartesian closed.

**Exercise 2.2.20.** Check that the example given in [Pal03] also works (*mutatis mutandis*) for  $\mathbf{Pos}$  to show that  $\mathbf{Pos}/_X$  is not always cartesian closed.

**Example 2.2.21.** A presheaf category  $\widehat{\mathcal{C}}$  is cartesian closed, provided the index category  $\mathcal{C}$  is small. To see what the exponential of presheaves  $P$  and  $Q$  ought to be, we can use the Yoneda lemma. If  $Q^P$  exists, then by Yoneda and the adjunction  $(- \times P) \dashv (-^P)$ , we would have, for all  $c \in \mathcal{C}$ ,

$$Q^P(c) \cong \mathbf{Nat}(yc, Q^P) \cong \mathbf{Nat}(yc \times P, Q) .$$

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

$$Q^P(c) = \mathbf{Nat}(yc \times P, Q) .$$

(This is indeed contravariant in  $c$ !) The evaluation morphism  $E : Q^P \times P \rightarrow Q$  is the natural transformation whose component at  $c$  is

$$\begin{aligned} E_c : \mathbf{Nat}(yc \times P, Q) \times Pc &\rightarrow Qc , \\ E_c : \langle \eta, x \rangle &\mapsto \eta_c(1_c, x) . \end{aligned}$$

The transpose of a natural transformation  $\phi : R \times P \rightarrow Q$  is the natural transformation  $\tilde{\phi} : R \rightarrow Q^P$  whose component at  $c$  is the function that maps  $z \in Rc$  to the natural transformation  $\tilde{\phi}_c z : yc \times P \rightarrow Q$ , whose component at  $b \in \mathcal{C}$  is

$$\begin{aligned} (\tilde{\phi}_c z)_b : \mathcal{C}(b, c) \times Pb &\rightarrow Qb , \\ (\tilde{\phi}_c z)_b : \langle f, y \rangle &\mapsto \phi_b((Rf)z, y) . \end{aligned}$$

**Exercise 2.2.22.** 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  $\mathbf{Graph}$  is cartesian closed, because it is the presheaf category  $\mathbf{Set}^{\rightarrow}$ . 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 2.2.23.** This exercise is for those with some background in linear algebra. Let  $\mathbf{Vec}$  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  $\mathbf{Vec}$ .

Later in this chapter, we will meet some further examples of CCCs with a more topological flavor:

- 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  $X$  (Section 2.8).
- Sheaves for the “+–topology” on a small category  $\mathcal{C}$  with (stable) sums  $A + B$ .
- Dana Scott’s category  $\mathbf{Equ}$  of equilogical spaces (Section 2.8).

## 2.3 Interpretation of the $\lambda$ -calculus in a CCC

We now consider semantic aspects of the  $\lambda$ -calculus and  $\lambda$ -theories. Suppose  $\mathbb{T}$  is a  $\lambda$ -theory 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 this purpose, we assume that a CCC structure on  $\mathcal{C}$  has been chosen.)

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

The interpretation is then 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 as follows:

$$\llbracket \Gamma \mid \langle t, u \rangle : A \times B \rrbracket = \langle \llbracket \Gamma \mid t : A \rrbracket, \llbracket \Gamma \mid u : B \rrbracket \rangle : \llbracket \Gamma \rrbracket \rightarrow \llbracket A \rrbracket \times \llbracket B \rrbracket$$

$$\llbracket \Gamma \mid \mathbf{fst} \, t : A \rrbracket = \pi_1 \circ \llbracket \Gamma \mid t : A \times B \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket A \rrbracket$$

$$\llbracket \Gamma \mid \mathbf{snd} \, t : A \rrbracket = \pi_2 \circ \llbracket \Gamma \mid t : A \times B \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket B \rrbracket.$$

- The interpretation of application and  $\lambda$ -abstraction is as follows:

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

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 2.3.1.** An interpretation of a  $\lambda$ -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  $\mathcal{C}$ ,

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

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  $\lambda$ -theory and  $\llbracket - \rrbracket$  is 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*

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

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

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

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  $\lambda$ -calculus, related to the Curry-Howard correspondence of section ??, relating  $\lambda$ -calculus to the proof theory of 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  $\Gamma$  (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  $A$  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  $\lambda$ -theory and  $\llbracket - \rrbracket$  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  $1 \rightarrow \llbracket A \rrbracket$  in  $\mathcal{C}$ . Thus for all  $\mathbb{T}$ -models  $\llbracket - \rrbracket$ ,*

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

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

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

**Example 2.3.5.** 1. A model of an algebraic theory  $\mathbb{A}$  (extended to a  $\lambda$ -theory  $\mathbb{A}^\lambda$  as in Example 2.1.3) when 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*

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

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  $\llbracket A \rightarrow B \rrbracket = \llbracket B \rrbracket^{\llbracket A \rrbracket}$  (this is discussed in more detail in the next section).

A point to note is that such a model is entirely determined by the interpretation of the *basic* types and terms – *i.e.* the algebra – and the rest of the interpretation is “standard” in the sense that  $\llbracket A \rightarrow B \rrbracket = \llbracket B \rrbracket^{\llbracket A \rrbracket}$ . So in particular, our models are *not* the “Henkin models” that one sometimes sees in the literature.

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 presheaf models at all! (And then, by Theorem 2.6.6, 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* (see [Sco80]).
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 2.3.6.** How are models of a (not necessarily propositional)  $\lambda$ -theory  $\mathbb{T}$  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?)

## 2.4 Functorial semantics

In Chapter ?? we saw how an algebraic theory gives rise to a category with finite products, and its algebras, or models, then correspond to functors preserving finite products on the theory-category. We then arranged the traditional relationship between syntax and semantics 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  $\lambda$ -theories and their models in CCCs. The first step is to build the *classifying category*  $\mathcal{C}_{\mathbb{T}}$  from a  $\lambda$ -theory  $\mathbb{T}$ . This is again constructed from the theory itself as a “syntactic” category, as follows:

**Definition 2.4.1.** For any  $\lambda$ -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  $A \rightarrow B$  are terms in context (of length one):

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

where two such terms  $x : A \mid s : B$  and  $x : A \mid s' : B$  are to represent the same morphism when  $\mathbb{T}$  proves  $x : A \mid s = s' : B$ . Note that longer contexts are not required, because we have product types  $A_1 \times \cdots \times A_n$ .

- Composition of the terms

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

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

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

- The identity morphism on  $A$  is the term  $[x : A \mid x : A]$  (up to “ $\alpha$ -renaming” of variables).

**Proposition 2.4.2.** *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 simply 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 the term

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

This morphism is indeed unique, because we always have the equation

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

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

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

Given morphisms

$$z : C \mid a : A, \quad z : C \mid b : B,$$

the term

$$z : C \mid \langle a, b \rangle : A \times B$$

represents the unique morphism satisfying

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

Indeed, if  $\mathbf{fst} t = a$  and  $\mathbf{snd} t = b$  for some  $t$ , then we have

$$t = \langle \mathbf{fst} t, \mathbf{snd} t \rangle = \langle a, b \rangle.$$

as required.

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

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

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

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

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

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

Indeed, we have:

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

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

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

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

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

Therefore,

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

as claimed.

□

The syntactic category  $\mathcal{C}_{\mathbb{T}}$  allows us to replace a  $\mathbb{T}$ -model  $\llbracket - \rrbracket$  in a CCC  $\mathcal{C}$  with a functor  $M : \mathcal{C}_{\mathbb{T}} \rightarrow \mathcal{C}$ . More precisely, we have the following.

**Lemma 2.4.3.** *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*

$$M(\mathbf{B}) = \llbracket \mathbf{B} \rrbracket, \quad M(\mathbf{c}) = \llbracket \mathbf{c} \rrbracket : 1 \rightarrow \llbracket C \rrbracket = M(C), \quad (2.6)$$

for all basic types  $\mathbf{B}$  and basic constants  $\mathbf{c} : C$ . 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 structural induction on types and terms with (2.6) as the base case, and using soundness, Proposition 2.3.2, for well-definedness on equivalence classes. Note that the uniqueness up to natural isomorphism uses the fact that all of the morphisms of  $\mathcal{C}_{\mathbb{T}}$  are given by terms. □

We then also have the expected functorial semantics theorem:

**Theorem 2.4.4.** *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*

$$\text{Mod}_{\lambda}(\mathbb{T}, \mathcal{C}) \simeq \text{CCC}(\mathcal{C}_{\mathbb{T}}, \mathcal{C}), \quad (2.7)$$

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  $\llbracket - \rrbracket$  in a CCC  $\mathcal{C}$  and a CCC functor  $f : \mathcal{C} \rightarrow \mathcal{D}$ , there is an induced model  $\llbracket - \rrbracket^f$  in  $\mathcal{D}$ , given by the interpretation  $\llbracket A \rrbracket^f = f \llbracket A \rrbracket$ . This is again 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  $A \rightarrow B$  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 is no obvious candidate for a map

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

when all we have are the following 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 therefore 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  $h_A^{-1} : \llbracket A \rrbracket^N \rightarrow \llbracket A \rrbracket^M$  in the contravariant positions, in order to get things to line up:

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

This suffices to at least get a *category* of models  $\mathbf{Mod}_\lambda(\mathbb{T}, \mathcal{C})$ , 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  $\lambda$ -models in  $\mathbf{Set}$ , say,  $\mathbf{Mod}_\lambda(\mathbb{A}^\lambda)$  is still the (wide but non-full) subcategory of isomorphisms of conventional (algebraic)  $\mathbb{A}$ -models

$$\mathbf{Mod}_\lambda(\mathbb{A}^\lambda) \hookrightarrow \mathbf{Mod}(\mathbb{A}).$$

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 now shown:

**Proposition 2.4.6.** *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 (2.7) 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 2.3.2).

**Corollary 2.4.7** (Completeness). *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  $1 \rightarrow \llbracket A \rrbracket^M$  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  $\lambda$ -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*  $\tau : \mathbb{S} \rightarrow \mathbb{T}$  of a  $\lambda$ -theory  $\mathbb{S}$  into a  $\lambda$ -theory  $\mathbb{T}$  is given by the following data:

1. For each basic type  $A$  in  $\mathbb{S}$  a type  $\tau A$  in  $\mathbb{T}$ . 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  $C$  in  $\mathbb{S}$  a term  $\tau c$  of type  $\tau C$  in  $\mathbb{T}$ . 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  $\mathbb{S}$ : 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  $\tau : \mathbb{S} \rightarrow \mathbb{T}$  is the same thing as a model of  $\mathbb{S}$  in  $\mathcal{C}_{\mathbb{T}}$ , 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 2.5.1.** Let  $\lambda\text{Thr}$  be the category whose objects are  $\lambda$ -theories and morphisms are translations between them.

In this way, we obtain an *isomorphism of sets*,

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

which is not only natural in  $\mathbb{T}$ , but also 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  $\iota_{\mathbb{S}} : \mathbb{S} \rightarrow \mathbb{S}$ . We can enrich this to an isomorphism of *groupoids* by defining syntactic isomorphisms between translations in  $\text{Hom}_{\lambda\text{Thr}}(\mathbb{S}, \mathbb{T})$  in a fairly obvious way so that they correspond bijectively to  $\mathbb{S}$ -model homomorphisms in  $\text{Mod}_{\lambda}(\mathbb{S}, \mathcal{C}_{\mathbb{T}})$ , which in turn correspond to natural isomorphisms between CCC functors in  $\text{Hom}_{\text{CCC}}(\mathcal{C}_{\mathbb{S}}, \mathcal{C}_{\mathbb{T}})$ , by Theorem 2.4.4,

$$\text{Hom}_{\lambda\text{Thr}}(\mathbb{S}, \mathbb{T}) \cong \text{Mod}_{\lambda}(\mathbb{S}, \mathcal{C}_{\mathbb{T}}) \simeq \text{Hom}_{\text{CCC}}(\mathcal{C}_{\mathbb{S}}, \mathcal{C}_{\mathbb{T}}) .$$

The equivalence  $\text{Hom}_{\lambda\text{Thr}}(\mathbb{S}, \mathbb{T}) \simeq \text{Hom}_{\text{CCC}}(\mathcal{C}_{\mathbb{S}}, \mathcal{C}_{\mathbb{T}})$  suggests that the functor  $\mathcal{C}_{(-)} : \lambda\text{Thr} \rightarrow \text{CCC}$  participates in an equivalence of categories,

$$\lambda\text{Thr} \simeq \text{CCC} ,$$

between  $\lambda$ -theories and cartesian closed categories.

Indeed, let  $\mathcal{C}$  be a small cartesian closed category. There is a  $\lambda$ -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  $f : A \rightarrow B$  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

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

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

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

They satisfy the following axioms:

$$\begin{aligned} u : \ulcorner 1 \urcorner \mid \mathsf{T} * &= u : \ulcorner 1 \urcorner \\ z : \ulcorner A \times B \urcorner \mid \mathsf{P}_{A,B} \langle \ulcorner \pi_1 \urcorner z, \ulcorner \pi_2 \urcorner z \rangle &= z : \ulcorner A \times B \urcorner \\ w : \ulcorner A \urcorner \times \ulcorner B \urcorner \mid \langle \ulcorner \pi_1 \urcorner (\mathsf{P}_{A,B} w), \ulcorner \pi_2 \urcorner (\mathsf{P}_{A,B} w) \rangle &= w : \ulcorner A \urcorner \times \ulcorner B \urcorner \\ f : \ulcorner B^A \urcorner \mid \mathsf{E}_{A,B} (\lambda x : \ulcorner A \urcorner . (\ulcorner \mathsf{ev}_{A,B} \urcorner (\mathsf{P}_{A,B} \langle f, x \rangle))) &= f : \ulcorner B^A \urcorner \\ f : \ulcorner A \urcorner \rightarrow \ulcorner B \urcorner \mid \lambda x : \ulcorner A \urcorner . (\ulcorner \mathsf{ev}_{A,B} \urcorner (\mathsf{P}_{A,B} \langle (\mathsf{E}_{A,B} f), x \rangle)) &= f : \ulcorner A \urcorner \rightarrow \ulcorner B \urcorner \end{aligned}$$

The purpose of the constants  $\mathsf{T}$ ,  $\mathsf{P}_{A,B}$ ,  $\mathsf{E}_{A,B}$ , 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  $\mathbb{S}$  proves

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

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

$$\begin{array}{ccc} \mathbb{S} & \xrightleftharpoons[\sigma]{\tau} & \mathbb{T} \end{array}$$

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

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

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

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

where  $\mathbf{CCC}$  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 the syntactic category, which maps a  $\lambda$ -theory  $\mathbb{S}$  to a small cartesian closed category  $\mathcal{C}_{\mathbb{S}}$ . This extends to a functor

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

because a translation  $\tau : \mathbb{S} \rightarrow \mathbb{T}$  induces a functor  $\mathcal{C}_{\tau} : \mathcal{C}_{\mathbb{S}} \rightarrow \mathcal{C}_{\mathbb{T}}$  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  $x : 1 \mid c : A$  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 2.5.2.** *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  $\mathbb{S} \in \lambda\mathbf{Thr}$  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  $\eta_{\mathcal{C}} : \mathcal{C} \rightarrow \mathcal{C}_{\mathbb{L}(\mathcal{C})}$ , 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  $\mathbb{S}$ , we define a translation  $\tau_{\mathbb{S}} : \mathbb{S} \rightarrow \mathbb{L}(\mathcal{C}_{\mathbb{S}})$ . 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

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

In the other direction we define a translation  $\sigma_{\mathbb{S}} : \mathbb{L}(\mathcal{C}_{\mathbb{S}}) \rightarrow \mathbb{S}$  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  $\ulcorner A \urcorner \rightarrow \ulcorner B \urcorner$  then

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

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

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

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  $X = 1$  then  $\tau_{\mathbb{S}}(\sigma_{\mathbb{S}}1) = 1$ .
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  $A = 1$  then  $\tau_{\mathbb{S}}(\sigma_{\mathbb{S}}\ulcorner 1 \urcorner) = 1$ . 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  $\tau_{\mathbb{S}}(\sigma_{\mathbb{S}}\ulcorner A \urcorner) = \ulcorner A \urcorner$ .
  - (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  $\mathbf{P}_{A,B}$ .
  - (d) The case  $A = B \rightarrow C$  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  $X = Y \rightarrow Z$  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  $\lambda\text{Thr}$  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),*

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

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 2.5.4.** In the proof of Theorem 2.5.2 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  $\eta : 1_{\text{CCC}} \Longrightarrow \mathcal{C} \circ \mathbb{L}$  which is an equivalence of categories. What about the translation  $\epsilon_{\mathbb{T}} : \mathbb{T} \rightarrow \mathbb{L}(\mathcal{C}_{\mathbb{T}})$ —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  $\lambda$ -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 a map  $e : A \rightarrow B^A$  is a pointwise surjection, then every map  $f : B \rightarrow B$  has a fixed point.*

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

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

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

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

Among the consequences of this theorem, as stated in [Law69], are: Cantor’s theorem (Corollary 1.2); Gödel’s incompleteness theorem (Theorem 3.3); and Tarski’s undefinability of truth (Theorem 3.2). These are all derived from the contrapositive form of Lawvere’s fixed point theorem 2.5.5: if a certain object  $B$  has an endomap with no fixed points, then for no  $A$  can there be a pointwise surjection  $A \rightarrow B^A$ . To infer Cantor’s theorem, for instance: in **Set** the contrapositive form of Theorem 2.5.5 implies that there is no pointwise surjection from a set  $A$  to its powerset  $\mathcal{P}A \cong 2^A$ , because the “negation” map  $\neg : 2 \rightarrow 2$  has no fixed points. (The same argument works in any topos, see [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.

## 2.6 Embedding theorems and completeness

We have considered the  $\lambda$ -calculus as a common generalization of both propositional logic, modeled by poset CCCs such as Boolean and Heyting algebras, and equational logic, modeled by finite product categories. Accordingly, there are then two different notions of “provability”, as discussed in Remark 2.3.3; namely, the derivability of a closed term  $\vdash a : A$ , and the derivability of an equation between two (not necessarily closed) terms of the same type  $\Gamma \vdash s = t : A$ . With respect to the semantics, there are then two different corresponding notions of soundness and completeness: for “inhabitation” of types, and for equality of terms. We consider special cases of these notions in more detail below.

### Conservativity

With regard to the former notion, inhabitation, one can consider the question of how it compares with simple provability in *propositional logic*: e.g. a positive propositional formula  $\phi$  in the variables  $p_1, p_2, \dots, p_n$  obviously determines a type  $\Phi$  in the corresponding  $\lambda$ -theory  $\mathbb{T}(X_1, X_2, \dots, X_n)$  over  $n$  basic type symbols. What is the relationship between provability in positive propositional logic,  $\text{PPL} \vdash \phi$ , and inhabitation in the associated  $\lambda$ -theory,  $\mathbb{T}(X_1, X_2, \dots, X_n) \vdash t : \Phi$ ? Let us call this the question of *conservativity* of  $\lambda$ -calculus over PPL. According to the basic idea of the Curry-Howard correspondence from Section ??, the  $\lambda$ -calculus is essentially the “proof theory of PPL”. So one should expect that starting from an inhabited type  $\Phi$ , a derivation of a term  $\mathbb{T}(X_1, X_2, \dots, X_n) \vdash t : \Phi$  should result in a corresponding proof of  $\phi$  in PPL just by “rubbing out the proof terms”. Conversely, given a provable formula  $\vdash \phi$ , one should be able to annotate a proof of it in PPL to obtain a derivation of a term  $\mathbb{T}(X_1, X_2, \dots, X_n) \vdash t : \Phi$  in the  $\lambda$ -calculus (although perhaps not the same term that one started with, if the proof was obtained from rubbing out a term).

We can make this idea precise semantically as follows. Write  $|\mathcal{C}|$  for the poset reflection of a category  $\mathcal{C}$ , that is, the left adjoint to the inclusion  $i : \mathbf{Pos} \hookrightarrow \mathbf{Cat}$ , and let  $\eta : \mathcal{C} \rightarrow |\mathcal{C}|$  be the unit of the adjunction.

**Lemma 2.6.1.** *If  $\mathcal{C}$  is cartesian closed, then so is  $|\mathcal{C}|$ , and  $\eta : \mathcal{C} \rightarrow |\mathcal{C}|$  preserves the CCC structure.*

*Proof.* Exercise! □

**Exercise 2.6.2.** Prove Lemma 2.6.1.

**Corollary 2.6.3.** *The syntactic category  $\text{PPC}(p_1, p_2, \dots, p_n)$  of the positive propositional calculus on  $n$  propositional variables is the poset reflection of the syntactic category  $\mathcal{C}_{\mathbb{T}(X_1, X_2, \dots, X_n)}$  of the  $\lambda$ -theory  $\mathbb{T}(X_1, X_2, \dots, X_n)$ ,*

$$|\mathcal{C}_{\mathbb{T}(X_1, X_2, \dots, X_n)}| \cong \text{PPC}(p_1, p_2, \dots, p_n).$$

*Proof.* We already know that  $\mathcal{C}_{\mathbb{T}(X_1, X_2, \dots, X_n)}$  is the free cartesian closed category on  $n$  generating objects, and that  $\text{PPC}(p_1, p_2, \dots, p_n)$  is the free cartesian closed poset on  $n$  generating elements. From the universal property of  $\mathcal{C}_{\mathbb{T}(X_1, X_2, \dots, X_n)}$ , we get a CCC map

$$\mathcal{C}_{\mathbb{T}(X_1, X_2, \dots, X_n)} \longrightarrow \text{PPC}(p_1, p_2, \dots, p_n)$$

taking generators to generators, and it extends along the quotient map to  $|\mathcal{C}_{\mathbb{T}(X_1, X_2, \dots, X_n)}|$  by the universal property of the poset reflection. Thus it suffices to show that the quotient map preserves, and indeed creates, the CCC structure on  $|\mathcal{C}_{\mathbb{T}(X_1, X_2, \dots, X_n)}|$ . But that follows from Lemma 2.6.1. □

**Remark 2.6.4.** Corollary 2.6.3 can be extended to other systems of type theory and logic, with further operations such as CCCs with sums  $0, A + B$  (“bicartesian closed categories”), and the full intuitionistic propositional calculus IPC with the logical operations  $\perp$  and  $p \vee q$ . We leave this as a topic for the interested student.

## Completeness

As was the case for equational theories and propositional logic, the fact that there is a generic model (Proposition 2.4.6) allows the general completeness theorem stated in Corollary 2.4.7 to be specialized to various classes of special models, via embedding (or “representation”) theorems, this time for CCCs, rather than for finite product categories or Boolean/Heyting algebras. We shall consider three such cases: “variable” models, Kripke models, and topological models. In each case, an “embedding theorem” of the form:

*Every CCC embeds into one of the special form  $\mathcal{X}$ .*

gives rise to a completeness theorem of the form:

For all  $\lambda$ -theories  $\mathbb{T}$ , if  $1 \rightarrow \llbracket A \rrbracket^M$  in all  $\mathbb{T}$ -models  $M$  in all  $\mathcal{X}$ , then  $\mathbb{T} \vdash a : A$ ,  
and if  $\llbracket a \rrbracket^M = \llbracket b \rrbracket^M : 1 \rightarrow \llbracket A \rrbracket$  in all  $\mathbb{T}$ -models  $M$  in all  $\mathcal{X}$ , then  $\mathbb{T} \vdash a = b : A$ .

This of course follows the same pattern that we saw for the simpler “proof relevant” case of equational (*i.e.* finite product) theories, and the even simpler “proof irrelevant” case of propositional logic, but now the proofs of some of the embedding theorems for CCCs require more sophisticated methods.

## Variable models

By a *variable model* of the  $\lambda$ -calculus we mean one in a CCC of the form  $\widehat{\mathbb{C}} = \mathbf{Set}^{\mathbb{C}^{\text{op}}}$ , i.e. presheaves on a (small) “index category”  $\mathbb{C}$ . We regard such a model as “varying over  $\mathbb{C}$ ”, just as we saw earlier that a presheaf of groups on *e.g.* the simplex category  $\Delta$  may be seen both as a simplicial group—a simplicial object in the category of groups—and as a group object in the category  $\mathbf{Set}^{\Delta^{\text{op}}}$  of simplicial sets.

The basic embedding theorem that we shall use in specializing Proposition 2.4.6 to such variable models is the following, which is one of the fundamental facts of categorical semantics.

**Lemma 2.6.5.** *For any small cartesian closed category  $\mathbb{C}$ , the Yoneda embedding*

$$y : \mathbb{C} \hookrightarrow \mathbf{Set}^{\mathbb{C}^{\text{op}}}$$

*preserves the cartesian closed structure.*

This is of course the “categorified” analogue of Lemma ??, which we used for the Kripke completeness of the positive propositional calculus PPC.

*Proof.* We can just evaluate  $yA(X) = \mathbb{C}(X, A)$ . It is clear that  $y1(X) = \mathbb{C}(X, 1) \cong 1$  naturally in  $X$ , and that  $y(A \times B)(X) = \mathbb{C}(X, A \times B) \cong \mathbb{C}(X, A) \times \mathbb{C}(X, B) \cong (yA \times yB)(X)$  for all  $A, B, X$ , naturally in all three arguments. For  $B^A \in \mathbb{C}$ , we then have

$$y(B^A)(X) = \mathbb{C}(X, B^A) \cong \mathbb{C}(X \times A, B) \cong \widehat{\mathbb{C}}(y(X \times A), yB) \cong \widehat{\mathbb{C}}(yX \times yA, yB),$$

since  $y$  is full and faithful and, as we just showed, preserves  $\times$ . But now recall that the exponential  $Q^P$  of presheaves  $P, Q$  is defined at  $X$  by the specification

$$Q^P(X) = \widehat{\mathbb{C}}(yX \times P, Q).$$

So, continuing where we left off,  $\widehat{\mathbb{C}}(yX \times yA, yB) = yB^{yA}(X)$ , and we’re done.  $\square$

For an early version of the following theorem (and much more), see the nice paper [Sco80] by Dana Scott.

**Theorem 2.6.6.** *For any  $\lambda$ -theory  $\mathbb{T}$ , we have the following:*

(i) *A type  $A$  is inhabited,*

$$\mathbb{T} \vdash a : A$$

*if, and only if, for every a small category  $\mathbb{C}$ , in every  $\mathbb{T}$ -model  $\llbracket - \rrbracket$  in presheaves  $\widehat{\mathbb{C}}$ , there is a point*

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

(ii) For any terms  $\Gamma \mid s, t : A$ ,

$$\mathbb{T} \vdash (\Gamma \mid s = t : A)$$

if, and only if,

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

for every presheaf model.

*Proof.* We simply specialize the general completeness statement of Corollary 2.4.7 to CCCs of the form  $\widehat{\mathbb{C}}$  using Lemma 2.6.5, together with the fact that the Yoneda embedding is full (and therefore reflects inhabitation) and faithful (and therefore reflects satisfaction of equations).  $\square$

## 2.7 Kripke models

By a Kripke model of (a theory  $\mathbb{T}$  in) the  $\lambda$ -calculus, we mean a model  $\llbracket - \rrbracket$  in the sense of Definition 2.3.1 in a presheaf CCC of the form  $\mathbf{Set}^K$  for a poset  $K$ , *i.e.* a variable model in the sense of the previous section, where the domain of variation is just a *poset*, rather than a proper category. As with Kripke models of propositional logic, we can regard such a model as varying through (branching) time, over a causally ordered state space, or some other (partially-)ordered parameter space. Note that we use “*covariant* presheaves”, *i.e.* functors  $K \rightarrow \mathbf{Set}$ , to model such variable sets, as is more customary in Kripke semantics. By Theorem 2.4.4, such a model  $(K, \llbracket - \rrbracket)$  is essentially the same thing as a CCC functor  $M : \mathcal{C}_{\mathbb{T}} \rightarrow \mathbf{Set}^K$ , taking values in “variable sets”. Regarding the  $\lambda$ -calculus as the proof theory of the propositional calculus via the Curry-Howard correspondence (Section ??), it is perhaps not surprising that it should be (inhabitation) *complete* with respect to such Kripke models, in light of Theorem ??. Completeness with respect to *equations between terms* is entirely another matter, though; while true, the proof is far from a simple generalization of other known results. It can be seen as a verification that the usual notion of  $\beta\eta$ -equivalence is the “right” notion of equality for proofs.

**Example 2.7.1** (Algebraic theories). Before considering such questions, however, let us first spell out explicitly what such a Kripke model looks like for the simple example of a theory  $\mathbb{T}$  of an object with a distinguished element, and a commutative binary operation,

$$\mathbb{T} = (\mathbf{B}, \mathbf{u} : \mathbf{B}, * : \mathbf{B} \times \mathbf{B} \rightarrow \mathbf{B}, \mathbf{x} * \mathbf{y} = \mathbf{y} * \mathbf{x}).$$

There is one basic type symbol  $\mathbf{B}$ , a constant  $\mathbf{u} : \mathbf{B}$ , a binary operation symbol  $* : \mathbf{B} \times \mathbf{B} \rightarrow \mathbf{B}$ , and a single equation  $x, y : \mathbf{B} \mid x * y = y * x : \mathbf{B}$ .

Let  $K$  be a poset with ordering relation  $j \leq k$  for  $j, k \in K$ . Unwinding the general definition for this special case, a *Kripke model*  $M$  of  $\mathbb{T}$  over  $K$  then consists, first, of a family of sets  $(M_k)_{k \in K}$ , equipped with functions

$$m_{j,k} : M_j \rightarrow M_k \quad (\text{for all } j \leq k \in K),$$

satisfying the “compatibility conditions”:

$$m_{k,k} = 1_{M_k}, \quad m_{j,k} \circ m_{i,j} = m_{i,k} \quad (\text{for all } j \leq k \in K).$$

This is of course exactly a functor  $M : K \rightarrow \mathbf{Set}$ , as the interpretation  $M = \llbracket \mathbf{B} \rrbracket$  of the basic type symbol  $\mathbf{B}$ . Such a variable set  $M$  may be thought of as a “set that is changing through time”, in that its elements  $m_j \in M_j$  develop and change at different stages  $j \leq k \in K$ . Note that, while new elements may appear at later stages, and distinct elements may become equal, once present, an element never vanishes, nor do elements ever split apart, because the functions  $m_{j,k} : M_j \rightarrow M_k$  are of course single-valued.

Next, for each  $k \in K$  we have an element

$$u_k : M_k,$$

and these should satisfy

$$m_{j,k}(u_j) = u_k \quad (\text{for all } i \leq j \leq k \in K).$$

That is to say, we have an element or “point”  $u : 1 \rightarrow M$  of  $M$  as a “variable set” in  $\mathbf{Set}^K$ .

Finally, for all  $k \in K$  we need functions

$$s_k : M_k \times M_k \rightarrow M_k$$

satisfying

$$m_{j,k}(s_j(x, y)) = s_k(m_{j,k}(x), m_{j,k}(y)) \quad (\text{for all } j \leq k \in K \text{ and } x, y \in M_j).$$

This is of course just a natural transformation  $s : M \times M \rightarrow M$ , as the interpretation  $s = \llbracket * \rrbracket$  of the operation symbol  $* : \mathbf{B} \times \mathbf{B} \rightarrow \mathbf{B}$ . The idea is that the  $*$ -product of two elements changes along with those elements, which one sees more clearly by writing  $*$  for  $s$ :

$$m_{j,k}(x * y) = m_{j,k}(x) * m_{j,k}(y) \quad (\text{for all } j \leq k \in K \text{ and } x, y \in M_j).$$

In other word, it doesn’t matter “when” one takes the  $*$ -product.

Finally, the interpretation  $(M, u, s) = \llbracket \mathbf{B}, \mathbf{u}, * \rrbracket$  should satisfy the equation  $x, y : \mathbf{B} \mid x * y = y * x : \mathbf{B}$ , meaning that

$$s_k(x, y) = s_k(y, x) \quad (\text{for all } k \in K).$$

This is because two natural transformations are equal just if all of their components are equal. Thus, in sum, a Kripke model of this theory  $\mathbb{T}$  is just a model of the underlying algebraic theory in the functor category  $\mathbf{Set}^K$ , which is of course the same thing as a functor from  $K$  to the usual category of  $\mathbb{T}$ -models in  $\mathbf{Set}$ ,

$$\mathbf{Mod}_{\mathbb{T}}(\mathbf{Set}^K) = \mathbf{Mod}_{\mathbb{T}}(\mathbf{Set})^K.$$

**Example 2.7.2** (Higher-order theories). A theory involving a “higher-order” operation, such as the section  $s : (D \rightarrow D) \rightarrow D$  in (the theory of) a reflexive type (Example 2.1.7) is no more “non-standard” than an algebraic one, once we recall how function types are interpreted, namely *not pointwise*. Let  $D = \llbracket \mathbf{D} \rrbracket$  be the interpretation of the basic type  $\mathbf{D}$ , so that  $\llbracket \mathbf{D} \rightarrow \mathbf{D} \rrbracket = D^D : K \rightarrow \mathbf{Set}$  is a presheaf exponential. At each  $k \in K$ , we then have,

$$(D^D)_k = \mathbf{Set}^K(D \times K(k, -), D).$$

Now observe that this set is trivial except on the upset  $\uparrow k$ , because  $K(k, j)$  is empty unless  $k \leq j$ , so that  $\mathbf{Set}^K(D \times K(k, j), D) = 1$  except when  $j \in \uparrow k$ . On  $\uparrow k$ , it consists of natural transformations

$$\mathbf{Set}^{\uparrow k}(D \uparrow k, D \uparrow k),$$

where  $D \uparrow k : \uparrow k \rightarrow \mathbf{Set}$  is  $D$  restricted to the upset  $\uparrow k \subseteq K$ , *i.e.* the composite

$$\uparrow k \hookrightarrow K \xrightarrow{D} \mathbf{Set}.$$

Given any such natural transformation  $\vartheta : D \uparrow k \rightarrow D \uparrow k$ , and any  $k \leq j$ , the action of the functor,

$$(D^D)_k \rightarrow (D^D)_j$$

on  $\vartheta$  is simply to restrict it further to  $\uparrow j \subseteq \uparrow k$ , thus taking  $\vartheta$  to

$$\vartheta \uparrow j : D \uparrow j \rightarrow D \uparrow j.$$

This is just the same function as  $\vartheta$ , but with the restricted domain of definition  $\uparrow j \subseteq \uparrow k$ . Note that the effect of this restriction may be to identify elements  $\vartheta$ , and that not every element defined at  $j$  need be the restriction of one defined at  $k$  for  $j \leq k$ , so the transition maps need be neither injective nor surjective.

The section  $s : (D \rightarrow D) \rightarrow D$  therefore takes, at each  $k \in K$ , such a  $\vartheta : D \uparrow k \rightarrow D \uparrow k$  to an element  $s_k(\vartheta) \in D_k$ , respecting the restrictions  $\uparrow j \subseteq \uparrow k$  in the sense that

$$d_{k,j}s_k(\vartheta) = s_j(\vartheta \uparrow j) \in D_j,$$

where  $d_{k,j} : D_k \rightarrow D_j$  is the action of the functor  $D : K \rightarrow \mathbf{Set}$ .

In this way, the presheaf exponential  $D^D : K \rightarrow \mathbf{Set}$  is entirely determined by the “base-case”  $D : K \rightarrow \mathbf{Set}$ , and is still a “full function space” at each  $k \in K$ , but the functorial action in  $k$  requires it to not be just  $D_k^{D_k}$  (which for a reflexive type would then be trivial at all  $k \in K$ ). Rather, it must take the entire segment  $\uparrow k$  into account—much in the way that  $k \Vdash \varphi \Rightarrow \psi$  was determined for Kripke models of the intuitionistic propositional calculus IPC by considering all  $j \geq k$ . (Indeed, one can explicitly formulate the Kripke semantics for simple type theory in the usual Kripke-forcing style  $k \Vdash a : A$ , *cf.* [AGH21] and Section ?? below.)

The proof of the following completeness theorem relies on a deep result from topos theory (for the proof of which, see [Joh03, §xy]). We state it in the following form:

**Theorem 2.7.3** (Joyal-Tierney, [JT84]). *For every Grothendieck topos  $\mathcal{E}$  there is a localic topos  $\mathbf{Sh}(L)$  and a connected, locally connected geometric morphism  $c : \mathbf{Sh}(L) \rightarrow \mathcal{E}$ .*

This theorem implies in particular that, for every small CCC  $\mathcal{C}$  there is a poset  $K$  and a fully faithful CCC functor  $\mathcal{C} \hookrightarrow \mathbf{Set}^K$ . From this, the completeness of Kripke semantics then follows easily:

**Theorem 2.7.4** (Kripke completeness for  $\lambda$ -calculus). *For any  $\lambda$ -theory  $\mathbb{T}$ :*

- (i) *A type  $A$  is inhabited just if it has a point  $1 \rightarrow \llbracket A \rrbracket$  in every Kripke model  $(K, \llbracket - \rrbracket)$ .*
- (ii) *Two terms are provably equal,  $\mathbb{T} \vdash (\Gamma \mid s = t : A)$ , just if they are equal in every Kripke model  $(K, \llbracket - \rrbracket)$ ,*

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

In the following chapter, we shall see that this result holds as well for *dependent* type theory with the  $\Sigma$ ,  $\Pi$ , and  $\mathbf{Eq}$  type-formers. For the proof, see [AR11], as well as [AGH21].

**Remark 2.7.5** (For readers familiar with topos theory). Let us see how to get from Theorem 2.7.3 to the fact used here, that for every small CCC  $\mathcal{C}$  there is a poset  $K$  and a fully faithful, CCC functor  $\varphi : \mathcal{C} \hookrightarrow \mathbf{Set}^K$ . First, compose the Yoneda embedding  $y : \mathcal{C} \hookrightarrow \widehat{\mathcal{C}}$  with the inverse image  $c^* : \widehat{\mathcal{C}} \hookrightarrow \mathbf{Sh}(L)$  of the Joyal-Tierney cover, which is also fully faithful and CCC. Then compose further with the inclusion  $i_* : \mathbf{Sh}(L) \hookrightarrow \mathbf{Set}^{L^{\text{op}}}$  of sheaves into presheaves, which is also fully faithful and CCC. So we can take  $K = L^{\text{op}}$  to get the desired CCC embedding  $\varphi = i_* \circ c^* \circ y : \mathcal{C} \hookrightarrow \mathbf{Set}^K$ . See [Awo00, AR11] for more details.

## 2.8 Topological models

From presheaves to posets to spaces to sheaves.

### Posets

Since the category  $\mathbf{Pos}$  is cartesian closed, we can take models of  $\lambda$ -theories there. Are such poset models sufficient to test for provability? The answer depends in general on the kinds of theories: Plotkin [Plo73] shows that for theories with one basic type, no basic terms, and no equations, the models in the category  $\mathbf{Set}$  with the base type interpreted as a *finite* set are already sufficient. And Friedman [Fri75] showed that the single model with one countably infinite base type is also sufficient. For theories with basic terms (but still no equations), other results are known for the category  $\mathbf{Pos}$ ; see [Sim95] for a summary.

We shall show here that for *arbitrary* theories, with basic types, basic terms, and equations, there are enough models in the category  $\mathbf{dopFib}$  of posets and discrete opfibrations, provided these are taken relative to an arbitrary base poset  $K$ .

**Definition 2.8.1.** A *discrete opfibration* of posets is a monotone map  $\pi : P \rightarrow K$  with the property that, for every  $p \in P$  and  $\pi p \leq k \in K$ , there is a unique  $p \leq q \in P$  with  $\pi q = k$ .



This “lifting property” can be equivalently reformulated as saying that every commutative square as follows has a unique diagonal filler, where  $2 = (0 \leq 1)$ .

$$\begin{array}{ccc} 1 & \longrightarrow & P \\ 0 \downarrow & \nearrow & \downarrow \pi \\ 2 & \longrightarrow & K \end{array}$$

**Lemma 2.8.2.** *The (full!) subcategory of all such maps*

$$\mathbf{dopFib}/_K \hookrightarrow \mathbf{Pos}/_K$$

*is equivalent to  $\mathbf{Set}^K$ . In particular, this category is therefore cartesian closed.*

For the proof, one can consider the universal discrete opfibration (with small fibers)  $u : \mathbf{Set}^\bullet \rightarrow \mathbf{Set}$  in  $\mathbf{CAT}$ , the category of large categories. A covariant presheaf  $P : K \rightarrow \mathbf{Set}$  then fits into a pullback diagram

$$\begin{array}{ccc} \int P & \longrightarrow & \mathbf{Set}^\bullet \\ \pi \downarrow \lrcorner & & \downarrow u \\ K & \xrightarrow{P} & \mathbf{Set} \end{array}$$

with the *category of elements*  $\pi : \int_K P \rightarrow K$  on the left, and indeed, every discrete opfibration  $p : D \rightarrow K$  arises in this way for an essentially unique  $P : K \rightarrow \mathbf{Set}$ , namely the one with  $P(k) = p^{-1}(k)$ .

**Exercise 2.8.3.** Fill in the details of the proof just sketched that  $\mathbf{Set}^K \simeq \mathbf{dopFib}/_K$ .

**Exercise 2.8.4.** Show that the inclusion  $\mathbf{dopFib}/_K \hookrightarrow \mathbf{Pos}/_K$  is always full, as claimed in Lemma 2.8.2.

This provides another useful perspective on the functor category  $\mathbf{Set}^K$ . Indeed, one can reformulate the Kripke semantics for simple type theory entirely in terms of discrete opfibrations  $\pi : P \rightarrow K$  in place of (covariant) presheaves  $P : K \rightarrow \mathbf{Set}$ . This will be particularly useful when we consider the semantics of *dependent* type theory in the next chapter.

**Corollary 2.8.5.** *The categories  $\mathbf{dopFib}/_K$  are sufficient for  $\lambda$ -theories: if an equation  $\Gamma \mid s = t : A$  is not provable in  $\mathbb{T}$ , then there is a  $\mathbb{T}$ -model in discrete opfibrations over a poset  $K$  in which the equation fails.*

The analogous statement regarding inhabitation of course also holds. The “fibrational” point of view is pursued in [AR11], which also includes details of the dependently typed case.

**Exercise 2.8.6.** Show that the category  $\mathbf{Set}$  is *not* sufficient for arbitrary  $\lambda$ -theories, with basic types, terms and equations. Is  $\mathbf{Pos}$ ? (*Hint:* Give a Kripke counter-model of triviality for reflexive domains.)

## Sheaves

The category **Pos** of posets may be cartesian closed, but its slices  $\mathbf{Pos}/_P$  are in general not so (by an argument similar to the one given in [Pal03]). By contrast, the (wide but not full) subcategory **dopFib** of posets and discrete opfibrations is *not* cartesian closed (proof!), whereas its slices  $\mathbf{dopFib}/_K \simeq \mathbf{Set}^K$  always are so. Something similar happens with topological spaces: the category **Top** of all spaces and continuous maps is itself not cartesian closed, nor are its slices, but there is a (wide but not full) subcategory  $\mathbf{LocHom} \hookrightarrow \mathbf{Top}$  the slices of which are always cartesian closed, even though **LocHom** itself is not. As in the case of **dopFib**, this is most easily seen by showing that each slice  $\mathbf{LocHom}/_X$  is actually equivalent to a functor category known to be cartesian closed, namely the category  $\mathbf{Sh}(X)$  of *sheaves* on the space  $X$ .

### A little topos theory: Local homeomorphisms.

**Definition 2.8.7.** A sheaf on a space  $X$  is a presheaf  $F : (\mathcal{O}X)^{\text{op}} \rightarrow \mathbf{Set}$  that satisfies the following “patching” condition: for every open cover  $U = \bigcup_{i \in I} U_i$  the canonical map

$$FU \rightarrow \prod_i FU_i \rightrightarrows \prod_{i,j} F(U_i \cap U_j)$$

is an equalizer.

Explanation and reformulation in terms of Hom-sets.

**Proposition 2.8.8.** *The full subcategory  $\mathbf{Sh}(X) \hookrightarrow \mathbf{Set}^{\mathcal{O}(X)^{\text{op}}}$  is cartesian closed, with the structure inherited from presheaves.*

*Proof.* fill in! □

**Definition 2.8.9.** A continuous function  $f : Y \rightarrow X$  is a *local homeomorphism* if for every  $y \in Y$  there is an open set  $y \in U$  such that (i) the image  $f(U) \subseteq X$  is open, and (ii) the restriction  $f|_U : U \rightarrow f(U)$  is a homeomorphism.

Let  $\mathbf{LocHom} \hookrightarrow \mathbf{Top}$  be the subcategory of spaces and local homeomorphisms, and  $\mathbf{LocHom}/_X$  the slice category. Thus a map  $f : (Y, p) \rightarrow (Z, q)$  of local homeomorphisms over  $X$  is a commutative triangle with  $p : Y \rightarrow X$  and  $q : Z \rightarrow X$  local homeomorphisms. One can show that in fact every merely *continuous* map  $g : Y \rightarrow Z$  making a commutative triangle  $q \circ g = p$  is also a local homeomorphism (exercise!).

**Proposition 2.8.10.** *There is an equivalence of categories  $\mathbf{Sh}(X) \simeq \mathbf{LocHom}/_X$ . In particular, every slice  $\mathbf{LocHom}/_X$  is a cartesian closed category.*

*Proof.* fill in! □

It is of some interest to know that semantics in spaces and local homeomorphisms is already sufficient for theories in the  $\lambda$ -calculus, but it is also of use to know that the  $\lambda$ -calculus can be used as an *internal language* to reason about sheaves over a space! The completeness theorem is proved using a refinement of the Joyal-Tierney covering theorem 2.7.3 due to Moerdijk [?], see [Awo00, ?].

## Equiological spaces

See [?].



# 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.
- [Awo00] Steve Awodey. Topological representation of the  $\lambda$ -calculus. *Mathematical Structures in Computer Science*, 10:81–96, 2000.
- [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>.
- [Fri75] H. Friedman. Equality between functionals. In R. Parikh, editor, *Logic Colloquium*. Springer-Verlag, New York, 1975.
- [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.
- [Pal03] Erik Palmgren. Groupoids and local cartesian closure. 08 2003. unpublished.

- [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.
- [Sco80] 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.