Introduction to Categorical Logic

[DRAFT: March 29, 2023]

Steve Awodey   Andrej Bauer

Contents

Chapter 3

The λ-calculus

3.1 Curry-Howard as categorification

Consider the following natural deduction proof in propositional calculus.

$$\dfrac{\dfrac{\dfrac{\dfrac{[(A\wedge B)\wedge(A\Rightarrow B)]^1}{A\wedge B}}{A} \qquad \dfrac{[(A\wedge B)\wedge(A\Rightarrow B)]^1}{A\Rightarrow B}}{B}}{(A\wedge B)\wedge(A\Rightarrow B)\Rightarrow B}\ {\scriptstyle (1)}$$

This deduction shows that

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

But so does the following:

$$\dfrac{\dfrac{\dfrac{[(A\wedge B)\wedge(A\Rightarrow B)]^1}{A\Rightarrow B} \qquad \dfrac{\dfrac{[(A\wedge B)\wedge(A\Rightarrow B)]^1}{A\wedge B}}{A}}{B}}{(A\wedge B)\wedge(A\Rightarrow B)\Rightarrow B}\ {\scriptstyle (1)}$$

As does:

$$\dfrac{\dfrac{\dfrac{[(A\wedge B)\wedge(A\Rightarrow B)]^1}{A\wedge B}}{B}}{(A\wedge B)\wedge(A\Rightarrow B)\Rightarrow B}\ {\scriptstyle (1)}$$

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

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

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

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

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

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

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

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

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

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

3.2 Cartesian closed categories

Exponentials

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

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

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

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

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

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

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

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

$$\begin{tikzcd} B^A & B^A \times A \arrow[r, "\epsilon"] & B \\ C \arrow[u, dotted, "\widetilde{f}"] & C \times A \arrow[u, "\widetilde{f} \times 1_A"] \arrow[ur, "f"'] & \end{tikzcd}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

Two characterizations of CCCs

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

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

has a right adjoint

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

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

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

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

as indicated below.

$$\begin{tikzcd} B^A \times A \arrow[r, "\epsilon_B"] \arrow[d, "g^A \times 1_A"'] & B \arrow[d, "g"] \\ C^A \times A \arrow[r, "\epsilon_C"'] & C \end{tikzcd} \tag{3.1}$$

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

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

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

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

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

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

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

□

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

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

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

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

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

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

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

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

$$\begin{tikzcd} & C \arrow[dl, "g"'] \arrow[d, "\langle f, g \rangle"] \arrow[dr, "f"] & \\ A & A \times B \arrow[l, "\pi_0"] \arrow[r, "\pi_1"'] & B \end{tikzcd}$$

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

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

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

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

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

Some proper CCCs

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

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

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

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

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

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

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

The foregoing examples are instances of the following general situation.

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

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

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

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

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

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

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

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

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

□

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- Etale spaces over a base space X. This category can be described as consisting of *local homeomorphisms* $f : Y \to X$ and commutative triangles over X between such maps. It is equivalent to the category $\mathsf{Sh}(X)$ of *sheaves* on X. See [**?**, ch.n].

- Various subcategories of topological spaces (sequential spaces, compactly-generated spaces). Cf. [**?**].

- Dana Scott's category Equ of equilogical spaces [**?**].

3.3 Simple type theory

The λ-calculus is the abstract theory of functions, just like group theory is the abstract theory of symmetries. There are two basic operations that can be performed with functions. The first one is the *application* of a function to an argument: if f is a function and a is an argument, then fa is the application of f to a, also called the *value* of f at a. The second operation is *abstraction*: if x is a variable and t is an expression in which x may appear, then there is a function f defined by the equation

$$fx = t\ .$$

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

$$x \mapsto t\ ,$$

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

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

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

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

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

$$\lambda x.\, (xx)$$

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

$$t : A\ .$$

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

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

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

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

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

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

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

In words, this means:

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

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

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

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

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

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

A *typing judgment* is a judgment of the form

$$\Gamma \mid t : A$$

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

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

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

The type of a variable is determined by the context:

$$\frac{}{x_1 : A_1, \ldots, x_i : A_i, \ldots, x_n : A_n \mid x_i : A_i} \;(1 \leq i \leq n)$$

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

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

The typing rules for pairs and projections are:

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

The typing rules for application and λ-abstraction are:

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

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

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

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:

1. Equality is an equivalence relation:

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

2. The weakening rule:

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

3. Unit type:

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

4. Equations for product types:

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

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

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

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

5. Equations for function types:

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

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

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

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

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

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

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

We summarize the preceding definitions.

Definition 3.3.1. A *(simply-typed) signature* for the λ-calculus is given by a set of *basic types* and a set of *basic constants* together with their types. *A simply-typed λ-theory* is a simply-typed signature together with a set of equations.

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

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

with the following equations:

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

These are just the familiar axioms for a group.

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

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

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

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

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

There are no equations.

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

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

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

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

Example 3.3.6. An example of a λ-theory is readily found in the theory of programming languages. The mini-programming language PCF is a simply-typed λ-calculus with a basic type `nat` for natural numbers, and a basic type `bool` of Boolean values,

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

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

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

The equational axioms of PCF are:

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

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

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

satisfying the equation

$$f : \mathtt{D} \to \mathtt{D} \mid \mathtt{r}\,(\mathtt{s}\,f) = f : \mathtt{D} \to \mathtt{D} \tag{3.2}$$

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

$$x : \mathtt{D} \mid \mathtt{s}\,(\mathtt{r}\,x) = x : \mathtt{D} \tag{3.3}$$

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

Untyped λ-calculus

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

$$t ::= v \mid t_! \, t_2 \mid \lambda x.\, t \;.$$

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

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

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

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

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

The rules of equality are as follows:

1. Equality is an equivalence relation:

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

2. The weakening rule:

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

3. Equations for application and λ-abstraction:

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

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

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

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

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

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

$$\Gamma \mid u = t \tag{3.4}$$

is translated to the judgment

$$\Gamma^* \mid u^* = t^* : \mathtt{D}\,. \tag{3.5}$$

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

Bibliography

[AK08] Steve Awodey and Kohei Kishida. Topology and modality: Topological semantics for first-order modal logic. *The Review of Symbolic Logic*, 1:146–166, 2008.

[AR11] S. Awodey and F. Rabe. Kripke semantics for Martin-Löf's extensional type theory. *Logical Methods in Computer Science*, 7(3):1–25, 2011.

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

[MH92] Michael Makkai and Victor Harnik. Lambek's categorical proof theory and Läuchli's abstract realizability. *Journal of Symbolic Logic*, 57(1):200–230, 1992.

[ML84] Per Martin-Löf. *Intuitionistic type theory*, volume 1 of *Studies in Proof Theory*. Bibliopolis, 1984.

[Sco70] Dana Scott. Constructive validity. In M. Laudet, D. Lacombe, L. Nolin, and M. Schützenberger, editors, *Symposium on Automatic Demonstration*, volume 125, pages 237–275. Springer-Verlag, 1970.

[Tai68] William W. Tait. Constructive reasoning. In *Logic, Methodology and Philos. Sci. III (Proc. Third Internat. Congr., Amsterdam, 1967)*, pages 185–199. North-Holland, Amsterdam, 1968.