Notes on Type Theory

[DRAFT: October 20, 2025]

Steve Awodey

with contributions from Andrej Bauer

Contents

Chapter 1

Introduction to Type Theory

1.1 A little history

We begin with a few historical remarks, intended to correct a common misconception about the origins of type theory. For an excellent survey of the history of modern type theory see [Coq22].

The history of type theory is closely tied to the history of logic, to which it is closely related, but distinct. Modern logic emerged together with modern algebra in the 19th century, as something like the algebra of "propositions", as opposed to that of numbers or quantities. As emphasized by Frege, the distinctive feature of logic was the notion of *truth*, which establishes its connection to language, thought, judgement, and other anthropocentric notions—as opposed to, say, numbers and sets, which could be regarded as existing independently of human activity. Although Frege himself strove to establish logic as an objective science, he struggled to define its basic objects in a way that did not rely on their symbolic (one would now say "syntactic") representations. *Objects* were determined as things that could be *named*, possibly by complex symbolic expressions. The basic notion of a *function* was, for Frege, something that derived from a complex name for an object by allowing a constituent name to be replaced by another (think of forming an expression for a polynomial function from an algebraic expression for a number). In this way, the basis of logic was tied to the relation between symbolic expressions and their *meaning* (German *Bedeutung*).

In addition to names of objects, and functional expressions regarded as fragmentary or incomplete names, there were sentences, which were treated simply as names for objects of a special kind, for which Frege coined the term *truth value* (German: *Wahrheitswert*). Functions whose values were truth values – whose expressions were therefore fragmentary sentences (i.e. predicates, or formulas with variables) – were called *concepts* (German: *Begriffe*). In this way, Frege's system of logic was based on (i) objects, including truth values, and functions, including concepts, all of which were in the objective realm of "things", and (ii) their symbolic or linguistic expressions, which were regarded as being in a separate realm. Mediating between the two realms was a third one called that of "senses" (German:

	Symbols	Senses	Things
Total	names, sentences	propositions	objects, truth values
Partial	formulas, predicates	properties	functions, concepts

Table 1.1: Frege's logical inventory

Sinne), which might now be called the *intensions* and included things like *propositions*, which Frege called "thoughts" (German: *Gedanken*).

In this way, certain kinds of *things* (like functions) were inferred, or assumed, to correspond to certain kinds of *expressions*. Accordingly, there were also assumed to be higher-order functions, which resulted from complex expressions by removing the expressions for functions and allowing these arguments to vary (a quantifier is an example of such a higher function). Frege introduced a systematic use of different kinds of variables, notation for variable binding, and other devices to permit the correct formal manipulation of expressions denoting not only objects, but also functions of several arguments, functions of functions, etc. He insisted, moreover, on a strict regimentation of the entities denoted by such expressions. A function of functions could no more take an object as argument than could a noun replace a verb to make a sentence. In this way, Frege's universe of logical entities was partitioned into a rigid hierarchy of disjoint kinds that Russell later called *logical types*.

To be a bit more precise, if we let o denote the type of all objects (including truth values), and $A \to B$ the type of functions from type A to type B, then Frege's system of all logical types T is generated simply by the rules:

$$T ::= \ o \mid (T_1, \ldots, T_n) \to o$$

where $(T_1, \ldots, T_n) \to o$ represents the type of o-valued functions in several arguments of types $T_1, \ldots, T_n$, respectively. (This display can be read as a recursive specification as follows: o is a type; if $T_1, \ldots, T_n$ are types, then $(T_1, \ldots, T_n) \to o$ is a type; and nothing else is a type.)

According to Russell [**?**], a *type* may be defined as "the range of significance of a *propositional function*", which corresponds roughly to Frege's partition of all functions according to what arguments they can take, with the ones taking no arguments being the objects. Whereas for Frege, the values of such functions were arbitrary objects, for Russell they were restricted to being *propositions*, thus corresponding (roughly[1]) to Frege's concepts. Representing the type of propositions by p, Russell's hierarchy of logical types is then generated by the similar rules:

$$T ::= \ o \mid p \mid (T_1, \ldots, T_n) \to p$$

which is to say, all (higher-order) relations on objects and propositions. Of course, Russell did not rest with this, but also introduced a further "ramification", determined by the

[1]We are suppressing the subtlety that Russell's functions were *intensional*, at least in the original formulation, and so took values in propositions, rather than truth values.

quantificational structure of the expression specifying a given (propositional) function. This was despite the fact that the unramified theory already sufficed to block the particular inconsistency in Frege's system that Russell had discovered [**?**]; nonetheless, Russell was worried that other inconsistencies might yet result without the more elaborate ramified theory of types.

The main point if these remarks is that Frege's type theory, which roughly agrees with what is now called *simple type theory*, was in fact *not* motivated by Russell's discovery of an inconsistency, but rather by a principled consideration of the nature of functions and their relationship to the symbolic expressions by which they can be determined. Unfortunately, Frege later effectively violated those restrictions by introducing the notion of the *extension* (German: *Wertverlauf*) of a function to play the role of a proxy *object*. For ϕ a concept, the extension $\hat{x}\phi$ was essentially the *set* of objects satisfying the concept (the extension of a general function was its "course of values", something like its "graph"). It was this assumption of extensions that led to the inconsistency in his system, via the infamous Law V, which in modern predicate logical notation reads innocently enough as an "axiom of extensionality",

$$\hat{x}\phi = \hat{x}\psi \;\Leftrightarrow\; \forall x(\phi = \psi). \tag{V}$$

It is indeed ironic that Frege, who first formulated, and strenuously insisted on, the natural rules of logical types, ultimately fell victim to violating those very rules.

1.2 Proof relevance

An important aspect of Frege's logic that distinguished it from the algebraic tradition of the time was his emphasis on a rigorous formal system of *derivations*, specified by *rules of inference* that made reference only to the outward logical ("syntactic") form of the expressions, and not to what they were assumed to mean. Frege regarded such formal "gap-free" proofs as essential in determining the logical character of a judgement, unlike some later logicians who regarded the logical status of a judgement as a property of the expression alone, possibly determined by consideration of external "semantic" interpretations. The *constructive* tradition in logic and foundations can be seen as descending from Frege's invention of a formal deductive system for determining the truth of a judgement, and his insistence on the importance of this notion; the related idea of *proof-relevance* in constructive logic and modern theoretical computer science is arguably further evidence for the validity of his point of view. The interaction between logic (with its emphasis on *truth*) and type theory (which emphasizes *proofs*) is encapsulated in constructive logic and type theory by the *Curry-Howard correspondence*. In these notes, we shall attempt to highlight this relationship from yet another point of view: the interaction between conventional algebraic structures and what we shall call *categorified* algebraic structures.

In a bit more detail:

- The *Curry-Howard correspondence* is sometimes presented as a somewhat mysterious connection between (say, propositional) logic and (simple) type theory, according

to which the "meaning" of a propositional formula is not just a truth-value (or a truth table of values), but rather the collection of its proofs. The Propositions-as-Types/Proofs-as-Terms (or Proofs-as-Programs) paradigm is then a proof-theoretic (or computational) alternative to Tarskian, truth-value semantics. The same correspondence also extends to first-order logic and dependent type theory, as we shall see below.

- The algebraic/categorical version of this correspondence is then as follows: not only does propositional *logic* interpret into Boolean and Heyting algebras (and first- and higher-order logic in (pre)toposes), but we also have categorical semantics of the associated *type theories*, like the λ-calculus which is modeled by (locally) cartesian closed categories, such as categories of (pre)sheaves.

- The *poset* algebra of truth-values used for the semantics of propositional or predicate logic (e.g. the Boolean algebra $\{0, 1\}$) is seen to be the *poset reflection* of a suitably structured, proper *category* (e.g. Set), which models a type theory that, in turn, is the "proof-relevant" version of the logic. The general scheme can be represented as follows, with the first row being the proof-relevant version of the first:

Type Theory	Category
Logic	Algebra

  Indeed, a third axis could be added for propositional versus predicate logic and simple versus dependent type theories:

Logic	Algebra	Type Theory	Category
Propositional	Boolean algebra	Simple	CCC
Predicate	Boolean category	Dependent	LCCC

- The relationship between *validity* and *provability*, which is classically described by the relationship between logic and type theory, is described categorically by the (adjoint) notions of categorical generalization and "poset reflection" between (structured) posets and categories. In this way, the Curry-Howard correspondence relates to the idea of "categorification": a structured category whose poset reflection is a given structured poset. For example, a category with finite products is a categorification of a $\wedge$-semilattice, and the category Set is a categorification of the Boolean algebra $\{0, 1\}$.

Finally, some new ideas have recently deepened this perspective: the original Curry-Howard paradigm, relating truth-value semantics (model theory) with type-theoretic syntax (proof theory), has turned out to capture only the first two levels of an infinite hierarchy of levels of structure. These levels are not merely cumulative, but are related by inclusion, truncation, (co-)reflection, and other operations. The importance of "proof-relevance" that underlies the Propositions-as-Types idea is essentially just one special case of the *coherence*

issue that arises everywhere in higher category theory. And the once-bold replacement of *truth-values* and *sets* by *types* in constructive logic and the foundations of computation parallels the replacement of discrete structures (sheaves) by "higher" ones (stacks) in algebra and geometry—except that we have now learned that the gap between the levels is not just a single step, but rather an infinite hierarchy of levels of structure, each just as significant as the first step.

These insights are reflected in current categorical logic in the recent extension from algebraic logic (level 0) and topos theory (level 1) to higher topos theory and homotopy type theory (level ∞). The latter are the focus of much current research, and the unification of the various earlier topics that has been achieved already shows how much we have learned about what happens in passing from 0 to 1, by passing from the finite to the infinite.

1.3 The Curry-Howard correspondence

Consider the following natural deduction proof in propositional calculus. This deduction shows that

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

But so does the following: As does:

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:

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.

1.4 Categorification

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

$$\pi_2(x)(\pi_1(\pi_1(x))) : (A \wedge B) \wedge (A \Rightarrow B) \longrightarrow B,$$
$$\pi_2(\pi_1(x)) : (A \wedge B) \wedge (A \Rightarrow B) \longrightarrow B.$$

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

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 simple type theory—as the proof theory of propositional logic, and as a formal system for manipulating functions—is sometimes also referred to as the "Curry-Howard correspondence" [Sco70, ML84, Tai68]. From the categorical point of view, it expresses an equivalence between two cartesian closed categories: that of *proofs* in propositional logic and that of *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 next chapter, we shall consider this remarkable correspondence in more detail, as well as some extensions of the basic case to λ-calculus, respectively cartesian closed categories, with sums, with natural numbers objects, and with modal operators. In the subsequent chapter, it will be seen that this correspondence extends even further to proofs in quantified predicate logic via dependent type theory and locally cartesian closed categories, and far beyond.

1.5 Completeness via representation theorems

As an example of the sort of reasoning that we shall extend from logic to type theory by "categorification", we sketch the proof of the Kripke completeness theorem for Intuitionistic Propositional Logic (IPL) via Joyal's representation theorem for Heyting algebras. For a fuller exposition see [Awo, §2.1]. We begin with a basic system without the coproducts $\bot$ or $\phi \vee \psi$, and thus also without negation $\neg\phi = \phi \Rightarrow \bot$, which we shall therefore call the *positive propositional calculus* (a non-standard designation).

1.5.1 Positive propositional calculus

Classically, implication $\phi \Rightarrow \psi$ can be defined by $\neg\phi \vee \psi$, but in categorical logic we prefer to consider $\phi \Rightarrow \psi$ as an *exponential* of ψ by ϕ defined as right adjoint to the conjunction $(-) \wedge \phi$, applied to to the argument ψ. Since this makes sense without negation $\neg\phi$ or joins $\phi \vee \psi$, we can study just the cartesian closed fragment separately, and then add those other operations later. The same approach will be used for type theory.

Definition 1.5.1. The *positive propositional calculus* PPC is the subsystem of the full propositional calculus (see [Awo, §2.1]) containing just (finite) conjunction and implication. So PPC is the set of all propositional formulas ϕ constructed from propositional variables $p_1, p_2, ...$, a constant $\top$ for *true*, and the binary connectives of conjunction $\phi \wedge \psi$ and implication $\phi \Rightarrow \psi$.

The system of deduction for PPC has one form of judgement

$$\phi_1, \ldots, \phi_m \vdash \phi$$

where the formulas $\phi_1, \ldots, \phi_m$ are called the *assumptions* (or *hypotheses*) and ϕ is the *conclusion*. The assumptions are regarded as a (finite) set; so they are unordered, have no repetitions, and may be empty. *Deductive entailment*, also denoted $\Phi \vdash \phi$, is a relation between finite sets of formulas Φ and single formulas ϕ, and is defined as the smallest such relation satisfying the following rules:

1. Hypothesis:
$$\frac{}{\Phi \vdash \phi} \text{ if } \phi \in \Phi$$
2. Truth:
$$\frac{}{\Phi \vdash \top}$$
3. Conjunction:
$$\frac{\Phi \vdash \phi \qquad \Phi \vdash \psi}{\Phi \vdash \phi \wedge \psi} \qquad \frac{\Phi \vdash \phi \wedge \psi}{\Phi \vdash \phi} \qquad \frac{\Phi \vdash \phi \wedge \psi}{\Phi \vdash \psi}$$
4. Implication:
$$\frac{\Phi, \phi \vdash \psi}{\Phi \vdash \phi \Rightarrow \psi} \qquad \frac{\Phi \vdash \phi \Rightarrow \psi \qquad \Phi \vdash \phi}{\Phi \vdash \psi}$$

A *proof* of a judgement $\Phi \vdash \phi$ is a *finite* tree built from the above inference rules the root of which is $\Phi \vdash \phi$, and the leaves of which are either the Truth rule or an instance of the Hypothesis rule. A judgment $\Phi \vdash \phi$ is *provable* if it has a proof.

Remark 1.5.2. An alternate form of presentation for proofs in natural deduction that is more, well, natural uses trees of formulas, rather than of judgements, with leaves labelled by *assumptions* ϑ that may also occur in *cancelled* form $[\vartheta]$. Thus for example the introduction and elimination rules for conjunction would be written in the form:

$$\frac{\begin{array}{cc}\Phi & \Phi \\ \vdots & \vdots \\ \phi & \psi\end{array}}{\phi \wedge \psi} \qquad \frac{\begin{array}{c}\Phi \\ \vdots \\ \phi \wedge \psi\end{array}}{\phi} \qquad \frac{\begin{array}{c}\Phi \\ \vdots \\ \phi \wedge \psi\end{array}}{\psi}$$

An example of a proof tree with (some) cancelled assumptions is the above rule of implication introduction, which takes the form: A proof tree in which all the assumptions have been cancelled represents a derivation of an unconditional judgement such as $\vdash \phi$.

We will have a better way to record such proofs using the λ-calculus in the next chapter.

As a category, PPC is a *preorder* under the relation $\phi \vdash \psi$ of logical entailment. As usual, it will be convenient to pass to the *poset reflection* of the preorder by identifying ϕ and ψ when $\phi \dashv\vdash \psi$. This poset category is called the *Lindenbaum-Tarski* algebra of the system PPC, and we denote it by

$$\mathcal{C}_{\mathsf{PPC}}\,.$$

The conjunction $\phi \wedge \psi$ is a greatest lower bound of ϕ and ψ in $\mathcal{C}_{\mathsf{PPC}}$, because $\phi \wedge \psi \vdash \phi$ and $\phi \wedge \psi \vdash \psi$, and for all ϑ, if $\vartheta \vdash \phi$ and $\vartheta \vdash \psi$ then $\vartheta \vdash \phi \wedge \psi$. Since binary products in a poset are the same thing as greatest lower bounds, we see that $\mathcal{C}_{\mathsf{PPC}}$ has all binary products; and of course $\top$ is a terminal object, so $\mathcal{C}_{\mathsf{PPC}}$ is a $\wedge$-semilattice. We have already remarked that implication is right adjoint to conjunction in the sense that for any ϕ, there is an adjunction between the monotone maps,

$$(-) \wedge \phi \;\dashv\; \phi \Rightarrow (-) : \mathcal{C}_{\mathsf{PPC}} \longrightarrow \mathcal{C}_{\mathsf{PPC}}\,. \tag{1.1}$$

Therefore $\phi \Rightarrow \psi$ is an exponential in $\mathcal{C}_{\mathsf{PPC}}$. The counit of the adjunction (the "evaluation" arrow) is the entailment

$$(\phi \Rightarrow \psi) \wedge \phi \vdash \psi\ ,$$

i.e. the familiar logical rule of *modus ponens*.

We therefore have the following:

Proposition 1.5.3. *The poset $\mathcal{C}_{\mathsf{PPC}}$ of positive propositional calculus is cartesian closed.*

We will use this fact to show that the positive propositional calculus is *deductively complete* with respect to the following notion of *Kripke semantics* [**?**].

Definition 1.5.4 (Kripke semantics)**.** We summarize this briefly as follows:

1. A *Kripke model* is a poset K (the "worlds") equipped with a relation

   $$k \Vdash p$$

   between elements $k \in K$ and propositional variables p, such that for all $j \in K$,

   $$j \leq k,\ k \Vdash p \quad \text{implies} \quad j \Vdash p\,. \tag{1.2}$$

2. Given a Kripke model $(K, \Vdash)$, extend the relation $\Vdash$ to all formulas ϕ in PPC by defining the relation of *holding in a world $k \in K$* inductively by the following conditions:

   $$\begin{array}{lll} k \Vdash \top & \text{always}, & \\ k \Vdash \phi \wedge \psi & \text{iff} & k \Vdash \phi \text{ and } k \Vdash \psi\,, \\ k \Vdash \phi \Rightarrow \psi & \text{iff} & \text{for all } j \leq k, \text{ if } j \Vdash \phi, \text{ then } j \Vdash \psi\,. \end{array} \tag{1.3}$$

3. Finally, say that ϕ *holds in the Kripke model* $(K, \Vdash)$, written

$$K \Vdash \phi$$

if $k \Vdash \phi$ for all $k \in K$. (One sometimes also says that ϕ *holds on* the poset K if $K \Vdash \phi$ for all such Kripke relations $\Vdash$ on K.)

Theorem 1.5.5 (Kripke completeness for PPC)**.** *A propositional formula ϕ is provable from the rules of deduction for PPC if, and only if, $K \Vdash \phi$ for all Kripke models $(K, \Vdash)$,*

$$\mathsf{PPC} \vdash \phi \qquad \textit{iff} \qquad K \Vdash \phi \quad \textit{for all } (K, \Vdash).$$

For the proof, we first require the following.

Lemma 1.5.6. *For any poset P, the poset $\mathsf{Down}(P)$ of all downsets in P, ordered by inclusion, is cartesian closed. Moreover, the downset embedding,*

$$\downarrow(-) : P \longrightarrow \mathsf{Down}(P)$$

preserves any CCC structure that exists in P.

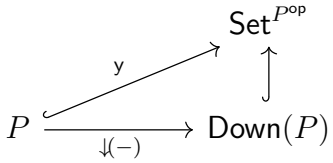
Proof. The total downset P is obviously terminal, and for any downsets $S, T \in \mathsf{Down}(P)$, the intersection $S \cap T$ is also closed down, so we have the products $S \wedge T = S \cap T$. For the exponential, let

$$S \Rightarrow T = \{p \in P \mid \downarrow(p) \cap S \subseteq T\}. \tag{1.4}$$

Then for any downset Q we have

$$\begin{aligned} Q \subseteq S \Rightarrow T \quad &\text{iff} \quad \text{for all } q \in Q, \quad q \in S \Rightarrow T\,, \\ &\text{iff} \quad \text{for all } q \in Q, \quad \downarrow(q) \cap S \subseteq T\,, \\ &\text{iff} \quad \textstyle\bigcup_{q \in Q} (\downarrow(q) \cap S) \subseteq T\,, \\ &\text{iff} \quad (\textstyle\bigcup_{q \in Q} \downarrow(q)) \cap S \subseteq T\,, \\ &\text{iff} \quad Q \cap S \subseteq T\,. \end{aligned}$$

The preservation of CCC structure by $\downarrow(-) : P \longrightarrow \mathsf{Down}(P)$ follows from its preservation by the Yoneda embedding, of which $\downarrow(-)$ is a factor,



Indeed, we can identify $\mathsf{Down}(P)$ with the subcategory $\mathsf{Sub}(1) \hookrightarrow \mathsf{Set}^{P^{\mathsf{op}}}$ of subobjects of the terminal presheaf 1, and the result then follows easily by using the left adjoint left inverse sup of the inclusion

$$\mathsf{sup} \dashv i : \mathsf{Sub}(1) \hookrightarrow \mathsf{Set}^{P^{\mathsf{op}}}\,,$$

to be considered later (*cf.* Lemma 2.6.1).

But it is also easy enough to check the preservation of CC structure directly: preservation of the limits 1, $p \wedge q$ are immediate from the definitions. Suppose $p \Rightarrow q$ is an exponential in P; then for any downset D we have:

$$\begin{aligned} D \subseteq {\downarrow}(p \Rightarrow q) \quad &\text{iff} \quad d \in {\downarrow}(p \Rightarrow q) \text{ , for all } d \in D \\ &\text{iff} \quad d \leq p \Rightarrow q \text{ , for all } d \in D \\ &\text{iff} \quad d \wedge p \leq q \text{ , for all } d \in D \\ &\text{iff} \quad {\downarrow}(d \wedge p) \subseteq {\downarrow}(q) \text{ , for all } d \in D \\ &\text{iff} \quad {\downarrow}(d) \cap {\downarrow}(p) \subseteq {\downarrow}(q) \text{ , for all } d \in D \\ &\text{iff} \quad D \subseteq {\downarrow}(p) \Rightarrow {\downarrow}(q) \end{aligned}$$

where the last line is by (1.4). Now take D to be ${\downarrow}(p \Rightarrow q)$ and ${\downarrow}(p) \Rightarrow {\downarrow}(q)$ respectively (or just apply Yoneda!). (Note that in line (3) we assumed that $d \wedge p$ exists for all $d \in D$; this can be avoided by a slightly more complicated argument.) □

We can now give the proof of the completeness theorem. It follows a standard pattern, which we will see again.

Proof. (of Theorem 1.5.5)

1. The syntactic category $\mathcal{C}_{\mathsf{PPC}}$ is a CCC, with $\top = 1$, $\phi \times \psi = \phi \wedge \psi$, and $\psi^{\phi} = \phi \Rightarrow \psi$. In fact, it is evidently the free cartesian closed poset on the generating set $\mathsf{Var} = \{p_1, p_2, \dots\}$ of propositional variables.

2. By Step 1 and the fact that $\mathsf{Down}(K)$ is cartesian closed, Lemma 1.5.6, a CCC functor $\mathcal{C}_{\mathsf{PPC}} \to \mathsf{Down}(K)$ is just an arbitrary map $\mathsf{Var} \to \mathsf{Down}(K)$. But this is just a (Kripke) model $(K, \Vdash)$, as in (1.2).

3. Thus we have a bijective correspondence between Kripke relations $\Vdash : K^{\mathsf{op}} \times \mathsf{Var} \longrightarrow \mathbb{2}$, arbitrary maps $\mathsf{Var} \to \mathsf{Down}(K)$, CCC functors $[\![-]\!] : \mathcal{C}_{\mathsf{PPC}} \to \mathsf{Down}(K)$, and monotone maps (also called) $\Vdash : K^{\mathsf{op}} \times \mathcal{C}_{\mathsf{PPC}} \longrightarrow \mathbb{2}$:

$$\cfrac{\cfrac{\cfrac{\Vdash : K^{\mathsf{op}} \times \mathsf{Var} \longrightarrow \mathbb{2}}{[\![-]\!] : \mathsf{Var} \longrightarrow \mathbb{2}^{K^{\mathsf{op}}} \cong \mathsf{Down}(K)}}{[\![-]\!] : \mathcal{C}_{\mathsf{PPC}} \longrightarrow \mathsf{Down}(K) \cong \mathbb{2}^{K^{\mathsf{op}}}}}{\Vdash : K^{\mathsf{op}} \times \mathcal{C}_{\mathsf{PPC}} \longrightarrow \mathbb{2}}$$

   where we use the poset $\mathbb{2}$ to classify downsets in the poset K, or equivalently, upsets in K^{op} (the contravariance will be convenient in Step 6). Here we are using the CCC structure of the category of posets. Note that the monotonicity of $\Vdash$ in the last line yields both of the conditions

$$j \leq k \,,\; k \Vdash \phi \implies j \Vdash \phi$$

and

$$k \Vdash \phi\,,\ \phi \vdash \psi \implies k \Vdash \psi\,.$$

4. Moreover, the CCC preservation of the map $[\![-]\!]$ in the third line yields the Kripke forcing conditions (1.3) (exercise!).

5. For any model $(K, \Vdash)$, by the adjunction in (3) we then have

$$K \Vdash \phi \iff [\![\phi]\!] = K\,,$$

where $K \subseteq K$ is the maximal downset.

6. Because the downset embedding $\downarrow$ preserves the CCC structure (by Lemma 1.5.6), $\mathcal{C}_{\mathsf{PPC}}$ has a *canonical model*, namely the special case of (3) with $K = \mathcal{C}_{\mathsf{PPC}}$ and $\Vdash$ resulting from the transposition:

$$\frac{\downarrow(-)\ :\ \mathcal{C}_{\mathsf{PPC}} \longrightarrow \mathsf{Down}(\mathcal{C}_{\mathsf{PPC}}) \cong 2^{\mathcal{C}_{\mathsf{PPC}}^{\mathsf{op}}}}{\Vdash\ :\ \mathcal{C}_{\mathsf{PPC}}^{\mathsf{op}} \times \mathcal{C}_{\mathsf{PPC}} \longrightarrow 2}$$

7. Now observe that for the Kripke relation $\Vdash$ in (6) we therefore have $\Vdash\ = \vdash$, since it is the transpose of the downset embedding, and the poset $\mathcal{C}_{\mathsf{PPC}}$ is ordered by $\phi \vdash \psi$. So the canonical model $(\mathcal{C}_{\mathsf{PPC}}, \Vdash)$ is *logically generic*, in the sense that

$$\phi \Vdash \psi \quad \iff \quad \phi \vdash \psi\,.$$

Thus in particular,

$$\mathcal{C}_{\mathsf{PPC}} \Vdash \phi \quad \iff \quad \mathsf{PPC} \vdash \phi\,.$$

The case of a general $(K, \Vdash)$ now follows easily. □

Exercise 1.5.7. Verify the claim in (4) that CCC preservation of the transpose $[\![-]\!]$ of $\Vdash$ yields the Kripke forcing conditions (1.3).

Exercise 1.5.8. Give a Kripke countermodel to show that $\mathsf{PPC} \nvdash (\phi \Rightarrow \psi) \Rightarrow \phi$.

1.5.2 Heyting algebras

Let us now extend the positive propositional calculus to the full intuitionistic propositional calculus. This involves adding the finite coproducts 0 and $p \vee q$ to the notion of a cartesian closed poset, to arrive at the general notion of a Heyting algebra. Heyting algebras are to intuitionistic logic as Boolean algebras are to classical logic: each is an algebraic description of the corresponding logical calculus. We shall review both the algebraic and the logical points of view; as we shall see, many aspects of the theory of Boolean algebras carry over to Heyting algebras. For instance, in order to prove the Kripke completeness of the full system of intuitionistic propositional calculus, we will need an alternative to Lemma 1.5.6, because the Yoneda embedding does not in general preserve coproducts. For that we will again use a version of the Stone representation theorem (see [Awo, §2.7]), this time in a generalized form due to Joyal.

Distributive lattices

Recall first that a (bounded) *lattice* is a poset that has finite limits and colimits. In other words, a lattice $(L, \leq, \wedge, \vee, 1, 0)$ is a poset $(L, \leq)$ with distinguished elements $1, 0 \in L$, and binary operations of meet $\wedge$ and join $\vee$, satisfying for all $x, y, z \in L$,

$$0 \leq x \leq 1 \qquad \frac{z \leq x \qquad z \leq y}{z \leq x \wedge y} \qquad \frac{x \leq z \qquad y \leq z}{x \vee y \leq z}$$

A *lattice homomorphism* is a function $f : L \to K$ between lattices which preserves finite limits and colimits, i.e., $f0 = 0$, $f1 = 1$, $f(x \wedge y) = fx \wedge fy$, and $f(x \vee y) = fx \vee fy$. The category of lattices and lattice homomorphisms is denoted by Lat.

Lattices are an algebraic theory, and can be axiomatized equationally in a signature with two distinguished elements 0 and 1 and two binary operations $\wedge$ and $\vee$, satisfying the following equations:

$$\begin{aligned} (x \wedge y) \wedge z &= x \wedge (y \wedge z)\ , & (x \vee y) \vee z &= x \vee (y \vee z)\ , \\ x \wedge y &= y \wedge x\ , & x \vee y &= y \vee x\ , \\ x \wedge x &= x\ , & x \vee x &= x\ , \\ 1 \wedge x &= x\ , & 0 \vee x &= x\ , \\ & x \wedge (y \vee x) = x = (x \wedge y) \vee x\ . \end{aligned} \tag{1.5}$$

The partial order on L is then determined by

$$x \leq y \iff x = x \wedge y\ .$$

Exercise 1.5.9. Show that in a lattice we also have $x \leq y$ if and only if $x \vee y = y$.

A lattice is *distributive* if the following distributive laws hold:

$$\begin{aligned} (x \vee y) \wedge z &= (x \wedge z) \vee (y \wedge z)\ , \\ (x \wedge y) \vee z &= (x \vee z) \wedge (y \vee z)\ . \end{aligned} \tag{1.6}$$

It turns out that if one distributive law holds then so does the other [Joh82, I.1.5].

Definition 1.5.10. A *Heyting algebra* is a cartesian closed lattice. This means that a Heyting algebra $\mathcal{H}$ has a binary operation of *implication* $x \Rightarrow y$, satisfying the following condition, for all $x, y, z \in \mathcal{H}$:

$$\frac{z \leq x \Rightarrow y}{z \wedge x \leq y}$$

A *Heyting algebra homomorphism* is a lattice homomorphism $f : \mathcal{K} \to \mathcal{H}$ between Heyting algebras that preserves implication, i.e., $f(x \Rightarrow y) = (fx \Rightarrow fy)$. The category of Heyting algebras and their homomorphisms is denoted by Heyt. (*Caution*: unlike Boolean

algebras, the subcategory of lattices consisting of Heyting algebras and their homomorphisms is *not full*.)

Heyting algebras can be axiomatized equationally as a set H with two distinguished elements 0 and 1 and three binary operations $\wedge$, $\vee$ and $\Rightarrow$. The equations for a Heyting algebra are the ones listed in (1.5), as well as the following ones for $\Rightarrow$.

$$\begin{aligned} (x \Rightarrow x) &= 1 \,, \\ x \wedge (x \Rightarrow y) &= x \wedge y \,, \\ y \wedge (x \Rightarrow y) &= y \,, \\ (x \Rightarrow (y \wedge z)) &= (x \Rightarrow y) \wedge (x \Rightarrow z) \,. \end{aligned} \tag{1.7}$$

For a proof, see [Joh82, I.1], where one can also find a proof that every Heyting algebra is distributive (exercise!).

Exercise 1.5.11. Show that every Heyting algebra is indeed a distributive lattice.

Example 1.5.12. We know from Lemma 1.5.6 that for any poset P, the poset $\mathsf{Down}(P)$ of all downsets in P, ordered by inclusion, is cartesian closed. Moreover, we know that

$$\mathsf{Down}(P) \cong \mathsf{Pos}(P^{\mathsf{op}}, \mathbb{2}) \,,$$

the latter regarded as a poset with the pointwise ordering on the monotone maps $P^{\mathsf{op}} \to \mathbb{2}$ (*i.e.* the natural transformations). The assignment takes a map $f : P^{\mathsf{op}} \to \mathbb{2}$ to the filter-kernel $f^{-1}(1) \subseteq P^{\mathsf{op}}$, which is therefore an upset in P^{op}, and so a downset in P.

Since $\mathbb{2}$ is a lattice, we can take joins $f \vee g$ in $\mathsf{Pos}(P^{\mathsf{op}}, \mathbb{2})$ pointwise, in order to get joins in $\mathsf{Down}(P) \cong \mathsf{Pos}(P^{\mathsf{op}}, \mathbb{2})$, which then correspond to (set theoretic) unions of the corresponding downsets $f^{-1}(1) \cup g^{-1}(1)$. Thus for any poset P, the lattice $\mathsf{Down}(P)$ is a Heyting algebra, with the downsets ordered by inclusion, and the (contravariant) classifying maps $P^{\mathsf{op}} \to \mathbb{2}$ ordered pointwise:

Proposition 1.5.13. *For any poset P, the poset $\mathsf{Down}(P)$ of all downsets in P ordered by inclusion is a Heyting algebra. The lattice operations of meet and join agree with the set-theoretic ones of intersection and union. The Heyting implication $S \Rightarrow T$ is given as in* (1.4) *by:*

$$S \Rightarrow T \;=\; \{p \in P \mid {\downarrow}(p) \cap S \subseteq T\}.$$

Of course, one can also compose the classifying maps with the negation iso $\neg : \mathbb{2} \xrightarrow{\sim} \mathbb{2}$ to get $\mathsf{Down}(P) \cong \mathsf{Pos}(P, \mathbb{2})$, with *covariant* classifying maps $P \to \mathbb{2}$ for the downsets, using the ideal-kernels $f^{-1}(0) \subseteq P$ instead of the filters; but then the ordering on $\mathsf{Pos}(P, \mathbb{2})$ will be the *reverse pointwise ordering* of maps $f : P \to \mathbb{2}$.

Example 1.5.14. For any topological space X, the poset of open sets $\mathcal{O}(X)$ is a Heyting algebra, with the lattice operations inherited from the powerset, and the Heyting implication given by

$$U \Rightarrow V = \bigcup \{W \in \mathcal{O}X \mid W \cap U \subseteq V\} \,.$$

Intuitionistic propositional calculus

There is an obvious forgetful functor $U : \mathsf{Heyt} \to \mathsf{Set}$ mapping a Heyting algebra to its underlying set, and a homomorphism of Heyting algebras to the underlying function. Because Heyting algebras are also models of an equational theory, there is a left adjoint $H \dashv U$, which is the usual "free" construction for algebras, mapping a set X to the free Heyting algebra $H(X)$ generated by it. As for all algebraic structures, the construction of $H(X)$ can be performed in two steps: first, define a set $H[X]$ of formal expressions in the signature, and then quotient it by an equivalence relation generated by the equations (1.5) and (1.7).

In more detail, let $H[X]$ be the set of formal expressions generated inductively by the following rules:

1. Generators: if $x \in X$ then $x \in H[X]$.

2. Constants: $\bot, \top \in H[X]$.

3. Connectives: if $\phi, \psi \in H[X]$ then $(\phi \wedge \psi), (\phi \vee \psi), (\phi \Rightarrow \psi) \in H[X]$.

We then impose an equivalence relation $\sim$ on $H[X]$, defined as the smallest equivalence relation containing all instances of the axioms (1.5) and (1.7) and closed under substitution of equals for equals (sometimes called the smallest *congruence*). This then forces the quotient

$$H(X) = H[X]/_\sim$$

to be a Heyting algebra, as is easily checked.

We define the action of the functor H on morphisms as usual: a function $f : X \to Y$ is mapped to the Heyting algebra homomorphism $H(f) : H(X) \to H(Y)$ (well-)defined (on equivalence classes) by

$$H(f)\bot = \bot\,, \qquad H(f)\bot = \bot\,, \qquad H(f)x = fx\,,$$
$$H(f)(\phi * \psi) = (H(f)\phi) * (H(f)\psi)\,,$$

where $*$ stands for $\wedge$, $\vee$ or $\Rightarrow$.

The inclusion of generators $\eta_X : X \to UH(X)$ into the underlying set of the free Heyting algebra $H(X)$ is then the component at X of a natural transformation $\eta : \mathsf{1}_{\mathsf{Set}} \Longrightarrow U \circ H$, which is of course the unit of the adjunction $H \dashv U$. To see this, consider a Heyting algebra $\mathcal{K}$ and an arbitrary function $f : X \to U\mathcal{K}$. Then the Heyting algebra homomorphism $\overline{f} : H(X) \to \mathcal{K}$ is defined in the evident way, by

$$\overline{f}\bot = \bot\,, \qquad \overline{f}\bot = \bot\,, \qquad \overline{f}x = fx\,,$$
$$\overline{f}(\phi * \psi) = (\overline{f}\phi) * (\overline{f}\psi)\,,$$

where, again, $*$ stands for $\wedge$, $\vee$ or $\Rightarrow$. The map $\overline{f}$ then makes the following triangle in Set commute:

$$\begin{array}{ccc} X & \xrightarrow{\eta_X} & UH(X) \\ & \searrow^{f} & \downarrow{\scriptstyle U\overline{f}} \\ & & U\mathcal{K} \end{array}$$

The homomorphism $\overline{f} : H(X) \to \mathcal{K}$ is the unique one with this property, because any two homomorphisms from $H(X)$ that agree on generators must clearly be equal (formally, this can be proved by induction on the structure of the expressions in $H[X]$).

We can now define the *intuitionistic propositional calculus* IPC to be the free Heyting algebra $H(p_0, p_1, \dots)$ on countably many generators $\{p_0, p_1, \dots\}$, called *atomic propositions* or *propositional variables*. This is a somewhat unorthodox definition from a logical point of view—normally we would start from a *deductive calculus* consisting of a formal language, entailment judgements, and rules of inference. But of course, by now, we realize that the two approaches are essentially equivalent.

Having said that, let us also briefly describe IPC in the conventional way: The formulas of IPC are built inductively as usual from propositional variables p_0, p_1, $\dots$, constants false $\bot$ and true $\top$, and binary operations $\wedge$, disjunction $\vee$ and implication $\Rightarrow$.

The rules are those of the positive calculus 1.5.1, together with the following:

5. Falsehood:
$$\frac{\Phi \vdash \bot}{\Phi \vdash \phi}$$

6. Disjunction:
$$\frac{\Phi \vdash \phi}{\Phi \vdash \phi \vee \psi} \qquad \frac{\Phi \vdash \psi}{\Phi \vdash \phi \vee \psi} \qquad \frac{\Phi \vdash \phi \vee \psi \qquad \Phi, \phi \vdash \theta \qquad \Phi, \psi \vdash \theta}{\Phi \vdash \theta}$$

For the purpose of deduction, we define $\neg\phi := \phi \Rightarrow \bot$ and $\phi \Leftrightarrow \psi := (\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)$.

Then let $\mathcal{C}_{\mathsf{IPC}}$ be the poset reflection of the formulas of IPC, preordered by entailment $\phi \vdash \psi$. The elements of $\mathcal{C}_{\mathsf{IPC}}$ are thus equivalence classes $[\phi]$ of formulas, where two formulas ϕ and ψ are equivalent if both $\phi \vdash \psi$ and $\psi \vdash \phi$ are provable in natural deduction,

$$[\phi] = [\psi] \iff \phi \dashv\vdash \psi\,.$$

This syntactic category $\mathcal{C}_{\mathsf{IPC}}$ is then easily seen to be the *free Heyting algebra* on the countably many generators $\{p_0, p_1, \dots\}$,

$$\mathcal{C}_{\mathsf{IPC}} \cong H(p_0, p_1, \dots)\,,$$

just as the corresponding Lindenbaum-Tarski algebra $\mathcal{C}_{\mathsf{PPC}}$ was seen to be the free CCC poset on the propositional variables.

Classical propositional calculus

Let us have a brief look at the theory of classical propositional logic from the current point of view, *i.e.* as a special kind of Heyting algebra. An element $x \in L$ of a lattice L is said to be *complemented* when there exists $y \in L$ such that

$$x \wedge y = 0 \,, \qquad\qquad x \vee y = 1 \,.$$

We say that y is the *complement* of x. In a distributive lattice, the complement of x is unique if it exists. Indeed, if both y and z are complements of x then

$$y \wedge z = (y \wedge z) \vee 0 = (y \wedge z) \vee (y \wedge x) = y \wedge (z \vee x) = y \wedge 1 = y \,,$$

hence $y \leq z$. A symmetric argument shows that $z \leq y$, therefore $y = z$. The complement of x, if it exists, is denoted by $\neg x$.

A *Boolean algebra* may be defined as a distributive lattice in which every element is complemented. In other words, a Boolean algebra B has a *complementation operation* $\neg : B \to B$ which satisfies, for all $x \in B$,

$$x \wedge \neg x = 0 \,, \qquad\qquad x \vee \neg x = 1 \,. \tag{1.8}$$

The full subcategory of Lat consisting of Boolean algebras is denoted by BA.

Exercise 1.5.15. Prove that every Boolean algebra is a Heyting algebra. (*Hint*: how is implication encoded in terms of negation and disjunction in classical logic?)

In a Heyting algebra, not every element is complemented. However, we can still define a *pseudo complement* or *negation* operation $\neg$ by

$$\neg x = (x \Rightarrow 0) \,,$$

Then $\neg x$ is the largest element for which $x \wedge \neg x = 0$. While in a Boolean algebra $\neg\neg x = x$, in a Heyting algebra we only have $x \leq \neg\neg x$ in general. An element x of a Heyting algebra for which $x = \neg\neg x$ is called *regular*.

Exercise 1.5.16. Derive the following properties of negation in a *Heyting* algebra:

$$\begin{aligned} x &\leq \neg\neg x \,, \\ \neg x &= \neg\neg\neg x \,, \\ x \leq y &\Rightarrow \neg y \leq \neg x \,, \\ \neg\neg(x \wedge y) &= \neg\neg x \wedge \neg\neg y \,. \end{aligned}$$

Exercise 1.5.17. We know from Example 1.5.14 that the topology $\mathcal{O}(X)$ of any topological space X is a Heyting algebra. Describe in topological language the implication $U \Rightarrow V$, the negation $\neg U$, and the regular elements $U = \neg\neg U$ in $\mathcal{O}X$.

Exercise 1.5.18. Show that for a Heyting algebra H, the regular elements of H form a Boolean algebra $H_{\neg\neg} = \{x \in H \mid x = \neg\neg x\}$. Here $H_{\neg\neg}$ is viewed as a subposet of H. Hint: negation $\neg'$, conjunction $\wedge'$, and disjunction $\vee'$ in $H_{\neg\neg}$ are expressed as follows in terms of negation, conjunction and disjunction in H, for $x, y \in H_{\neg\neg}$:

$$\neg' x = \neg x \ , \qquad x \wedge' y = \neg\neg(x \wedge y) \ , \qquad x \vee' y = \neg\neg(x \vee y) \ .$$

From logical point of view, the *classical propositional calculus* CPC is obtained from the intuitionistic propositional calculus by the addition of either one of the following additional rules.

7. Classical logic:

$$\frac{}{\Phi \vdash \phi \vee \neg\phi} \qquad \frac{\Phi \vdash \neg\neg\phi}{\Phi \vdash \phi}$$

Identifying logically equivalent formulas of CPC, we obtain a poset $\mathcal{C}_{\mathsf{CPC}}$ ordered by logical entailment. This poset is, of course, the *free Boolean algebra* on the countably many generators $\{p_0, p_1, \dots\}$. The free Boolean algebra can be constructed just as the free Heyting algebra above, either equationally, or in terms of deduction. The equational axioms for a Boolean algebra are the axioms for a lattice (1.5), the distributive laws (1.6), and the complement laws (1.8).

Exercise* 1.5.19. Is $\mathcal{C}_{\mathsf{CPC}}$ isomorphic to the Boolean algebra $\mathcal{C}_{\mathsf{IPC}_{\neg\neg}}$ of the regular elements of $\mathcal{C}_{\mathsf{IPC}}$?

Exercise 1.5.20. Show that in a Heyting algebra H, one has $\neg\neg x = x$ for all $x \in H$ if, and only if, $y \vee \neg y = 1$ for all $y \in H$. *Hint*: half of the equivalence is easy. For the other half, observe that the assumption $\neg\neg x = x$ means that negation is an order-reversing bijection $H \to H$. It therefore transforms joins into meets and vice versa, and so the *De Morgan laws* hold:

$$\neg(x \wedge y) = \neg x \vee \neg y \ , \qquad \neg(x \vee y) = \neg x \wedge \neg y \ .$$

Together with $y \wedge \neg y = 0$, the De Morgan laws easily imply $y \vee \neg y = 1$. See [Joh82, I.1.11].

Kripke semantics for IPC

Let us now prove the Kripke completeness of IPC, extending Theorem 1.5.5, namely:

Theorem 1.5.21 (Kripke completeness for IPC)**.** *Let $(K, \Vdash)$ be a Kripke model,* i.e. *a poset K equipped with a* forcing relation *$k \Vdash p$ between elements $k \in K$ and propositional variables p, satisfying*

$$j \leq k, \ k \Vdash p \quad \textit{implies} \quad j \Vdash p. \tag{1.9}$$

Extend $\Vdash$ *to all formulas* ϕ *in* IPC *by defining*

$$\begin{array}{lll} k \Vdash \top & \textit{always,} & \\ k \Vdash \bot & \textit{never,} & \\ k \Vdash \phi \wedge \psi & \textit{iff} & k \Vdash \phi \textit{ and } k \Vdash \psi\,, \qquad (1.10) \\ k \Vdash \phi \vee \psi & \textit{iff} & k \Vdash \phi \textit{ or } k \Vdash \psi\,, \qquad (1.11) \\ k \Vdash \phi \Rightarrow \psi & \textit{iff} & \textit{for all } j \leq k\textit{, if } j \Vdash \phi\textit{, then } j \Vdash \psi\,. \end{array}$$

Finally, write $K \Vdash \phi$ *if* $k \Vdash \phi$ *for all* $k \in K$*.*

A propositional formula ϕ *is then provable from the rules of deduction for* IPC *if, and only if,* $K \Vdash \phi$ *for all Kripke models* $(K, \Vdash)$*. Briefly:*

$$\mathsf{IPC} \vdash \phi \quad \textit{iff} \quad K \Vdash \phi \textit{ for all } (K, \Vdash).$$

Let us first see that we cannot simply reuse the proof from Theorem 1.5.5 for the positive fragment PPC, because the downset (Yoneda) embedding that we used there

$$\downarrow \,: \mathcal{C}_{\mathsf{PPC}} \hookrightarrow \; \mathsf{Down}(\mathcal{C}_{\mathsf{PPC}}) \tag{1.12}$$

would not preserve the coproducts $\bot$ and $\phi \vee \psi$. Indeed, $\downarrow(\bot) \neq \emptyset$, because it contains $\bot$ itself! And in general $\downarrow(\phi \vee \psi) \neq \downarrow(\phi) \cup \downarrow(\psi)$, because the righthand side need not contain, e.g., $\phi \vee \psi$.

Instead, we will generalize the Stone Representation theorem [Awo, §2.6] from Boolean algebras to Heyting algebras using a theorem due to A. Joyal (cf. [MR95, MH92]). First, recall that the Stone representation provides, for any Boolean algebra $\mathcal{B}$, an injective Boolean homomorphism into a powerset,

$$\mathcal{B} \rightarrowtail \mathcal{P}X\,.$$

For X we take the set of prime filters, which we can identify with the homset of Boolean homomorphisms $\mathsf{BA}(\mathcal{B}, \mathbb{2})$ by taking the filter-kernel $f^{-1}(1) \subseteq \mathcal{B}$ of a homomorphism $f : \mathcal{B} \to \mathbb{2}$. The injective homomorphism $\eta : \mathcal{B} \rightarrowtail \mathcal{P}(\mathsf{BA}(\mathcal{B}, \mathbb{2}))$ is then given by:

$$\eta(b) = \{F \,|\, b \in F\} = \{f : \mathcal{B} \to \mathbb{2} \,|\, f(b) = 1\}\,.$$

Now, the set $\mathsf{BA}(\mathcal{B}, \mathbb{2})$ can be regarded as a (discrete) poset, and since the inclusion $\mathsf{Set} \hookrightarrow \mathsf{Pos}$ as discrete posets is left adjoint to the forgetful functor $|-| : \mathsf{Pos} \to \mathsf{Set}$, for the powerset $\mathcal{P}(\mathsf{BA}(\mathcal{B}, \mathbb{2}))$ we have

$$\mathcal{P}(\mathsf{BA}(\mathcal{B}, \mathbb{2})) \cong \mathsf{Set}(\mathsf{BA}(\mathcal{B}, \mathbb{2}), 2) \cong \mathsf{Pos}(\mathsf{BA}(\mathcal{B}, \mathbb{2}), \mathbb{2}) \cong \mathbb{2}^{\mathsf{BA}(\mathcal{B}, \mathbb{2})}$$

where the latter is the exponential in the cartesian closed category Pos. Composing with the Stone representation $\eta : \mathcal{B} \rightarrowtail \mathcal{P}(\mathsf{BA}(\mathcal{B}, \mathbb{2}))$ and transposing in Pos,

$$\frac{\mathcal{B} \rightarrowtail \mathcal{P}(\mathsf{BA}(\mathcal{B}, \mathbb{2})) \cong \mathbb{2}^{\mathsf{BA}(\mathcal{B}, \mathbb{2})}}{\mathsf{BA}(\mathcal{B}, \mathbb{2}) \times \mathcal{B} \to \mathbb{2}}$$

we arrive at the (monotone) evaluation map

$$\mathsf{eval} : \mathsf{BA}(\mathcal{B}, \mathbb{2}) \times \mathcal{B} \to \mathbb{2}. \tag{1.13}$$

Finally, recall that the category of Boolean algebras is full in the category DLat of distributive lattices, so that

$$\mathsf{BA}(\mathcal{B}, \mathbb{2}) = \mathsf{DLat}(\mathcal{B}, \mathbb{2})\,.$$

Now for any *Heyting algebra* $\mathcal{H}$ (or indeed any distributive lattice), the homset $\mathsf{DLat}(\mathcal{H}, \mathbb{2})$, ordered pointwise, is isomorphic to the *poset* of all prime filters in $\mathcal{H}$ ordered by inclusion, again by taking $h : \mathcal{H} \to \mathbb{2}$ to its (filter) kernel $h^{-1}\{1\} \subseteq \mathcal{H}$. In particular, when $\mathcal{H}$ is not Boolean, the poset $\mathsf{DLat}(\mathcal{H}, \mathbb{2})$ is no longer discrete, since prime filters in a Heyting algebra need not be maximal. Indeed, recall that Proposition **??** described the prime filters in a Boolean algebra $\mathcal{B}$ as those with a classifying map $f : \mathcal{B} \to \mathbb{2}$ that is a lattice homomorphism and therefore those with a complement $f^{-1}(0) \subseteq \mathcal{B}$ that is a (prime) ideal. In the Boolean case, these were also the *maximal* filters, because the preservation of Boolean negation $\neg b$ allowed us to deduce that for every $b \in \mathcal{B}$, exactly one of b or $\neg b$ must be in such a filter F. In a Heyting algebra, however, the last condition need not obtain; and indeed prime filters in a Heyting algebra need not be maximal.

The transpose in Pos of the evaluation map,

$$\mathsf{eval} : \mathsf{DLat}(\mathcal{H}, \mathbb{2}) \times \mathcal{H} \to \mathbb{2}. \tag{1.14}$$

is again a monotone map

$$\eta \,:\, \mathcal{H} \longrightarrow \mathbb{2}^{\mathsf{DLat}(\mathcal{H},\mathbb{2})}, \tag{1.15}$$

which takes $p \in \mathcal{H}$ to the "evaluation at p" map $f \mapsto f(p) \in \mathbb{2}$, i.e.,

$$\eta_p(f) = f(p) \qquad \text{for } p \in \mathcal{H} \text{ and } f : \mathcal{H} \to \mathbb{2}\,.$$

As before (cf. Example 1.5.12), the poset $\mathbb{2}^{\mathsf{DLat}(\mathcal{H},\mathbb{2})}$ (ordered pointwise) may be identified with the downsets in the poset $\mathsf{DLat}(\mathcal{H}, \mathbb{2})^{\mathsf{op}}$, ordered by inclusion, which recall from Example 1.5.12 is always a Heyting algebra. Thus, in sum, for any Heyting algebra $\mathcal{H}$, we have a monotone map,

$$\eta : \mathcal{H} \longrightarrow \mathsf{Down}(\mathsf{DLat}(\mathcal{H}, \mathbb{2})^{\mathsf{op}})\,, \tag{1.16}$$

generalizing the Stone representation from Boolean to Heyting algebras.

Theorem 1.5.22 (Joyal)**.** *Let $\mathcal{H}$ be a Heyting algebra. There is an injective homomorphism of Heyting algebras*

$$\mathcal{H} \;\rightarrowtail\; \mathsf{Down}(J)$$

into the Heyting algebra of downsets in a poset J.

Note that in this form, the theorem literally generalizes the Stone representation theorem: when $\mathcal{H}$ is Boolean we can take J to be discrete, and then $\mathsf{Down}(J) \cong \mathsf{Pos}(J, \mathbb{2}) \cong \mathsf{Set}(J, 2) \cong \mathcal{P}(J)$ is Boolean, whence the Heyting embedding is also Boolean.

The proof will again use the transposed evaluation map,

$$\eta : \mathcal{H} \longrightarrow 2^{\mathsf{DLat}(\mathcal{H},2)} \cong \mathsf{Down}(\mathsf{DLat}(\mathcal{H},\mathbb{2})^{\mathsf{op}})$$

which, as before, is injective, by the Prime Ideal Theorem (see [Awo] Lemma **??**). We will use the latter in the following form due to Birkhoff.

Lemma 1.5.23 (Prime Ideal Theorem)**.** *Let D be a distributive lattice, $I \subseteq D$ an ideal, and $x \in D$ with $x \notin I$. There is a prime ideal $I \subseteq P \subset D$ with $x \notin P$.*

Proof. As in the proof of Lemma **??**, it suffices to prove it for the case $I = (0)$. This time, we use Zorn's Lemma: a poset in which every chain has an upper bound has maximal elements. Consider the poset $\mathcal{I}\backslash x$ of "ideals I without x", $x \notin I$, ordered by inclusion. The union of any chain $I_0 \subseteq I_1 \subseteq ...$ in $\mathcal{I}\backslash x$ is clearly also in $\mathcal{I}\backslash x$, so we have (at least one) maximal element $M \in \mathcal{I}\backslash x$. We claim that $M \subseteq D$ is prime. To that end, take $a, b \in D$ with $a \wedge b \in M$. If $a, b \notin M$, let $M[a] = \{n \leq m \vee a \mid m \in M\}$, the ideal join of M and $\downarrow(a)$, and similarly for $M[b]$. Since M is maximal without x, we therefore have $x \in M[a]$ and $x \in M[b]$. Thus let $x \leq m \vee a$ and $x \leq m' \vee b$ for some $m, m' \in M$. Then $x \vee m' \leq m \vee m' \vee a$ and $x \vee m \leq m \vee m' \vee b$, so taking meets on both sides gives

$$(x \vee m') \wedge (x \vee m) \leq (m \vee m' \vee a) \wedge (m \vee m' \vee b) = (m \vee m') \vee (a \wedge b).$$

Since the righthand side is in the ideal M, so is the left. But then $x \leq x \vee (m \wedge m')$ is also in M, contrary to our assumption that $M \in \mathcal{I}\backslash x$. □

Proof of Theorem 1.5.22. As in (1.16), let $J^{\mathsf{op}} = \mathsf{DLat}(\mathcal{H}, \mathbb{2})$ be the poset of prime filters in $\mathcal{H}$, and consider the transposed evaluation map (1.16),

$$\eta : \mathcal{H} \longrightarrow \mathsf{Down}(\mathsf{DLat}(\mathcal{H},\mathbb{2})^{\mathsf{op}}) \cong 2^{\mathsf{DLat}(\mathcal{H},2)} \tag{1.17}$$

given by $\eta(p) = \{F \,|\, p \in F \text{ prime}\} \cong \{f : \mathcal{H} \to \mathbb{2} \,|\, f(p) = 1\}$.

Clearly $\eta(0) = \emptyset$ and $\eta(1) = \mathsf{DLat}(\mathcal{H}, \mathbb{2})$, and similarly for the other meets and joins, so η is a lattice homomorphism. Moreover, if $p \neq q \in \mathcal{H}$ then, as in the proof of **??**, we have that $\eta(p) \neq \eta(q)$, by the Prime Ideal Theorem (Lemma 1.5.23). Thus it only remains to show that

$$\eta(p \Rightarrow q) \;=\; \eta(p) \Rightarrow \eta(q)\,.$$

Unwinding the definitions, this means that, for all $f \in \mathsf{DLat}(\mathcal{H}, \mathbb{2})$,

$$f(p \Rightarrow q) = 1 \quad \text{iff} \quad \text{for all } g \geq f,\ g(p) = 1 \text{ implies } g(q) = 1. \tag{1.18}$$

Equivalently, for all prime filters $F \subseteq \mathcal{H}$,

$$p \Rightarrow q \in F \quad \text{iff} \quad \text{for all prime } G \supseteq F,\ p \in G \text{ implies } q \in G. \tag{1.19}$$

Now if $p \Rightarrow q \in F$, then for all (prime) filters $G \supseteq F$, also $p \Rightarrow q \in G$, and so $p \in G$ implies $q \in G$, since $(p \Rightarrow q) \wedge p \leq q$.

Conversely, suppose $p \Rightarrow q \notin F$, and we seek a prime filter $G \supseteq F$ with $p \in G$ but $q \notin G$. Consider the filter

$$F[p] = \{x \wedge p \leq h \in \mathcal{H} \mid x \in F\},$$

which is the join of F and $\uparrow(p)$ in the poset of filters. If $q \in F[p]$, then $x \wedge p \leq q$ for some $x \in F$, whence $x \leq p \Rightarrow q$, and so $p \Rightarrow q \in F$, contrary to assumption; thus $q \notin F[p]$. By the Prime Ideal Theorem again (applied to the distributive lattice $\mathcal{H}^{\mathsf{op}}$) there is a prime filter $G \supseteq F[p]$ with $q \notin G$. □

The proof of the Kripke completeness theorem 1.5.21 now proceeds as in the case of PPC: given a formula ϕ such that $K \Vdash \phi$ in every Kipke model $(K, \Vdash)$, then in particular, for the universal model $J = \mathsf{DLat}(\mathsf{IPC}, \mathbb{2})^{\mathsf{op}}$, ordered by inclusion of downsets, we must have $J \Vdash \phi$. This means that for the embedding (1.17), we have $\eta\phi = \mathsf{IPC} = \eta\top$, the maximal downset, and so by injectivity of η, we must have $\top \vdash \phi$. The converse follows from the universality of IPC as the free Heyting algebra.

The classical case of "truth tables" results by considering arbitrary assignments v from $\mathsf{Var} = \{p_0, p_1, \dots\}$ to truth values $\mathbb{2}$, which correspond to Boolean homomorphisms from the free Boolean algebra $\mathcal{B}(p_0, p_1, \dots)$, and therefore to maximal filters in $\mathcal{B}(p_0, p_1, \dots)$. Joyal's representation theorem then agrees with that of Stone, and the resulting completeness theorem is then exactly the classical one for CPC: a propositional formula is provable in CPC if, and only if, its truth value computes to 1 by the usual "truth-table rules" (i.e. the definition of a Boolean homomorphism) under every assignment $v : \{p_0, p_1, \dots\} \to \mathbb{2}$ of truth values to the propositional letters it contains.

Corollary 1.5.24 (Completeness for classical PC)**.** *The classical propositional calculus is deductively complete with respect to truth-value semantics ("truth tables").*

Exercise 1.5.25. Give a Kripke countermodel to show that the Law of Excluded Middle $\phi \vee \neg\phi$ is not provable in IPC.

Topological semantics for IPC

Finally, we recall the topological interpretation of IPC, because it will also be generalized to type theory. It is clear how to interpret IPC into a topological space X: each formula ϕ is assigned to an open set $[\![\phi]\!] \in \mathcal{O}X$ in such a way that $[\![-]\!]$ is a homomorphism of Heyting algebras.

Definition 1.5.26. A *topological model* of IPC consists of a space X and a function

$$[\![-]\!] : \mathsf{Var} \to \mathcal{O}(X)$$

from the propositional variables $\mathsf{Var} = \{p_0, p_1, \dots\}$ to open sets of X. The interpretation is then extended to all formulas,

$$[\![-]\!] : \mathsf{IPC} \to \mathcal{O}(X),$$

by setting:

$$\begin{aligned}
[\![\top]\!] &= X \\
[\![\bot]\!] &= \emptyset \\
[\![\phi \wedge \psi]\!] &= [\![\phi]\!] \cap [\![\psi]\!] \\
[\![\phi \vee \psi]\!] &= [\![\phi]\!] \cup [\![\psi]\!] \\
[\![\phi \Rightarrow \psi]\!] &= [\![\phi]\!] \Rightarrow [\![\psi]\!] \,.
\end{aligned}$$

where the Heyting implication $[\![\phi]\!] \Rightarrow [\![\psi]\!]$ in $\mathcal{O}X$, is defined as in Example 1.5.14 as

$$[\![\phi]\!] \Rightarrow [\![\psi]\!] \;=\; \bigcup \big\{ U \in \mathcal{O}X \;\big|\; U \cap [\![\phi]\!] \subseteq [\![\psi]\!] \big\} \;.$$

Joyal's representation theorem 1.5.22 then easily implies that IPC is sound and complete with respect to topological semantics.

Corollary 1.5.27. *A formula ϕ is provable in IPC if, and only if, it holds in every topological interpretation $[\![-]\!]$ into a space X, briefly:*

$$\mathsf{IPC} \vdash \phi \qquad \textit{iff} \qquad [\![\phi]\!] = X \textit{ for all spaces } X \,.$$

Proof. Put the Alexandroff topology on the downsets of prime filters in the Heyting algebra IPC. □

Exercise 1.5.28. Give a topological countermodel to show that the Law of Double Negation $\neg\neg\phi \Rightarrow \phi$ is not provable in IPC.

1.6 Outline

Here is a preliminary outline of the course:

1. Introduction (week 1)
   (a) Logic and type theory
   (b) Proof relevance, Curry-Howard, categorification
   (c) Soundness and completeness via embedding and representation theorems
2. Simply Typed Lambda-Calculus (weeks 2-5)
   (a) Lambda theories and their models in CCCs
   (b) Classifying category and functorial semantics
   (c) Completeness in CCCs
   (d) Poset and Kripke semantics
   (e) Further topics:
      i. Topological models, spaces and local homeomorphisms.
      ii. H-Sets, sheaves, realizability
      iii. Domains
      iv. NNOs
      v. The untyped lambda-calculus
      vi. Modalities and monads
      vii. Monoidal closed categories and linear type theory
      viii. Normalization
3. Dependent Type Theory (weeks 6-9)
   (a) Dependent types
      i. Sigma, Pi, and Equality types
      ii. Beck-Chavalley
      iii. Hyperdoctrines
   (b) Locally cartesian closed categories
      i. The slice lemma
      ii. H-sets
      iii. Presheaves
      iv. Local homeomorphisms
   (c) Completeness in LCCCs
      i. Kripke semantics

   ii. Topological semantics
   iii. Kripke-Joyal forcing
   (d) W-types
      i. Polynomial endofunctors
      ii. Initial algebras
   (e) Coherence:
      i. CwFs, natural models
      ii. Universes and the Beck-Chevalley
   (f) Further topics:
      i. Equilogical spaces
      ii. Final coalgebras and coinduction
      iii. Impredicativity, realizability models
      iv. CwA's, comprehension categories
      v. Setoids and quotient types
      vi. Normalization and decidability of equality

4. Homotopy Type Theory (weeks 10-13)
   (a) Identity types
      i. UIP
      ii. function extensionality
   (b) Fibrations
   (c) Hofmann-Streicher universes
   (d) Univalence
   (e) The H-levels: Prop, Set, Gpd, ...
   (f) Further topics:
      i. The groupoid model
      ii. Algebraic weak factorization systems
      iii. Homotopy-initial algebras
      iv. Synthetic homotopy theory: $\pi_1(S^1)$

5. Student Presentations (week 14)
   (a) 2 talks / class meeting = 4 presentations

Chapter 2

Simple Type Theory

2.1 The λ-calculus

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

$$fx = t\ .$$

Here we gave the name f to the newly formed function, which takes an argument x to the value t. But we could have expressed the same function without giving it a name; this is sometimes written as

$$x \mapsto t\ ,$$

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

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

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

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

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

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

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

$$t : T$$

and it is one of the primitive notions of type theory (meaning that it is not defined). To computer scientists, the idea of expressions having types is familiar from programming languages; whereas mathematicians can think of types as sets and read $t : A$ as $t \in A$ (at least to get started).

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

$$\begin{aligned} \text{Basic types} \quad & \mathtt{B} ::= \mathtt{B}_0 \mid \mathtt{B}_1 \mid \mathtt{B}_2 \cdots \\ \text{Simple types} \quad & A ::= \mathtt{B} ::= \mathtt{1} \mid A_1 \times A_2 \mid A_1 \to A_2. \end{aligned}$$

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

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

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

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

In words, this means:

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

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

6. if s and t are terms then $s\, t$ is a term, called an *application*,

7. if x is a variable, A is a type, and t is a term, then $\lambda x : A\,.\, t$ is a term, called a *λ-abstraction*.

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

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

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

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

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

A *typing judgment* is a judgment of the form

$$\Gamma \mid t : A$$

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

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

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

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

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

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

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

- The typing rules for pairs and projections are:

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

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

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

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

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

the judgment that they are equal is written as

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

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

- Equality is an equivalence relation:

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

- The substitution rule:

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

- The weakening rule:

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

- Unit type:

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

- Equations for product types:

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

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

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

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

- Equations for function types:

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

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

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

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

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

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

Simply-typed λ-theories. Apart from the above rules for equality, which are part of the λ-calculus, we might want to impose additional equations between terms. In this case we speak of a λ-*theory*. Thus, a λ-theory $\mathbb{T}$ is given by a set of basic types and a set of basic constants, called the *signature*, and a set of *equations* of the form

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

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

We summarize the preceding definitions.

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

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

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

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

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

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

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

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

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

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

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

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

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

There are no equations.

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

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

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

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

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

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

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

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

The equational axioms of PCF are:

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

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

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

satisfying the equation

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

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

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

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

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

Untyped λ-calculus

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

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

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

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

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

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

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

The rules of equality are as follows:

1. Equality is an equivalence relation:

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

2. The weakening rule:

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

3. Equations for application and λ-abstraction:

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

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

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

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

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

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

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

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

is translated to the judgment

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

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

2.2 Cartesian closed categories

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

Exponentials

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

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

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

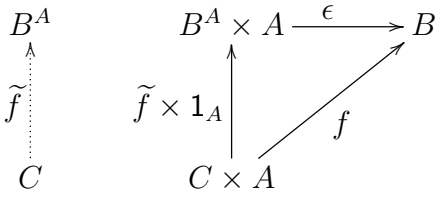
Similarly, given monoids $K, M \in \mathsf{Mon}$, there is a natural monoid structure on the set $\mathsf{Hom}(K, M)$, defined pointwise by

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

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

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

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



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

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

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

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

In Pos 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, which is easily seen to be monotone. 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*, 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 $\epsilon : B^A \times A \to B$). We say that $\mathcal{C}$ *has exponentials* if every object is exponentiable. A *cartesian closed category (ccc)* is a category that has all finite products and exponentials.

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

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

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

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

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

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

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

Two characterizations of CCCs

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

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

has a right adjoint

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

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

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

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

as indicated below.

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

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

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

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 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:*

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

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.

□

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 a 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 $\mathsf{1} \in \mathcal{C}$ 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 $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$, 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 $f : C \to A$, $g : C \to B$, $h : C \to A \times B$,*
$$\pi_1 \circ \langle f, g \rangle = f \ , \qquad \pi_2 \circ \langle f, g \rangle = g \ , \qquad \langle \pi_1 \circ h, \pi_2 \circ h \rangle = h \ .$$
3. *For all $f : C \times A \to B$, $g : C \to B^A$,*
$$\epsilon \circ (\widetilde{f} \times \mathsf{1}_A) = f \ , \qquad (\epsilon \circ (g \times \mathsf{1}_A))^{\sim} = g \ .$$

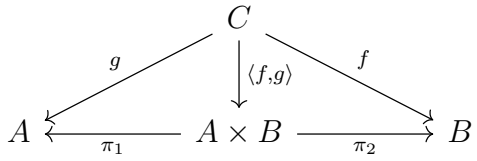
where for $e : E \to E'$ and $f : F \to F'$ we define

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

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

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

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



Suppose $h : C \to A \times B$ is another morphism such that $f = \pi_1 \circ h$ 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 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 $\Pi_{i\in I}\,\mathcal{C}_i$ 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

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,

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

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

Example 2.2.12. The category Cat of all small categories is cartesian closed. The exponential of small categories $\mathcal{C}$ and $\mathcal{D}$ is the category $\mathcal{D}^{\mathcal{C}}$ of functors, with natural transformations as arrows (see **??**). Note that if $\mathcal{D}$ is a groupoid (all arrows are isos), then so is $\mathcal{D}^{\mathcal{C}}$. It follows that the category of groupoids is full (even as a 2-category) in Cat. Since limits of groupoids in Cat are also groupoids, the inclusion of the full subcategory $\mathsf{Grpd} \hookrightarrow \mathsf{Cat}$ preserves limits. It also preserves the CCC structure.

Example 2.2.13. The same reasoning as in the previous example shows that the full subcategory $\mathsf{Pos} \hookrightarrow \mathsf{Cat}$ of all small posets and monotone maps is also cartesian closed, and the (limit preserving) inclusion $\mathsf{Pos} \hookrightarrow \mathsf{Cat}$ also preserves exponentials. Note that the (non-full) forgetful functor $U : \mathsf{Pos} \to \mathsf{Set}$ does not, and that $U(Q^P) \subseteq (UQ)^{UP}$ is in general a *proper* subset.

Exercise 2.2.14. Show that there is a full and faithful functor $D : \mathsf{Set} \to \mathsf{Poset}$ that preserves finite limits as well as exponentials. Note the similarity to the example $\mathsf{Grpd} \hookrightarrow \mathsf{Cat}$.

The foregoing examples are instances of the following general situation.

Proposition 2.2.15. *Let $\mathcal{E}$ be a CCC and $i : \mathcal{S} \hookrightarrow \mathcal{E}$ a full subcategory with finite products and a left adjoint reflection $L : \mathcal{E} \to \mathcal{S}$ preserving finite products. Suppose moreover that for any two objects A, B in $\mathcal{S}$, the exponential iB^{iA} is again in $\mathcal{S}$. Then $\mathcal{S}$ has all exponentials, and these are preserved by i.*

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

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

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

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

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

Proposition 2.2.16. *Let $\mathcal{E}$ be a CCC and $i : \mathcal{S} \hookrightarrow \mathcal{E}$ a full subcategory with finite products and a right adjoint reflection $R : \mathcal{E} \to \mathcal{S}$. 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 the inverse image of the one associated to the map $|X| \to X$ of locales (or in this case, spaces), from the discrete space on the set of points of X.

Exercise 2.2.17. Which of the foregoing examples follows from which of the previous two propositions?

Example 2.2.18. For any set X, the slice category $\mathsf{Set}/_X$ is cartesian closed. The product of $f : A \to X$ and $g : B \to X$ is the pullback $A \times_X B \to X$, which can be constructed as the set of pairs

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

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

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

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

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

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

Exercise 2.2.19. Prove that $\mathsf{Set}/_X$ is always cartesian closed. (Hint: Use the fact that $\mathsf{Set}/_X \simeq \mathsf{Set}^X$, 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 catesian 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 Pos to show that $\mathsf{Pos}/_X$ is not always cartesian closed.

Example 2.2.21. 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 can use the Yoneda lemma. If Q^P exists, then by Yoneda and the adjunction $(- \times P) \dashv (-^P)$, we would have, for all $c \in \mathbb{C}$,

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

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

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

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

$$E_c : \mathsf{Nat}(\mathsf{y}c \times P, Q) \times Pc \to Qc \ ,$$
$$E_c : \langle \eta, x \rangle \mapsto \eta_c \langle \mathsf{1}_c, x \rangle \ .$$

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

$$(\widetilde{\phi}_c z)_b : \mathcal{C}(b, c) \times Pb \to Qb \ ,$$
$$(\widetilde{\phi}_c z)_b : \langle f, y \rangle \mapsto \phi_b \langle (Rf)z, y \rangle \ .$$

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

Exercise 2.2.23. This exercise is for those 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.

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* $f : Y \to X$ and commutative triangles over X between such maps. It is equivalent to the category $\mathsf{Sh}(X)$ 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 Equ of equilogical spaces (Section 2.8).

2.3 Interpretation of the λ-calculus in a CCC

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

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

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

  (For this purpose, we assume that a CCC structure on $\mathcal{C}$ has been chosen.)

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

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

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

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

  and the empty context is interpreted as the terminal object,

$$[\![\cdot]\!] = \mathtt{1}\,.$$

- A typing judgment

$$\Gamma \mid t : A$$

  will be interpreted as a morphism

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

  The interpretation is defined inductively by the following rules:

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

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

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

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

- The interpretation of projections and pairs is as follows:

$$\begin{aligned}
[\![\Gamma \mid \langle t, u\rangle : A \times B]\!] &= \langle [\![\Gamma \mid t : A]\!], [\![\Gamma \mid u : B]\!]\rangle : [\![\Gamma]\!] \to [\![A]\!] \times [\![B]\!] \\
[\![\Gamma \mid \mathtt{fst}\, t : A]\!] &= \pi_1 \circ [\![\Gamma \mid t : A \times B]\!] : [\![\Gamma]\!] \to [\![A]\!] \\
[\![\Gamma \mid \mathtt{snd}\, t : A]\!] &= \pi_2 \circ [\![\Gamma \mid t : A \times B]\!] : [\![\Gamma]\!] \to [\![B]\!]\,.
\end{aligned}$$

- The interpretation of application and λ-abstraction is as follows:

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

  where $\epsilon : \llbracket A \to B \rrbracket \times \llbracket A \rrbracket \to \llbracket B \rrbracket$ is the evaluation morphism for $\llbracket B \rrbracket^{\llbracket A \rrbracket}$ and

$$(\llbracket \Gamma, x : A \mid t : B \rrbracket)^{\sim}$$

  is the transpose of the morphism

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

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

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

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 λ-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 $\llbracket - \rrbracket$,

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

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 λ-calculus, related to the Curry-Howard correspondence of section 1.3, relating λ-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 Γ (consider the case where $\Gamma = x : A, \dots$). We say that a type A is *inhabited* (by a closed term) when there is some $\vdash a : A$, and regard an inhabited type A as one that is *provable*. There is then a different notion of soundness related to this notion of provability.

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

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

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

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

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

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

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

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

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

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 $[\![A \to B]\!] = [\![B]\!]^{[\![A]\!]}$. 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 [Sco80b]).

3. A (positive) propositional theory $\mathbb{T}$ may be regarded as a λ-theory, and a model in a cartesian closed poset P is then the same thing as before: an interpretation of the atomic propositions $p_1, p_2, \ldots$ of $\mathbb{T}$ as elements $[\![p_1]\!], [\![p_2]\!], \ldots \in P$, such that the axioms $\phi_1, \phi_2, \ldots$ of $\mathbb{T}$ are all sent to $1 \in P$ by the extension of $[\![-]\!]$ to all formulas,

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

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

2.4 Functorial semantics

In Chapter **??** we saw how 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 λ-theories and their models in CCCs. The first step is to build the *classifying category* $\mathcal{C}_{\mathbb{T}}$ from a λ-theory $\mathbb{T}$. This is again constructed from the theory itself as a "syntactic" category, as follows:

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

- The objects of $\mathcal{C}_{\mathbb{T}}$ are the types of $\mathbb{T}$.
- Arrows $A \to B$ are terms in context (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

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

  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 "α-renaming" of variables).

Proposition 2.4.2. *The syntactic category $\mathcal{C}_{\mathbb{T}}$ built from a λ-theory is cartesian closed.*

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

- The terminal object is the unit type $\mathtt{1}$. For any type A the unique morphism $!_A : A \to \mathtt{1}$ is the term

  $$x : A \mid * : \mathtt{1} \,.$$

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

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

  is an axiom for the terms of unit type $\mathtt{1}$.

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

$$z : A \times B \mid \mathtt{fst}\, z : A\ , \qquad\qquad z : A \times B \mid \mathtt{snd}\, z : B\ .$$

  Given morphisms

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

  the term

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

  represents the unique morphism satisfying

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

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

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

  as required.

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

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

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

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

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

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

  Indeed, we have:

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

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

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

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

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

  Therefore,

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

  as claimed.

□

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

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

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

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

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

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

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

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

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

Proof. The only thing remaining to show is that, given a model $[\![-]\!]$ in a CCC $\mathcal{C}$ and a CCC functor $f : \mathcal{C} \to \mathcal{D}$, there is an induced model $[\![-]\!]^f$ in $\mathcal{D}$, given by the interpretation $[\![A]\!]^f = f[\![A]\!]$. This is 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 $A \to B$ prevents us from specifying a non-iso homomorphism of models $h : M \to N$ by the obvious recursion on the type structure.

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

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

when all we have are the following induced maps:

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

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

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

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

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

We shall consider other solutions to the problem of contravariance below.

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

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

For the record, we therefore have now shown:

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

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

Of course, the syntactic model $[-]$ is the one associated under (2.7) to the identity functor $\mathcal{C}_\mathbb{T} \to \mathcal{C}_\mathbb{T}$, *i.e.* it is the *universal* one. It therefore satisfies an equation just in case the equation holds in *all* models, by the classifying property of $\mathcal{C}_\mathbb{T}$, and the preservation of satisfaction of equations by CCC functors (Proposition 2.3.2).

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

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

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

2.5 The internal language of a CCC

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

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

1. For each basic type A in $\mathbb{S}$ a type τA in $\mathbb{T}$. The translation is then extended to all types by the rules
$$\tau \mathsf{1} = \mathsf{1}\,, \qquad \tau(A \times B) = \tau A \times \tau B\,, \qquad \tau(A \to B) = \tau A \to \tau B\,.$$
2. For each basic constant c of type C in $\mathbb{S}$ a term τc of type τC in $\mathbb{T}$. The translation of terms is then extended to all terms by the rules
$$\begin{aligned} \tau(\mathtt{fst}\, t) &= \mathtt{fst}\,(\tau t)\,, & \tau(\mathtt{snd}\, t) &= \mathtt{snd}\,(\tau t)\,, \\ \tau\langle t, u\rangle &= \langle \tau t, \tau u\rangle\,, & \tau(\lambda x : A\,.\, t) &= \lambda x : \tau A\,.\, \tau t\,, \\ \tau(t\, u) &= (\tau t)(\tau u)\,, & \tau x &= x \quad (\text{if } x \text{ is a variable})\,. \end{aligned}$$

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

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

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

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

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

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

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

which is 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 $\iota_\mathbb{S} : \mathbb{S} \to \mathbb{S}$. We can enrich this to an isomorphism of *groupoids* by defining syntactic isomorphisms between translations in $\mathsf{Hom}_{\lambda\mathsf{Thr}}(\mathbb{S}, \mathbb{T})$ in a fairly obvious way so that they correspond bijectively to $\mathbb{S}$-model homomorphisms in $\mathsf{Mod}_\lambda(\mathbb{S}, \mathcal{C}_\mathbb{T})$, which in turn correspond to natural isomorphisms between CCC functors in $\mathsf{Hom}_{\mathsf{CCC}}(\mathcal{C}_\mathbb{S}, \mathcal{C}_\mathbb{T})$, by Theorem 2.4.4,

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

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

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

between λ-theories and cartesian closed categories.

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

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

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

4. For all morphisms $f : A \to B$, $g : B \to C$, and $h : A \to C$ such that $h = g \circ f$, there is an axiom
$$x : \ulcorner A \urcorner \mid \ulcorner h \urcorner x = \ulcorner g \urcorner (\ulcorner f \urcorner x) : \ulcorner C \urcorner \, .$$

5. There is a constant
$$\mathtt{T} : 1 \to \ulcorner \mathtt{1} \urcorner \, ,$$
and for all $A, B \in \mathcal{C}$ there are constants
$$\mathtt{P}_{A,B} : \ulcorner A \urcorner \times \ulcorner B \urcorner \to \ulcorner A \times B \urcorner \, , \qquad \mathtt{E}_{A,B} : (\ulcorner A \urcorner \to \ulcorner B \urcorner) \to \ulcorner B^A \urcorner \, .$$
They satisfy the following axioms:
$$\begin{aligned}
u : \ulcorner \mathtt{1} \urcorner \mid \mathtt{T} * &= u : \ulcorner \mathtt{1} \urcorner \\
z : \ulcorner A \times B \urcorner \mid \mathtt{P}_{A,B} \langle \ulcorner \pi_1 \urcorner z, \ulcorner \pi_2 \urcorner z \rangle &= z : \ulcorner A \times B \urcorner \\
w : \ulcorner A \urcorner \times \ulcorner B \urcorner \mid \langle \ulcorner \pi_1 \urcorner (\mathtt{P}_{A,B} w), \ulcorner \pi_2 \urcorner (\mathtt{P}_{A,B} w) \rangle &= w : \ulcorner A \urcorner \times \ulcorner B \urcorner \\
f : \ulcorner B^A \urcorner \mid \mathtt{E}_{A,B} (\lambda x : \ulcorner A \urcorner \, . \, (\ulcorner \mathsf{ev}_{A,B} \urcorner (\mathtt{P}_{A,B} \langle f, x \rangle))) &= f : \ulcorner B^A \urcorner \\
f : \ulcorner A \urcorner \to \ulcorner B \urcorner \mid \lambda x : \ulcorner A \urcorner \, . \, (\ulcorner \mathsf{ev}_{A,B} \urcorner (\mathtt{P}_{A,B} \langle (\mathtt{E}_{A,B} f), x \rangle)) &= f : \ulcorner A \urcorner \to \ulcorner B \urcorner
\end{aligned}$$

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

Furthermore, an *equivalence of theories* $\mathbb{S}$ and $\mathbb{T}$ is a pair of translations
$$\mathbb{S} \underset{\sigma}{\overset{\tau}{\rightleftarrows}} \mathbb{T}$$
such that, for any type A in $\mathbb{S}$ and any type B in $\mathbb{T}$,
$$\sigma(\tau A) \cong A \, , \qquad\qquad \tau(\sigma B) \cong B \, .$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

□

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

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

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

This is mediated by an adjunction,

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

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

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

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

Lawvere's fixed point theorem

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

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

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

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

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

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

Among the consequences of this theorem, as stated in [Law69], are: Cantor's theorem (Corollary 1.2); Gödel's incompleteness theorem (Theorem 3.3); and Tarski's indefinability of truth (Theorem 3.2). These are all derived from the contrapositive form 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 $A \to B^A$. 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 $\neg : 2 \to 2$ 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 λ-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 ϕ in the variables $p_1, p_2, ..., p_n$ obviously determines a type Φ in the corresponding λ-theory $\mathbb{T}(X_1, X_2, ..., X_n)$ over n basic type symbols. What is the relationship between provability in positive propositional logic, $\mathsf{PPL} \vdash \phi$, and inhabitation in the associated λ-theory, $\mathbb{T}(X_1, X_2, ..., X_n) \vdash t : \Phi$? Let us call this the question of *conservativity* of λ-calculus over PPL. According to the basic idea of the Curry-Howard correspondence from Section 1.3, the λ-calculus is essentially the "proof theory of PPL". So one should expect that starting from an inhabited type Φ, a derivation of a term $\mathbb{T}(X_1, X_2, ..., X_n) \vdash t : \Phi$ should result in a corresponding proof of ϕ 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 $\mathbb{T}(X_1, X_2, ..., X_n) \vdash t : \Phi$ in the λ-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 $i : \mathsf{Pos} \hookrightarrow \mathsf{Cat}$, and let $\eta : \mathcal{C} \to |\mathcal{C}|$ be the unit of the adjunction.

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

Proof. Exercise! □

Exercise 2.6.2. Prove Lemma 2.6.1.

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

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

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

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

taking generators to generators, and it extends along the quotient map to $|\mathcal{C}_{\mathbb{T}(X_1,X_2,...,X_n)}|$ 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 $|\mathcal{C}_{\mathbb{T}(X_1,X_2,...,X_n)}|$. 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 $\bot$ 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 λ-theories $\mathbb{T}$, if $1 \to [\![A]\!]^M$ in all $\mathbb{T}$-models M in all $\mathcal{X}$, then $\mathbb{T} \vdash a : A$,
and if $[\![a]\!]^M = [\![b]\!]^M : 1 \to [\![A]\!]$ 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 λ-calculus we mean one in a CCC of the form $\widehat{\mathbb{C}} = \mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}$, 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 Δ may be seen both as a simplicial group—a simplicial object in the category of groups—and as a group object in the category $\mathsf{Set}^{\Delta^{\mathsf{op}}}$ 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*

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

preserves the cartesian closed structure.

This is of course the "categorified" analogue of Lemma 1.5.6, which we used for the Kripke completeness of the positive propositional calculus PPC.

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

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

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

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

So, continuing where we left off, $\widehat{\mathbb{C}}(\mathsf{y}X \times \mathsf{y}A, \mathsf{y}B) = \mathsf{y}B^{\mathsf{y}A}(X)$, and we're done. □

For an early version of the following theorem (and much more), see the nice paper [Sco80b] by Dana Scott.

Theorem 2.6.6. *For any λ-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 $[\![-]\!]$ in presheaves $\widehat{\mathbb{C}}$, there is a point

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

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

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

if, and only if,

$$[\![\Gamma \vdash s : A]\!] = [\![\Gamma \vdash t : A]\!] : [\![\Gamma]\!] \longrightarrow [\![A]\!]$$

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). □

Exercise 2.6.7. Show that not every presheaf topos $\mathsf{Set}^{\mathbb{C}^{\mathrm{op}}}$ admits a CCC embedding into a category of the form Set/X for a set X (you may assume the fact that the theory of a reflexive type (Example 2.1.7), is *not* trivial).

2.7 Kripke models

By a Kripke model of (a theory $\mathbb{T}$ in) the λ-calculus, we mean a model $[\![-]\!]$ in the sense of Definition 2.3.1 in a presheaf CCC of the form 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 Kirpke 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 $K \to \mathsf{Set}$, to model such variable sets, as is more customary in Kripke semantics. By Theorem 2.4.4, such a model $(K, [\![-]\!])$ is essentially the same thing as a CCC functor $M : \mathcal{C}_{\mathbb{T}} \to \mathsf{Set}^K$, taking values in "variable sets". Regarding the λ-calculus as the proof theory of the propositional calculus via the Curry-Howard correspondence (Section 1.3), it is perhaps not surprising that it should be (inhabitation) *complete* with respect to such Kripke models, in light of Theorem 1.5.5. 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,

$$\mathbb{T} \;=\; \big(\mathtt{B},\ \mathtt{u} : \mathtt{B}, * : \mathtt{B} \times \mathtt{B} \to \mathtt{B},\ \mathtt{x} * \mathtt{y} = \mathtt{y} * \mathtt{x}\big)\,.$$

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

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

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

satisfying the "compatibility conditions":

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

This is of course exactly a functor $M : K \to \mathsf{Set}$, as the interpretation $M = [\![\mathtt{B}]\!]$ of the basic type symbol $\mathtt{B}$. 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 $m_{j,k} : M_j \to M_k$ are of course single-valued.

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

$$u_k : M_k\,,$$

and these should satisfy

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

That is to say, we have an element or "point" $u : 1 \to M$ of M as a "variable set" in Set^K.

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

$$s_k : M_k \times M_k \to M_k$$

satisfying

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

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

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

In other word, it doesn't matter "when" one takes the $*$-product.

Finally, the interpretation $(M, u, s) = [\![\mathtt{B}, \mathtt{u}, *]\!]$ should satisfy the equation $x, y : \mathtt{B} \,|\, x * y = y * x : \mathtt{B}$, meaning that

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

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 Set^K, which is of course the same thing as a functor from K to the usual category of $\mathbb{T}$-models in Set,

$$\mathsf{Mod}_{\mathbb{T}}(\mathsf{Set}^K) \;=\; \mathsf{Mod}_{\mathbb{T}}(\mathsf{Set})^K\,.$$

Example 2.7.2 (Higher-order theories)**.** A theory involving a "higher-order" operation, such as the section $s : (D \to D) \to D$ 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 $D = [\![\mathtt{D}]\!]$ be the interpretation of the basic type $\mathtt{D}$, so that $[\![\mathtt{D} \to \mathtt{D}]\!] = D^D : K \to \mathsf{Set}$ is a presheaf exponential. At each $k \in K$, we then have,

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

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 $\mathsf{Set}^K\big(D \times K(k,j), D\big) = 1$ except when $j \in \uparrow k$. On $\uparrow k$, it consists of natural transformations

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

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

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

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

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

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

$$\vartheta{\uparrow}j : D{\uparrow}j \longrightarrow D{\uparrow}j\,.$$

This is just the same function as ϑ, but with the restricted domain of definition $\uparrow j \subseteq \uparrow k$. Note that the effect of this restriction may be to identify elements ϑ, 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 $s : (D \to D) \to D$ therefore takes, at each $k \in K$, such a $\vartheta : D{\uparrow}k \longrightarrow D{\uparrow}k$ 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 $d_{k,j} : D_k \to D_j$ is the action of the functor $D : K \to \mathsf{Set}$.

In this way, the presheaf exponential $D^D : K \to \mathsf{Set}$ is entirely determined by the "base-case" $D : K \to \mathsf{Set}$, 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.* [AGH24] 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 $\mathsf{Sh}(L)$ and a connected, locally connected geometric morphism $c : \mathsf{Sh}(L) \to \mathcal{E}$.*

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

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

(i) A type A is inhabited just if it has a point $1 \to [\![A]\!]$ in every Kripke model $(K, [\![-]\!])$.

(ii) Two terms are provably equal, $\mathbb{T} \vdash (\Gamma \mid s = t : A)$, just if they are equal in every Kripke model $(K, [\![-]\!])$,

$$[\![s]\!] = [\![t]\!] : [\![\Gamma]\!] \longrightarrow [\![A]\!] .$$

In the following chapter, we shall see that this result holds as well for *dependent* type theory with the Σ, Π, and Eq type-formers. For the proof, see [AR11], as well as [AGH24].

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 $\varphi : \mathcal{C} \hookrightarrow \mathsf{Set}^K$. First, compose the Yoneda embedding $\mathsf{y} : \mathcal{C} \hookrightarrow \widehat{\mathcal{C}}$ with the inverse image $c^* : \widehat{\mathcal{C}} \hookrightarrow \mathsf{Sh}(L)$ of the Joyal-Tierney cover, which is also fully faithful and CCC. Then compose further with the inclusion $i_* : \mathsf{Sh}(L) \hookrightarrow \mathsf{Set}^{L^{\mathsf{op}}}$ of sheaves into presheaves, which is also fully faithful and CCC. So we can take $K = L^{\mathsf{op}}$ to get the desired CCC embedding $\varphi = i_* \circ c^* \circ \mathsf{y} : \mathcal{C} \hookrightarrow \mathsf{Set}^K$. See [Awo00, AR11] for more details.

2.8 Topological models

From presheaves to posets to spaces to sheaves.

Posets

Since the category Pos is cartesian closed, we can take models of λ-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 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 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 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 $\pi : P \to K$ 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 $\mathbb{2} = (0 \leq 1)$.

$$\begin{array}{ccc} 1 & \longrightarrow & P \\ {\scriptstyle 0}\downarrow & \nearrow & \downarrow{\scriptstyle \pi} \\ \mathbb{2} & \longrightarrow & K \end{array} \tag{2.9}$$

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

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

is equivalent to Set^K*. In particular, this category is therefore cartesian closed.*

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

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

with the *category of elements* $\pi : \int_K P \to K$ on the left, and indeed, every discrete opfibration $p : D \to K$ arises in this way from an essentially unique $P : K \to \mathsf{Set}$, namely the one with $P(k) = p^{-1}(k)$. Moreover, every pullback of a discrete fibration (such as u is a discrete fibration, as is easily seen by considering lifting (2.9).

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

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

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

Corollary 2.8.5. *The categories* $\mathsf{dopFib}/_K$ *are sufficient for* λ*-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 Set is *not* sufficient for arbitrary λ-theories, with basic types, terms and equations. Is 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 $\mathsf{Pos}/_P$ 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 $\mathsf{dopFib}/_K \simeq \mathsf{Set}^K$ always are so. Something similar happens with topological spaces: the category Top of all spaces and continuous maps is itself not even cartesian closed (unlike Pos), nor are its slices, but there is a (wide but not full) subcategory $\mathsf{LocHom} \hookrightarrow \mathsf{Top}$ the *slices* of which are always cartesian closed, even though the total category LocHom itself is not. As in the case of dopFib, this is most easily seen by showing that each slice $\mathsf{LocHom}/_X$ is actually equivalent to a functor category known to be cartesian closed, namely the category $\mathsf{Sh}(X)$ of *sheaves* on the space X.

A little topos theory: sheaves and local homeomorphisms.

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

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

is an equalizer.

In words, given a family of elements $f_i : \mathsf{y}U_i \to F$ that "match" on the overlaps,

$$f_i|_{U_j} = f_j|_{U_i} : \mathsf{y}(U_i \cap U_j) \to F,$$

there is a unique "amalgamation" $f : \mathsf{y}U \to F$ that restricts to the given family on the cover, $f|_{U_i} = f_i : \mathsf{y}U_i \to F$. For example, in the case of two open sets U, V, this condition says exactly that the following *pushout* diagram in $\mathcal{O}(X)$

$$\begin{array}{ccc} U \cap V & \longrightarrow & V \\ \downarrow & & \downarrow \\ U & \longrightarrow & U \cup V \end{array} \tag{2.10}$$

is taken by F to a *pullback* in Set:

$$\begin{array}{ccc} F(U \cap V) & \longleftarrow & FV \\ \uparrow & & \uparrow \\ FU & \longleftarrow & F(U \cup V) \end{array} \tag{2.11}$$

A basic example of a sheaf is the presheaf $\mathsf{Top}(-, Z) : \mathcal{O}(X)^{\mathsf{op}} \to \mathsf{Set}$ of continuous functions into a fixed space Z, where the open sets $U \subseteq X$ are regarded as subspaces and therefore objects in Top. Indeed, given a family of continuous functions $f_i : U_i \to Z$ that match on the intersections $U_i \cap U_j$, we can define an amalgamation $f : U \to Z$ by setting $f(x) = f_i(x)$ for *some* i with $x \in U_i$ (which exists since $U = \bigcup_{i \in I} U_i$) and the specification will be unique by the matching condition.

It is not difficult to prove the following fact directly from the elementary definition 2.8.7.

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

Exercise 2.8.9. Prove this by showing that Z^Y is a sheaf as soon as Z is a sheaf, by analyzing the exponential as $Z^Y(U) \cong \mathsf{Hom}(\mathsf{y}U \times Y, Z)$.

There is an equivalent perspective on sheaves over X that is often useful, namely as certain spaces over X via certain "generalized covering spaces" $p : Y \to X$ called local homeomorphisms.

Definition 2.8.10. A continuous function $p : Y \to X$ is a *local homeomorphism* if for every $y \in Y$ there is an open set $y \in U \subseteq Y$ such that (i) the image $p(U) \subseteq X$ is open, and (ii) the restriction $p|_U : U \to p(U)$ is a homeomorphism.

Let $\mathsf{LocHom} \hookrightarrow \mathsf{Top}$ be the subcategory of spaces and local homeomorphisms, and $\mathsf{LocHom}/_X$ the slice category. Thus a map $f : (Y, p) \to (Z, q)$ of local homeomorphisms over X is a commutative triangle in Top with $p : Y \to X$ and $q : Z \to X$ local homeomorphisms.

$$\begin{array}{ccc} Y & \xrightarrow{\quad f \quad} & Z \\ & {\scriptstyle p} \searrow \quad \swarrow {\scriptstyle q} & \\ & X & \end{array}$$

One can show that in fact every merely *continuous* map $f : Y \to Z$ making a commutative triangle $q \circ f = p$ is also a local homeomorphism (exercise!), so that this definition is indeed the slice category $\mathsf{LocHom}/_X$, the inclusion of which is *full* in $\mathsf{Top}/_X$.

Given any space "over X" via a continuous map $p_Y : Y \to X$, we can define the *presheaf of local sections* $\Gamma(Y)$ by

$$\Gamma(Y)(U) = \mathsf{Top}/_X(U, Y) \qquad \text{for } U \hookrightarrow X\,.$$

That is, we regard the open set $U \subseteq X$ as a space over X via its inclusion $U \hookrightarrow X$, and consider all "local sections of Y over U", i.e. continuous maps over X from $U \hookrightarrow X$ to $p_Y : Y \to X$. The presheaf $\Gamma(Y)$ is now easily seen to be a sheaf (much like the example $\mathsf{Top}(-, Z)$ above). In this way we have a functor

$$\Gamma : \mathsf{Top}/_X \longrightarrow \mathsf{Set}^{\mathcal{O}(X)^{\mathsf{op}}}$$

which in fact factors through the full subcategory of sheaves $\mathsf{Sh}(X) \hookrightarrow \mathsf{Set}^{\mathcal{O}(X)^{\mathsf{op}}}$. There is also a functor coming back

$$\Lambda : \mathsf{Set}^{\mathcal{O}(X)^{\mathsf{op}}} \longrightarrow \mathsf{Top}/_X,$$

(called the *bundle of germs*), which factors through the full subcategory of local homeomorphisms $\mathsf{LocHom}/_X \hookrightarrow \mathsf{Top}/_X$. The local homeomorphism $\Lambda(P) \to X$ has the total space

$$\Lambda(P) = \coprod_{x \in X} \mathsf{germ}_x(P),$$

where the "stalk of germs at x" $\mathsf{germ}_x(P)$ is defined by

$$\mathsf{germ}_x(P) = \operatorname*{colim}_{U \ni x} PU\,,$$

the colimit being taken over the filter of all open sets U with $x \in U$. The space $\Lambda(P)$ is topologized by declaring as basic opens the images of all partial sections $s : U \to \Lambda(P)$ over X (for $U \subseteq X$ open).

The situation is summarized in the following proposition, for a detailed proof of which, see [MM92, Ch. II].

Proposition 2.8.11. *The bundle of germs functor Λ, which takes a presheaf P on the space X to the local homeomorphism $\Lambda(P) \to X$, is left adjoint to the presheaf of sections functor Γ. The images of these functors are the full subcategories of sheaves $\mathsf{Sh}(X)$, for Γ, and local homeomorphisms $\mathsf{LocHom}/_X$, for Λ. The inclusions have adjoints: a left adjoint $\mathsf{a} \dashv i$ for $\mathsf{Sh}(X)$ and a right adjoint $j \dashv \mathsf{b}$ for $\mathsf{LocHom}/_X$.*

$$\begin{array}{ccc}
\mathsf{Set}^{\mathcal{O}(X)^{\mathsf{op}}} & \underset{\Lambda}{\overset{\Gamma}{\rightleftarrows}} & \mathsf{Top}/_X \\
\mathsf{a} \downarrow \quad \uparrow i & & j \uparrow \quad \downarrow \mathsf{b} \\
\mathsf{Sh}(X) & \underset{\Lambda}{\overset{\Gamma}{\rightleftarrows}} & \mathsf{LocHom}/_X
\end{array} \tag{2.12}$$

Thus there is an equivalence of categories $\mathsf{Sh}(X) \simeq \mathsf{LocHom}/_X$.

One immediate consequence is that every slice $\mathsf{LocHom}/_X$ is cartesian closed, by Proposition 2.8.8. Another application is an explicit description of the left adjoint *sheafification* functor $\mathsf{a} = \Gamma \circ \Lambda$, which is seen to preserve finite limits, since each of its factors does so. Another application that will be of use in the semantics of the λ-calculus with sums in the next section is the description of the coproduct of two local homeomorphisms $A \to X$ and $B \to X$ as $A + B \to X$, constructed in $\mathsf{Top}/_X$ in the obvious way.

Corollary 2.8.12. *The sheafified Yoneda embedding* $\mathsf{ay} : \mathcal{O}(X) \to \mathsf{Sh}(X)$ *is still full and faithful and still preserves all limits and exponentials. It now also preserves joins,*

$$\mathsf{ay}U \cup \mathsf{ay}V \cong \mathsf{ay}(U \cup V) \quad \textit{in } \mathsf{Sub}_{\mathsf{Sh}(X)}(1)\,,$$

and similarly for all joins $U = \bigcup_{i\in I} U_i$ *in* $\mathcal{O}(X)$*. Indeed, the factorization*

$$\mathsf{ay} : \mathcal{O}(X) \to \mathsf{Sub}_{\mathsf{Sh}(X)}(1)$$

is an isomorphism of complete Heyting algebras.

Proof. To give a sketch: chasing around the diagram (2.12), we can regard the sheafified Yoneda embedding ay as being given by $\Lambda \circ \mathsf{y} : \mathcal{O}(X) \to \mathsf{LocHom}/_X$, which is just the functor

$$(U \subseteq X) \longmapsto (U \hookrightarrow X)\,,$$

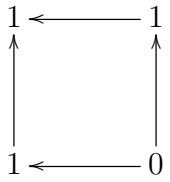
taking an open subset U of X to the inclusion of the open subspace $U \hookrightarrow X$, which is obviously a local homeomorphism. The join of a family $U_i \hookrightarrow X$ of subobjects of 1 in $\mathsf{LocHom}/_X$ is computed there by first taking the coproduct in $\mathsf{Top}/_X$ to give a local homoeomorphism $\coprod_i U_i \to X$ with a disjoint sum of spaces as its domain, and then the image factorization $\coprod_i U_i \twoheadrightarrow \bigcup_i U_i \rightarrowtail X$ to give the open set inclusion $\bigcup_i U_i \hookrightarrow X$, which is the inclusion of the join of the U_i in $\mathcal{O}(X)$. □

Exercise 2.8.13. Show that every representable functor $\mathsf{y}(U)$ is a sheaf. Conclude that the "sheafified" Yoneda embedding $\mathsf{a} \circ \mathsf{y} : \mathcal{O}X \to \mathsf{Sh}(X)$ is fully faithful and injective on objects.

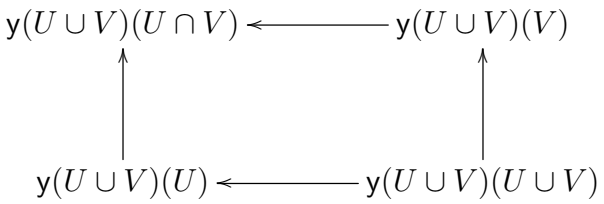
Let us consider a simple example of the preservation of joins by comparing the presheaf join $\mathsf{y}U \cup \mathsf{y}V$ with the sheaf $\mathsf{y}(U \cup V)$, in the case where neither $U \subseteq V$ nor $V \subseteq U$. We can evaluate the presheaf $\mathsf{y}U \cup \mathsf{y}V$ at the pushout diagram (2.10) to get the following diagram of sets,

$$\begin{array}{ccc} (\mathsf{y}U \cup \mathsf{y}V)(U \cap V) & \longleftarrow & (\mathsf{y}U \cup \mathsf{y}V)(V) \\ \uparrow & & \uparrow \\ (\mathsf{y}U \cup \mathsf{y}V)(U) & \longleftarrow & (\mathsf{y}U \cup \mathsf{y}V)(U \cup V) \end{array}$$

which evaluates to the following,



since $(U \cup V)$ is "in" neither $\mathsf{y}U$ nor $\mathsf{y}V$, and so not in their join $\mathsf{y}U \cup \mathsf{y}V$ (where being "in the presheaf" means that the presheaf is not empty at that argument). This diagram is clearly not a pullback, as required for $\mathsf{y}U \cup \mathsf{y}V$ to be a sheaf. On the other hand, evaluating $\mathsf{y}(U \cup V)$ instead results in the evident pullback:



which has $1 = \{*\}$ at all four corners.

Remark 2.8.14. An application of Corollary 2.8.12 is a completeness theorem for full intuitionistic *propositional* logic with respect to categories of sheaves $\mathsf{Sh}(X)$, using the sheafified Yoneda embedding in place of Joyal's representation theorem, which we used for completeness with respect to *pre*sheaves. We leave it to the reader to fill in the details.

Exercise 2.8.15. Fill in the details. (*Hint*: The downset embedding of a Heyting algebra H into the complete Heyting algebra of its ideals $\mathsf{Idl}(H)$ is an injective Heyting homomorphism, and there is an isomorphism of complete Heyting algebras $\mathsf{Idl}(H) \cong \mathcal{O}\mathsf{Spec}(H)$, where the space $\mathsf{Spec}(H)$ is the *prime spectrum* of H, a generalization of the Stone space of a Boolean algebra.)

As a further corollary of Proposition 2.8.11 we have a completeness theorem analogous to Corollary 2.8.5 for models of the λ-calculus in categories of the form $\mathsf{Sh}(X) \simeq \mathsf{LocHom}/_X$, by first deriving completeness for categories of sheaves $\mathsf{Sh}(X)$ from the same covering theorem 2.7.3 that was used for Kripke completeness of the λ-calculus; see Remark 2.7.5. (The completeness theorem with respect to *spaces* rather than posets is proved using a refinement of the Joyal-Tierney covering theorem 2.7.3 due to Moerdijk [**?**], also see [Awo00, **?**].) We will state this specialization where it is needed below for the semantics of λ-calculus with sums. While it is of interest to know that semantics in spaces and local homeomorphisms is sufficient for theories in the λ-calculus, it is also of practical use simply to know that the λ-calculus can be used as an *internal language* to reason about sheaves over a space—a setting with many applications in everyday mathematics.

Equilogical spaces

See [**?**].

2.9 Extensions of the λ-calculus

We conclude our study of simple type theory by considering a few extensions of the basic λ-calculus:

1. λ-Calculus with sums.
2. Natural numbers objects.
3. Higher-order logic.
4. Modal operators.

The first three of these are discussed in much more detail in the book [LS88]. These extensions are not λ-theories as previously defined, but instead involve further operations on types, which may be regarded as (proof-relevant versions of) general rules of inference. Their categorical semantics require additional structures on a CCC.

2.9.1 λ-Calculus with sums

We can extend the Curry-Howard correspondence between positive propositional calculi and CCCs by adding "sums" to the CCCs to obtain a categorified version of Heyting algebras, or intuitionistic propositional calculi, which we shall call *BiCartesian Closed Categories* (BiCCCs),

$$\frac{\mathsf{BiCCC}}{\mathsf{CCC}} = \frac{\mathsf{IPC}}{\mathsf{PPC}}.$$

The internal language of such categories will be a simple type theory given by adding "stable sums" 0 and $A + B$ to the λ-calculus. Following [LS88, ADHS01, FDCB02], the additional rules required are the following.

1. The *types* are extended by adding the type constructors $\mathtt{0}$ and $A + B$, so we now have:

   $$\text{Simple types} \quad A ::= \mathtt{B} \mid \mathtt{1} \mid A_1 \times A_2 \mid A_1 \to A_2 \mid \mathtt{0} \mid A_1 + A_2$$

   with the expected formation rules.

2. For the *terms* we now have

   $$\text{Terms} \quad t ::= v \mid c \mid * \mid \langle t_1, t_2 \rangle \mid \mathtt{fst}\, t \mid \mathtt{snd}\, t \mid t_1\, t_2 \mid \lambda x : A\,.\, t \mid !\, t \mid \mathtt{inl}\, t \mid \mathtt{inr}\, t \mid [x.t_1, x.t_2]u$$

   The copairs $[x.t_1, x.t_2]u$ are sometimes called "cases", and the variable x is bound. Their typing rules, and those for the injections $!\, t$, $\mathtt{inl}\, t$, and $\mathtt{inr}\, t$, are:

   $$\frac{\Gamma \mid u : \mathtt{0}}{\Gamma \mid !\, u : C} \qquad \frac{\Gamma \mid a : A}{\Gamma \mid \mathtt{inl}\, a : A + B} \qquad \frac{\Gamma \mid b : B}{\Gamma \mid \mathtt{inr}\, b : A + B}$$

   $$\frac{\Gamma, x : A \mid s : C \qquad \Gamma, y : B \mid t : C \qquad \Gamma \mid u : A + B}{\Gamma \mid [x.s, y.t]u : C}$$

3. The *equations* for these terms are as follows.

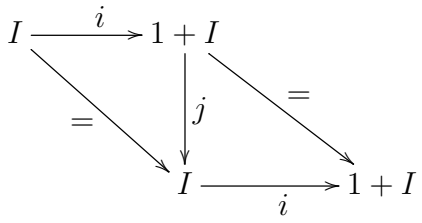
$$\overline{z : C \mid z = !\, u : C} \qquad \overline{[x.a, y.b](\mathtt{inl}\, s) = a[s/x] : C} \qquad \overline{[x.a, y.b](\mathtt{inr}\, t) = b[t/y] : C}$$

$$\overline{u = [x.\mathtt{inl}\, x, y.\mathtt{inr}\, y]u : A + B}$$

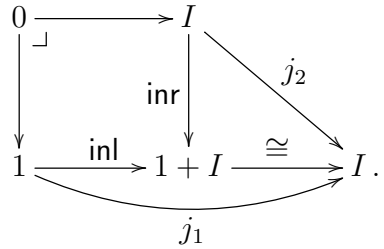
$$\overline{v\big([x.s, y.t]u\big) = [x.vs, y.vt]u : D}$$

   The last equation is a "distributivity" law, in which $v : C \to D$ and $x, y \notin \mathsf{FV}(v)$.

4. For an example of a *theory* in the λ-calculus with sums, consider the notion of an *infinite* object I. According to R. Dedekind [Ded88], an object is *infinite* if it admits an injective mapping to a proper subobject. This condition can be captured in a BiCCC by requiring an isomorphism $1 + I \cong I$. Specifically, we require maps and equations as follows:



   For then $j = [j \circ \mathsf{inl}, j \circ \mathsf{inr}] : 1 + I \to I$ for unique maps $j_1 = j \circ \mathsf{inl} : 1 \to I$ and $j_2 = j \circ \mathsf{inr} : I \to I$, whence $j_2 : I \to I$ is injective (as a composite of injections), and there is at least one element $j_1 : 1 \to I$ that is not in its image, by the disjointness of coproducts,



Exercise 2.9.1. Write down the theory of infinite objects in the λ-calculus with sums and prove that every model in a BiCCC is indeed Dedekind infinite. Also formulate the theory of a "successor algebra" as an object X equipped with a point $x : 1 \to X$ and an endomorphism $s : X \to X$. Prove that the natural numbers are initial among all successor algebras in Set (with the evident definition of algebraic homomorphisms). Show from this that the natural numbers are Dedekind infinite.

We now have the expected extension of the foregoing results for λ-calculi and CCCs to the case of λ-calculi with sums and BiCCCs, namely:

Proposition 2.9.2. *For any theory $\mathbb{T}$ in λ-calculi with sums, there is a (syntactic) BiCCC $\mathcal{B}_{\mathbb{T}}$ that classifies $\mathbb{T}$-models in arbitrary BiCCCs $\mathcal{B}$,*

$$\mathsf{BiCCC}(\mathcal{B}_{\mathbb{T}}, \mathbb{B}) \simeq \mathsf{Mod}(\mathbb{T}, \mathcal{B})\,.$$

In particular, the λ-calculus with sums is complete (in the sense of Corollary 2.4.7) with respect to models in bicartesian closed categories.

The proof is analogous to the previous case, although some care is required with the initial object, the disjointness of sums, and their stability under products with a fixed object (see [LS88, FDCB02]). There is also an internal language correspondence as in Section 2.5 that we need not spell out. An interesting question considered in [FDCB02] is that of *type isomorphisms*, as a generalization of elementary algebraic equations. For example ...

Remark 2.9.3 (Variable set completeness)**.** Completeness of λ-calculus with respect to arbitrary presheaf categories $\mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}$ is apparently more difficult to generalize to λ-calculus with sums than was the general categorical completeness theorem, Proposition 2.9.2. This is because, although such categories $\mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}$ have very well-behaved (indeed, freely added) coproducts, the Yoneda embedding does not preserve the coproducts that may exist in the index category $\mathbb{C}$. In the "proof-irrelevant" propositional case, we solved this problem using a more sophisticated embedding theorem due to Joyal, Theorem 1.5.22. One may conjecture that something similar could hold in the present case: we can embed a BiCCC $\mathcal{B}$ into the category of presheaves on all "biCartesian" functors $M : \mathcal{B} \to \mathsf{Set}$ (playing the role of the prime filters in a Heyting algebra); however, we would still need to show that this analogous "evaluation embedding" also preserves all exponentials; whether this holds is an open question.

We leave the question of completeness with respect to variable sets aside for now, and briefly consider a different approach to the semantics of BiCCCs using *sheaves*, for which one can show completeness using well-known results (e.g. [FS99]).

Definition 2.9.4. Let $\mathcal{B}$ be a BiCCC. A presheaf $F : \mathcal{B}^{\mathsf{op}} \to \mathsf{Set}$ is called a *sheaf* (for the $+$-topology) if it preserves finite products. Explicitly, a sheaf F is a contravariant functor that takes the finite coproducts in $\mathcal{B}$ to products in Set (via the canonical maps),

$$\begin{aligned} F(0) &\cong 1\,, \\ F(A+B) &\cong FA \times FB\,. \end{aligned}$$

A *morphism of sheaves* $f : G \to F$ is just a natural transformation.

Lemma 2.9.5. *By definition, the category of sheaves is a full subcategory. The inclusion $i : \mathsf{Sh}(\mathcal{B}) \hookrightarrow \widehat{\mathcal{B}}$ has a left adjoint,*

$$\mathsf{a} : \widehat{\mathcal{B}} \longrightarrow \mathsf{Sh}(\mathcal{B}),$$

called sheafification*, which, moreover, preserves all finite limits.*

Proof. Since finite products commute with limits, it is easy to see that the FP-functors are closed under all limits. So by the adjoint functor theorem, we see that the full subcategory of FP-functors is reflective in the category of all Set-valued functors, as we have already shown in Section **??**. That the reflector a preserves finite limits can be shown by analyzing the sheafification functor a in terms of the so-called Grothedieck +-construction; see [MM92, III.5]. □

Proposition 2.9.6. *We require the following facts about the subcategory category*

$$\mathsf{Sh}(\mathcal{B}) \hookrightarrow \widehat{\mathcal{B}}$$

of +-sheaves on a BiCCC $\mathcal{B}$.

1. *The representable functors $\mathsf{y}B : \mathcal{B}^{\mathsf{op}} \to \mathsf{Set}$ are all sheaves; in particular, the terminal object $1 = \mathsf{y}1$ is a sheaf.*

2. *The sheafified Yoneda embedding $\mathsf{ay} : \mathcal{B} \to \mathsf{Sh}(\mathcal{B})$ is fully faithful, and preserves finite coproducts.*

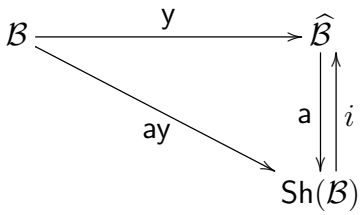
3. *If F, G are sheaves, so is their product $F \times G$, and if G is a sheaf and F a presheaf, then the presheaf exponential G^F is a sheaf.*

Thus in particular, the fully faithful functor $\mathsf{ay} : \mathcal{B} \hookrightarrow \mathsf{Sh}(\mathcal{B})$ preserves the BiCCC structure.

Proof. 1. For any object $B \in \mathcal{B}$ we have:

$$\begin{aligned} \mathsf{y}B(0) &= \hom(0, B) = 1\,, \\ \mathsf{y}B(A_1 + A_2) &= \hom(A_1 + A_2, B) \cong \hom(A_1, B) \times \hom(A_2, B) \\ &= \mathsf{y}B(A_1) \times \mathsf{y}B(A_2)\,. \end{aligned}$$

2. To see that $\mathsf{ay} : \mathcal{B} \to \mathsf{Sh}(\mathcal{B})$ is fully faithful, note that by (1) we have $\mathsf{y} \cong i \circ (\mathsf{ay}) : \mathcal{B} \to \widehat{\mathcal{B}}$, which is fully faithful, and $i : \mathsf{Sh}(\mathcal{B}) \to \widehat{\mathcal{B}}$ is so as well.



To see that ay preserves sums, for any sheaf F, we have:

$$\begin{aligned} \mathsf{Sh}(\mathcal{B})(\mathsf{ay}(0), F) &\cong \widehat{\mathcal{B}}(\mathsf{y}(0), iF) \cong iF(0) \cong 1\,, \\ \mathsf{Sh}(\mathcal{B})(\mathsf{ay}(A + B), F) &\cong \widehat{\mathcal{B}}(\mathsf{y}(A + B), iF) \cong iF(A + B) \cong iF(A) \times iF(B)\,. \end{aligned}$$

3. If F, G are sheaves then $F \times G$ is one as well, since, as presheaves,

$$\begin{aligned}(F \times G)(0) &\cong F0 \times G0 \cong 1 \times 1 \cong 1\,,\\ (F \times G)(A+B) &\cong F(A+B) \times G(A+B) \cong (FA \times FB) \times (GA \times GB)\\ &\cong (F \times G)(A) \times (F \times G)(B)\,.\end{aligned}$$

If G is a sheaf and F a presheaf, then as presheaves,

$$\begin{aligned}(iG^F)(0) &\cong \widehat{\mathcal{B}}\big(\mathsf{y}(0) \times F, iG\big) \cong \mathsf{Sh}(\mathcal{B})\big(\mathsf{a}(\mathsf{y}(0) \times F), G\big)\\ &\cong \mathsf{Sh}(\mathcal{B})\big(\mathsf{ay}(0) \times \mathsf{a}F, G\big) \cong \mathsf{Sh}(\mathcal{B})\big(\mathsf{a}0, G\big) \cong 1\,.\\ (iG^F)(A+B) &\cong \widehat{\mathcal{B}}\big(\mathsf{y}(A+B) \times F, iG\big) \cong \widehat{\mathcal{B}}\big(\mathsf{ay}(A+B) \times F, G\big)\\ &\cong \widehat{\mathcal{B}}\big((\mathsf{ay}(A) + \mathsf{ay}(B)) \times F, G\big)\\ &\cong \widehat{\mathcal{B}}\big((\mathsf{ay}(A) \times F) + (\mathsf{ay}(B) \times F), G\big)\\ &\cong \widehat{\mathcal{B}}\big(\mathsf{ay}(A) \times F, G\big) \times \widehat{\mathcal{B}}\big(\mathsf{ay}(B) \times F, G\big)\\ &\cong \widehat{\mathcal{B}}\big(\mathsf{y}(A) \times F, iG\big) \times \widehat{\mathcal{B}}\big(\mathsf{y}(B) \times F, iG\big)\\ &\cong iG^F(A) \times iG^F(B)\,.\end{aligned}$$

□

As a result of the forgoing embedding theorem, the completeness of the basic λ-calculus with respect to presheaves can be extended to completeness of the λ-calculus *with sums* with respect to categories of *sheaves* (although we have not yet defined these except in the special cases of topological spaces and $+$ sheaves on a category with stable, finite coproducts). To that end, we define a *model* in a category $\mathsf{Sh}(\mathbb{C}, +)$ of sheaves (for the $+$ topology) on a small category $\mathbb{C}$ with stable finite coproducts to be a BiCCC functor from $\mathcal{B}_{\mathbb{T}}$, the classifying BiCCC of a theory $\mathbb{T}$, into $\mathsf{Sh}(\mathbb{C}, +)$. We then have the following completeness theorem.

Proposition 2.9.7 (Completeness of λ-Calculus with Sums)**.** *For a theory $\mathbb{T}$ in the λ-calculus with sums,*

1. *a type A is inhabited in every model in a category of sheaves $\mathsf{Sh}(\mathbb{C}, +)$ iff there is a closed term $a : A$,*

2. *an equation $\Gamma \mid s = t : A$ holds in every model in a category of sheaves $\mathsf{Sh}(\mathbb{C}, +)$ iff there is a proof of it from the equations of $\mathbb{T}$.*

There is a more general notion of a sheaf for a "Grothendieck topology" on a small category $\mathbb{C}$, of which the $+$-topology on a category with sums is a special case, and the foregoing proposition then generalizes to that case, but we shall not pursue this further here.

Exercise 2.9.8. Fill in the details of the proof of Proposition 2.9.7.

Finally, we can again apply the Joyal-Tierney covering theorem 2.7.3 to obtain completeness of the λ-calculus *with sums* with respect to categories $\mathsf{Sh}(X)$ of sheaves on a *space*:

Corollary 2.9.9 (Topological Completeness of λ-Calculus with Sums)**.** *For a theory $\mathbb{T}$ in the λ-calculus with sums,*

1. *a type A is inhabited in every model in a category of sheaves on a space $\mathsf{Sh}(X)$ iff there is a closed term $a : A$,*

2. *an equation $\Gamma \mid s = t : A$ holds in every model in a category of sheaves on a space $\mathsf{Sh}(X)$ iff there is a proof of it from the equations of $\mathbb{T}$.*

Of course, the result can also be formulated equivalently in terms of BiCCCs of the form $\mathsf{LocHom}/_X$ rather than $\mathsf{Sh}(X)$, where the coproducts of local homeomorphisms $A \to X$ and $B \to X$ are more easily constructed naively as $A + B \to X$ in the underlying category Top/X (by Proposition 2.8.11).

Remark 2.9.10. It may also be asked whether there is a *single space* X such that $\mathsf{Sh}(X)$ is sufficient for all theories in the λ-calculus with sums, as has been shown e.g. for intuitionistic first-order logic IFOL with respect to just sheaves on the real line $\mathbb{R}$ [].

2.9.2 Natural numbers objects

Using sums we can describe infinite objects $X + 1 \cong X$, but we cannot describe the *free* such objects, such as the *initial* one, without having a general induction principle with respect to other such objects. Such inductive type are more conveniently formulated in dependent type theory, as we shall do in the next chapter, but we can also formulate them in simple type theory by adding new recursion operations, see Lambek-Scott [LS88]. . This leads to the important notion of a *natural numbers object*: an initial infinite object.

2.9.3 Higher-order logic

This example presumes familiarity with the results of Chapter **??**, or at least with the basic categorical approach to first-order logic as presented in [MM92, **?**].

The approach to IHOL presented here is closely tied to *topos theory*, which is to be treated in greater depth in Chapter **??**. Also see Lambek-Scott [LS88].

Remark 2.9.11.

2.9.4 Modalities

Recall first the propositional modal logics IS4, IS5 with adjoints, natural deduction using Bierman-dePaiva, Kavvos.

Example of a CCC with a "modal operator": pointed sets for partial functions and the lifting monad.

Summarize Moggi's modal λ-calculus. Also see Shulman.

See [Sco80b, Sco80a] for more on the λ-calculus

Still ToDo: Normalization use Lambek-Scott, NbE use Altenkirch

Chapter 3

Dependent Type Theory

The Curry-Howard correspondence from Chapter 1.3 can be extended to natural deduction proofs in *first-order* logic, providing an extension of the "propositions as types/proofs as terms" idea from propositional logic to first-order logic (see [Sco70, How80] . In addition to simple types $A, B, ...$ representing propositions, one then has *dependent* types $x : A \vdash B(x)$ representing "propositional functions" or predicates. In addition to the simple type formers $A \times B$ and $A \to B$, one has dependent type formers $\Sigma_{x:A} B(x)$ and $\Pi_{x:A} B(x)$, representing the quantified propositions $\exists_{x:A} B(x)$ and $\forall_{x:A} B(x)$. As before, these types may have different terms $s, t : \Pi_{x:A} B(x)$, resulting from different proofs of the corresponding propositions, so that the calculus of terms again records more information than mere *provability*. Also as before, the resulting abstract structure turns out to be one that is shared by other categories *not* arising from logic—and now the coincidence is even more remarkable, because the structure at issue is a much more elaborate one. Where proofs in the propositional calculus gave rise to a cartesian closed category, the category of proof terms of first-order logic will be seen to be *locally cartesian closed*, a mathematical structure also shared by sheaves on a space, Grothendieck toposes, categories of fibrations, and other important examples.

Before stating a formal dependent type theory, we begin by infomally "categorifying" first-order logic with an abstraction (due to Lawvere [Law70]) called a *hyperdoctrine*. A hyperdoctrine is a contravariant functor $P : \mathcal{C}^{\mathsf{op}} \to \mathsf{Cat}$ (see Section 3.1), and there are in particular both poset-valued and "proper" category-valued ones. The former correspond to propositional and predicate logic, while the latter correspond more closely to dependent type theory, where the individual value categories $P(C)$ may be proper cartesian closed categories (rather than just Heyting algebras or CCC posets). Moreover, the reindexing functors along all projections $p_A : X \times A \to A$ in the index category $\mathcal{C}$ of contexts are also required to admit both left and right adjoints $\Sigma_A \dashv p_A^* \dashv \Pi_A$, according to Lawvere's adjoint analysis of quantification. An important *difference* between hyperdoctrines and dependent type theories, however, is that the indexing category of contexts in dependent type theory has not just finite products, but also some additional structure resulting from an operation of *context extension*, which takes as input a type in context $\Gamma \vdash A$ and returns a new context $(\Gamma, x : A)$, together with a substitution arrow $(\Gamma, x : A) \to \Gamma$. This is taking

the "propositions-as-types" idea even more seriously, by allowing every proposition $\Gamma \vdash \varphi$ in first-order logic to form a new type $\{\Gamma \vdash \varphi\}$, thus turning the objects $A \in P(C)$ in the value-categories of hyperdoctrine $(\mathcal{C}, P)$ into arrows $\{A\} \to C$ in $\mathcal{C}$.[1]

3.1 Hyperdoctrines

Given an algebraic signature, let $\mathcal{C}$ be the category of contexts, with (non-dependent) tuples of typed variables $\Gamma = (x_1 : C_1, ..., x_n : C_n)$ as objects, and as arrows $\gamma : \Delta \to \Gamma$ the n-tuples of terms $c_1 : C_1, \ldots, c_n : C_n$, all in context $\Delta = (y_1 : D_1, ..., y_m : D_m)$,

$$\Delta \vdash c_i : C_i\,, \quad 1 \leq i \leq n\,.$$

Composition is given by substitution of terms for variables,

$$\gamma \circ \delta = (c_1[d_1/y_1, \ldots, d_m/y_m], \ldots, c_n[d_1/y_1, \ldots, d_m/y_m],)$$

for $\delta = (d_1, \ldots, d_m) : \mathsf{E} \to \Delta$ with $\mathsf{E} = (z_1 : E_1, ..., z_k : E_k)$, and the identity arrows are the variables themselves (terms are identified up to α-renaming of variables, as in Lawvere algebraic theories, see Chapter **??**). The category $\mathcal{C}$ then has all finite products, essentially given by tupling.

For each object Γ, let $P(\Gamma)$ be the *poset* of all first-order formulas $(\Gamma \mid \varphi)$, ordered by entailment $\Gamma \mid \varphi \vdash \psi$ and identified up to provable equivalence $\Gamma \mid \varphi \dashv\vdash \psi$. Substitution of a term $\sigma : \Delta \to \Gamma$ into a formula $(\Gamma \mid \varphi)$ then determines a morphism of posets $\sigma^* : P(\Gamma) \to P(\Delta)$, which also preserves all of the propositional operations,

$$\sigma^*(\varphi \wedge \psi) = \varphi[\sigma/x] \wedge \psi[\sigma/x] = \sigma^*(\varphi) \wedge \sigma^*(\psi), \qquad \text{etc.}$$

(Exercise!). Moreover, since substitutions into formulas and terms commute with each other, $\tau^*\sigma^*\varphi = \varphi[\sigma \circ \tau/x]$, this action is *strictly* functorial, and so we have a contravariant functor

$$P : \mathcal{C}^{\mathsf{op}} \longrightarrow \mathsf{Heyt}$$

from the category of contexts to the category of Heyting algebras.

Now consider the quantifiers $\exists$ and $\forall$. Given a projection of contexts $p_X : \Gamma \times X \to \Gamma$, in addition to the pullback functor

$$p_X^* : P(\Gamma) \longrightarrow P(\Gamma \times X)$$

induced by weakening, there are the operations of quantification

$$\exists_X, \forall_X : P(\Gamma \times X) \longrightarrow P(\Gamma)\,.$$

By the rules for the quantifiers, these are indeed left and right adjoints to weakening,

$$\exists_X \dashv p_X^* \dashv \forall_X\,.$$

The Beck-Chevalley rules assert that substitution commutes with quantification, in the sense that $(\forall_x \varphi)[s/y] = \forall_x(\varphi[s/y])$, and similarly for $(\exists_x \varphi)$.

[1] [Law70] does just this.

Definition 3.1.1. A *(posetal) hyperdoctrine* consists of a Cartesian category $\mathcal{C}$ together with a contravariant functor

$$P : \mathcal{C}^{\mathsf{op}} \longrightarrow \mathsf{Heyt}\,,$$

such that for each $f : D \to C$ the action maps $f^* = Pf : PC \to PD$ have both left and right adjoints

$$\exists_f \dashv f^* \dashv \forall_f$$

that satisfy the Beck-Chavalley conditions.

Exercise 3.1.2. Verify that the syntax of first-order logic can indeed be organized into a hyperdoctrine in the way just described.

Examples

1. We just described the syntactic example of first-order logic. Indeed, for each first-order theory $\mathbb{T}$ there is an associated hyperdoctrine $(\mathcal{C}_\mathbb{T}, P_\mathbb{T})$, with the types and terms of $\mathbb{T}$ as the category of contexts $\mathcal{C}_\mathbb{T}$, and the formulas (in context) of $\mathbb{T}$ as "predicates", i.e. the elements of the Heyting algebras $\varphi \in P_\mathbb{T}(\Gamma)$. A general hyperdoctrine can be regarded as an abstraction of this example.

2. A hyperdoctrine on the index category $\mathcal{C} = \mathsf{Set}$ is given by the powerset functor

   $$\mathcal{P} : \mathsf{Set}^{\mathsf{op}} \longrightarrow \mathsf{Heyt}\,,$$

   which is represented by the Heyting algebra $\mathbb{2}$, in the sense that for each set I one has

   $$\mathcal{P}(I) \cong \mathsf{Hom}(I, \mathbb{2})\,.$$

   Similarly, for any *complete* Heyting algebra H in place of $\mathbb{2}$, there is a hyperdoctrine H-Set, with

   $$P_\mathsf{H}(I) \cong \mathsf{Hom}(I, \mathsf{H})\,.$$

   The adjoints to precomposition along a map $f : J \to I$ are given by

   $$\begin{aligned} \exists_f(\varphi)(i) &= \bigvee_{j \in J} (f(j) = i) \wedge \varphi(j)\,, \\ \forall_f(\varphi)(i) &= \bigwedge_{j \in J} (f(j) = i) \Rightarrow \varphi(j)\,, \end{aligned}$$

   where the value of $x = y$ in H is defined to be $\bigvee\{\top \mid x = y\}$.

   We leave it as an exercise to verify that this is hyperdoctrine, in particular to show that the Beck-Chevalley conditions are satisfied.

   Exercise 3.1.3. Show this.

3. For a related example, let $\mathbb{C}$ be any small index category and $\mathcal{C} = \widehat{\mathbb{C}}$, the category of presheaves on $\mathbb{C}$. An internal Heyting algebra H in $\mathcal{C}$, i.e. a functor $\mathbb{C}^{\mathsf{op}} \to \mathsf{Heyt}$, is said to be *internally complete* if, for every $I \in \mathcal{C}$, the transpose $\mathsf{H} \to \mathsf{H}^I$ of the projection $\mathsf{H} \times I \to \mathsf{H}$ has both left and right adjoints. Such an internally complete Heyting algebra determines a (representable) hyperdoctrine $P_{\mathsf{H}} : \mathcal{C} \to \mathsf{Set}$ just as for the case of $\mathcal{C} = \mathsf{Set}$, by setting $P_{\mathsf{H}}(C) = \mathcal{C}(C, \mathsf{H})$.

4. For any Heyting category $\mathcal{H}$ let $\mathsf{Sub}(C)$ be the Heyting algebra of all subobjects $S \rightarrowtail C$ of the object C. The presheaf $\mathsf{Sub} : \mathcal{H}^{\mathsf{op}} \to \mathsf{Heyt}$, with action by pullback, is then a hyperdoctrine, essentially by the definition of a Heyting category.

Remark 3.1.4 (Lawvere's Law)**.** In any hyperdoctrine $(\mathcal{C}, P)$, for each object $C \in \mathcal{C}$, we can determine an equality relation $=_C$ in each $P(C \times C)$, namely by setting

$$(x =_C y) \;=\; \exists_{\Delta_C}(\top)\,,$$

where $\Delta_C : C \to C \times C$ is the diagonal, $\exists_{\Delta_C} \dashv \Delta_C^*$, and $\top \in P(C)$. Displaying variables for clarity, if $\rho(x, y) \in P(C \times C)$ then $\Delta_C^* \rho(x, y) = \rho(x, x) \in PC$ is the contraction of the different variables, and the adjunction $\exists_{\Delta_C} \dashv \Delta_C^*$ can be formulated as the following two-way rule,

$$\frac{x : C \mid \top \vdash \rho(x, x)}{x : C, y : C \mid (x =_C y) \vdash \rho(x, y)} \tag{3.1}$$

which expresses that $(x =_C y)$ is the least reflexive relation on C. See [Law70] and Exercise **??** above.

Exercise 3.1.5. Prove the standard first-order laws of equality from the above hyperdoctrine formulation of Lawvere's Law (3.1).

Proper hyperdoctrines

Now let us consider some hyperdoctrines of a different kind. For any set I, let Set^I be the category of *families of sets* $(A_i)_{i \in I}$, with families of functions $(g_i : A_i \to B_i)_{i \in I}$ as arrows, and for $f : J \to I$ let us reindex along f by the precomposition functor $f^* : \mathsf{Set}^I \to \mathsf{Set}^J$, with

$$f^*((A_i)_{i \in I})_j = A_{f(j)}\,.$$

Thus we have a contravariant functor

$$P : \mathsf{Set}^{\mathsf{op}} \to \mathsf{Cat}$$

with $P(I) = \mathsf{Set}^I$ and $f^*(A : I \to \mathsf{Set}) = A \circ f : J \to \mathsf{Set}$.

Lemma 3.1.6. *The precomposition functors $f^* : \mathsf{Set}^I \to \mathsf{Set}^J$ have both left and right adjoints $f_! \dashv f^* \dashv f_*$ which can be computed by the formulas:*

$$\begin{aligned} f_!(A)_i &= \coprod_{j \in f^{-1}\{i\}} A_j \,, \\ f_*(A)_i &= \prod_{j \in f^{-1}\{i\}} A_j \,, \end{aligned} \tag{3.2}$$

for $A = (A_j)_{j \in J}$. Moreover, these functors satisfy the Beck-Chevally conditions.

A closely related example uses the familiar equivalence of categories $\mathsf{Set}^I \simeq \mathsf{Set}/_I$, where now the adjoints

$$f_! \dashv f^* \dashv f_* : \mathsf{Set}/_J \longrightarrow \mathsf{Set}/_I$$

to reindexing along $f : J \to I$ are (post-)composition, pullback, and "push-forward", respectively. In this case, the action of the pseudofunctor P is not strictly functorial, as it was for the case of $P(I) = \mathsf{Set}^I$. Note that the Beck-Chevalley conditions for such Cat-valued functors should now also be stated as (canonical) isomorphisms, rather than equalities as they were for poset-valued functors. In this way, when the individual categories $P(I)$ are proper, and not just posets, the entire hyperdoctrine structure may be weakened to include (coherent) isomorphisms, both in the functorial action of P, and in the B-C conditions. We will not spell out the required coherences here, but the interested reader may look up the corresponding notion of an *indexed-category*, which is a Cat-valued *pseudofunctor* (see [**?**, B1.2]).

Example 3.1.7. Another example of a "proper" hyperdoctrine, with values in non-posetal (large!) categories, is the category of presheaves construction $\widehat{\mathbb{C}} = \mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}$, where:

$$\begin{aligned} \mathcal{P} : \mathsf{Cat}^{\mathsf{op}} &\longrightarrow \mathsf{CAT} \,, \\ \mathbb{C} &\longmapsto \widehat{\mathbb{C}} \,. \end{aligned}$$

Here the action of $\mathcal{P}$ may be assumed to be *strictly* functorial, because it's given by precomposition. Nonetheless the B-C conditions must be stated as natural isos, because the adjoints $F_! \dashv F^* \dashv F_* : \widehat{\mathbb{D}} \longrightarrow \widehat{\mathbb{C}}$ for $F : \mathbb{D} \to \mathbb{C}$ are given by left and right Kan extensions, which need not be strictly functorial.

We shall consider several more examples of proper hyperdoctrines below. The internal logic of such categories generalizes and "categorifies" first-order logic, and is better described as dependent type theory. Proper hyperdoctrines $P : \mathcal{C}^{\mathsf{op}} \to \mathsf{Cat}$ are roughly related to dependent type theory in the way that posetal ones $P : \mathcal{C}^{\mathsf{op}} \to \mathsf{Pos}$ are related to FOL. There are actually two distinct aspects of this generalization: (1) the individual categories of "predicates" $P(C)$ are proper categories rather than mere posets, (2) the variation over the index category $\mathcal{C}$ of contexts (and its adjoints) is weakened accordingly to pseudo-functoriality. Each of these aspects plays an important role in dependent type theory and its categorical semantics.

First-Order Logic	Dependent Type Theory
Propositional Logic	Simple Type Theory

3.2 Dependently-typed lambda-calculus.

We give a somewhat informal specification of the syntax of the *dependently-typed λ-calculus* (see [Hof95, AG] for a more detailed exposition).

Dependent type theories have four standard forms of judgement

$$A : \mathtt{type}\,, \quad A \equiv B : \mathtt{type}\,, \quad a : A\,, \quad a \equiv b : A\,.$$

We refer to the triple equality relation $\equiv$ in these judgements as *definitional (or judgemental) equality*. It should not be confused with the notions of *(extensional and intensional) propositional equality* to be introduced below. A judgement J of one of the four above kinds can also be made relative to a *context* Γ of variable declarations, a situation that we indicate by writing $\Gamma \vdash J$. When stating deduction rules for such judgements we make use of standard conventions to simplify the exposition, such as omitting the (part of the) context that is common to premisses and conclusions of the rule.

To formulate the rules, we revisit the rules of simple type theory from Section 2.1 and adjust them as follows.

Judgements: The basic kinds of judgements are:

$$\Gamma\ \mathtt{ctx}\,, \qquad \Gamma \vdash A\ \mathtt{type}\,, \qquad \Gamma \vdash a : A\,.$$

along with the judgemental equalities of each kind:

$$\Gamma \equiv \Delta\ \mathtt{ctx} \qquad A \equiv B\ \mathtt{type} \qquad a \equiv b : A\,,$$

each of which are assumed to satisfy the usual laws of equality.

Contexts: These are formed by the rules:

$$\frac{}{(\cdot)\ \mathtt{ctx}} \qquad\qquad \frac{\Gamma \vdash A\ \mathtt{type}}{\Gamma, x : A\ \mathtt{ctx}}$$

Here it is assumed that x is a fresh variable, not already occurring in Γ. Note that, unlike in the simple type theory of the previous chapter, the order of the types occurring in a context now matters, since types to the right may depend on ones to their left.

Types: In addition to the usual *simple types*, generated from *basic types* by formation of *products* and *function types*, we may also have some *basic types in context*,

$$\text{Basic dependent types} \quad \Gamma_1 \vdash \mathtt{B}_1,\ \Gamma_2 \vdash \mathtt{B}_2,\ \cdots$$

where the contexts Γ need not be basic. Further dependent types are formed from the basic ones by the *sum* Σ and *product* Π *type formers*, using the *formation rules*:

$$\frac{\Gamma, x : A \vdash B\ \mathtt{type}}{\Gamma \vdash \Sigma_{x:A} B\ \mathtt{type}} \qquad \frac{\Gamma, x : A \vdash B\ \mathtt{type}}{\Gamma \vdash \Pi_{x:A} B\ \mathtt{type}}$$

Terms: As for the simple types, we assume there is a countable set of *variables* $x, y, z, \ldots$. We are also given a set of *basic constants*. The set of *terms* is then generated from variables and basic constants by the following grammar, just as for simple types:

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

The rules for deriving typing judgments are much as for simple types. They are of course assumed to hold in any context Γ.

- Each basic constant $\mathtt{c}_i$ has a uniquely determined type C_i (not necessarily basic):

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

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

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

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

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

- The typing rules for pairs and projections now take the form:

$$\frac{a : A \qquad b : B(a)}{\langle a, b \rangle : \Sigma_{x:A} B} \qquad \frac{c : \Sigma_{x:A} B}{\mathtt{fst}\, c : A} \qquad \frac{c : \Sigma_{x:A} B}{\mathtt{snd}\, c : B(\mathtt{fst}\, c)}$$

  We write e.g. $B(a)$ rather than $B[a/x]$ to indicate a substitution of the term a for the variable x in the type B. Similarly, we may write $\Sigma_{x:A} B(x)$ to emphasize the possible occurence of the variable x in B. We treat $A \times B$ as another way of writing $\Sigma_{x:A} B$, when the variable $x : A$ does not occur in the type B.

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

$$\frac{t : \Pi_{x:A} B \qquad a : A}{t\, a : B(a)} \qquad\qquad \frac{x : A \vdash t : B}{(\lambda x : A\,.\, t) : \Pi_{x:A} B}$$

  We treat $A \to B$ as another way of writing $\Pi_{x:A} B$ when the variable $x : A$ does not occur in the type B.

The (β and η) equations between these terms are just as they were for simple types:

- Equations for unit type:

$$\frac{}{t \equiv * : \mathtt{1}}$$

- Equations for sum types:

$$\frac{u \equiv v : A \qquad s \equiv t : B(a)}{\langle u, s\rangle \equiv \langle v, t\rangle : \Sigma_{x:A} B}$$

$$\frac{s \equiv t : \Sigma_{x:A} B}{\mathtt{fst}\, s \equiv \mathtt{fst}\, t : A} \qquad\qquad \frac{s \equiv t : \Sigma_{x:A} B}{\mathtt{snd}\, s \equiv \mathtt{snd}\, t : A}$$

$$\frac{}{t \equiv \langle \mathtt{fst}\, t, \mathtt{snd}\, t\rangle : \Sigma_{x:A} B} \qquad (\eta\text{-rule})$$

$$\frac{}{\mathtt{fst}\, \langle s, t\rangle \equiv s : A} \qquad\qquad \frac{}{\mathtt{snd}\, \langle s, t\rangle \equiv t : A} \qquad (\beta\text{-rule})$$

- Equations for product types:

$$\frac{s \equiv t : \Pi_{x:A} B \qquad u \equiv v : A}{s\, u \equiv t\, v : B}$$

$$\frac{x : A \vdash t \equiv u : B}{(\lambda x : A\,.\, t) \equiv (\lambda x : A\,.\, u) : \Pi_{x:A} B}$$

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

$$\frac{}{\lambda x : A\,.\, (t\, x) \equiv t : \Pi_{x:A} B} \text{ if } x \notin \mathsf{FV}(t) \qquad (\eta\text{-rule})$$

Equality types: Just as for first-order logic, for each type A we have a primitive *equality type*:

$$x, y : A \vdash \mathtt{Eq}_A(x, y)\ \mathtt{type}\,.$$

This is called *propositional equality*. For convenience, we may sometimes also write $x =_A y$ for $\mathtt{Eq}_A(x, y)$. Although they will turn out to be logically equivalent, the reader is warned not to confuse propositional and judgemental equality $x \equiv y : A$.

The formation, introduction, elimination, and computation rules for equality types are as follows:

$$\frac{s : A \qquad t : A}{s =_A t \ \mathtt{type}} \qquad\qquad \frac{a : A}{\mathtt{refl}_a : (a =_A a)}$$

$$\frac{p : s =_A t}{s \equiv t : A} \qquad\qquad \frac{p : s =_A t}{p \equiv \mathtt{refl}_s : (s =_A s)}$$

The elimination rule is known as *equality reflection*. We may say that two elements $s, t : A$ are *propositionally equal* if the type $s =_A t$ is inhabited. Thus the equality reflection rule says that if two terms are propositionally equal then they are judgementally equal.

Exercise 3.2.1. Show that two terms are propositionally equal if, and only if, they are judgementally equal.

Remark 3.2.2 (Identity types)**.** The formulation of the rules for equality just given is known as the *extensional* theory. There is also an *intensional* version, with different elimination (and computation) rules, to be considered in the next chapter. To help maintain the distinction between these three (!) different relations, the intensional version is sometimes called the *identity type* and written $\mathtt{Id}_A(s, t)$ instead. See [AG] for details.

Remark 3.2.3 (Variant rules for sum types)**.** Another formulation of the rules for Σ-types using a single dependent elimination rule is as follows:

$$\frac{z : \Sigma_{x:A} B \vdash C \ \mathtt{type} \qquad x : A, y : B(x) \vdash c(x, y) : C(\langle x, y\rangle)}{z : \Sigma_{x:A} B \vdash \mathtt{split}(z, c) : \Sigma_{x:A} B}$$

with the associated computation rule:

$$\frac{z : \Sigma_{x:A} B \vdash C \ \mathtt{type} \qquad x : A, y : B(x) \vdash c(x, y) : C(\langle x, y\rangle)}{x : A, y : B(x) \vdash \mathtt{split}(\langle x, y\rangle, c) \equiv c(x, y) : C(\langle x, y\rangle)}$$

These rules permit one to derive the simple elimination terms $\mathtt{fst}\, c$ and $\mathtt{snd}\, c$, and to prove the above computation rules for them. The η-rule is derived using a dependent elimination involving the $\mathtt{Eq}$-type.

Exercise 3.2.4. Prove the simple elimination rules for sum-types (involving $\mathtt{fst}\, c$ and $\mathtt{snd}\, c$) from the dependent ones (involving $\mathtt{split}$).

Remark 3.2.5 (The type-theoretic axiom of choice)**.** One of the oldest problems in the foundations of mathematics is the logical status of the Axiom of Choice. Is it a "Law of Logic"? A mathematical fact about sets? A falsehood with paradoxical consequences?

Per Martin-Löf discovered that the rules of constructive type theory that we have just presented actually suffice to *decide* this question in favor of "Law of Logic" in a certain sense [ML84] (see also [Tai68]). Since the statement of the type theoretic axiom of choice goes (slightly) beyond standard first-order logic, this arguably provides a resolution that also clarifies why the problem remained open for so long in conventional mathematics.

Under propositions as types, reading Σ as "there exists" and Π as "for all", a type such a $\Pi_{x:A}\Sigma_{y:B}R(x,y)$ can be regarded as a stating a proposition—in this case, "for all $x : A$ there is a $y : B$ such that $R(x,y)$". By Curry-Howard, such a "proposition" is then provable if it has a closed term $t : \Pi_{x:A}\Sigma_{y:B}R(x,y)$, which then corresponds to a proof, by unwinding the rules that constructed the term, and observing that they correspond to the usual natural deduction rules for first-order logic.

Of course, the rules of construction for terms correspond to provability only under a certain "constructive" conception of validity (see [Sco70]). Stated as follows,

$$\Pi_{x:A}\Sigma_{y:B}R(x,y) \to \Sigma_{f:A\to B}\Pi_{x:A}R(x,fx)\,, \tag{3.3}$$

the "type theoretic axiom of choice" may sound like the classical axiom of choice under the propositions as types interpretation, but this type is actually *provable* in (constructive) type theory, rather than being an axiom!

Exercise 3.2.6. Prove the type theoretic axiom of choice (3.3) from the rules for sum and product types given here.

3.2.1 Interaction of Eq with Σ and Π

The type theoretic axiom of choice Example 3.2.5, can be seen as a distributivity law for Σ and Π. It is in fact an *isomorphism of types*: there are terms going both ways, the composites of which are propositionally (and therefore definitionally!) equal to the identity maps (i.e. $\lambda x : X.\, x : X \to X$). It is natural to ask, how do the other type formers interact?

Consider first the result of combining Eq-types with Σ. We can show that for $s, t : A \times B$ there is always a term,

$$\mathtt{Eq}_{A\times B}(s,t) \to \big(\mathtt{Eq}_A(\mathtt{fst}\, s, \mathtt{fst}\, t) \times \mathtt{Eq}_B(\mathtt{snd}\, s, \mathtt{snd}\, t)\big)\,,$$

Moreover, there is a term in the other direction as well, and the composites are propositionally equal to the identity. By equality reflection, it therefore follows that these types are also *syntactically isomorphic*, in the sense just described. The same is true for dependent sums, although this is a bit more awkward to state, owing to the fact that $\mathtt{snd}\,(s) : B(\mathtt{fst}\, s)$ and $\mathtt{snd}\, t : B(\mathtt{fst}\,(t)$. However, since the first projection gives a term $p : \mathtt{Eq}_A(\mathtt{fst}\, s, \mathtt{fst}\, t)$ we have $\mathtt{fst}\, s \equiv \mathtt{fst}\, t$ and therefore $B(\mathtt{fst}\, s) \equiv B(\mathtt{fst}\, t)$, so that $\mathtt{Eq}_{B(\mathtt{fst}\, s)}(\mathtt{snd}\, s, \mathtt{snd}\, t)$ makes sense, and is in fact judgementally equal to $\mathtt{Eq}_{B(\mathtt{fst}\, t)}(\mathtt{snd}\, s, \mathtt{snd}\, t)$, so we can write:

$$\mathtt{Eq}_{\Sigma_{x:A}B}(s,t) \to \Sigma_{p:\mathtt{Eq}_A(\mathtt{fst}\, s, \mathtt{fst}\, t)}\mathtt{Eq}_{B(\mathtt{fst}\, s)}(\mathtt{snd}\, s, \mathtt{snd}\, t)\,,$$

Moreover, since $\mathtt{Eq}_{B(\mathtt{fst}\, s)}(\mathtt{snd}\, s, \mathtt{snd}\, t)$ does not depend on $p : \mathtt{Eq}_A(\mathtt{fst}\, s, \mathtt{fst}\, t)$, this actially rewrites to:

$$\mathtt{Eq}_{\Sigma_{x:A}B}(s,t) \to \mathtt{Eq}_A(\mathtt{fst}\, s, \mathtt{fst}\, t) \times \mathtt{Eq}_{B(\mathtt{fst}\, s)}(\mathtt{snd}\, s, \mathtt{snd}\, t)\,.$$

Moreover, these types are also isomorphic.

For Π-types, given terms $f, g : A \to B$, we can form a term of type

$$\mathtt{Eq}_{A\to B}(f,g) \to \Pi_{x:A}\mathtt{Eq}_B(fx, gx)\,.$$

and again, this is an isomorphism of types. The corresponding law for dependent functions $f, g : \Pi_{x:A}B$ takes the more perspicuous form

$$\mathtt{Eq}_{\Pi_{x:A}B}(f,g) \to \Pi_{x:A}\mathtt{Eq}_{B(x)}(fx, gx)\,.$$

And again, this is also an iso. Note that these last two isomorphisms say that two functions are equal just if they are so "pointwise". This principle is called *Function Extensionality.*

Finally, let us consider equality of equality types. Given any terms $a, b : C$ and $p, q : \mathtt{Eq}_C(a,b)$, what more can be said? The principle called *Uniqueness of Identity Proofs* (UIP) asserts that there is always a term of type

$$\mathtt{Eq}_{\mathtt{Eq}_C(a,b)}(p,q)\,.$$

Is there an argument for this principle, analogous to those for the equalities of terms of types Σ and Π? We shall return to this question in the setting of intensional type theory in the next chapter.

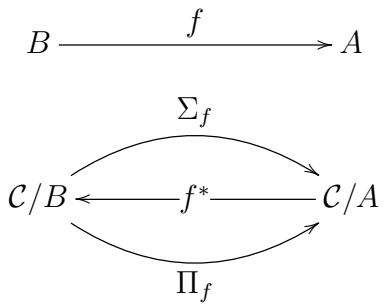
Exercise 3.2.7. Prove that extensional type theory satisfies (UIP).

3.3 Locally cartesian closed categories

Recall the following proposition from **??**.

Proposition 3.3.1 (and Definition)**.** *The following conditions on a category $\mathcal{C}$ with a terminal object 1 are equivalent:*

1. *Every slice category $\mathcal{C}/A$ is cartesian closed.*
2. *For every arrow $f : B \to A$ the (post-) composition functor $\Sigma_f : \mathcal{C}/B \to \mathcal{C}/A$ has a right adjoint f^*, which in turn has a right adjoint Π_f.*



Such a category is called locally cartesian closed.

The notation $\Sigma_f \dashv f^* \dashv \Pi_f$ of course anticipates the interpretation of DTT. A common alternate notation is $f_! \dashv f^* \dashv f_*$.

Proof. Suppose every slice of $\mathcal{C}$ is cartesian closed. It suffices to consider the case of (2) with $A = 1$, and to show that the functor $B^* : \mathcal{C} \to \mathcal{C}/_B$ with $B^*(X) = (\pi_2 : X \times B \to B)$ has both left and right adjoints $\Sigma_B \dashv B^* \dashv \Pi_B : \mathcal{C}/_B \to \mathcal{C}$. For Σ_B we can just take the forgetful functor. For $\Pi_B(p : X \to B)$ we use the CCC structure in $\mathcal{C}$ to form the map $p^B : X^B \to B^B$, which we then pull back along the point $'1_B' : 1 \to B^B$ that is the transpose of the identity map $1_B : B \to B$.

$$\begin{array}{ccc} \Pi_B(X) & \longrightarrow & X^B \\ \downarrow & \lrcorner & \downarrow{\scriptstyle p^B} \\ 1 & \xrightarrow['1_B']{} & B^B \end{array}$$

Conversely, if $\mathcal{C}$ is LCC, then in every slice $\mathcal{C}/_X$ we can define the product of $A \to X$ and $B \to X$ as $A \times_X B = \Sigma_A A^* B$ and the exponential as $(B^A)_X = \Pi_A A^* B$. The universal properties are then easily checked. □

Exercise 3.3.2. Verify the details of the proof just sketched for Proposition 3.3.1.

Basic examples of LCCCs

1. We have already seen the hyperdoctrine $\mathcal{C} = \mathsf{Set}$ and $P : \mathsf{Set}^{\mathsf{op}} \to \mathsf{Cat}$ where $P(I) = \mathsf{Set}^I$, with action of $f : J \to I$ on $A : I \to \mathsf{Set}$ by precomposition $f^*A = A \circ f : J \to \mathsf{Set}$, which is strictly functorial. There is an equivalent hyperdoctrine with the slice category $\mathsf{Set}/_I$ as the "category of predicates" and action by pullback $f^* : \mathsf{Set}/_I \to \mathsf{Set}/_J$. The equivalence of categories

$$\mathsf{Set}^I \simeq \mathsf{Set}/_I$$

allows us to use post-composition as the left adjoint $f_! : \mathsf{Set}/_J \to \mathsf{Set}/_I$, rather than the coproduct formula in (3.2). Indeed, this hyperdoctrine structure arises immediately from the locally cartesian closed character of Set. We have the same for any other LCC $\mathcal{E}$, namely the pair $(\mathcal{E}, \mathcal{E}/_{(-)})$ determines a hyperdoctrine, with the action of $\mathcal{E}/_{(-)}$ by pullback, and the left and right adjoints coming from the LCC structure.

2. Another familiar example of a hyperdoctrine arising from LCC structure is presheaves on a small category $\mathbb{C}$, where for the slice category $\widehat{\mathbb{C}}/_X$ we have another category of presheaves, namely

$$\widehat{\mathbb{C}}/_X \cong \widehat{\textstyle\int_{\mathbb{C}} X}\,, \tag{3.4}$$

on the category of elements $\int_{\mathbb{C}} X$. For a natural transformation $f : Y \to X$ we have a functor $\int f : \int Y \to \int X$, which (as usual) induces a triple of adjoints on presheaves,

$$(\textstyle\int f)_! \dashv (\int f)^* \dashv (\int f)_* : \widehat{\int Y} \longrightarrow \widehat{\int X}\,,$$

where the middle functor $(\int f)^*$ is precomposition with $\int f$, which preserves all (co)limits. These satisfy the Beck-Chevalley conditions (up to isomorphism), because this indexed category is equivalent to the one coming from the LCC structure (3.4), which we know satisfies them.

Note that each of the categories $\widehat{\mathbb{C}}/_X$ is therefore also Cartesian closed, so that $\widehat{\mathbb{C}}$ is indeed LCC by Proposition 3.3.1. Moreover each $\widehat{\mathbb{C}}/_X$ also has coproducts $0, X + Y$, so it is a "categorified" Heyting algebra—although we don't make that part of the definition of a hyperdoctrine.

3. An instructive example of a hyperdoctrine that is *not* an LCC is the subcategory of Pos of posets and monotone maps, which we already met in Section **??**, with the "predicates" being the discrete fibrations. For each poset K, let us take as the category of predicates $P(K)$ the full subcategory $\mathsf{dFib}\_K \hookrightarrow \mathsf{Pos}/_K$ consisting of the *discrete fibrations*: monotone maps $p : X \to K$ with the "unique lifting property": for any x and $k \leq p(x)$ there is a unique $x' \leq x$ with $p(x') = k$. Since each category $\mathsf{dFib}/_K$ is equivalent to a category of presheaves $\mathsf{Set}^{K^{\mathrm{op}}}$, and pullback along any monotone $f : J \to K$ preserves discrete fibrations, and moreover commutes with the equivalences to the presheaf categories and the precomposition functor $f^* : \hat{K} \to \hat{J}$, we have a hyperdoctrine if only the Beck-Chevalley conditions hold. We leave this as an exercise for the reader. Finally, observe that dFib cannot be an LCC, simply because it does not have a terminal object; however, every slice of course does one, and so every slice $\mathsf{dFib}/_K$ is a CCC, and therefore also an LCC (since a slice of a slice is a slice).

4. An example formally similar to the foregoing is the non-full subcategory $\mathsf{LocHom} \hookrightarrow \mathsf{Top}$ of topological spaces and local homeomorphisms between them, which also lacks a terminal object, but each slice of which $\mathsf{LocHom}/X \simeq \mathsf{Sh}(X)$ is equivalent to the topos of *sheaves* on the space X, and is therefore CCC (and so LCCC).

5. Fibrations of groupoids. Another, similar, example of a hyperdoctrine not arising simply from an LCCC is the category Grpd of groupoids and homomorphisms, which is not LCC (cf. [Pal03]). We can however take as the category of predicates $P(G)$ the full subcategory $\mathsf{Fib}(G) \hookrightarrow \mathsf{Grpd}/_G$ consisting of the *fibrations* into G: homomorphisms $p : H \to G$ with the "iso lifting property": for any $h \in H$ and $\gamma : g \cong p(h)$ there is some $\vartheta : h' \cong h$ with $p(\vartheta) = \gamma$. Now each category $\mathsf{Fib}(G)$ is biequivalent to a category of presheaves of groupoids $\mathsf{Fib}(G) \simeq \mathsf{Grpd}^{G^{\mathrm{op}}}$. It is not so easy to show that this is a (bicategorical) hyperdoctrine; see [HS98]. This example will be important in the next chapter as a model of *intensional* dependent type theory. The category Cat, with *iso*-fibrations as the "predicates", has a similar character.

Exercise 3.3.3. 1. Verify that the pullback of a discrete fibration $X \to K$ along a monotone map $f : J \to K$ exists in Pos, and is again a discrete fibration.

2. Verify the equivalence of categories $\mathsf{dFib}(K) \simeq \mathsf{Set}^{K^{\mathsf{op}}}$.

3. Show the Beck-Chavelley conditions for the indexed category of discrete fibrations of posets.

Exercise 3.3.4. Let $P : \mathcal{C}^{\mathsf{op}} \to \mathsf{Cat}$ be a hyperdoctrine for which there are equivalences $PC \simeq \mathcal{C}/C$, naturally in C, with respect to the left adjoints $\Sigma_f : \mathcal{C}/A \to \mathcal{C}/B$ for all $f : A \to B$ in $\mathcal{C}$. Show that $\mathcal{C}$ is then LCC.

Exercise 3.3.5. Show that any LCCC $\mathcal{C}$, regarded as a hyperdoctrine, has equality in the sense of Remark 3.1.4.

3.4 Functorial semantics of DTT in LCCCs

We begin by describing a "naive" interpretation of dependent type theory in a locally cartesian closed category which, although not strictly sound, is nonetheless useful and intuitive. In particular, it extends the functorial semantics of simple type theory in CCCs that we developed in the last chapter in a natural way. In a subsequent section, we shall "strictify" the interpretation to one that is fully correct, but technically somewhat more complicated. See Remark 3.4.4 below.

Contexts Γ are interpreted as objects $[\![\Gamma]\!]$, and dependent types $\Gamma \vdash A$ as morphisms into the context $[\![\Gamma]\!]$. To begin, let $\mathcal{C}$ be an LCCC, and interpret the empty context as the terminal object, $[\![\cdot]\!] = 1$. Then to each closed basic type $\cdot \vdash \mathtt{B}$, we assign a type $[\![\mathtt{B}]\!]$ and interpret $[\![\cdot \vdash \mathtt{B}]\!] : [\![\mathtt{B}]\!] \to 1$. Proceeding by recursion, given any type in context $\Gamma \vdash A$, we shall have

$$[\![\Gamma \vdash A]\!] : [\![\Gamma, A]\!] \longrightarrow [\![\Gamma]\!]\,,$$

abbreviating $\Gamma, x : A$ to Γ, A. A basic dependent type $\Gamma \vdash B$ is interpreted by specifying a map $[\![\Gamma \vdash B]\!] : [\![\Gamma, B]\!] \longrightarrow [\![\Gamma]\!]$, where the interpretation $[\![\Gamma]\!]$ is assumed to have been already given. Note that in this way, we also interpret the operation of *context extension*, by taking the domain of the interpretation of a type in context.

Weakening a type in context $\Gamma \vdash C$ to one $\Gamma, A \vdash C$ is interpreted as

$$[\![\Gamma, A \vdash C]\!] = p^*[\![\Gamma \vdash C]\!],$$

that is, the lefthand vertical map in the following pullback square, where the substitution $p : [\![\Gamma, A]\!] \to [\![\Gamma]\!]$ is the canonical projection $p = [\![\Gamma \vdash A]\!]$.

$$\begin{array}{ccc} [\![\Gamma, A, C]\!] & \longrightarrow & [\![\Gamma, C]\!] \\ {\scriptstyle [\![\Gamma, A \vdash C]\!]}\downarrow & & \downarrow{\scriptstyle [\![\Gamma \vdash C]\!]} \\ [\![\Gamma, A]\!] & \xrightarrow[p]{} & [\![\Gamma]\!] \end{array} \tag{3.5}$$

Given $\Gamma, A \vdash B$, we may assume that we already have maps

$$[\![\Gamma, A, B]\!] \overset{[\![\Gamma, A \vdash B]\!]}{\longrightarrow} [\![\Gamma, A]\!] \overset{[\![\Gamma \vdash A]\!]}{\longrightarrow} [\![\Gamma]\!]\,,$$

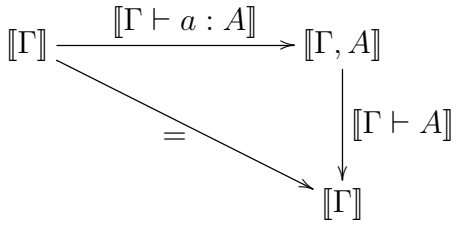
and we use the left and right adjoints to the pullback functor

$$[\![\Gamma \vdash A]\!]^* : \mathcal{C}/_{[\![\Gamma]\!]} \to \mathcal{C}/_{[\![\Gamma, A]\!]}$$

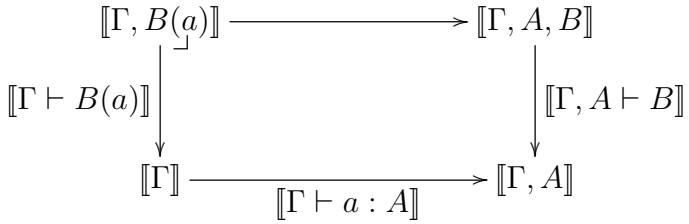
to interpret the eponymous type-forming operations:

$$\begin{aligned} [\![\Gamma \vdash \Sigma_{x:A} B]\!] &= \Sigma_{[\![\Gamma \vdash A]\!]}([\![\Gamma, A \vdash B]\!])\,, \\ [\![\Gamma \vdash \Pi_{x:A} B]\!] &= \Pi_{[\![\Gamma \vdash A]\!]}([\![\Gamma, A \vdash B]\!])\,. \end{aligned}$$

A term $\Gamma \vdash a : A$ is interpreted as a section:

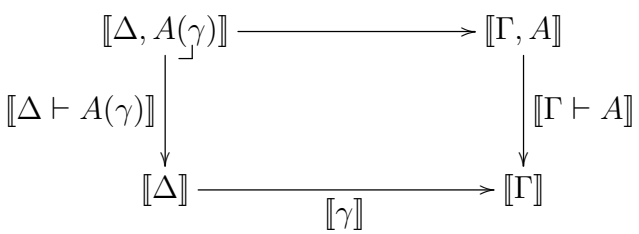


Finally, as in first-order logic, substitution of a term $\Gamma \vdash a : A$ for a variable $\Gamma, x : A$ in a dependent type $\Gamma, A \vdash B$ is interpreted by taking a pullback,



and similarly for substitution into terms.

More generally, given any substitution $\gamma : \Delta \to \Gamma$ (a tuple of terms $c_1, \ldots, c_n$ in context Δ of types those in $\Gamma = (x_1 : C_1, \ldots, x_n : C_n)$), we have a morphism $[\![\gamma]\!] : [\![\Delta]\!] \to [\![\Gamma]\!]$. Then, as in the substitution of a single term $\Gamma \vdash c : C$ for a variable $\Gamma, x : C$, we can obtain a pullback diagram along $[\![\gamma]\!]$:



The lefthand vertical map is then by definition the interpretation of the substituted type $\Delta \vdash A(\gamma)$. The interpretation of a substitution into a term $\Delta \vdash a(\gamma) : A(\sigma)$ is similarly induced by pullback.

Finally, we interpret an equality type $x : A, y : A \vdash \mathtt{Eq}_A(x, y)$ as the diagonal of the interpretation of A,

$$[\![x, y : A \vdash \mathtt{Eq}_A(x, y)]\!] = \Delta_{[\![A]\!]} : [\![A]\!] \longrightarrow [\![A]\!] \times [\![A]\!] .$$

As a map of contexts we then have

$$[\![A, A, \mathtt{Eq}_A(x, y)]\!] = [\![A]\!] \longrightarrow [\![A]\!] \times [\![A]\!] = [\![A, A]\!] .$$

For $\Gamma, A, A \vdash \mathtt{Eq}_A(x, y)$, with a context Γ, we take the diagonal of the map $[\![\Gamma \vdash A]\!] : [\![\Gamma, A]\!] \longrightarrow [\![\Gamma]\!]$ as an object in the slice category over $[\![\Gamma]\!]$.

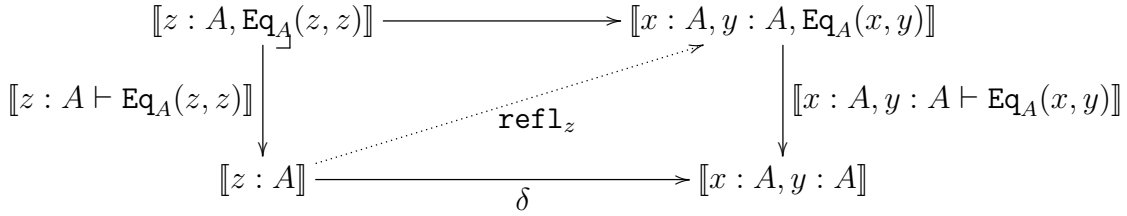
Proposition 3.4.1 ([See84])**.** *The rules of dependent type theory are sound with respect to the naive interpretation in any LCCC,* modulo *the following Remark 3.4.4.*

Proof. One needs to check that all of the typing judgements from Section 3.2 are sound, but much of this work has already been done in the simply typed case, in virtue of Proposition 3.3.1. One thing that we cannot take over from that case is the interpretation of the Eq-types, so let us check those rules by way of example.

The Formation rule (in the empty context) is clearly satified by the stated interpretation:

$$[\![x : A, y : A \vdash \mathtt{Eq}_A(x, y)]\!] = \Delta_{[\![A]\!]} : [\![A]\!] \longrightarrow [\![A]\!] \times [\![A]\!] . \tag{3.6}$$

For the Introduction rule, we pull back the dependent type $[\![x : A, y : A \vdash \mathtt{Eq}_A(x, y)]\!]$ along the substitution $\delta : [\![z : A]\!] \to [\![x : A, y : A]\!]$ (the diagonal of $[\![A]\!]$ interpreting the variable contraction $\langle z/x, z/y \rangle$) to obtain the diagram:



The term $[\![z : A \vdash \mathtt{refl}_z : \mathtt{Eq}_A(z, z)]\!]$ is to be a section of the left hand vertical map, or equivalently, the indicated diagonal. This can be taken to be the identity arrow of $[\![A]\!]$ by the specification of $[\![x : A, y : A \vdash \mathtt{Eq}_A(x, y)]\!]$ in (3.6). The Elimination and Computation Rules are left as an exercise. □

Exercise 3.4.2. Complete the verification of the Eq-rules.

Exercise 3.4.3. Verify either the Σ or the Π rules.

Remark 3.4.4 (Coherence)**.** Although it is correct according to the intuition of slice categories in an LCCC, the naive interpretation of substitution as pullback leads to a well-known *coherence problem* [Hof95] for the interpretation of dependent type theory, arising from the fact that pullbacks are only determined up to (canonical) isomorphism. For example, in the foregoing situation, given another substitution $\Phi \vdash \delta : \Delta$, we obtain the two-pullback diagram:

$$\begin{array}{ccccc}
[\![\Phi, A(\gamma)(\delta)]\!] & \longrightarrow & [\![\Delta, A(\gamma)]\!] & \longrightarrow & [\![\Gamma, A]\!] \\
\downarrow{\scriptstyle [\![\Phi \vdash A(\gamma)(\delta)]\!]} & & \downarrow{\scriptstyle [\![\Delta \vdash A(\gamma)]\!]} & & \downarrow{\scriptstyle [\![\Gamma \vdash A]\!]} \\
[\![\Phi]\!] & \xrightarrow[[\![\Phi \vdash \delta : \Delta]\!]]{} & [\![\Delta]\!] & \xrightarrow[[\![\Delta \vdash \gamma : \Gamma]\!]]{} & [\![\Gamma]\!]
\end{array}$$

However, the composition of substitutions $[\![\Delta \vdash \gamma : \Gamma]\!] \circ [\![\Phi \vdash \delta : \Delta]\!]$ across the bottom is the map $[\![\Phi \vdash \gamma(\delta) : \Gamma]\!]$, and so there is another option for the vertical map on the left, namely the single pullback:

$$\begin{array}{ccc}
[\![\Phi, A(\gamma(\delta))]\!] & \longrightarrow & [\![\Gamma, A]\!] \\
\downarrow{\scriptstyle [\![\Phi \vdash A(\gamma(\delta))]\!]} & & \downarrow{\scriptstyle [\![\Gamma \vdash A]\!]} \\
[\![\Phi]\!] & \xrightarrow[[\![\Phi \vdash \delta(\gamma) : \Gamma]\!]]{} & [\![\Gamma]\!]
\end{array}$$

Since pullbacks are unique up to iso, there is of course a canonical isomorphism

$$[\![\Phi, A(\gamma)(\delta)]\!] \;\cong\; [\![\Phi, A(\gamma(\delta))]\!]$$

over the base $[\![\Phi]\!]$, but there is no reason for these two objects (and their associated projections) to be the same. To put the matter succinctly, the action of substitution between contexts on dependent types is *strictly* functorial, but the action of pullback on slice categories is only a *pseudo*functor.

The naive LCCC interpretation, with substitution as pullback, is thus only sound "up to (canonical) isomorphism". The problem becomes more acute in the case of intensional type theory, where the interpretation of certain type formers is not even determined up to isomorphism. We shall consider one solution to this problem in detail in Section 4.4 below. For the remainder of this chapter on extensional type theory, however, we can continue to work "up to (canonical) isomorphism" without worrying about coherence.

As was done for simple type theory in Section 2.5, we can also again develop the relationship between the type theory and its models using the framework of *functorial semantics*. This is now a common generalization of λ-theories, modeled in CCCs, and first-order logic, modeled in Heyting categories. The first step is to build a syntactic

classifying category $\mathcal{C}_{\mathbb{T}}$ from a theory $\mathbb{T}$ in dependent type theory, which we then show classifies $\mathbb{T}$-models in LCCCs. We omit the now essentially routine details (given the analogous cases already considered), and merely state the main result, the proof of which is also analogous to the previous cases. A detailed treatment can be found in the seminal paper [See84].

Theorem 3.4.5. *For any theory* $\mathbb{T}$ *in dependent type theory, the locally cartesian closed syntactic category* $\mathcal{C}_{\mathbb{T}}$ *classifies* $\mathbb{T}$*-models, in the sense that for any locally cartesian closed category* $\mathcal{C}$ *there is an equivalence of categories*

$$\mathsf{Mod}\big(\mathbb{T}, \mathcal{C}\big) \;\simeq\; \mathsf{LCCC}\big(\mathcal{C}_{\mathbb{T}}\,, \mathcal{C}\big)\,, \tag{3.7}$$

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

As a corollary, again as before, we have that dependent type theory is *complete* with respect to the semantics in locally cartesian closed categories, in virtue of the syntactic construction of the classifying category $\mathcal{C}_{\mathbb{T}}$. Specifically, any theory $\mathbb{T}$ has a canonical interpretation $[-]$ in the syntactic category $\mathcal{C}_{\mathbb{T}}$ which is *logically generic* in the sense that, for any terms $\Gamma \vdash s : A$ and $\Gamma \vdash t : A$, we have

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

Thus, for the record, we have:

Proposition 3.4.6. *For any dependently typed theory* $\mathbb{T}$*,*

$$\mathbb{T} \vdash (\Gamma \vdash u \equiv t : A) \quad \textit{if, and only if,} \quad \mathcal{C}_{\mathbb{T}} \models (\Gamma \vdash u \equiv t : A)\,.$$

Of course, the syntactic model $[-]$ in $\mathcal{C}_{\mathbb{T}}$ is the one associated under (3.7) to the identity functor $\mathcal{C}_{\mathbb{T}} \to \mathcal{C}_{\mathbb{T}}$, *i.e.* it is the *universal* one. It therefore satisfies an equation just in case the equation holds in *all* models, by the classifying property of $\mathcal{C}_{\mathbb{T}}$, and the preservation of satisfaction of equations by LCCC functors (as in Proposition 2.3.2).

Corollary 3.4.7. *For any dependently typed theory* $\mathbb{T}$*,*

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

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

Remark 3.4.8. In the current setting of *extensional* dependent type theory, soundness and completeness with respect to inhabitation is actually to the same for equations, because $\vdash u \equiv t : A$ just if $\vdash e : \mathtt{Eq}_A(u, t)$ for some (closed) term e, and similarly on the semantic side.

The embedding and completeness theorems of the previous chapter with respect to general presheaf models, Kripke models, and topological and sheaf semantics can also be extended to dependently typed theories. See [AR11, Awo00] for details. The internal language construction and Theorem 2.5.2, associating a dependently typed theory with an LCCC, also extend from simple to dependent type theory. Indeed, this is the main result of [See84]. In view of this result, we hereafter move back and forth freely between syntactic (*i.e.*, type theoretic) and semantic (*i.e.*, categorical) statements and proofs.

Exercise 3.4.9. In the internal logic of an LCCC $\mathcal{E}$, show that the category of types in context $\Gamma \in \mathcal{E}$ is equivalent to the slice category $\mathcal{E}/_\Gamma$. (*Hint*: use `Eq`-types.)

3.5 Inductive types

3.5.1 Sum types

Recall from Chapter 2 the following rules for sum types $0, A+B$ in STT, with term-formers $!\,t$, $\mathtt{inl}\,t$, and $\mathtt{inr}\,t$ and $[x.t_1, x.t_2]u$.

1. The Introduction and Elimination rules are:

$$\frac{\Gamma \mid a : A}{\Gamma \mid \mathtt{inl}\, a : A + B} \qquad \frac{\Gamma \mid b : B}{\Gamma \mid \mathtt{inr}\, b : A + B}$$

$$\frac{\Gamma \mid u : \mathtt{0}}{\Gamma \mid !\, u : C} \qquad \frac{\Gamma, x : A \mid s : C \qquad \Gamma, y : B \mid t : C \qquad \Gamma \mid u : A + B}{\Gamma \mid [x.s, y.t]u : C}$$

2. The Computation rules are the following *equations*.

$$\overline{z : C \mid z \equiv !\, u : C} \qquad \overline{[x.a, y.b](\mathtt{inl}\, s) \equiv a[s/x] : C} \qquad \overline{[x.a, y.b](\mathtt{inr}\, t) \equiv b[t/y] : C}$$

$$\overline{u \equiv [x.\mathtt{inl}\, x, y.\mathtt{inr}\, y]u : A + B}$$

$$\overline{v\big([x.s, y.t]u\big) \equiv [x.vs, y.vt]u : D}$$

To reformulate these in a form suitable for *dependent* types, we shall change the Elimination rule to allow for a type C *varying over* the type being defined. The idea is that the simple elimination rule for $A + B$ determines a (normal) function of the form $A + B \to C$, while the dependent eliminator determines a "dependent function" $z : A+B \vdash s(z) : C(z)$, i.e. a section s,

$$\begin{array}{ccc} & & C \\ & {\scriptstyle s}\nearrow & \downarrow \\ A + B\,. & & \end{array}$$

We designate the former as *recursion* and the latter as *induction*, for reasons which will be become clear shortly, but for now, one can think of $C \to A + B$ as a type-family or "predicate" on $A + B$. Let us consider first the special of the "Boolean truth-values" $\mathtt{Bool} = 1 + 1$.

The Booleans. As a special case of sum types, consider the following rules for the type $\mathtt{Bool} = 1 + 1$, with the obvious renaming of the term-formers (cf. [**?**, Section 5.1])

- Formation rule.
$$\mathtt{Bool}\ \mathtt{type}\,.$$
- Introduction rules.
$$\mathtt{false} : \mathtt{Bool}\,, \qquad \mathtt{true} : \mathtt{Bool}\,.$$
- Elimination rule.
$$\frac{x : \mathtt{Bool} \vdash C(x)\ \mathtt{type} \qquad c_0 : C(\mathtt{false}) \qquad c_1 : C(\mathtt{true})}{x : \mathtt{Bool} \vdash \mathsf{ind}_{\mathtt{Bool}}(x, c_0, c_1) : C(x)}$$
- Computation rules.
$$\mathsf{ind}_{\mathtt{Bool}}(\mathtt{false}, c_0, c_1) \equiv c_0 : C(\mathtt{false})\,, \qquad \mathsf{ind}_{\mathtt{Bool}}(\mathtt{true}, c_0, c_1) \equiv c_1 : C(\mathtt{true}) \quad (\beta)$$
$$\frac{x : \mathtt{Bool} \vdash c(x) : C(x)\ \mathtt{type}}{x : \mathtt{Bool} \vdash \mathsf{ind}_{\mathtt{Bool}}(x, c(\mathtt{false}), c(\mathtt{true})) \equiv c(x) : C(x)} \quad (\eta)$$

A basic property of dependent elimination rules is now the following.

Proposition 3.5.1. *The η-rule can be derived from the other rules.*

Proof. We use the dependent $\mathtt{Eq}$-type for $C(x)$ as follows:
By $\mathtt{Eq}$-elim it suffices to show that there is a term
$$x : \mathtt{Bool} \vdash e : \mathtt{Eq}_{C(x)}\big(\mathsf{ind}_{\mathtt{Bool}}(x, c(\mathtt{false}), c(\mathtt{true})), c(x)\big)\,.$$
By $\mathtt{Bool}$-elim it therefore suffices to have terms
$$e_0 : \mathtt{Eq}_{C(\mathtt{false})}\big(\mathsf{ind}_{\mathtt{Bool}}(\mathtt{false}, c(\mathtt{false}), c(\mathtt{true})), c(\mathtt{false})\big)\,,$$
$$e_1 : \mathtt{Eq}_{C(\mathtt{true})}\big(\mathsf{ind}_{\mathtt{Bool}}(\mathtt{true}, c(\mathtt{false}), c(\mathtt{true})), c(\mathtt{true})\big)\,.$$
But by the β-rules, we have
$$\mathsf{ind}_{\mathtt{Bool}}(\mathtt{false}, c(\mathtt{false}), c(\mathtt{true})) \equiv c(\mathtt{false})$$
$$\mathsf{ind}_{\mathtt{Bool}}(\mathtt{true}, c(\mathtt{false}), c(\mathtt{true})) \equiv c(\mathtt{true})\,,$$
and so we can take
$$e_0 := \mathtt{refl} : \mathtt{Eq}_{C(\mathtt{false})}\big(c(\mathtt{false}), c(\mathtt{false})\big)\,,$$
$$e_1 := \mathtt{refl} : \mathtt{Eq}_{C(\mathtt{true})}\big(c(\mathtt{true}), c(\mathtt{true})\big)\,.$$
□

Exercise 3.5.2. Formulate the corresponding dependent rules for sum types $A + B$ and prove the η-rule from the others.

Exercise 3.5.3. Assuming the dependent rules for sum types $A + B$, prove the simple rules, including the η and distributivity computation rules.

3.5.2 Natural numbers

First, recall the following definition, which can be stated in any category with a terminal object. It is usually stated with additional parameters, but this is not required in the case of an LCCC (see [LS88] for a discussion).

Definition 3.5.4 (Lawvere [Law63])**.** A *natural numbers object* in an LCCC $\mathcal{E}$ is an object N equipped with the structure $0 : 1 \to \mathsf{N}$ and $s : \mathsf{N} \to \mathsf{N}$, and initial in the category of such structures.

Spelling this out: given any object $A \in \mathcal{E}$ together with morphisms $a : 1 \to A$ and $f : A \to A$, there is a unique map

$$u : \mathsf{N} \to A\,,$$

making the following diagram in $\mathcal{E}$ commute.

$$\begin{array}{ccccc} 1 & \xrightarrow{a} & A & \xrightarrow{f} & A \\ \| & & \uparrow{\scriptstyle u} & & \uparrow{\scriptstyle u} \\ 1 & \xrightarrow[0]{} & \mathsf{N} & \xrightarrow[s]{} & \mathsf{N} \end{array}$$

Of course, this is just a categorical way of stating the usual "definition by recursion" property of the natural numbers. It can be shown to imply the usual Peano axioms in an elementary topos (see [LS88]).

In type theory, we can formulate the corresponding notion as an inductive type. Indeed, the type Nat of natural numbers is the paradigmatic inductive type. The familiar rules for Nat in *simple* type theory are as follows.

- Formation rule.

$$\mathsf{Nat}\ \mathtt{type}$$

- Introduction rules.

$$\mathtt{zero} : \mathsf{Nat} \qquad n : \mathsf{Nat} \vdash \mathtt{succ}(n) : \mathsf{Nat}$$

- Simple elimination rule.

$$\frac{C\ \mathtt{type} \qquad c_0 : C \qquad x : C \vdash c(x) : C}{n : \mathsf{Nat} \vdash \mathtt{rec}(n, c_0, c) : C}$$

Note that in the conclusion we treat $c := \lambda x : C.\, c(x)$ as a term of type $C \to C$.

- Computation rules.

$$\begin{aligned} \mathtt{rec}(\mathtt{zero}, c_0, c) &\equiv c_0 : C \\ \mathtt{rec}(\mathtt{succ}(n), c_0, c) &\equiv c(\mathtt{rec}(n, c_0, c)) : C \end{aligned}$$

The function $\mathtt{rec}_{(c_0,c)} := \lambda n : \mathsf{Nat}.\, \mathtt{rec}(n, c_0, c) : \mathsf{Nat} \to C$ then satisfies the usual recursion equations:

$$\begin{aligned} \mathtt{rec}_{(c_0,c)}(\mathtt{zero}) &\equiv \mathtt{rec}(\mathtt{zero}, c_0, c) \equiv c_0 : C\,, \\ \mathtt{rec}_{(c_0,c)}(\mathtt{succ}(n)) &\equiv \mathtt{rec}(\mathtt{succ}(n), c_0, c) \\ &\equiv c(\mathtt{rec}(n, c_0, c) \\ &\equiv c(\mathtt{rec}_{(c_0,c)}(n)) : C\,. \end{aligned}$$

Note that this specification follows Definition 3.5.4 of an NNO pretty closely, until we come to the *uniqueness* of the function $\mathtt{rec}_{(c_0,c)} : \mathsf{Nat} \to C$. In order to show that $\mathtt{rec}_{(c_0,c)}$ is the *unique* function satisfying the Computation rules, we would need to add an appropriate η-rule (see Exercise 3.5.9 below). Alternately, we can strengthen the elimination rule to a dependent one, in order to allow the type $\mathsf{Eq}_{\mathsf{Nat}}$ to appear in the conclusion, as we did for the example of `Bool` in Proposition 3.5.1. Such a dependent elimination rule corresponds (under propositions as types) to the familiar rule of "proof by induction": if for some property of natural numbers $P(n)$, we have $P(0)$, and if $P(n)$ implies $P(n+1)$ for all n, then $P(n)$ holds for all n. Reformulating this familiar principle in *dependent* type theory with an explicit proof term for the inference from n to $n+1$ results in a more powerful recursion schema with parameters.

Dependent elimination. The rules for Nat in dependent type theory use the same formation and introduction rules as above, but provide for eliminating into a type *family*, parametrized by natural numbers.

- Dependent elimination rule.

$$\frac{n : \mathsf{Nat} \vdash C(n)\ \mathtt{type} \qquad c_0 : C(\mathtt{zero}) \qquad n : \mathsf{Nat}, x : C(n) \vdash c(n, x) : C(\mathtt{succ}(n))}{n : \mathsf{Nat} \vdash \mathtt{ind}(n, c_0, c) : C(n)}$$

  Note that in the conclusion we now treat $c := \lambda n : \mathsf{Nat}\, \lambda x : C(n).\, c(n, x)$ as a term of type $\Pi_{n:\mathsf{Nat}}.\, C(n) \to C(\mathtt{succ}(n))$.

- Dependent computation rules.

$$\begin{aligned} \mathtt{ind}(\mathtt{zero}, c_0, c) &\equiv c_0 : C(\mathtt{zero})\,, \\ \mathtt{ind}(\mathtt{succ}(n), c_0, c) &\equiv c(n, \mathtt{ind}(n, c_0, c)) : C(\mathtt{succ}(n))\,. \end{aligned}$$

Proposition 3.5.5. *In the classifying category $\mathcal{C}_{\mathsf{Nat}}$ with the dependent elimination and computation rules just stated, the type Nat equipped with $\mathtt{zero} : \mathsf{Nat}$ and $\mathtt{succ} : \mathsf{Nat} \to \mathsf{Nat}$ is a natural numbers object.*

Proof. Let A be any type with a distinguished point $a : A$ and an endo map $f : A \to A$. We need to find a unique map

$$u : \mathsf{Nat} \to A\,,$$

making the following diagram commute in $\mathcal{C}_{\mathsf{Nat}}$.

$$\begin{array}{ccccc} 1 & \xrightarrow{a} & A & \xrightarrow{f} & A \\ \| & & \uparrow{\scriptstyle u} & & \uparrow{\scriptstyle u} \\ 1 & \xrightarrow[\mathtt{zero}]{} & \mathsf{Nat} & \xrightarrow[\mathtt{succ}]{} & \mathsf{Nat} \end{array}$$

We therefore require a term $n : \mathsf{Nat} \vdash u(n) : A$ with:

$$\begin{aligned} u(\mathtt{zero}) &\equiv a : A\,, \\ n : \mathsf{Nat} &\vdash u(\mathtt{succ}(n)) \equiv f(u(n)) : A\,. \end{aligned} \tag{3.8}$$

In the (dependent) elimination rule, we can take $C(n)$ to be A (the constant family), and c_0 to be a, and for $n : \mathsf{Nat}, x : A$ we let $c(n, x)$ be $f(x)$. In the conclusion we then obtain $n : \mathsf{Nat} \vdash \mathtt{ind}(n, a, f) : A$, which we take to be the required map $u : \mathsf{Nat} \to A$. The computation rules then clearly provide the required equations (3.8).

We then use the equality type to prove uniqueness. Namely, suppose that we also have $n : \mathsf{Nat} \vdash v(n) : A$ with

$$\begin{aligned} v(\mathtt{zero}) &\equiv a : A\,, \\ n : \mathsf{Nat} &\vdash v(\mathtt{succ}(n)) \equiv f(v(n)) : A\,. \end{aligned} \tag{3.9}$$

We wish to show $u \equiv v : \mathsf{Nat} \to \mathsf{Nat}$. By equality reflection, it suffices to show $\mathtt{Eq}_{\mathsf{Nat}\to\mathsf{Nat}}(u, v)$, and by function extensionality (3.2.1), it is therefore enough to show $\Pi_{n:\mathsf{Nat}}.\mathtt{Eq}_{\mathsf{Nat}}(u(n), v(n))$. Thus we will be done by the following dependent elimination with $C(n) = \mathtt{Eq}_{\mathsf{Nat}}(u(n), v(n))$, once we have found suitable terms e_0 and $e(n, x)$.

$$\frac{\begin{array}{l} e_0 : \mathtt{Eq}_{\mathsf{Nat}}(u(\mathtt{zero}), v(\mathtt{zero})) \\ n : \mathsf{Nat}, x : \mathtt{Eq}_{\mathsf{Nat}}(u(n), v(n)) \vdash e(n, x) : \mathtt{Eq}_{\mathsf{Nat}}\big(u(\mathtt{succ}(n)), v(\mathtt{succ}(n))\big) \end{array}}{n : \mathsf{Nat} \vdash \mathtt{ind}(n, e_0, e) : \mathtt{Eq}_{\mathsf{Nat}}(u(n), v(n))}$$

But since $u(\mathtt{zero}) \equiv a \equiv v(\mathtt{zero})$ by (3.8) and (3.9), we can take $e_0 := \mathtt{refl}$. By the same, we also have $u(\mathtt{succ}(n)) \equiv f(u(n))$ and $v(\mathtt{succ}(n)) \equiv f(v(n))$, and by the assumption $x : \mathtt{Eq}_{\mathsf{Nat}}(u(n), v(n))$ we have $u(n) \equiv v(n)$ and thus $f(u(n)) \equiv f(v(n))$. So for $e(n, x)$ we can again take a suitable $\mathtt{refl}$. □

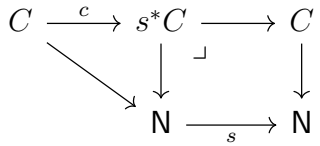
Conversely, we also have the following result:

Proposition 3.5.6. *Let $\mathcal{E}$ be an LCCC with a natural numbers object,*

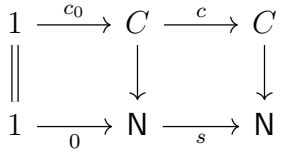
$$1 \xrightarrow{0} \mathsf{N} \xrightarrow{s} \mathsf{N}\,.$$

Then $(\mathsf{N}, 0, s)$ satisfies the Formation, Introduction, (dependent) Elimination, and (dependent) Computation rules for the type Nat.

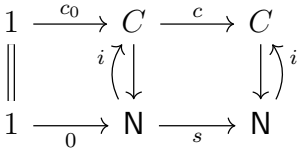
Proof. The Formation and Introduction rules are immediate. For the Elimination rule, suppose given an interpretation of the premises, namely: a dependent type $[\![\mathsf{N} \vdash C]\!] : C \to \mathsf{N}$, a section $c_0 : 1 \to 0^*C$ of the pullback of C over $0 : 1 \to \mathsf{N}$, and a map c as in the following diagram:



Then we have, equivalently, a commutative diagram:



By the universal property of N as an NNO, there is a (unique!) section $i : \mathsf{N} \to C$ commuting with the NNO structure maps.



Taking $[\![n : \mathsf{Nat} \vdash \mathtt{ind}(n, c_0, c)]\!] := i$ then satisfies the dependent Elimination and Computation rules. (Why is $i : \mathsf{N} \to C$ a *section* of $C \to \mathsf{N}$?) □

Remark 3.5.7. We shall see that, in dependent type theory, the foregoing two propositions are typical of inductive types in general: a structured type (S, s) is *initial* if and only if every type family $T \to S$ equipped with the same kind of structure (T, t) over (S, s) has a structure preserving section. We shall make this precise in terms of *algebras for (polynomial) endofunctors* in the next section.

Exercise 3.5.8. Use the dependent rules for Nat to define the addition function $+ : \mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Nat}$ in such a way that

$$m + 0 \equiv m\,,$$
$$m + \mathtt{succ}(n) \equiv \mathtt{succ}(m + n)\,.$$

Exercise 3.5.9. Formulate a (simple) η-rule for Nat that allows one to prove the analog of Proposition 3.5.5 from just the simple elimination and computation rules (including your new η-rule).

3.5.3 Algebras for endofunctors

Let $\mathcal{E}$ be an LCCC with sums, and consider the endofunctor $F : \mathcal{E} \to \mathcal{E}$ with $F(X) = 1+X$. As usual, an *algebra* for F is an object A equipped with a map $a : F(A) \to A$, and a homomorphism of F-algebras $h : (A, a) \to (B, b)$ is a map $h : A \to B$ commuting with the algebra structure maps:

$$\begin{array}{ccc} FA & \xrightarrow{Fh} & FB \\ {\scriptstyle a}\downarrow & & \downarrow{\scriptstyle b} \\ A & \xrightarrow[h]{} & B \end{array} \tag{3.10}$$

In this particular case, such an algebra $a : FA = 1 + A \to A$ corresponds to a unique "successor algebra" structure $a = [a_0, a_s]$ where:

$$1 \xrightarrow{a_0} A \xrightarrow{a_s} A\,,$$

and an F-algebra homomorphism is just a successor algebra homomorphism. It follows that a *natural numbers object* is the same thing as an *initial F-algebra*, i.e. an initial object in the category F-Alg of F-algebras and their homomorphisms.

More generally, one can consider the category of algebras for *any* endofunctor $F : \mathcal{E} \to \mathcal{E}$, but there need not always be an initial one, in light of the following fact.

Lemma 3.5.10 (Lambek)**.** *Given $F : \mathcal{E} \to \mathcal{E}$, if $i : F(I) \to I$ is an initial F-algebra, then the map i is an isomorphism.*

Exercise 3.5.11. Prove Lambek's lemma and conclude that not every endofunctor has an initial algebra.

When an initial algebra does exists, it can be regarded as a generalized "inductive type", in view of the following.

Proposition 3.5.12. *Let $F : \mathcal{E} \to \mathcal{E}$ and let $i : F(I) \to I$ be an initial F-algebra. Let $p : C \to I$ be a family over I with an F-algebra structure $c : FC \to C$ making the following diagram commute.*

$$\begin{array}{ccc} FC & \xrightarrow{c} & C \\ {\scriptstyle Fp}\downarrow & & \downarrow{\scriptstyle p} \\ FI & \xrightarrow[i]{} & I \end{array}$$

Then there is a section $s : I \to C$ that is an algebra homomorphism.

Conversely, if $a : FA \to A$ is an algebra such that every algebra $(C, c) \to (A, a)$ over it has an algebra section, then (A, a) is initial in the category F-Alg.

Proof. Since (I, i) is initial, there is an algebra homomorphism $h : (I, i) \to (C, c)$. Since homomorphisms compose, $p \circ h : I \to I$ is also one. But then $p \circ h = 1_I$ since homomorphisms from initial algebras are unique, and (I, i) is initial.

Conversely, suppose every algebra over (A, a) has an (algebra) section, and let (B, b) be any algebra. We need a (unique) algebra homomorphism $u : (A, a) \to (B, b)$. Consider the projection $p_1 : A \times B \to A$ as an algebra over (A, a) with structure map $(aFp_1,\, bFp_2) : F(A \times B) \to A \times B$.

$$\begin{array}{ccc} F(A \times B) & \xrightarrow{(aFp_1, bFp_2)} & A \times B \\ {\scriptstyle Fp_1}\downarrow & & \downarrow{\scriptstyle p_1} \\ FA & \xrightarrow[a]{} & A \end{array}$$

Let $s : A \to A \times B$ be an algebra section, so that $u := p_2 \circ s : A \to B$ is a homomorphism. We claim that u is unique. Indeed, given any homomorphism $t : A \to B$, consider the equalizer $e : E(t, u) \rightarrowtail A$, which of course is the pullback of the diagonal $\Delta_B : B \rightarrowtail B \times B$ along the map $(t, u) : A \to B \times B$. It will suffice to equip $E(t, u)$ with an algebra structure map $\varepsilon : F(E(t, u)) \to E(t, u)$ over $a : FA \to A$, for then we shall have an algebra section $s : A \to E(t, u)$, whence $e : E(t, u) \cong A$, and so $t = u$, since $e : E(t, u) \rightarrowtail A$ is the equalizer.

$$\begin{array}{ccc} F(E(t,u)) & \overset{\varepsilon}{\cdots\cdots\to} & E(t,u) \\ {\scriptstyle Fe}\downarrow & & {\scriptstyle s}\uparrow\downarrow{\scriptstyle e} \\ FA & \xrightarrow[a]{} & A \end{array}$$

We claim that $a \circ Fe : F(E(t, u)) \to FA \to A$ precomposes equally with $t, u : A \to B$, which will suffice for ε. But this follows by a chase around the following diagram, recalling that t and u are homomorphisms, and $te = ue$.

$$\begin{array}{ccc} F(E(t,u)) & \overset{\varepsilon}{\cdots\cdots\to} & E(t,u) \\ {\scriptstyle Fe}\downarrow & & \downarrow{\scriptstyle e} \\ FA & \xrightarrow[a]{} & A \\ {\scriptstyle Ft}\downarrow\;\downarrow{\scriptstyle Fu} & & {\scriptstyle t}\downarrow\;\downarrow{\scriptstyle u} \\ FB & \xrightarrow[b]{} & B \end{array}$$

□

Exercise 3.5.13. Reformulate and prove Proposition 3.5.12 in dependent type theory, using `Eq`-types for the equalizer.

Polynomial endofunctors. One class of endofunctors $F : \mathsf{Set} \to \mathsf{Set}$ for which initial algebras *do* exist are the (finitary) *polynomial functors*,

$$F(X) = C_0 + C_1 \times X + \cdots + C_n \times X^n\,, \tag{3.11}$$

where $C_0, \ldots, C_n$ are sets and $X^k = X \times \ldots \times X$ is the k-fold product. The endofunctor $1 + X$ for the natural numbers $\mathbb{N}$ was of course of this kind. An algebra (A, a) for e.g. the

functor $1 + X + X^2$ will be a *pregroup structure* on the set A,

$$[a_0, a_1, a_2] : 1 + A + A^2 \longrightarrow A\,,$$

corresponding to an element $a_0 \in A$, a unary operation $a_1 : A \to A$ and a binary operation $a_2 : A \times A \to A$. We say "pregroup structure" to emphasize that no equations are required to hold; this is just an interpretation of the "signature" of a group.

An algebra $a : F(A) \to A$ for the functor (3.11) would thus be a "conventional" algebraic structure on A (in the sense of universal algebra) consisting of C_0-many "constants" $1 \to A$ and C_1-many "unary operations" $A \to A$, ..., and C_n-many n-ary "operations" $A^n \to A$. Note that algebra homomorphisms in the sense of (3.10) are just homomorphisms in the usual sense of algebraic structures.

Proposition 3.5.14. *Any finitary polynomial functor $F : \mathsf{Set} \to \mathsf{Set}$ such as* (3.11) *has an initial algebra.*

An elementary proof would proceed by forming the "term algebra" A consisting of all expressions of the form $a_{c_k}(t_1, \ldots, t_k)$, where $c_k \in C_k$, and the t_k are "previously" formed terms of the same kind. A more abstract proof (that also generalizes to other settings) is as follows:

Proof. By [Awo10, 10.13], it suffices to show that F preserves ω-colimits, for then the colimit of the sequence

$$0 \to F0 \to FF0 \to \ldots$$

will be an initial algebra. The coproduct $C_0 + C_1 \times X + \cdots + C_n \times X^n$ preserves all of the colimits preserved by the monomials $C_k \times X^k$, and each of these preserves the colimits preserved by the functor $X^k = X \times \ldots \times X$, which includes the filtered ones like ω. □

The proof obviously generalizes to a much larger class of endofunctors, including ones on categories other than Set (see e.g. [Awo10, 10.14]). Rather than pursuing this topic further, however (for which, see [AR94]), we want to consider a type-theoretic reformulation that captures a range of inductive types with good properties. Let us first apply Proposition 3.5.12, and state the resulting rules for an initial algebra of a polynomial functor, for example:

$$F(X) = A + B \times X + C \times X^2\,,$$

as a simplified version of (3.11) (the general case will be covered below). Let us write $s : F(I) \to I$ for an initial F-algebra (assuming it exists). Then we have the following rules:

The assumption that the initial algebra exists takes the form:

- I-formation rule.

$$\frac{A\ \mathtt{type} \qquad B\ \mathtt{type} \qquad C\ \mathtt{type}}{I\ \mathtt{type}}$$

And I also will come with an algebra structure $s : F(I) \to I$, given by:

- I-introduction rules for the operation $s : A + B \times I + C \times I^2 \longrightarrow I$,

$$\frac{a : A}{s_0(a) : I} \qquad \frac{b : B\,, i : I}{s_1(b, i) : I} \qquad \frac{c : C\,, i : I\,, j : I}{s_2(c, i, j) : I}$$

We also have an induction principle, as in Proposition 3.5.12, for any F-algebra $f : F(X) \to X$ over $F(I) \to I$:

- I-elimination rule.

$$\frac{\begin{array}{l} i : I \vdash X(i)\ \mathtt{type} \\ a : A \ \vdash f_0(a) : X(s_0(a)) \\ b : B,\, i : I,\, x : X(i) \ \vdash f_1(b, x) : X(s_1(b, i)) \\ c : C,\, i, j : I,\, x : X(i),\, y : X(j) \ \vdash \ f_2(c, x, y) : X(s_2(c, i, j)) \end{array}}{i : I \vdash \mathtt{rec}(i, f_0, f_1, f_2) : X(i)}$$

And, of course, there is a computation rule, resulting from first introducing and then eliminating, which says that the following diagram commutes.

$$\begin{array}{ccc} F(X) & \xrightarrow{\quad f \quad} & X \\ {\scriptstyle \mathtt{rec}}\uparrow\downarrow & & \downarrow\uparrow{\scriptstyle \mathtt{rec}} \\ FI & \xrightarrow[\quad s \quad]{} & I \end{array}$$

- I-computation rules.

$$\begin{aligned} a : A \ &\vdash \mathtt{rec}(s_0(a), f_0, f_1, f_2) \ \equiv \ f_0(a) : X(s(a))\,, \\ b : B,\, i : I \ &\vdash \mathtt{rec}(s_1(b, i), f_0, f_1, f_2) \ \equiv \ f_1(b, \mathtt{rec}(i, \vec{f}\,)) : X(s_1(b, i))\,, \\ c : C,\, i, j : I \ &\vdash \mathtt{rec}(s_2(c, i, j), f_0, f_1, f_2) \ \equiv \ f_2(c, \mathtt{rec}(i, \vec{f}\,), \mathtt{rec}(j, \vec{f}\,)) : X(s_2(c, i, j))\,. \end{aligned}$$

Exercise 3.5.15. Specialize the foregoing to the case $F(X) = 1 + X$ and derive the rules for Nat from Section 3.5.2.

3.5.4 W-types

The rules just given for initial algebras for polynomial functors are a bit unwieldy as the degree of the polynomial F increases, but they simplify when stated in a more general form, which can actually be applied in any LCCC $\mathcal{E}$. Indeed, let $p : B \to A$ be any map in $\mathcal{E}$, regarded as a type family $a : A \vdash B(a)$. We can form the (generalized) polynomial endofunctor $P : \mathcal{E} \to \mathcal{E}$ as:

$$P(X) = \Sigma_{a:A} X^{B(a)} \ = \ A_! \circ p_* \circ B^*(X)\,, \tag{3.12}$$

as indicated in the following:

$$\begin{array}{ccccc} X & \longleftarrow & X \times B & & P(X) \\ & & \downarrow & & \downarrow \\ & & B & \xrightarrow[p]{} & A \end{array}$$

Observe that $B^* = p^* \circ A^*$, so that

$$\Sigma_A \, p_* \, B^* \, (X) = \Sigma_A \, p_* \, p^* A^*(X) = \Sigma_A (A^* X)^p \, ,$$

which justifies the polynomial notation $\Sigma_{a:A} X^{B(a)}$, since the type $B(a)$ is the fiber of $p : B \to A$ at $a : A$.

Definition 3.5.16 (cf. [MP00])**.** A (semantic) W-*type* in a locally cartesian closed category $\mathcal{E}$ is an initial algebra for a polynomial endofunctor $P : \mathcal{E} \to \mathcal{E}$ associated to a map $p : B \to A$, as in (3.12).

We shall see that W-types can be used to introduce a wide class of inductive types in dependent type theory. The following rules for W-types are due to [**?**]. To state them more perspicuously, for a fixed type family $x : A \vdash B(x)$ we may write W instead of $\mathsf{W}_{x:A} B(x)$.

- W-formation rule.

$$\frac{A \ \mathtt{type} \qquad x : A \vdash B(x) \ \mathtt{type}}{\mathsf{W}_{x:A} B(x) \ \mathtt{type}}$$

- W-introduction rule.

$$\frac{a : A \qquad t : B(a) \to \mathsf{W}}{\mathtt{wsup}(a, t) : \mathsf{W}}$$

- W-elimination rule.

$$\frac{\begin{array}{l} w : \mathsf{W} \vdash C(w) \ \mathtt{type} \\ x : A, \ u : B(x) \to \mathsf{W}, \ v : \Pi_{y:B(x)} C(u(y)) \ \vdash \ c(x, u, v) : C(\mathtt{wsup}(x, u)) \end{array}}{w : \mathsf{W} \vdash \mathtt{wind}(w, c) : C(w)}$$

- W-computation rule.

$$\frac{\begin{array}{l} w : \mathsf{W} \vdash C(w) \ \mathtt{type} \\ x : A, \ u : B(x) \to \mathsf{W}, \ v : \Pi_{y:B(x)} C(u(y)) \ \vdash \ c(x, u, v) : C(\mathtt{wsup}(x, u)) \end{array}}{\begin{array}{l} x : A, \ u : B(x) \to \mathsf{W} \vdash \mathtt{wind}(\mathtt{wsup}(x, u), c) \ \equiv \\ \qquad\qquad c(x, u, \lambda y. \mathtt{wind}(u(y), c)) : C(\mathtt{wsup}(x, u)) \end{array}}$$

Informally, the W-type for a family $x : A \vdash B(x)$ can be regarded as the free algebra for a signature with A-many operations, each of (possibly infinite) arity $B(a)$ – and *no* equations. Indeed, the premisses of the formation rule above can be thought of as specifying a signature that has the terms $a : A$ as the operations themselves, and (the cardinality of) the type $B(a)$ as the "arity" of $a : A$. Then, the introduction rule specifies an element of the free algebra, namely $\mathtt{wsup}(a, t) : \mathsf{W}$, where $t : B(a) \to \mathsf{W}$. The elimination rule then states that W is the *initial algebra* of the assocated polynomial functor $\Sigma_{a:A} X^{B(a)}$.

Proposition 3.5.17. *An object* W *satisfies the rules for* W*-types if, and only if, it is an initial algebra for the polynomial functor* $P(X) = \Sigma_{a:A} X^{B(a)}$.

Proof. We use Proposition 3.5.12. Suppose W satisfies the rules for W types above. From the introduction rule, we have a P-algebra structure $\mathtt{wsup} : P(\mathsf{W}) \to \mathsf{W}$. Let $C \to \mathsf{W}$ with a P-algebra structure $c : P(C) \to C$ over $\mathtt{wsup} : P(\mathsf{W}) \to \mathsf{W}$. This is exactly what the premises of the elimination rule say, so by the conclusion of that rule there is a section $w : \mathsf{W} \vdash \mathtt{wind}(w, c) : C(w)$, which is a P-algebra homomorphism by the computation rule. The converse is left as an exercise. □

Exercise 3.5.18. Complete the proof of Proposition 3.5.17.

Remark 3.5.19 (cf. [AGS17])**.** The foregoing (dependent) W-elimination rule implies what may be called the *simple* W-elimination rule:

$$\frac{C\ \mathtt{type} \qquad x : A, v : B(x) \to C \vdash c(x, v) : C}{w : \mathsf{W} \vdash \mathtt{wrec}(w, c) : C}$$

This can be recognized as a recursion principle for maps from W into P-algebras, since the premisses of the rule describe exactly a type C equipped with a structure map $c : PC \to C$. For this special case of the elimination rule, the corresponding computation rule again states that the function

$$\lambda w.\mathtt{wrec}(w, c) : \mathsf{W} \to C\,,$$

where $c(x, v) = c(\langle x, v\rangle)$ for $x : A$ and $v : B(x) \to C$, is a P-algebra homomorphism. Moreover, this homomorphism can then be shown to be (definitionally) *unique*, using $\mathtt{Eq}$-types, the elimination rule, and the reflection rule, as in the proof of Proposition 3.5.12. The converse implication also holds: one can derive the general W-elimination rule from the simple elimination rule and the following η-rule.

$$\frac{\begin{array}{l} C : \mathtt{type} \qquad w : \mathsf{W} \vdash h(w) : C \\ x : A, v : B(x) \to C \vdash c(x, v) : C \\ x : A\,, u : B(x) \to W \vdash h\,(\mathtt{wsup}(x, u)) = c(x, \lambda y.hu(y)) : C \end{array}}{w : \mathsf{W} \vdash h(w) \equiv \mathtt{wrec}(w, c) : C}$$

This rule states the uniqueness of the $\mathtt{wrec}$ term among algebra maps. Overall, we therefore have that induction and recursion are inter-derivable in the present theory with extensional

Eq-types:

Induction	$\Leftrightarrow$	**Recursion**
Dependent elimination		Simple elimination
Dependent computation		Simple computation + η-rule

Examples of W-types. We conclude by noting that many familiar inductive types can be reduced to W-types. We mention the following examples, among many others (see [**?**], [**?**], [**?**], [**?**], [**?**], [**?**]):

1. *Natural numbers.* The usual rules for Nat as an inductive type can be derived from its formalization as a W-type. Consider the signature determined by the map $\mathtt{inl} : 1 \to 1 + 1$ (say): it has two operations, one of which has arity 0 and one of which has arity 1, since these are the pullbacks of the map $\mathtt{inl} : 1 \to 1 + 1$ along the two points $1 \rightrightarrows 1 + 1$. To present this in type theory, we need a type family over Bool (say) with the types 0 and 1 as its values. Consider the family

$$v : \mathtt{Bool} \vdash \mathtt{Eq}_{\mathtt{Bool}}(v, \mathtt{true}) \ \ \mathtt{type},$$

which has the values:

$$\mathtt{Eq}_{\mathtt{Bool}}(v, \mathtt{true}) \cong \begin{cases} 1 & v = \mathtt{true} \\ 0 & v = \mathtt{false} \end{cases}$$

Indeed, one can show that the projection $\Sigma_{v:\mathtt{Bool}}\mathtt{Eq}_{\mathtt{Bool}}(v, \mathtt{true}) \to \mathtt{Bool}$ is isomorphic to the map $\mathtt{true} : 1 \to \mathtt{Bool}$ over Bool.

The corresponding polynomial functor can then be defined as

$$\begin{aligned} P(X) &= \Sigma_{v:\mathtt{Bool}}\, \mathtt{Eq}_{\mathtt{Bool}}(v, \mathtt{true}) \to X \\ &= 1 + X\,. \end{aligned}$$

The corresponding W type is then the initial algebra of $P(X) = 1 + X$, namely the type Nat of natural numbers,

$$\mathsf{Nat} = \mathsf{W}_{v:\mathtt{Bool}}\, \mathtt{Eq}_{\mathtt{Bool}}(v, \mathtt{true})\,.$$

The canonical element $\mathtt{zero} : \mathsf{Nat}$ and the successor function $\mathtt{succ} : \mathsf{Nat} \to \mathsf{Nat}$ result from the two cases of introduction rule,

$$\frac{b : \mathtt{Bool} \qquad t : \mathtt{Eq}_{\mathtt{Bool}}(b, \mathtt{true}) \to \mathsf{Nat}}{\mathtt{wsup}(b, t) : \mathsf{Nat}}$$

namely:

$$\frac{\mathtt{false} : \mathtt{Bool} \qquad t : (\mathtt{Eq}_{\mathtt{Bool}}(\mathtt{false}, \mathtt{true}) \to \mathsf{Nat})}{\mathtt{wsup}(\mathtt{false}, t) : \mathsf{Nat}}$$

and

$$\frac{\mathtt{true} : \mathsf{Bool} \qquad t : (\mathsf{Eq}_{\mathsf{Bool}}(\mathtt{true}, \mathtt{true}) \to \mathsf{Nat})}{\mathtt{wsup}(\mathtt{true}, t) : \mathsf{Nat}}$$

Thus we can take

$$\begin{aligned} \mathtt{zero} &:= \lambda t.\, \mathtt{wsup}(\mathtt{false}, t) : 1 \to \mathsf{Nat}\,, \\ \mathtt{succ} &:= \lambda t.\, \mathtt{wsup}(\mathtt{true}, t) : \mathsf{Nat} \to \mathsf{Nat}\,. \end{aligned}$$

2. *Second number class.* As shown in [**?**], the second number class can be obtained as a W-type determined by the polynomial functor

$$P(X) = \mathsf{1} + X + (\mathsf{Nat} \to X)\,.$$

This has algebras with three operations, one of arity 0, one of arity 1, and one of arity (the cardinality of) Nat.

3. *Lists.* The type $\mathsf{List}(A)$ of finite lists of elements of some type A can be built as a W-type determined by the polynomial functor

$$P(X) = \mathsf{1} + A \times X\,,$$

associated to the map $! + A : 0 + A \to 1 + 1$. We refer to [**?**] for details.

Exercise 3.5.20. If a signature for an algebraic theory has no constants, then the free algebra on the empty set 0 will itself be empty, as can be seen by considering the term algebra construction of the free algebra $F(0)$. Something similar is true for W-types (say, in Set): if $p : B \twoheadrightarrow A$ is an epimorphism, then $P(0) = \Sigma_{a:A}\, 0^{B(a)} \cong 0$, and so $\mathsf{W}_{a:A}\, B(a) \cong 0$. Prove this.

3.6 Propositional truncation

Even under the Propositions-as-Types (PaT) conception there are certain types P that are *proof irrelevant* in the sense that for any $p, q : P$, we have $\mathtt{Eq}_P(p, q)$ (meaning that we have a term $t : \mathtt{Eq}_P(p, q)$, and so $p \equiv q$). For example, the type 1 has this property, as does 0. Let us call this such special types *propositional*, which is *definable* by

$$\mathsf{IsProp}(P) \;=\; \Pi_{p,q:P} \mathtt{Eq}_P(p, q)\,.$$

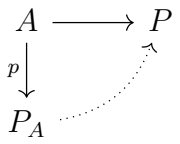
This condition is equivalent to $P \cong P \times P$.

Exercise 3.6.1. Prove this: a type P is propositional if and only if $P \cong P \times P$ (canonically) and, moreover, if and only if the unique map $P \to 1$ is a monomorphism.

The propositions are easily seen to be closed under finite products $P \times Q$, and if $x : X \vdash P(x)$ is a family of propositions, then $\Pi_{x:X} A(x)$ is also a proposition. Finally, if P is a proposition, then so is $A \to P$ for any A.

Exercise 3.6.2. Prove the last three statements.

It therefore makes sense to expect that for any type A, there could be a universal propositional approximation $p : A \to P_A$, with the universal property that every map $A \to P$ with P a proposition factors (uniquely) through $p : A \to P_A$, as in:



This is equivalent to saying that for any proposition P, the map $P^p : P^{P_A} \to P^A$ induced by precomposing with $p : A \to P_A$ is an iso. When it exists, we shall call such an object a *propositional truncation* of A, and denote it by $A \to [A]$.

Definition 3.6.3. Given a type A, a *propositional truncation* of A is a type $[A]$ equipped with a map $A \to [A]$ such that, for any proposition P, the canonical precomposition map

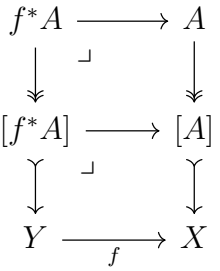
$$P^{[A]} \to P^A$$

is an isomorphism.

Example 3.6.4. In a category of presheaves $\mathsf{Set}^{\mathbb{C}^{\mathrm{op}}}$ the propositions are exactly the subobjects of 1, by Exercise 3.6.1. But since every map $A \to B$ in presheaves can be factored into an epi followed by a mono $A \twoheadrightarrow M \rightarrowtail B$, every object A has a propositional truncation $A \twoheadrightarrow [A] \rightarrowtail 1$. Moreover, since these factorizations are stable under pullback ($\widehat{\mathbb{C}}$ is regular), the propositional truncation operation $[A]$ commutes with pullback, in the sense that for $B \to 1$ we have $B^*[A] \cong [B^*A]$. More generally, for any $f : Y \to X$ and any $A \to X$, we have

$$f^*[A] \cong [f^*A]\,,$$

as in the following diagram,



as is seen by applying the foregoing remark to the presheaf category $\widehat{\mathbb{C}}/_X \;\cong\; \widehat{\int_{\mathbb{C}} X}$.

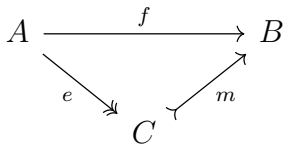
The "stability under pullback" of the operation $A \mapsto [A]$ means that it can be added to dependent type theory as a new type former, because it exists in any context and commutes with substitution. We shall formulate rules for $[A]$ in the next section 3.6.1. The propositional truncation $[A]$ of a type A may be regarded as "erasing (or ignoring)

the (computational) content" of A and treating it as a "mere truth-value": either A is inhabited or not, and all inhabitants $a : A$ are identified.

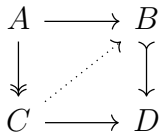
The reg-epi/mono factorizations in a regular category fit into an orthogonal factorization system in the following sense.

Definition 3.6.5. An *orthogonal factorization system* on a category $\mathcal{C}$ consists of two classes of arrows (E, M) such that:

1. The classes of maps $\mathsf{E}, \mathsf{M} \subseteq \mathcal{C}_1$ are closed under isos in the arrow category.
2. Every map $f : A \to B$ factors $f = m \circ e$ into $e \in \mathsf{E}$ followed by $m \in \mathsf{M}$,



3. Given any commutative square with an E-map on the left and an M-map on the right,



there is a unique *diagonal filler* (as indicated) making both triangles commute.

Exercise 3.6.6. Show that the epimorphisms and monomorphisms form an orthogonal factorization system on the category Set. Infer that the same is true for any presheaf category $\mathsf{Set}^{\mathbb{C}^{\mathrm{op}}}$.

3.6.1 Bracket types

The rules for *bracket types* $[A]$ in dependent type theory are as follows (cf. [**?**, **?**]):

- Formation rule.

$$\frac{A\ \mathtt{type}}{[A]\ \mathtt{type}}$$

- Introduction rules.

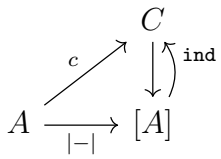
$$\frac{a : A}{|a| : [A]} \qquad \frac{a : [A],\ b : [A]}{\mathsf{eq}(a,b) : \mathtt{Eq}_{[A]}(a,b)}$$

- Elimination rule.

$$\frac{\begin{array}{l} z : [A] \vdash C(z)\ \mathtt{type} \qquad x : A \vdash c(x) : C(|x|) \\ x : A,\ u : C(|x|),\ v : C(|x|) \vdash\ p(x,u,v) : \mathtt{Eq}_{C(|x|)}(u,v) \end{array}}{z : [A] \vdash \mathtt{ind}(z,c,p) : C(z)}$$

- Computation rule.
$$x : A \vdash \mathtt{ind}(|x|, c, p) \equiv c(x) : C(|x|)$$

The Computation rule can be understood semantically as stating that, when it exists, the section $z : [A] \vdash \mathtt{ind}(z, c, p) : C(z)$ makes the following diagram commute.



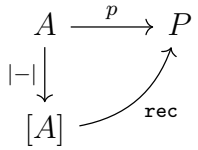
And the Elimination rule states that such a section exists if any $u, v : C(|x|)$ are always equal, so that $C(z)$ is a family of propositions.

The simple rules are perhaps easier to understand:

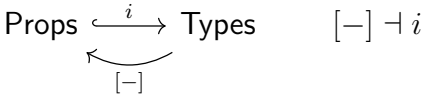
- Simple Elimination rule.
$$\frac{P\ \mathtt{type} \qquad x : A \vdash p(x) : P \qquad u : P,\ v : P \vdash\ q(u, v) : \mathtt{Eq}_P(u, v)}{z : [A] \vdash \mathtt{rec}(z, p, q) : P}$$

- Simple Computation rule.
$$x : A \vdash \mathtt{rec}(|x|, p, q) \equiv p(x) : P$$

The Simple Computation rule states that, when it exists, the map $z : [A] \vdash \mathsf{rec}(z, p, q) : P$ makes the following diagram commute.



And by the Simple Elimination rule, such a map $z : [A] \vdash \mathsf{rec}(z, p, q) : P$ exists whenever P is a proposition. Taken together with the Introduction rule, which says that $[A]$ is always a proposition, this clearly states that the inclusion $i : \mathsf{Props} \hookrightarrow \mathsf{Types}$ of the propositions into the types has the propositional truncation operation $[-]$ as a left adjoint.



Exercise 3.6.7. Prove the adjointness between the inclusion of propositions into types and the propositional truncation operation. Why doesn't one need to add an η computation rule to the simple Elimination rule to get the uniqueness of the eliminator required for the adjunction?

3.6.2 Propositions as [types]

In dependent type theory with the type-forming operations

$$0,\ 1,\ [A],\ A+B,\ \mathtt{Eq}_A,\ \Pi_{x:A}B,\ \Sigma_{x:A}B$$

the propositions in every context model *intuitionistic first-order logic* (IFOL), under the following definitions:

$$\begin{aligned} \top &= 1 \\ \bot &= 0 \\ \phi \wedge \psi &= \phi \times \psi \\ \phi \vee \psi &= [\phi + \psi] \\ \phi \Rightarrow \psi &= \phi \to \psi \\ \neg\phi &= \phi \to 0 \\ \forall x : A.\, \phi &= \Pi_{x:A}\, \phi \\ \exists x : A.\, \phi &= [\Sigma_{x:A}\, \phi] \\ x = y &= \mathtt{Eq}_A(x,y) \end{aligned} \tag{3.12}$$

The bracket is thus used to "rectify" the operations $+$ and Σ, because they lead out of propositions. The operations defined in (3.12) satisfy the usual rules for intuitionistic first-order logic, and the resulting system is then a hybrid of dependent type theory with first-order logic over each type. It can be described categorically as the internal logic of a regular LCCC with finite sums. This formulation is equivalent to the more customary one using both type theory and predicate logic (such as in [**?**]), despite the fact that the first-order logical operations on the propositions are here *defined* in terms of the type-theoretic operations on types, rather than being taken as primitive.

In addition to first-order logic, one can also use brackets to define *subset types*. For any type $\Gamma, x : A \vdash B$, the associated subset type

$$\Gamma \vdash \{x : A \,|\, B\}$$

is defined by

$$\{x : A \,|\, B\} = \Sigma_{x:A}\, [B(x)]\,.$$

Semantically, this can therefore be modeled by the image factorization:

$$\begin{array}{ccc} \Sigma_{x:A}\, B(x) & = & B \\ \downarrow & & \downarrow \\ \{x : A \,|\, B\} & = & [B] \\ \downarrow & & \downarrow \\ A & = & A \end{array}$$

This hybrid logic of dependent types with first-order logic over each type is a very expressive system, and is used in many modern settings [Uni13, **?**, **?**].

3.6.3 Completeness of propositions as types

We can use bracket types to compare conventional first-order logic with the propositions-as-types interpretation, and relate first-order provability to provability in dependent type theory (without brackets). Consider the standard propositions-as-types translation of first-order logic into type theory:

$$* : \mathsf{FOL} \to \mathsf{DTT}$$

For a single-sorted first-order theory $\mathbb{T}$, consisting of constants, function and relation symbols, the $*$-translation is determined by first fixing the translations of the basic sort of individuals, and the constants, function and relation symbols. The rest of the translation is then determined inductively in the expected way, using the type-forming operations in place of the corresponding logical ones.

For example,

$$(\forall x\, \exists y.\, R(x,y) \vee P(x))^* \;=\; \Pi_{x:I}\, \Sigma_{y:I}\, R^*(x,y) + P^*(x)\,,$$

where I is a basic type representing the domain of "individuals" (which is usually implicit in FOL), and the dependent types $x : I \vdash P^*(x)$ and $x : I, y : I \vdash R^*(x,y)$ interpret the relation symbols P and R.

If we then add a constant $a : \alpha^*$ for each axiom α, the translation ϕ^* of a provable, closed formula ϕ is then inhabited by a closed term, that is obtained from a straightforward translation of the IFOL proof of ϕ into type theory (recalling that the rules of type theory are proof-relevant versions of those of IFOL).

Thus we have:

$$\mathsf{IFOL}(\mathbb{T}) \vdash \phi \quad \text{implies} \quad \mathsf{DTT}(\mathbb{T}) \vdash \phi^*, \tag{3.13}$$

where by $\mathsf{DTT}(\mathbb{T}) \vdash \phi^*$ we mean that the type ϕ^* is inhabited in dependent type theory enriched with the basic types and constants needed for the translation $*$, and with constants inhabiting the translations of axioms of $\mathbb{T}$. We emphasize that we are *not* using the defined operations (3.12), but rather the standard "PAT" ones.

The question we want to consider now is the converse implication: if ϕ^* is "PAT-provable", i.e. inhabited in $\mathsf{DTT}(\mathbb{T})$, must ϕ be provable in the intuitionistic first-order theory $\mathbb{T}$? Note that functions of higher types may be used in a term inhabiting ϕ^*, so this is not merely a matter of tracing out proofs in first-order logic from those in DTT, as was the converse case.

Proofs of partial converses of (3.13), for different fragments of first-order logic, have been given by Martin-Löf ($\forall$, $\Rightarrow$ in [**?**]), Tait ($\forall$, $\to$, $\exists$, $\wedge$, $\neg$ in [**?**]), and Constable ($\forall$, $\to$, $\exists$, $\wedge$, $\vee$, $\neg$ in [**?**]). These results are for type theory with no equality types, and proceed from proofs of normalization. We state a result below that applies to type theory with (extensional) equality and a large fragment of first-order logic. The proof uses the bracket types translation.

Definition 3.6.8. A formula ϑ in first-order logic with equality is *stable* when it contains neither $\forall$ nor $\Rightarrow$, but negation $\neg$ is allowed as a special case of $\Rightarrow$. A first-order formula ϕ is *left-stable* when in every subformula of the form $\vartheta \Rightarrow \psi$, the formula ϑ is stable.

Theorem 3.6.9. *If ϕ is left-stable, then*

$$\mathsf{DTT}(\mathbb{T}) \vdash \phi^* \quad \textit{implies} \quad \mathsf{IFOL}(\mathbb{T}) \vdash \phi\,.$$

For the proof, see [**?**].

Finally, observe that every first-order formula ϕ is *classically* equivalent to one ϕ^s that is stable. The *stabilization* ϕ^s of ϕ is obtained by replacing in ϕ every $\forall x.\,\vartheta$ and $\vartheta \Rightarrow \psi$ by $\neg\exists x.\,\neg\vartheta$ and $\neg(\vartheta \wedge \neg\psi)$, respectively. The equivalence $\phi \Leftrightarrow \phi^s$ holds intuitionistically if $\phi = \psi^{\neg\neg}$ is the double-negation translation of a formula ψ. Therefore, the stabilized double-negation translation

$$(\phi^{\neg\neg})^s$$

takes a formula ϕ of first-order logic with equality to a stable one, for which provability in IFOL is equivalent to classical provability (in CFOL),

$$\mathsf{CFOL} \vdash \phi \quad \text{if and only if} \quad \mathsf{IFOL} \vdash (\phi^{\neg\neg})^s\,.$$

If we further compose the $(\neg\neg\mathrm{s})$-translation with the propositions-as-types translation $*$, we obtain a translation

$$\phi^+ \;:=\; ((\phi^{\neg\neg})^s)^*$$

which takes formulas of (classical) first-order logic into dependent type theory, in such a way that every formula ϕ is classically equivalent to one covered by Theorem 3.6.9.

Corollary 3.6.10. *The translation $\phi \mapsto \phi^+$ of first-order logic with equality into dependent type theory is sound and complete, in the sense that for every formula ϕ,*

$$\mathsf{CFOL} \vdash \phi \quad \textit{if and only if} \quad \mathsf{DTT} \vdash \phi^+\,.$$

Here $\mathsf{DTT} \vdash \phi^+$ means that the type ϕ^+ is inhabited.

Remark 3.6.11 (Thierry Coquand)**.** The following formula is not provable in intuitionistic first-order logic, but its $*$-translation is inhabited in dependent type theory, by an application of the axiom of choice.

$$(\forall x\,\exists y.\,R(x,y)) \Rightarrow \forall x, x'\,\exists y, y'.\,(R(x,y) \wedge R(x',y') \wedge (x = x' \Rightarrow y = y'))\,.$$

Theorem 3.6.9 therefore cannot be extended to full intuitionistic first-order logic.

Chapter 4

Homotopy Type Theory

The extensional dependent type theory of the previous chapter is in some ways a very natural system that admits an intuitively clear model in the locally cartesian closed category of sets and related categories. But for computational purposes, and specifically for the important application of type theory to proof checking in a computer proof assistant such as Agda or Lean, it has some serious defects: the equality relation between terms (or types) is not *decidable*: there is no algorithm that will determine whether two closed terms of a given type $s, t : A$ are (judgementally) equal $s \equiv t : A$. Indeed, there is no normalization procedure for reducing terms to normal forms—otherwise we could use it to decide whether two terms were equal by normalizing them and then comparing their normal forms. Relatedly, one cannot effectively decide whether a given type (e.g an equality type such as $\mathtt{Eq}_A(s,t)$) is inhabited (which would be a decision procedure for the *provability* of $s =_A t$), even given a candidate "proof term" $p : \mathtt{Eq}_A(s,t)$ (which would be a decision procedure for *being a proof*).

For this reason, the extensional system is often replaced in applications by a weaker one, called *intensional type theory*, which enjoys better computational behavior, such as decidability of equality and type-checking, and normalization. A good discussion of these and several related issues, such as canonicity and consistency can be found in Chapter 3 of the book [AG].

However, this is only one side of the story. The intensional theory was mainly a technical device for specialists in computational type theory (and a conceptual challenge from the semantic point of view) until around 2006, when it was discovered that this theory admitted a homotopical (and higher-categorical) interpretation, which led to the discovery of Homotopy type theory (HoTT) [Awo12]. This interpretation not only helped to clarify the intensional theory, and prove useful in investigating its computational properties, but also opened up a wide range of applications outside of the conventional areas of type theory, *vis.* computational and constructive mathematics. For, quite independently of such applications, the homotopical interpretation permits the use of intensional type theory as a powerful and expressive *internal language* for formal reasoning in homotopy theory and higher category theory, both highly abstract areas of mathematics, for which new and rigorous tools for calculation and proof are quite welcome. Moreover, the fortuitous fact that

this system also has the good computational behavior that it does has led to the use of computational proof assistants in homotopy theory and higher category theory, even ahead of some more down-to-earth branches of mathematics, where such exotic semantics were not needed.

The homotopical interpretation was already anticipated by a 2-dimensional one in the category of groupoids, a special case of a higher categorical model that already suffices to make some of the essential features of such models clear. Thus we shall briefly review this model below, after introducing the intensional theory, and before considering the general homotopical semantics using weak factorization systems. Such "weak" interpretations also bring to a head the coherence issues that we deferred in the previous chapter, and we conclude with one approach to strictifying such interpretations using natural models, aka, categories with families.

4.1 Identity types

We begin by recalling from Section 3.2 the rules for *equality* types in the *extensional* system: The formation, introduction, elimination, and computation rules for equality types were as follows:

$$\frac{s : A \qquad t : A}{s =_A t \ \ \mathtt{type}} \qquad\qquad \frac{a : A}{\mathtt{refl}(a) : (a =_A a)}$$

$$\frac{p : s =_A t}{s \equiv t : A} \qquad\qquad \frac{p : s =_A t}{p \equiv \mathtt{refl}(s) : (s =_A s)}$$

The *Identity types* in the intensional theory, also written $x =_A y$, or sometimes $\mathtt{Id}_A(x, y)$, have the same formation and introduction rules as the Equality types, but the *elimination rule* of "equality reflection" is replaced by the following elimination rule:

$$\frac{x : A, y : A, z : \mathtt{Id}_A(x, y) \vdash C(x, y, z) \ \mathtt{type}, \qquad x : A \vdash c(x) : C(x, x, \mathtt{refl}(x))}{x : A, y : A, z : \mathtt{Id}_A(x, y) \vdash \mathtt{J}(x, y, z, c) : C}$$

in which the variable x is bound in the occurance of c within the eliminator $\mathtt{J}$. The associated *computation rule* then becomes:

$$x : A \vdash \mathtt{J}(x, x, \mathtt{refl}(x), c) \equiv c(x) : C(x, x, \mathtt{refl}(x))$$

In HoTT, the elimination rule is called *path induction*, for reasons that will become clear.

To see how the elimination rule works, let us derive the basic laws of identity, namely reflexivity, symmetry, and transitivity, as well as Leibniz's Law the *indicernibility of identicals*, also known as the substitution of equals for equals.

- Reflexivity: states that $x =_A y$ is a reflexive relation, but this is just the $\mathtt{Id}$-formation and intro rules:

$$x : A, y : A \vdash x =_A y \ \ \mathtt{type}\ , \qquad x : A \vdash \mathtt{refl}(x) : x =_A x$$

- Symmetry: can be stated as $x : A, y : A, u : x =_A y \vdash ? : y =_A x$, which can be proved with an `Id`-elim as follows:

$$\frac{x : A \vdash \texttt{refl}(x) : x =_A x}{x : A, y : A, u : x =_A y \vdash \texttt{J}(x, y, u, \texttt{refl}) : y =_A x}$$

- Transitivity: we wish to show

$$x : A, y : A, z : A, u : x =_A y, v : y =_A z \vdash ? : x =_A z$$

  regarding $z : A$ as a fixed parameter, which we can move to the front of the context, we want to apply an `Id`-elim with respect to the assumption $u : x =_A y$, so we can set x to y, and look for a premiss of the form:

$$z : A, y : A, v : y =_A z \vdash ? : y =_A z$$

  We cannot simply take v, however, since the order of the types in the context is still wrong for `Id`-elim, but we can move the assumption $v : y =_A z$ to the right with a λ-abstraction to obtain

$$z : A, y : A \vdash \ \lambda v\,.v : y =_A z \to y =_A z\,,$$

  and now we *can* apply the planned `Id`-elim with respect to $u : x =_A y$ with the "motive" being $y =_A z \to x =_A z$ to obtain

$$z : A, y : A, x : A, u : x =_A y \vdash \texttt{J}(x, y, u, \lambda v.v) : y =_A z \to x =_A z$$

  from which follows the desired

$$x : A, y : A, z : A, u : x =_A y, v : y =_A z \vdash \texttt{J}(x, y, u, \lambda v.v)\, v : x =_A z\,.$$

- Substitution: to show

$$\frac{x : A \vdash C(x) \ \texttt{type}}{x : A, y : A, u : x =_A y \vdash ? : C(x) \to C(y)}$$

  it suffices to have a premiss of the form

$$x : A \vdash c(x) : C(x) \to C(x)$$

  for this, we can take $c(x) = \lambda z : C(x).\, z : C(x) \to C(x)$ to obtain

$$x : A, y : A, u : x =_A y \vdash \texttt{J}(x, y, u, x.\lambda z : C(x).z) : C(x) \to C(y)\,.$$

  Note that the variable x is bound in the `J` term.

Many more properties of Id-types and their associated J-terms are shown in the introductory texts [**?**, Rij25]. One key fact is that the higher identity types $\mathtt{Id}_{\mathtt{Id}_A(a,b)}(p,q)$ are no longer degenerate, but themselves may have terms that are non-identical, i.e. not propositionally equal, leading to so-called *higher types*. This "failure of UIP" (uniqueness of identity proofs) in the intensional system was first shown using the groupoid model, which sheds considerable light on the intensional system.

Exercise 4.1.1. Show that given any $a, b, c : A$ and $p : a =_A b$ and $q : b =_A c$, one can define a composite $p \cdot q : a =_A c$ (using the transitivity of $=_A$). Then show that, for any $p : a =_A b$, the symmetry term $\sigma(p) : b =_A a$ satisfies the (propositional) equation $\sigma(p) \cdot p = \mathtt{refl}$. Is either of $\sigma(p) \cdot p = \mathtt{refl}$ or $\mathtt{refl} \cdot \sigma(p) = \mathtt{refl}$ judgemental? What about associativity of $p \cdot q$

Exercise 4.1.2. Show that $p \cdot q$ from the previous exercise is (propositionally) associative.

Exercise 4.1.3. Show that any term $f : A \to B$ acts on identities $p : a =_A b$, in the sense that there is a term $\mathtt{ap}(f)(p) : fa =_B fb$. Is $\mathtt{ap}(f)$ "functorial" (in the evident sense)?

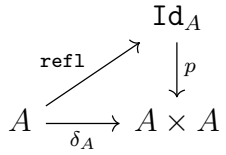
Exercise 4.1.4. Observe that the Substitution property means that the assignment

$$(a : A) \mapsto C(a) \ \mathtt{type}$$

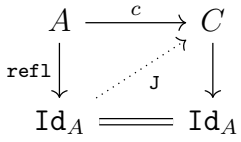
is functorial (in some sense). Is it strictly functorial?

4.1.1 The naive interpretation

We can try to interpret the (intensional) identity types in the naive way, as we did for extensional dependent type theory. This would give the formation and introduction rules as a type family $\mathtt{Id}_A \to A \times A$ with a partial section over the diagonal substitution $\delta_A : (x : A) \to (x : A, y : A)$.



where we are writing $\mathtt{Id}_A$ for the extended context $(A, A, \mathtt{Id}_A)$ and p for the dependent family $(x : A, y : A \vdash \mathtt{Id}_A(x, y))$. The elimination rule then takes the form:



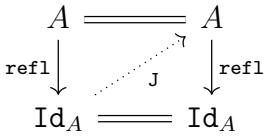
(4.1)

for any type family $C \to \mathtt{Id}_A$, with the computation rule asserting that the top triangle commutes (the bottom triangle commutes by the assumption tht J is a section of $C \to \mathtt{Id}_A$).

But now recall that in extensional type theory, *any* map $f : B \to A$ can be regarded as a type family over A, namely by taking the *graph factorization*

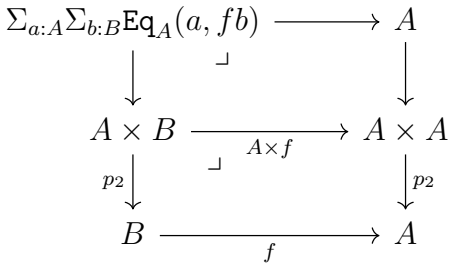
$$B \cong \Sigma_{a:A}\Sigma_{b:B}\mathtt{Eq}_A(a, fb) \longrightarrow B \times A.$$

So we can take the family C in the elimination to be $\mathtt{refl} : A \to \mathtt{Id}_A$, to obtain:



We therefore get an iso $A \cong \mathtt{Id}_A$, making the identity type isomorphic to the extensional equality type $\mathtt{Eq}_A = A \to A \times A$.

Exercise 4.1.5. Prove that in the extensional theory, the graph factorization does indeed make any map $f : B \to A$ isomorphic to a family of types over its codomain. *Hint:* Consider the following two-pullback diagram.



4.2 The groupoid model

Exercises 4.1.1 – 4.1.4 from the last section suggest an interpretation of the intensional version of dependent type theory, namely with types as *groupoids* and type families as *functors*. Such an interpretation was first given by [HS98] in order to show that the principle of Uniqueness of Identity Proofs (UIP) – which holds in the extensional theory – indeed fails in the intensional one. We shall briefly sketch this result here.

In order to give a model of intensional type theory we should define what it means to be *be* a model of (intensional) type theory. We will do this in section 4.4 below – for now we simply describe a single model in the category Gpd of groupoids, which will turn out to be an instance of the general notion. For the extensional theory, we defined a model simply to be an interpretation into an LCCC $\mathcal{E}$, with contexts Γ interpreted as objects of $\mathcal{E}$, substitutions $\sigma : \Delta \to \Gamma$ as arrows of $\mathcal{E}$, type families $\Gamma \vdash A$ as objects of $\mathcal{E}/_\Gamma$, and terms $\Gamma \vdash a : A$ as sections of the associated families. Substitution into families and terms was (weakly) interpreted' as pullback (in the sense that there was an unresolved coherence issue), and the Σ and Π type formers were adjoints to pullback. Finally, the equality type

$x : A, y : A \vdash \mathtt{Eq}_A(x, y)$ was interpreted as the diagonal $A \to A \times A$.

$$\begin{array}{ccc} (x : A, y : A, z : \mathtt{Eq}_A(x, y)) & = & A \\ \downarrow & & \downarrow \\ (x : A, y : A) & = & A \times A \end{array}$$

We may simplify the notation for category of contexts by using Σ-types, writing e.g. $(x : A, y : A, z : \mathtt{Eq}_A(x, y)) = \Sigma_{x:A}\Sigma_{y:A}\mathtt{Eq}_A(x, y)$ or even $\mathtt{Eq}_A$, and $(x : A, y : A) = A \times A$, etc.

The groupoid interpretation of the intensional theory is based on the idea that the identity type of a type (interpreted as a groupoid) G can be interpreted by the *path groupoid* of G, which we shall write as G^{I}.

Definition 4.2.1. If the groupoid $\mathsf{G} = G_0 \rightrightarrows G_1$ has objects G_0 and arrows G_1, the *path groupoid* $\mathsf{G}^{\mathsf{I}} = (|(\mathsf{G}^{\mathsf{I}})^{\mathsf{I}}| \rightrightarrows |\mathsf{G}^{\mathsf{I}}|)$ has as objects $|\mathsf{G}^{\mathsf{I}}| = G_1$, and as arrows $|(\mathsf{G}^{\mathsf{I}})^{\mathsf{I}}|$ the set of all commutative squares in G, with the obvious source and target maps.

In other words, the path groupoid is the arrow category $\mathsf{G}^{\downarrow}$. Recall that the category Gpd of (small) groupoids is a cartesian closed subcategory of Cat, and that there is a *walking arrow* groupoid I with exactly two objects and two (mutually inverse) non-identity arrows,

$$\mathsf{I} \quad = \quad (\, 0 \; \underset{}{\overset{\curvearrowleft}{\underset{\curvearrowright}{}}} \; 1 \,)$$

The notation G^{I} for the path groupoid is then correctly the exponential of G by I in Gpd (and in Cat). Observe that there are functors $\mathsf{dom}, \mathsf{cod} : \mathsf{G}^{\mathsf{I}} \rightrightarrows \mathsf{G}$, as well as one $\mathsf{id} : \mathsf{G} \to \mathsf{G}^{\mathsf{I}}$, making $\mathsf{G}^{\mathsf{I}} \rightrightarrows \mathsf{G}$ into an internal groupoid in Gpd, for any object G. This will be our interpretation of the identity type of the type interpreted by G.

More formally, we interpret:

- Contexts Γ: groupoids, i.e. objects of Gpd,
- Substitutions $\sigma : \Delta \to \Gamma$: homomorphisms of groupoids, i.e. arrows of Gpd,
- Types $\Gamma \vdash A$: functors $A : \mathsf{G} \to \mathsf{Gpd}$, where G interprets Γ,
- Terms $\Gamma \vdash a : A$: natural transformations $a : 1 \to A$ between functors, where 1 is the terminal functor in $\mathsf{Gpd}^{\mathsf{G}}$,
- Context extension $(\Gamma, A) \to \Gamma$: the Grothendieck construction $\int_{\mathsf{G}} A \to \mathsf{G}$.

In order to model the type formers Σ, Π, etc. of intensional type theory in Gpd, we must deal with the fact that Gpd is not locally cartesian closed, although it is cartesian closed. Recall that in order to model extensional type theory in presheaves we used the fact that $\mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}$ is always a CCC and that for any $P \in \mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}$ we have an equivalence

$$\mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}/_P \;\simeq\; \mathsf{Set}^{\int P^{\mathsf{op}}},$$

and thus every slice is also a CCC. For groupoids, something similar is the case, but instead of the full slice category Gpd/G we use the subcategory of *fibrations* $\mathsf{Fib}_{\mathsf{G}} \hookrightarrow \mathsf{Gpd}/\mathsf{G}$, for which we have an equivalence

$$\mathsf{Fib}_{\mathsf{G}} \;\simeq\; \mathsf{Gpd}^{\mathsf{G}} \;\simeq\; \mathsf{Gpd}(\mathsf{Set}^{\mathsf{G}^{\mathsf{op}}})\,,$$

Since the proof that Gpd is a CCC doesn't depend on the classical logic of Set, the category of internal groupoids in a topos like $\mathsf{Set}^{\mathsf{G}^{\mathsf{op}}}$ is also a CCC. Thus we have that $\mathsf{Fib}_{\mathsf{G}}$ is a CCC for any groupoid G.

Definition 4.2.2. A (split op-) fibration of groupoids $p : \mathsf{A} \to \mathsf{G}$ is a functor satisfying the condition: for every $a \in \mathsf{A}$ and $\gamma : pa \to g$ there is given a "lift" $\ell(a,\gamma) : a \to \tilde{g}$ with $p(\ell(a,\gamma)) = p$, and moreover,

1. $\ell(a, 1_{pa}) = 1_a : a \to a$,
2. for $\gamma' : g = p(\tilde{g}) \to h$, the lift of the composite is the composite of the lifts:

$$\ell(a, \gamma' \circ \gamma) = \ell(\tilde{g}, \gamma') \circ \ell(a,\gamma) : a \to \tilde{h}\,.$$

Proposition 4.2.3. *The category* $\mathsf{Fib}_{\mathsf{G}}$ *of fibrations of groupoids and functors* $f : \mathsf{A} \to \mathsf{B}$ *over* G *that preserve the lifts is equivalent to the functor category* $\mathsf{Gpd}^{\mathsf{G}}$.

The interpretation of the context extension $(\Gamma, A) \to \Gamma$ is to be projection $\int_{\mathsf{G}} A \to \mathsf{G}$ given by the Grothendieck construction, and this is indeed a fibration of groupoids. Indeed, the functor taking $A : \mathsf{G} \to \mathsf{Gpd}$ to $\int_{\mathsf{G}} A \to \mathsf{G}$ mediates the equivalence $\int : \mathsf{Gpd}^{\mathsf{G}} \simeq \mathsf{Fib}_{\mathsf{G}}$.

For the base change functors along a fibration $p : \mathsf{A} \to \mathsf{G}$, we then have left and right adjoints as follows:

$$\begin{array}{ccc} \mathsf{A} & & \mathsf{Gpd}^{\mathsf{A}} \xrightarrow[\sim]{\int} \mathsf{Fib}_{\mathsf{A}} \\ {\scriptstyle p}\downarrow & & {\scriptstyle p^*}\uparrow \qquad\qquad \Sigma \dashv p^* \dashv \Pi \\ \mathsf{G} & & \mathsf{Gpd}^{\mathsf{G}} \xrightarrow[\sim]{\int} \mathsf{Fib}_{\mathsf{G}} \end{array}$$

To show this, it needs to be shown that:

1. the pullback of a fibration is a fibration,
2. the composite of fibrations is a fibration,
3. there is a push-forward fibration of a fibration along a fibration, which is right adjoint to pullback.

The proof uses the CCC structure in the categories $\mathsf{Fib}_{\mathsf{A}} \simeq \mathsf{Gpd}^{\mathsf{A}} \simeq \mathsf{Gpd}(\mathsf{Set}^{\mathsf{A}})$ and is similar to the proof of the LCCC structure for a category of presheaves.

Identity types

To interpret the $\mathtt{Id}$-type of a type (interpreted as, say) $\mathsf{A} = (A_1 \rightrightarrows A_0)$ in Gpd (or indeed in any relative version $\mathsf{Gpd}(\mathsf{Set}^{\mathsf{G}})$), we shall use the path groupoid

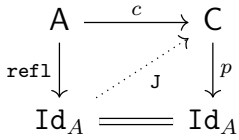
$$\mathtt{Id}_{\mathsf{A}} = \mathsf{A}^{\mathsf{I}} \to \mathsf{A} \times \mathsf{A}\,,$$

which is easily seen to be a fibration, and therefore corresponds to a functor $\mathtt{Id}_A : \mathsf{A} \times \mathsf{A} \to \mathsf{Gpd}$, namely that with *discrete groupoids* as its values:

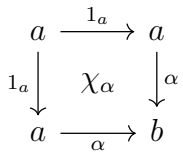
$$\mathtt{Id}_A(a, b) = \{p : a \to b\} \subseteq A_1\,.$$

Note that for two objects $a, b \in \mathsf{A}$ there may be many different arrows $f : a \to b$ in $\mathtt{Id}_A(a, b)$, but for two such parallel arrows $f, g : a \rightrightarrows b$ in A, regarded as objects in the path groupoid $\mathtt{Id}_A = \mathsf{A}^{\mathsf{I}}$, there need be no arrow between them in the (discrete) groupoid $\mathtt{Id}_A(a, b)$; and indeed, there will be one (which is then unique) just if $f = g$. Thus we will have the desired violation of UIP, once we have shown that this interpretation satisfies the rules for intensional $\mathtt{Id}$-types.

To show that, consider the diagram below, which we have already encountered as (4.6.2). We take any fibration $p : \mathsf{C} \to \mathtt{Id}_{\mathsf{A}}$ and any section $c : \mathsf{A} \to \mathsf{C}$ over the insertion of identity arrows into the path groupoid $\mathtt{refl} : \mathsf{A} \to \mathsf{A}^{\mathsf{I}} = \mathtt{Id}_A$, and we need a diagonal filler $\mathtt{J}$.



Since the diagram commutes by assumption, for any $a \in \mathsf{A}$ we have $pca = 1_a$. Let $\alpha : a \to b$ be any object in $\mathtt{Id}_A = \mathsf{A}^{\mathsf{I}}$ and observe that there is always an arrow $\chi_\alpha : 1_a \Rightarrow \alpha$ in $\mathtt{Id}_A = \mathsf{A}^{\mathsf{I}}$, namely $\chi_\alpha = (1_a, \alpha)$.



Since $p : \mathsf{C} \to \mathtt{Id}_{\mathsf{A}}$ is a fibration, there is a lift $\ell(ca, \chi_\alpha) : ca \to \tilde{\alpha}$. We then set

$$\mathtt{J}(\alpha) = \tilde{\alpha}$$

to obtain a functor $\mathtt{J} : \mathtt{Id}_{\mathsf{A}} \to \mathsf{A}$ making the two triangles in the diagram commute.

Exercise 4.2.4. Prove this!

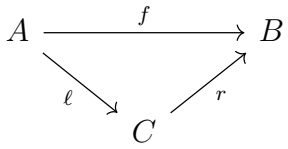
Exercise 4.2.5. Show that the composition of fibrations $\mathsf{B} \to \mathsf{A}$ and $\mathsf{A} \to \mathsf{G}$ is a fibration. (This will be used for the interpretation of the type $\Gamma \vdash \Sigma_{\mathsf{A}}\mathsf{B}$, where $\Gamma \vdash \mathsf{A}$ and $\Gamma, \mathsf{A} \vdash \mathsf{B}$.)

4.3 Weak factorization systems

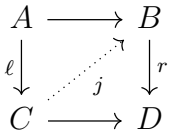
We can axiomatize the features of the groupoid model that allowed us to model intensional type theory using the notion of a *weak factorization system*, which is important in axiomatic homotopy theory. This is a weakening of the notion of an orthogonal factorization system from Definition 3.6.5:

Definition 4.3.1. An *weak factorization system (wfs)* on a category $\mathcal{C}$ consists of two classes of arrows (L, R) such that:

1. Every map $f : A \to B$ factors $f = r \circ \ell$ into $\ell \in \mathsf{L}$ followed by $r \in \mathsf{R}$,



2. Given any commutative square with an L-map on the left and an R-map on the right,



(4.2)

   there is a (not necessarily unique) *diagonal filler* j as indicated, making both triangles commute.

3. The classes L, R are closed under retracts in the arrow category $\mathcal{C}^{\downarrow}$.

Given such a wfs on a finitely complete category $\mathcal{C}$, we shall interpret the contexts and substitutions as the objects and arrows of $\mathcal{C}$, the type families as the right maps R, and the terms as sections of the right maps. The first part of the following is required for the interpretation of substitution, and the second part is used for context extension and Σ-types.

Lemma 4.3.2. *In a wfs* (L, R) *on a finitely complete category* $\mathcal{C}$*, the right maps are stable under pullback along all maps. (Dually, the left maps are stable under pushouts along all maps.) Moreover, both* L *and* R *are closed under composition.*

Before giving the proof, we develop an important aspect of wfs's: weak orthogonality. For any maps $f : A \to B$ and $g : C \to D$, let us write

$$f \pitchfork g$$

and say that f *is weakly orthogonal to* g if every commutative square with f on the left and g on the right has a diagonal filler j as in (4.2). We also say that "f has the left lifting

property with respect to g" and "g has the right lifting property with respect to f". More generally, for any class of arrows S in $\mathcal{C}$, write

$$\begin{aligned} S \pitchfork f &= s \pitchfork f \text{ for all } s \in S \\ f \pitchfork S &= f \pitchfork s \text{ for all } s \in S \end{aligned}$$

and let

$$\begin{aligned} S^{\pitchfork} &= \{f \mid S \pitchfork f\} \\ {}^{\pitchfork}S &= \{f \mid f \pitchfork S\}\,. \end{aligned}$$

Finally, let $S \pitchfork T$ mean that $s \pitchfork t$ for all $s \in S$ and $\mathsf{t} \in T$. Then in a wfs (L, R) we clearly have

$$\mathsf{L} \pitchfork \mathsf{R}\,,$$

by axiom 2, but in fact more is true:

Lemma 4.3.3. *Given two classes of maps (L, R) in a category $\mathcal{C}$ satisfying the factorization and diagonal filler axioms for a wfs above, L and R are also closed under retracts if and only if*

$$\mathsf{L}^{\pitchfork} = \mathsf{R} \quad \textit{and} \quad \mathsf{L} = {}^{\pitchfork}\mathsf{R}\,.$$

Proof. Suppose (L, R) is a wfs, so both classes are closed under retracts. We need to show that if $f : A \to B$ satisfies $\mathsf{L} \pitchfork \mathsf{f}$, then $f \in \mathsf{R}$ (the converse is already true by $\mathsf{L} \pitchfork \mathsf{R}$. Factor $f = r \circ \ell$ and consider the diagram

$$\begin{array}{ccc} A & = & A \\ {\scriptstyle \ell}\downarrow & \overset{j}{\nearrow} & \downarrow{\scriptstyle f} \\ C & \xrightarrow[r]{} & B \end{array} \tag{4.3}$$

which commutes and has a diagonal filler j, since $\ell \in \mathsf{L}$. We can rearrange (4.3) into a retract diagram as follows.

$$\begin{array}{ccccc} A & \xrightarrow{\ell} & C & \xrightarrow{j} & A \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle r} & & \downarrow{\scriptstyle f} \\ B & = & B & = & B \end{array} \tag{4.4}$$

Thus $f \in \mathsf{R}$, since R is closed under retracts. The argument for $\mathsf{L} = {}^{\pitchfork}\mathsf{R}$ is dual. We leave the converse as an exercise. □

Exercise 4.3.4. Show that (L, R) is a wfs if the factorization axiom holds and $\mathsf{L}^{\pitchfork} = \mathsf{R}$ and $\mathsf{L} = {}^{\pitchfork}\mathsf{R}$.

Proof. (of Lemma 4.3.2) Let $f : A \to B$ be in R and consider its pullback along any $B' \to B$:

$$\begin{array}{ccc} A' & \longrightarrow & A \\ {\scriptstyle f'}\downarrow & \lrcorner & \downarrow{\scriptstyle f} \\ B' & \longrightarrow & B \end{array} \tag{4.5}$$

To show $f' \in \mathsf{R}$, it suffices by Lemma 4.3.3 to show that f' has the right lifting property with respect to L, but this follows easily from f' being a pullback of f, which does. We leave the rest of the proof as an exercise. □

Exercise 4.3.5. Finish the proof of Lemma 4.3.2.

It now follows that we can interpret the structural rules of dependent type theory, as well as the context extension operation, using the R maps as the type families, just as we did using arbitrary maps in an lccc. The rules for Σ types will also be satisfied, since these essentially state that Σ is left adjoint to pullback along type families, and therefore closure of the right maps under composition means that they are closed under Σ-types.

Let us see that we can also interpret the rules for $\mathtt{Id}$-types. The formation rule for $\mathtt{Id}_A$ is interpreted by factoring the diagonal substitution $\delta : A \to A \times A$ into a left map followed by a right map:

$$\begin{array}{ccc} & & \mathtt{Id}_A \\ & \overset{\mathtt{refl}}{\nearrow} & \downarrow{\scriptstyle p} \\ A & \xrightarrow[\delta]{} & A \times A \end{array}$$

This also interprets the introduction rule, using the left map in the factorization as the interpretation of the $\mathtt{refl}$ term. For the elimination rule, suppose we have a type family $p : \mathsf{C} \to \mathtt{Id}_{\mathsf{A}}$ and a section $c : \mathsf{A} \to \mathsf{C}$ over $\mathtt{refl} : \mathsf{A} \to \mathtt{Id}_A$; then we need a diagonal filler $\mathtt{J}$.

$$\begin{array}{ccc} \mathsf{A} & \xrightarrow{c} & \mathsf{C} \\ {\scriptstyle \mathtt{refl}}\downarrow & \overset{\mathtt{J}}{\nearrow} & \downarrow{\scriptstyle p} \\ \mathtt{Id}_A & = & \mathtt{Id}_A \end{array}$$

But since $\mathtt{refl} : A \to \mathtt{Id}_A$ is a left map by the factorization, and $C \to \mathtt{Id}_A$ is a right map by the interpretation of type families as right maps, there is such a filler by the second axiom of wfs's. Thus we have already shown:

Proposition 4.3.6 ([AW09])**.** *In a fintely complete category $\mathcal{C}$ with a wfs, the rules of intensional identity types are soundly modeled by interpreting the type families as the right maps and the identity type $\mathtt{Id}_A$ as a factorization of the diagonal $\delta : A \to A \times A$ into a left map $\mathtt{refl} : A \to \mathtt{Id}_A$ followed by a right map $\mathtt{Id}_A \to A \times A$.*

This kind of interpretation includes many important "naturally occurring" examples involving *Quillen model categories*, which are categories equipped with two interlocking

wfs's (see [AW09]). The Π-types can also be interpreted in this way, if the right maps of the wfs pushforward along right maps, as is the case in examples such as right-proper Quillen model categories and Π-tribes in the sense of [Joy17]. Indeed, the groupoid model from the previous section was an instance of such a wfs: as the right maps one can take the isofibrations, and the left maps are then the equivalences that are injective on objects [GG08].

Remark 4.3.7 (Coherence)**.** The coherence issue that we have been postponing is now even more pressing, however, because the factorizations in a wfs need not be stable under pullback, and so the following (slightly schematic) substitution rule for the `Id` type former will not be soundly modeled, even up to isomorphism.

$$\frac{\Gamma \vdash A\,, \quad \Gamma \vdash a : A, \quad \Gamma \vdash b : A, \quad \sigma : \Delta \to \Gamma}{\Delta \vdash \mathtt{Id}_A(a,b)[\sigma] \equiv \mathtt{Id}_{A[\sigma]}(a[\sigma], b[\sigma])}$$

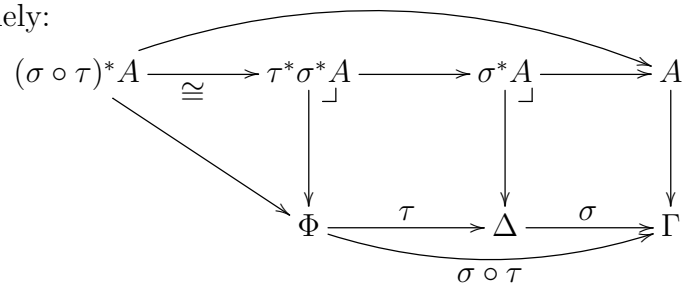
A similar problem occurs with respect to the `J`-term, which is also required to respect (certain) substitutions, but need not do so under this interpretation. One solution to this problem makes use of an algebraic structure on the weak factorization system called an *algebraic weak factorization system*. This approach is discussed in [GL23]. We shall develop a different solution in the next section.

4.4 Natural models

The semantics of DTT in LCCCs employed in the previous sections uses the "slice category" hyperdoctrine of an LCCC to interpret the dependent types. Thus the contexts Γ and substitutions $\sigma : \Delta \to \Gamma$ are interpreted as the objects and arrows of an LCCC $\mathcal{C}$, and the dependent types $\Gamma \vdash A$ and terms $\Gamma \vdash a : A$ are interpreted as objects $A \to \Gamma$ in the slice category $\mathcal{C}/\Gamma$ and their global sections $a : \Gamma \to A$ (over Γ). As we mentioned in Remark 3.4.4, however, there is a problem with this kind of semantics (as first pointed out by [Hof95]): as a hyperdoctrine, this interpretation is a *pseudo*functor $\mathcal{C}/ : \mathcal{C}^{\mathsf{op}} \to \mathsf{Cat}$, but the syntax of DTT produces an actual *presheaf* of types in context $\mathsf{Ty} : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$, since substitution into dependent types is *strictly* functorial with respect to composition of substitutions, in the sense that for a type in context $\Gamma \vdash A$ and substitutions $\sigma : \Delta \to \Gamma$ and $\tau : \Phi \to \Delta$ we have a *judgemental equality of types in context*,

$$\Phi \vdash (A[\sigma])[\tau] \equiv A[\sigma \circ \tau]\,,$$

rather than the (canonical) isomorphism $\cong$ fitting into the two-pullbacks diagram of the hyperdoctrine, namely:



A similar problem occurs in the Beck-Chavalley conditions, where the hyperdoctrine structure has only canonical isos, rather than the strict equalities that obtain in the syntax, such as

$$(\Pi_{x:A} B)[\sigma] \equiv (\Pi_{x:A[\sigma]} B[\sigma])\,.$$

Exactly the same problem occurs, of course, if we use only the right maps in a wfs, rather than all maps in the slice category of an LCC. And for the `Id`-type former, we have an even more acute probelm, as noted in Remark 4.3.7, since the factorization of the diagonal is not even determined up to isomorphism, and need not be stable under pullback

There are various different solutions to these "coherence problems" in the literature, some involving *strictifications* of the LCC slice-category (or wfs-pullback) hyperdoctrine (including both left- and right-adjoint strictifications [LW15, Hof94]), as well as other semantics altogether, such as categories-with-families [Dyb96], categories-with-attributes [**?**], and comprehension categories [**?**]. A solution based on the notion of a *universe* $\widetilde{U} \to U$ was first proposed by Voevodsky [**?**]; in [Awo16], the universe approach is combined with the notion of a *representable natural transformation* to determine the semantic notion of a *natural model*, as follows.

Definition 4.4.1. For a small category $\mathbb{C}$, a natural transformation $f : Y \to X$ of presheaves on $\mathbb{C}$ is called *representable* if for every $C \in \mathbb{C}$ and $x \in X(C)$, there is given a $p : D \to C$ and a $y \in Y(D)$ such that the following square is a pullback.

$$\begin{array}{ccc} \mathsf{y}D & \xrightarrow{y} & Y \\ {\scriptstyle \mathsf{y}p}\downarrow & \lrcorner & \downarrow{\scriptstyle f} \\ \mathsf{y}C & \xrightarrow[x]{} & X \end{array} \tag{4.6}$$

We will show that a representable natural transformation is essentially the same thing as a *category with families* in the sense of Dybjer [Dyb96]. Indeed, let us write the objects of $\mathbb{C}$ as $\Gamma, \Delta, \ldots$ and the arrows as $\sigma : \Delta \to \Gamma, \ldots$, thinking of $\mathbb{C}$ as a "category of contexts". Let $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ be a representable map of presheaves, and interpret the elements as

$$\begin{aligned} \mathsf{T}(\Gamma) &= \{A \mid \Gamma \vdash A\} \\ \dot{\mathsf{T}}(\Gamma) &= \{a \mid \Gamma \vdash a : A, \text{ for some } A\}, \end{aligned}$$

so that under Yoneda we have:

$$\begin{aligned} A \in \mathsf{T}(\Gamma) \quad &\text{iff} \quad \Gamma \vdash A \\ a \in \dot{\mathsf{T}}(\Gamma) \quad &\text{iff} \quad \Gamma \vdash a : A, \text{ where } \mathsf{t} \circ a = A \end{aligned}$$

as indicated in:

$$\begin{array}{ccc} & & \dot{\mathsf{T}} \\ & {\scriptstyle a}\nearrow & \downarrow{\scriptstyle \mathsf{t}} \\ \mathsf{y}\Gamma & \xrightarrow[A]{} & \mathsf{T} \end{array}$$

Thus we are regarding T as the *presheaf of types*, with $\mathsf{T}(\Gamma)$ the set of all types in context Γ, and $\dot{\mathsf{T}}$ as the *presheaf of terms*, with $\dot{\mathsf{T}}(\Gamma)$ the set of all terms in context Γ, while the component $\mathsf{t}_\Gamma : \dot{\mathsf{T}}(\Gamma) \to \mathsf{T}(\Gamma)$ is the typing of the terms in context Γ.

The naturality of $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ just means that for any substitution $\sigma : \Delta \to \Gamma$, we have an action on types and terms:

$$\begin{aligned} \Gamma \vdash A \quad &\mapsto \quad \Delta \vdash A\sigma \\ \Gamma \vdash a : A \quad &\mapsto \quad \Delta \vdash a\sigma : A\sigma\,. \end{aligned}$$

While, by functoriality, given any further $\tau : \Phi \to \Delta$, we have

$$(A\sigma)\tau = A(\sigma \circ \tau) \qquad (a\sigma)\tau = a(\sigma \circ \tau),$$

as well as

$$A1 = A \qquad a1 = a$$

for the identity substitution $1 : \Gamma \to \Gamma$.

Finally, the representability of the natural transformation $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ is exactly the operation of *context extension*: given any $\Gamma \vdash A$, by Yoneda we have the corresponding map $A : \mathsf{y}\Gamma \to \mathsf{T}$, and we let $p_A : \Gamma.A \to \Gamma$ be (the map representing) the pullback of t along A, as in (4.6). We therefore have a pullback square:

$$\begin{array}{ccc} \mathsf{y}\Gamma.A & \xrightarrow{q_A} & \dot{\mathsf{T}} \\ {\scriptstyle \mathsf{y}p_A}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{t}} \\ \mathsf{y}\Gamma & \xrightarrow[A]{} & \mathsf{T} \end{array} \tag{4.7}$$

where the map $q_A : \Gamma.A \to \dot{\mathsf{T}}$ now determines a term

$$\Gamma.A \vdash q_A : Ap_A.$$

In type theory, the term $q_A : \Gamma.A \to \dot{\mathsf{T}}$ corresponds to the rule of "assumption":

$$\frac{\Gamma \vdash A \ \mathtt{type}}{\Gamma, x : A \vdash x : A}$$

We may hereafter omit the y for the Yoneda embedding, letting the Greek letters serve to distinguish representable presheaves and their maps.

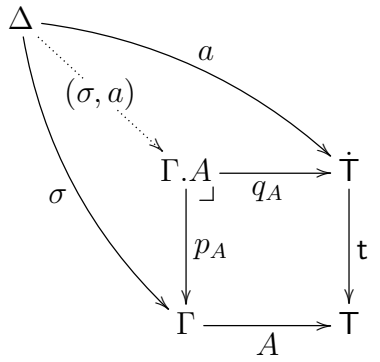
Exercise 4.4.2. Show that the fact that (4.4.4) is a pullback means that given any $\sigma : \Delta \to \Gamma$ and $\Delta \vdash a : A\sigma$, there is a map

$$(\sigma, a) : \Delta \to \Gamma.A,$$

and this operation satisfies the equations

$$\begin{aligned} p_A \circ (\sigma, a) &= \sigma \\ q_A(\sigma, a) &= a, \end{aligned}$$

as indicated in the following diagram.



Show moreover that the uniqueness of (σ, a) means that for any $\tau : \Delta' \to \Delta$ we also have:

$$(\sigma, a) \circ \tau = (\sigma \circ \tau, a\tau)$$
$$(p_A, q_A) = 1.$$

Comparing the foregoing with the definition of a category with families in [Dyb96], we have shown:

Proposition 4.4.3. *Let* $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ *be a representable natural transformation of presheaves on a small category* $\mathbb{C}$ *with a terminal object. Then* t *determines a category with families, with* $\mathbb{C}$ *as the contexts and substitutions,* $\mathsf{T}(\Gamma)$ *as the types in context* Γ*, and* $\dot{\mathsf{T}}(\Gamma)$ *as the terms in context* Γ*.*

Remark 4.4.4. A category with families is usually defined in terms of a presheaf

$$\mathsf{Ty} : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$$

of types on the category $\mathcal{C}$ of contexts, together with a presheaf

$$\mathsf{Tm}' : (\textstyle\int_{\mathcal{C}} \mathsf{Ty})^{\mathsf{op}} \to \mathsf{Set}$$

of typed-terms on the category $\int_{\mathcal{C}} \mathsf{Ty}$ of types-in-context. We are using the equivalence of categories, valid for any category of presheaves $\mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}$,

$$\mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}/_P \;\simeq\; \mathsf{Set}^{(\int_{\mathbb{C}} P)^{\mathsf{op}}}$$

between the slice category over a presheaf P and the presheaves on its category of elements $\int_{\mathbb{C}} P$, to turn the presheaf $\mathsf{Tm}' : (\int_{\mathcal{C}} \mathsf{Ty})^{\mathsf{op}} \to \mathsf{Set}$ into one $\mathsf{Tm} : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$ together with a map $\mathsf{Tm} \to \mathsf{Ty}$ in $\mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$.

We think of a representable map of presheaves on an arbitrary category $\mathbb{C}$ as a "type theory over $\mathbb{C}$", with $\mathbb{C}$ as the category of contexts and substitutions. We will show in Section 4.4.4 that such a map of presheaves is essentially determined by a class of maps in $\mathbb{C}$ that is closed under all pullbacks, corresponding to the "incoherent" interpretation of types in context as maps $A \to \Gamma$.

Definition 4.4.5. A *natural model of type theory* on a small category $\mathbb{C}$ is a representable map of presheaves $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$.

Exercise 4.4.6 (The natural model of syntax)**.** Let $\mathbb{T}$ be a dependent type theory and $\mathcal{C}_{\mathbb{T}}$ its category of contexts and substitutions. Define the presheaves $\mathsf{Ty} : {\mathcal{C}_{\mathbb{T}}}^{\mathsf{op}} \to \mathsf{Set}$ of types-in-context and $\mathsf{Tm} : {\mathcal{C}_{\mathbb{T}}}^{\mathsf{op}} \to \mathsf{Set}$ of terms-in-context, along with a natural transformation

$$\mathsf{tp} : \mathsf{Tm} \to \mathsf{Ty}$$

that takes a term to its type. Show that $\mathsf{tp} : \mathsf{Tm} \to \mathsf{Ty}$ is a natural model of type theory.

4.4.1 Modeling $1, \Sigma, \Pi$

Given a natural model $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$, we will make extensive use of the associated *polynomial endofunctor* $\mathsf{P}_{\mathsf{t}} : \hat{\mathbb{C}} \longrightarrow \hat{\mathbb{C}}$ (cf. [GK13]), defined by

$$\mathsf{P}_{\mathsf{t}} = \mathsf{T}_! \circ \mathsf{t}_* \circ \dot{\mathsf{T}}^* : \hat{\mathbb{C}} \longrightarrow \hat{\mathbb{C}}\,,$$

$$\begin{array}{ccc} \mathsf{Set}^{\mathbb{C}^{\mathsf{op}}} & \xrightarrow{\mathsf{P}_{\mathsf{t}}} & \mathsf{Set}^{\mathbb{C}^{\mathsf{op}}} \\ {\scriptstyle \dot{\mathsf{T}}^*}\downarrow & & \uparrow{\scriptstyle \mathsf{T}_!} \\ \mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}/_{\dot{\mathsf{T}}} & \xrightarrow[\mathsf{t}_*]{} & \mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}/_{\mathsf{T}} \end{array}$$

The action of P_{t} on an object X may be depicted:

$$\begin{array}{ccc} X \longleftarrow X \times \dot{\mathsf{T}} & & \mathsf{P}_{\mathsf{t}}(X) \\ \downarrow & & \downarrow \\ \dot{\mathsf{T}} & \xrightarrow[\mathsf{t}]{} & \mathsf{T} \end{array}$$

We call $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ the *signature* of P_{t} and briefly recall the following *universal mapping property* from [Awo16].

Lemma 4.4.7. *For any $p : E \to B$ in a locally cartesian closed category $\mathcal{E}$, the polynomial functor $\mathsf{P}_p : \mathcal{E} \to \mathcal{E}$ has the following universal property: for any objects $X, Z \in \mathcal{E}$, maps $f : Z \to \mathsf{P}_p(X)$ correspond bijectively to pairs of maps $f_1 : Z \to B$ and $f_2 : Z \times_B E \to Z$, as indicated below.*

$$\begin{array}{c} Z \xrightarrow{\quad f \quad} \mathsf{P}_p(X) \\ \hline\hline \\ \begin{array}{ccccc} X & \xleftarrow[f_2]{} & Z \times_B E & \longrightarrow & E \\ & & \downarrow & \lrcorner & \downarrow{\scriptstyle p} \\ & & Z & \xrightarrow[f_1]{} & B \end{array} \end{array} \tag{4.8}$$

The correspondence is natural in both X and Z, in the expected sense.

This universal property is also suggested by the conventional type theoretic notation, namely:

$$\mathsf{P}_p(X) = \Sigma_{b:B} X^{E_b}$$

The lemma can be used to determine the signature $p \cdot q$ for the composite $\mathsf{P}_p \circ \mathsf{P}_q$ of two polynomial functors, which is again polynomial, and for which we therefore have

$$\mathsf{P}_{p\cdot q} \;\cong\; \mathsf{P}_p \circ \mathsf{P}_q \,. \tag{4.9}$$

Indeed, let $p : B \to A$ and $q : D \to C$, and consider the following diagram resulting from applying the correspondence (4.10) to the identity arrow,

$$\langle a, c\rangle = 1_{\mathsf{P}_p(C)} : \mathsf{P}_p(C) \to \mathsf{P}_p(C)\,,$$

and taking Q to be the indicated pullback.

$$\begin{array}{ccccc} D & \longleftarrow & Q & & \\ {\scriptstyle q}\downarrow & \llcorner & \downarrow & \searrow^{p\cdot q} & \\ C & \xleftarrow[c]{} & \pi^*B & \longrightarrow & B \\ & & \downarrow & \lrcorner & \downarrow{\scriptstyle p} \\ & & \mathsf{P}_p(C) & \xrightarrow[a]{} & A \end{array} \tag{4.10}$$

The map $p \cdot q$ is then defined to be the indicated composite,

$$p \cdot q \;=\; a^* p \circ c^* q \,.$$

The condition (4.9) can then be checked using the correspondence (4.10) (also see [GK13]).

Definition 4.4.8. A natural model $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ over $\mathbb{C}$ will be said to *model* the type formers $\mathsf{1}, \Sigma, \Pi$ if there are pullback squares in $\hat{\mathbb{C}}$ of the following form,

$$\begin{array}{ccc} 1 & \longrightarrow & \dot{\mathsf{T}} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{t}} \\ 1 & \longrightarrow & \mathsf{T} \end{array} \qquad \begin{array}{ccc} \dot{\mathsf{T}}_2 & \longrightarrow & \dot{\mathsf{T}} \\ {\scriptstyle \mathsf{t}\cdot\mathsf{t}}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{t}} \\ \mathbb{T}_2 & \longrightarrow & \mathsf{T} \end{array} \qquad \begin{array}{ccc} \mathsf{P}_\mathsf{t}(\dot{\mathsf{T}}) & \longrightarrow & \dot{\mathsf{T}} \\ {\scriptstyle \mathsf{P}_\mathsf{t}(\mathsf{t})}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{t}} \\ \mathsf{P}_\mathsf{t}(\mathsf{T}) & \longrightarrow & \mathsf{T} \end{array} \tag{4.11}$$

where $\mathsf{t}\cdot\mathsf{t} : \dot{\mathsf{T}}_2 \to \mathsf{T}_2$ is determined by $\mathsf{P}_{\mathsf{t}\cdot\mathsf{t}} \cong \mathsf{P}_\mathsf{t} \circ \mathsf{P}_\mathsf{t}$ as in (4.9).

The terminology is justified by the following result from [Awo16].

Theorem 4.4.9 ([Awo16] Theorem XXX)**.** *Let* $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ *be a natural model. The associated category with families satisfies the usual rules for the type-formers* $\mathsf{1}, \Sigma, \Pi$ *just if* $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ *models the same in the sense of Definition 4.4.8.*

We only sketch the case of Π-types, but the other type formers will be treated in detail in Section **??**.

Proposition 4.4.10. *The natural model* $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ *models* Π*-types just if there are maps* λ *and* Π *making the following a pullback diagram.*

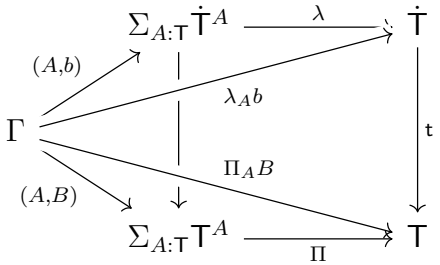
$$\begin{array}{ccc} \mathsf{P}_{\mathsf{t}}(\dot{\mathsf{T}}) & \xrightarrow{\lambda} & \dot{\mathsf{T}} \\ {\scriptstyle \mathsf{P}_{\mathsf{t}}(\mathsf{t})}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{t}} \\ \mathsf{P}_{\mathsf{t}}(\mathsf{T}) & \xrightarrow[\Pi]{} & \mathsf{T} \end{array} \tag{4.12}$$

Proof. Unpacking the definitions, we have $\mathsf{P}_{\mathsf{t}}(\mathsf{T}) = \Sigma_{A:\mathsf{T}}\mathsf{T}^A$, etc., so diagram (4.14) becomes:

$$\begin{array}{ccc} \Sigma_{A:\mathsf{T}}\dot{\mathsf{T}}^A & \xrightarrow{\lambda} & \dot{\mathsf{T}} \\ {\scriptstyle \Sigma_{A:\mathsf{T}}\mathsf{t}^A}\downarrow & & \downarrow{\scriptstyle \mathsf{t}} \\ \Sigma_{A:\mathsf{T}}\mathsf{T}^A & \xrightarrow[\Pi]{} & \mathsf{T} \end{array}$$

For $\Gamma \in \mathbb{C}$, maps $\Gamma \to \Sigma_{A:\mathsf{T}}\mathsf{T}^A$ correspond to pairs (A, B) with $A : \Gamma \to \mathsf{T}$ and $B : \Gamma, A \to \mathsf{T}$, and thus to $\Gamma \vdash A$ and $\Gamma, A \vdash B$. Similarly, a map $\Gamma \to \Sigma_{A:\mathsf{T}}\dot{\mathsf{T}}^A$ corresponds to a pair (A, b) with $\Gamma \vdash A$ and $\Gamma, A \vdash b : B$, the typing of b resulting from composing with the map

$$\Sigma_{A:\mathsf{T}}\mathsf{t}^A : \Sigma_{A:\mathsf{T}}\dot{\mathsf{T}}^A \to \Sigma_{A:\mathsf{T}}\mathsf{T}^A\,.$$



The composition across the top is then the term $\Gamma \vdash \lambda_{x:A} b$, the type of which is determined by composing with t and comparing with the composition across the bottom, namely $\Gamma \vdash \Pi_{x:A} B$. In this way, the lower horizontal arrow in the diagram models the Π-*formation rule*:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash \Pi_{x:A} B}$$

and the upper horizontal arrow, along with the commutativity of the diagram, models the Π-*introduction rule*:

$$\frac{\Gamma, A \vdash b : B}{\Gamma \vdash \lambda_{x:A} b : \Pi_{x:A} B}$$

The square (4.14) is a pullback just if, for every $(A, B) : \Gamma \to \Sigma_{A:\mathsf{T}}\mathsf{T}^A$ and every $t : \Gamma \to \dot{\mathsf{T}}$ with $\mathsf{t} \circ t = \Pi_A B$, there is a unique $(A, b) : \Gamma \to \Sigma_{A:\mathsf{T}}\dot{\mathsf{T}}^A$ with $b : B$ and $\lambda_A b = t$. In terms of the interpretation, given $\Gamma, A \vdash B$ and $\Gamma \vdash t : \Pi_{x:A} B$, there is a term $\Gamma, A \vdash t' : B$ with $\lambda_{x:A} t' = t$, and t' is unique with this property. This is just what is provided by the Π-*elimination rule*:

$$\frac{\Gamma, A \vdash B \qquad \Gamma \vdash t : \Pi_{x:A} B \qquad \Gamma \vdash x : A}{\Gamma, A \vdash t\, x : B}$$

in conjunction with the Π-*computation rules*:

$$\lambda_{x:A}(t\,x) = t : \Pi_A B$$
$$(\lambda_{x:A} b)\,x = b : B$$

□

4.4.2 Identity types

A natural model $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ in a presheaf category $\widehat{\mathbb{C}}$ was defined in [Awo16] to model both the extensional and intensional identity types of Martin-Löf type theory in terms of the existence of certain additional structures, which we briefly review. Condition (1) below captures the extensional equality types of Martin-Löf type theory. The condition given in *op.cit.* for the intensional case is replaced below by a simplification suggested by R. Garner.

Definition 4.4.11. Let $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ be a map in an lccc $\mathcal{E}$.

1. The map $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ is said to model the (extensional) *equality type former* if there are structure maps $(\mathtt{refl}, \mathsf{Eq})$ making a pullback square:

$$\begin{array}{ccc} \dot{\mathsf{T}} & \xrightarrow{\mathtt{refl}} & \dot{\mathsf{T}} \\ {\scriptstyle\delta}\downarrow & \lrcorner & \downarrow{\scriptstyle\mathsf{t}} \\ \dot{\mathsf{T}} \times_{\mathsf{T}} \dot{\mathsf{T}} & \xrightarrow[\mathsf{Eq}]{} & \mathsf{T} \end{array}$$

2. The map $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ is said to model the (intensional) *identity type former* if there are structure maps $(\mathsf{i}, \mathsf{Id})$ making a commutative square,

$$\begin{array}{ccc} \dot{\mathsf{T}} & \xrightarrow{\mathsf{i}} & \dot{\mathsf{T}} \\ {\scriptstyle\delta}\downarrow & & \downarrow{\scriptstyle\mathsf{t}} \\ \dot{\mathsf{T}} \times_{\mathsf{T}} \dot{\mathsf{T}} & \xrightarrow[\mathsf{Id}]{} & \mathsf{T} \end{array} \tag{4.13}$$

   together with a weak pullback structure J for the resulting comparison square, in the sense of (4.15) below.

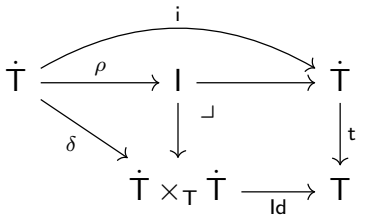
To describe the map J, let us see how (2) models identity types. Under the interpretation already described in Section **??** the maps `Id` and i in

$$\begin{array}{ccc} \dot{\mathsf{T}} & \xrightarrow{\mathsf{i}} & \dot{\mathsf{T}} \\ {\scriptstyle\delta}\downarrow & & \downarrow{\scriptstyle\mathsf{t}} \\ \dot{\mathsf{T}} \times_{\mathsf{T}} \dot{\mathsf{T}} & \xrightarrow[\mathsf{Id}]{} & \mathsf{T} \end{array}$$

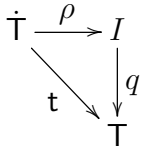
respectively, directly model the formation and introduction rules.

$$\begin{aligned} x, y : A &\vdash \mathtt{Id}_A(x, y) \\ x : A &\vdash \mathsf{i}(x) : \mathtt{Id}_A(x, x) \end{aligned}$$

Next, pull t back along Id to get an object I and a map $\rho : \dot{\mathsf{T}} \to \mathsf{I}$,



which commutes with the compositions to T as indicated below.



The map $\rho : \dot{\mathsf{T}} \to \mathsf{I}$, which can be interpreted as the substitution $(x) \mapsto (x, x, \mathsf{i}x)$, gives rise to a "restriction" natural transformation of polynomial endofunctors ([GK13]),

$$\rho^* : P_q \to P_{\mathsf{t}}\,,$$

evaluation of which at $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ results in the following commutative naturality square.

$$\begin{array}{ccc} P_q\dot{\mathsf{T}} & \xrightarrow{\rho^*_{\dot{\mathsf{T}}}} & P_{\mathsf{t}}\dot{\mathsf{T}} \\ {\scriptstyle P_q\mathsf{t}}\downarrow & & \downarrow{\scriptstyle P_{\mathsf{t}}\mathsf{t}} \\ P_q\mathsf{T} & \xrightarrow[\rho^*_{\mathsf{T}}]{} & P_{\mathsf{t}}\mathsf{T} \end{array} \tag{4.14}$$

Note that (4.14) is a pullback in the extensional case; here we require it to be a *weak* pullback by taking a section of the resulting "gap map". Explicitly, a *weak pullback structure* J is a section of the resulting comparison map.

$$P_q\dot{\mathsf{T}} \xrightarrow{\quad\mathsf{J}\quad} P_q\mathsf{T} \times_{P_{\mathsf{t}}\mathsf{T}} P_{\mathsf{t}}\dot{\mathsf{T}} \tag{4.15}$$

To show that this models the standard elimination rule, namely

$$\frac{x : A \vdash c(x) : C(\rho x)}{x, y : A, z : \mathtt{Id}_A(x, y) \vdash \mathsf{J}_c(x, y, z) : C(x, y, z)}$$

take any object $\Gamma \in \mathcal{C}$ and maps $(A, A, \mathtt{Id}_A \vdash C) : \Gamma \to P_q\mathsf{T}$ and $(A \vdash c) : \Gamma \to P_{\mathsf{t}}\dot{\mathsf{T}}$ with equal composites to $P_{\mathsf{t}}\mathsf{T}$, meaning that $A \vdash c : C(\rho x)$. Composing the resulting map

$$(A \vdash c(x) : C(\rho x)) : \Gamma \longrightarrow P_q\mathsf{T} \times_{P_{\mathsf{t}}\mathsf{T}} P_{\mathsf{t}}\dot{\mathsf{T}}$$

with $\mathsf{J} : P_q\mathsf{T} \times_{P_{\mathsf{t}}\mathsf{T}} P_{\mathsf{t}}\dot{\mathsf{T}} \to P_q\dot{\mathsf{T}}$ then indeed provides a term

$$x : A, y : A, z : \mathtt{Id}_A(x, y) \vdash \mathsf{J}_c(x, y, z) : C(x, y, z)\,.$$

The computation rule

$$x : A \vdash \mathsf{J}_c(\rho x) = c(x) : C(\rho x)$$

then says exactly that J is indeed a section of the comparison map (4.15).

Proposition 4.4.12 (R. Garner)**.** *The map* $\mathtt{t} : \dot{\mathsf{T}} \to \mathsf{T}$ *models intensional* identity types *just if there are maps* $(\mathsf{i}, \mathtt{Id})$ *making the diagram* (4.13) *commute, together with a weak pullback structure* J *for the resulting comparison square* (4.14)*.*

4.4.3 Path types

Definition 4.4.13. By an *interval* in an lccc $\mathcal{E}$ we simply mean a bipointed object $d_0, d_1 : 1 \rightrightarrows \mathsf{I}$. In terms of such an interval, we then define further:

1. For every object A, there is a *pathobject* factorization of the diagonal $\delta : A \to A \times A$, obtained by exponentiating A by $1 \rightrightarrows \mathsf{I} \to 1$,

$$\begin{array}{ccc} A & \xrightarrow{\ \rho\ } & A^{\mathsf{I}} \\ & \searrow^{\delta} & \downarrow{\scriptstyle \langle \varepsilon_0, \varepsilon_1 \rangle} \\ & & A \times A \end{array}$$

   where we write $\rho = A^{!_{\mathsf{I}}}$, and $\varepsilon_0 = A^{d_0}$, and $\varepsilon_1 = A^{d_1}$.

2. Similarly, and abusing notation slightly, for any $A \to X$ regarded as an object in the slice category $\mathcal{E}/_X$, we define the *relative pathobject factorization*

$$\begin{array}{ccc} A & \xrightarrow{\ \rho\ } & A^{\mathsf{I}} \\ & \searrow^{\delta} & \downarrow{\scriptstyle \langle \varepsilon_0, \varepsilon_1 \rangle} \\ & & A \times_X A \end{array}$$

   to be the pathobject factorization in $\mathcal{E}/_X$ with respect to the pulled-back interval $1_X \rightrightarrows X^*\mathsf{I} \to 1_X$, where we are using the pullback functor $X^* = !_X^* : \mathcal{E} \to \mathcal{E}/_X$ along $!_X : X \to 1$.

Lemma 4.4.14. *For any object $A \to X$ over any base X, the relative pathobject factorization*

$$\begin{array}{ccc} A & \xrightarrow{\rho} & A^{\mathsf{I}} \\ & \searrow^{\delta} & \downarrow{\scriptstyle \langle \varepsilon_0, \varepsilon_1 \rangle} \\ & & A \times_X A\,, \end{array}$$

is stable under pullback along any map $f : Y \to X$, in the sense that the factorization pulls back to the relative pathobject factorization over Y of the pullback $f^*A \to Y$, resulting in a canonical isomorphism over Y,

$$(f^*A)^{\mathsf{I}} \cong f^*(A^{\mathsf{I}})\,.$$

Definition 4.4.15. A natural model $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ will be said to *have path types* if there are structure maps $(\mathsf{Id}, \mathsf{j})$ making a pullback square,

$$\begin{array}{ccc} \dot{\mathsf{T}}^{\mathsf{I}} & \xrightarrow{\mathsf{j}} & \dot{\mathsf{T}} \\ {\scriptstyle \varepsilon}\downarrow & & \downarrow{\scriptstyle \mathsf{t}} \\ \dot{\mathsf{T}} \times_{\mathbb{T}} \dot{\mathsf{T}} & \xrightarrow[\mathsf{Id}]{} & \mathsf{T} \end{array} \tag{4.16}$$

where $\varepsilon = \langle \varepsilon_0, \varepsilon_1 \rangle : \dot{\mathsf{T}}^{\mathsf{I}} \to \dot{\mathsf{T}} \times_{\mathsf{T}} \dot{\mathsf{T}}$ is the relative pathobject of $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ over T.

In order to show that a natural model with path types also has intensional Identity types in the sense of Definition 2, we require an additional condition on the map $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$, namely that it is a "Hurewicz fibration", in the following sense.

Definition 4.4.16. A map $f : Y \to X$ will be called a *Hurewicz fibration* (with respect to an interval $d_0, d_1 : 1 \rightrightarrows \mathsf{I}$), if it has the right lifting property with respect to the 0-end inclusion of every cylinder $Z \times d_0 : Z \times 1 \to Z \times \mathsf{I}$. In detail, given any object Z and maps y and h as indicated below, there exists a diagonal filler $\tilde{h}$ making the following diagram commute.

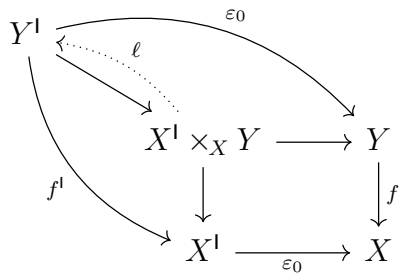
$$\begin{array}{ccc} Z \times 1 & \xrightarrow{y} & Y \\ {\scriptstyle Z \times d_0}\downarrow & \overset{\tilde{h}}{\nearrow} & \downarrow{\scriptstyle f} \\ Z \times \mathsf{I} & \xrightarrow[h]{} & X \end{array} \tag{4.17}$$

One regards $h : Z \times \mathsf{I} \to X$ as a homotopy between the maps $h_0, h_1 : Z \to X$ obtained by composing it with the two ends of the cylinder $Z \times 1 \rightrightarrows Z \times \mathsf{I}$, and $\tilde{h} : Z \times \mathsf{I} \to Y$ as a lift of h to the specified 0-end y. (Note that we do not need to require lifts against the 1-end inclusion $Z \times d_1 : Z \times 1 \to Z \times \mathsf{I}$.)

Proposition 4.4.17. *A map $f : Y \to X$ is a Hurewicz fibration just if the following diagram is a weak pullback.*

$$\begin{array}{ccc} Y^{\mathsf{I}} & \xrightarrow{\varepsilon_0} & Y \\ {\scriptstyle f^{\mathsf{I}}}\downarrow & & \downarrow{\scriptstyle f} \\ X^{\mathsf{I}} & \xrightarrow[\varepsilon_0]{} & X \end{array} \tag{4.18}$$

Proof. The diagram (4.18) is a weak pullback just if the comparison map to the actual pullback has a section $\ell : X^{\mathsf{I}} \times_X Y \to Y^{\mathsf{I}}$, as indicated below.

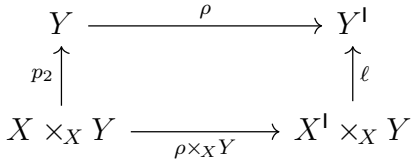


(4.19)

In terms of the so-called "Leibniz adjunction" $\otimes \dashv \Rightarrow$ (see [**?**]), the comparison map $\langle f^{\mathsf{I}}, \varepsilon_0 \rangle : Y^{\mathsf{I}} \to X^{\mathsf{I}} \times_X Y$ is the "pullback-hom", $\langle f^{\mathsf{I}}, \varepsilon_0 \rangle = d_0 \Rightarrow f$.

Now, an arbitrary map $g : A \to B$ has a section just if it lifts on the right against $0 \to Z$, for all objects Z (one can take $Z = B$). Thus $d_0 \Rightarrow f$ has a section just if $0_Z \pitchfork (d_0 \Rightarrow f)$ for all Z, which is equivalent by the adjunction to $(0_Z \otimes d_0) \pitchfork f$ for all Z. But we have $0_Z \otimes d_0 = Z \times d_0 : Z \to Z \times \mathsf{I}$. □

A section $\ell : X^{\mathsf{I}} \times_X Y \to Y^{\mathsf{I}}$ may be regarded as a "lifting operation" that takes a path $p : x_0 \rightsquigarrow x_1$ in X, and a point $y_0 \in Y$ over x_0, to a lifted path $\tilde{p} : y_0 \rightsquigarrow y$ in Y lying over p. A Hurewicz fibration $f : Y \to X$ will be called *normal* if it comes with such a lifting $\ell : X^{\mathsf{I}} \times_X Y \to Y^{\mathsf{I}}$ that takes an identity path in X to an identity path in Y, as indicated below.



(4.20)

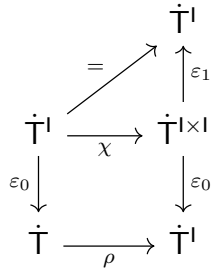
Now consider the following *axioms* for a natural model $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ with an interval $d_0, d_1 : 1 \rightrightarrows \mathsf{I}$.

(A1) $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ has path types, as in Definition 4.4.15.

(A2) $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ is a Hurewicz fibration, as in Definition 4.4.16.

We also assume that $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ is normal.

Lemma 4.4.18. *Assuming axioms (A1) and (A2), the map* $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ *has a* connection*: a map* $\chi : \dot{\mathsf{T}}^{\mathsf{I}} \to (\dot{\mathsf{T}}^{\mathsf{I}})^{\mathsf{I}} \cong \dot{\mathsf{T}}^{\mathsf{I} \times \mathsf{I}}$ *(over* T*) making the following commute.*



(4.21)

Moreover, the connection χ is normal *in the sense that $\chi\rho = \rho\rho$,*

$$\begin{array}{ccc} \dot{\mathsf{T}}^I & \xrightarrow{\chi} & \dot{\mathsf{T}}^{\mathsf{I}\times\mathsf{I}} \\ {\scriptstyle\rho}\uparrow & & \uparrow{\scriptstyle\rho} \\ \dot{\mathsf{T}} & \xrightarrow[\rho]{} & \dot{\mathsf{T}}^{\mathsf{I}}. \end{array} \tag{4.22}$$

Proof. We can use "box-filling" to construct the connection as the transpose of a certain diagonal filler

$$\tilde{\chi} : \dot{\mathsf{T}}^{\mathsf{I}} \times \mathsf{I} \to (\dot{\mathsf{T}}^{\mathsf{I}})$$

as follows. Given any pathspace $A^{\mathsf{I}} \to A \times A$ that is a Hurewicz fibration, consider a lifting problem of the form:

$$\begin{array}{ccc} 1 & \xrightarrow{r} & A^{\mathsf{I}} \\ {\scriptstyle\delta_0}\downarrow & \overset{\tilde{b}}{\nearrow} & \downarrow{\scriptstyle A^\partial} \\ \mathsf{I} & \xrightarrow[\langle q,p\rangle]{} & A \times A \end{array} \tag{4.23}$$

This can be regarded as a filler for an "open box" (p, q, r) in A to give a 2-cube $b : \mathsf{I}\times\mathsf{I} \to A$, which can be depicted as follows:

$$\begin{array}{ccc} q_0 & \xrightarrow{r} & p_0 \\ {\scriptstyle q}\downarrow & b & \downarrow{\scriptstyle p} \\ q_1 & \cdots\cdots\to & p_1 \end{array} \tag{4.24}$$

Now consider the case $q = \rho_{p_0} = r$ and take $\chi_p : \mathsf{I} \to A^{\mathsf{I}}$ to be the resulting box:

$$\begin{array}{ccc} p_0 & \xrightarrow{=} & p_0 \\ {\scriptstyle =}\downarrow & \chi_p & \downarrow{\scriptstyle p} \\ p_0 & \cdots\cdots\to & p_1 \end{array} \tag{4.25}$$

We apply this to the case where $A = \dot{\mathsf{T}}$ (over T), and with object of parameters $\dot{\mathsf{T}}^{\mathsf{I}}$, to obtain the following:

$$\begin{array}{ccc} \dot{\mathsf{T}}^{\mathsf{I}} \times 1 & \xrightarrow{r} & \dot{\mathsf{T}}^{\mathsf{I}} \\ {\scriptstyle \dot{\mathsf{T}}^{\mathsf{I}}\times\delta_0}\downarrow & \overset{\tilde{\chi}}{\nearrow} & \downarrow{\scriptstyle \dot{\mathsf{T}}^\partial} \\ \dot{\mathsf{T}}^{\mathsf{I}} \times \mathsf{I} & \xrightarrow[\langle q,p\rangle]{} & \dot{\mathsf{T}} \times \dot{\mathsf{T}} \end{array} \tag{4.26}$$

The maps p, q, r are as follows:

$$\begin{aligned} p &= \varepsilon \\ q &= \varepsilon(\rho\varepsilon_0 \times \mathsf{I}) \\ r &= \rho\varepsilon_0 \end{aligned}$$

Transposing provides the desired map $\chi : \dot{\mathsf{T}}^{\mathsf{I}} \to \dot{\mathsf{T}}^{\mathsf{I}\times\mathsf{I}}$ with evaluations $\chi(p)_0 = \rho p_0$ and $\chi(p)_1 = p$ for all $p : \dot{\mathsf{T}}^{\mathsf{I}}$. □

The Id-Elim rule now follows for any $A \to X$ that is classified by $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$, as follows. By (A2), $A \to X$ is Hurewicz (since it's a pullback of $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$), and by (A1) the pathtype $A^{\mathsf{I}} \to A \times_X A$ is also Hurewicz (since it's therefore a pullback of $\dot{\mathsf{T}}^{\mathsf{I}} \to \dot{\mathsf{T}} \times_{\mathsf{T}} \dot{\mathsf{T}}$). By Lemma 4.4.18 there is also a (normal) connection on $A \to X$ (again since it's a pullback of a map with a connection). Consider an Id-elimination problem as follows:

$$\begin{array}{ccc} A & \xrightarrow{c} & C \\ {\scriptstyle\rho}\downarrow & \overset{j}{\nearrow} & \downarrow{\scriptstyle\pi} \\ A^{\mathsf{I}} & \xrightarrow[=]{} & A^{\mathsf{I}} \end{array} \tag{4.27}$$

where $\pi : C \to A^{\mathsf{I}}$ is a pullback of $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$, and therefor a Hurewicz fibration, with (normal) lifting operation $\ell : (A^{\mathsf{I}})^{\mathsf{I}} \to C^{\mathsf{I}}$.

We argue first with elements to give the idea of the proof, before giving a diagrammatic version of the same argument. Thus take any $p : A^{\mathsf{I}}$, which is a path $p : p_0 \rightsquigarrow p_1$ in A. Applying the connection on A we obtain a (higher) path $\chi_p : \rho p_0 \rightsquigarrow p$ in A^{I}. Since the outer square of (4.27) commutes, for every $p_0 : A$ the term $cp_0 : C$ lies over ρp_0, thus we can lift χ_p to a path $\ell\langle \chi_p, cp_0\rangle : cp_0 \rightsquigarrow \tilde{p}$ in C, with the endpoint $\tilde{p}$ over p. We then set

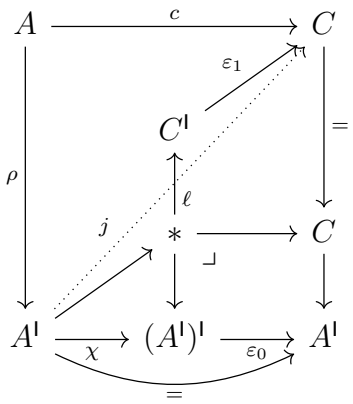
$$jp := \tilde{p} = \varepsilon_1 \ell \langle \chi_p, cp_0 \rangle \,,$$

which plainly makes the bottom triangle of (4.27) commute. Observe that in the case $p = \rho a$ for $a : A$, we have $\chi_{\rho a} = \rho_{\rho a}$ since χ is normal, and then for the lift we have $\ell\langle \chi_{\rho a}, ca\rangle = \ell\langle \rho_{\rho a}, ca\rangle = \rho ca$, since ℓ is normal. Whence $j\rho a = \varepsilon_1 \rho ca = ca$, as required for the top triangle of (4.27) to commute.

We summarize the result as follows.

Proposition 4.4.19. *Let $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ be a natural model in a category $\mathcal{E}$ with an interval $1 \rightrightarrows \mathsf{I}$, and assume axioms (A1) and (A2) above. Then $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ has Identity-types of the form $A \to A^{\mathsf{I}} \to A \times A$ validating the usual rules of intensional type theory.*

Proof. The foregoing "elementary" proof is depicted in the following diagram:



(4.28)

Where $* = (A^{\mathsf{I}})^{\mathsf{I}} \times_{A^{\mathsf{I}}} C$ is the indicated pullback, and the unlabelled map into it is the pair

$$\langle \chi, c\varepsilon_0 \rangle : A^{\mathsf{I}} \longrightarrow * \,.$$

We leave the diagram chase to the reader. □

Since the map $j = \varepsilon_1 \circ \ell \circ \langle \chi, c\varepsilon_0 \rangle$ is defined from others that are stable under pullback (themselves being pullbacks of a universal instance, defined from structure on $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$), the substitution condition with respect to a change of context $\sigma : X' \to X$ will obtain; that is we shall have:

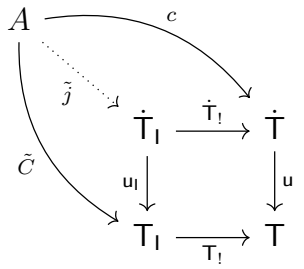
$$(j^c)_\sigma = j^{c_\sigma} : C_\sigma \,.$$

From this, it follows that we can determine a weak pullback structure map J as in (4.15), relating the polynomial functors associated with $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ and $\dot{\mathsf{T}}^{\mathsf{I}} \to \mathsf{T}$, evaluated at $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$, as in (4.14).

A simpler formulation of this condition is available in the case when the interval I is a *tiny* object, i.e. one for which the functor $(-)^{\mathsf{I}}$ of exponentiation has a right adjoint *root* functor $(-)_{\mathsf{I}}$, as obtains e.g. in the presheaves over the (finite product) category $\mathcal{C}$ of contexts. In that case, let us reformulate the elimination diagram (4.27) in the equivalent form:

$$\begin{array}{ccc} A & \xrightarrow{c} & \dot{\mathsf{T}} \\ {\scriptstyle\rho}\downarrow & \overset{j}{\nearrow} & \downarrow{\scriptstyle\mathsf{u}} \\ A^{\mathsf{I}} & \xrightarrow[C]{} & \mathsf{T} \end{array} \tag{4.29}$$

Then, writing $\rho = A^{!} : A \to A^{\mathsf{I}}$, for $! : \mathsf{I} \to 1$, we can transpose across the adjunction $(-)^{\mathsf{I}} \dashv (-)_{\mathsf{I}}$ to obtain the following equivalent problem:



(4.30)

Note that we are using the fact that the terminal object 1 is also tiny and $(-)_1 = \mathsf{id}$ is the identity functor, so $! : \mathsf{I} \to 1$ gives rise to a natural transformation $(-)_! : (-)_{\mathsf{I}} \to \mathsf{id}$.

The problem (4.30) can then be reformulated without reference to $A, \tilde{C}, c$ as stating that the inner square is a weak pullback, or, again equivalently, that there exists a section $\tilde{\mathsf{J}}$ of the comparison map $\langle \mathsf{u}_{\mathsf{I}}, \dot{\mathsf{T}}_! \rangle$ to the actual pullback,

$$\begin{array}{ccc} & \overset{\tilde{\mathsf{J}}}{\swarrow} & \\ \dot{\mathsf{T}}_{\mathsf{I}} & \longrightarrow & \mathsf{T}_{\mathsf{I}} \times_{\mathsf{T}} \dot{\mathsf{T}} \end{array} \tag{4.31}$$

This condition is clearly independent of any particular maps $A : X \to \mathsf{T}$, etc., into the object $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$, and therefore evidently satisfies the strict rule of coherence under substitution. Comparing the condition (4.31) with the similar (4.15), we have achieved a simplification in replacing the polynomial functors $P_{\mathsf{u}}, P_q : \hat{\mathcal{C}} \to \hat{\mathcal{C}}$ by the root functor $(-)_{\mathsf{I}} : \hat{\mathcal{C}} \to \hat{\mathcal{C}}$, which is available when an interval $1 \rightrightarrows \mathsf{I}$ is present in $\mathcal{C}$, and of course in replacing the *axioms* (4.13) and (4.15) by the axioms (A1) and (A2) above, from which (4.31) can be proven.

4.4.4 Strictification

Let $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$ be a natural model over a (small) finitely complete category $\mathbb{C}$, and consider the set

$$\mathcal{D}_{\mathsf{t}} \subseteq \mathbb{C}_1$$

of all $d : D' \to D$ in $\mathbb{C}$ that occur as pullbacks of t, as in

$$\begin{array}{ccc} \mathsf{y}D' & \longrightarrow & \dot{\mathsf{T}} \\ {\scriptstyle \mathsf{y}d}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{t}} \\ \mathsf{y}D & \longrightarrow & \mathsf{T} \end{array}$$

Clearly $\mathcal{D}_{\mathsf{t}}$ is closed under pullbacks in $\mathbb{C}$ and isos in the arrow category $\mathbb{C}^{\downarrow}$, and so $(\mathbb{C}, \mathcal{D}_{\mathsf{t}})$ is a *category with display maps* in the sense of [Tay99](§8.3). Such a pair $(\mathbb{C}, \mathcal{D})$ consisting of a finitely complete category $\mathbb{C}$ with a "stable" class of maps $\mathcal{D}$ can be used to basic model dependent type theory as a common generalization of semantics in an LCCC and in a category with an wfs. Additional conditions on $(\mathbb{C}, \mathcal{D})$ are of course needed to model the different type formers, leading to such notions as *clans* and *tribes* [Joy17].

Given a category with display maps $(\mathbb{C}, \mathcal{D})$ we can form the natural transformation $\mathsf{t}_{\mathcal{D}}$ over $\mathbb{C}$ simply by taking the coproduct of all the display maps $d : D' \to D$ in $\mathcal{D}$:

$$\begin{array}{ccc} \dot{\mathsf{T}}_{\mathcal{D}} & = & \coprod_{d \in \mathcal{D}} D' \\ {\scriptstyle \mathsf{t}_{\mathcal{D}}}\downarrow & & \downarrow{\scriptstyle \coprod_{d \in \mathcal{D}} d} \\ \mathsf{T}_{\mathcal{D}} & = & \coprod_{d \in \mathcal{D}} D \end{array} \tag{4.32}$$

The proof of following is an easy exercise.

Lemma 4.4.20. *The natural transformation $\mathsf{t}_{\mathcal{D}} : \dot{\mathsf{T}}_{\mathcal{D}} \to \mathsf{T}_{\mathcal{D}}$ is representable.*

Exercise 4.4.21. Give the proof!

We now have constructions going back and forth:

$$\mathsf{NaturalModels} \underset{\mathcal{D}_{\mathsf{t}}}{\overset{\mathsf{t}_{\mathcal{D}}}{\leftrightarrows}} \mathsf{Cats\ w/DisplayMaps}$$

By defining suitable morphisms on both sides, these operations can be made functorial, and the constructions become adjoint, $\mathsf{t}_{\mathcal{D}} \dashv \mathcal{D}_{\mathsf{t}}$. We can leave the details to the reader.

Finally, by considering the appropriate additional structures on a category with display maps in order to model the type formers (for example local cartesian closure or a right proper weak factorization system), one can refine the adjunction to give a proof of the following "strictification theorem" (cf. [LW15]):

Theorem 4.4.22 ([Awo16])**.** *If $(\mathbb{C}, \mathcal{D})$ is a Π-tribe in the sense of Joyal [Joy17] then $\mathsf{t}_{\mathcal{D}} : \dot{\mathsf{T}}_{\mathcal{D}} \to \mathsf{T}_{\mathcal{D}}$ is a natural model with the type constructors $1, \Sigma, \Pi, \mathtt{Id}$.*

4.5 Universes

We recall the notion of a *Hofmann-Streicher universe* [HS], as reformulated in [Awo24].

4.5.1 A realization $\dashv$ nerve adjunction

For any presheaf X on a small category $\mathbb{C}$, recall that the category of elements is the comma category,

$$\textstyle\int_{\mathbb{C}} X \;=\; \mathsf{y}_{\mathbb{C}}/_X \,,$$

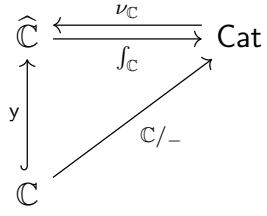
where $\mathsf{y}_{\mathbb{C}} : \mathbb{C} \to \mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}$ is the Yoneda embedding, which we may supress and write simply as $\mathbb{C}/_X$. The following can be found already in [Gro83](§28).

Proposition 4.5.1. *The category of elements functor $\int_{\mathbb{C}} : \widehat{\mathbb{C}} \longrightarrow \mathsf{Cat}$ has a right adjoint, which we denote*

$$\nu_{\mathbb{C}} : \mathsf{Cat} \longrightarrow \widehat{\mathbb{C}} \,.$$

For a small category $\mathbb{A}$, we call the presheaf $\nu_{\mathbb{C}}(\mathbb{A})$ the $\mathbb{C}$-nerve *of $\mathbb{A}$.*

Proof. As suggested by the name, the adjunction $\int_{\mathbb{C}} \dashv \nu_{\mathbb{C}}$ can be seen as the familiar "realization $\dashv$ nerve" construction with respect to the covariant post-composition functor $\mathbb{C}/_{\_} : \mathbb{C} \to \mathsf{Cat}$, as indicated below.



(4.33)

In detail, for $\mathbb{A} \in \mathsf{Cat}$ and $c \in \mathbb{C}$, let $\nu_{\mathbb{C}}(\mathbb{A})(c)$ be the Hom-set of functors,

$$\nu_{\mathbb{C}}(\mathbb{A})(c) = \mathsf{Cat}\big(\mathbb{C}/_c \,,\, \mathbb{A}\big) \,,$$

with contravariant action on $h : d \to c$ given by pre-composing a functor $P : \mathbb{C}/_c \to \mathbb{A}$ with the post-composition functor

$$\mathbb{C}/_h : \mathbb{C}/_d \longrightarrow \mathbb{C}/_c \,.$$

For the adjunction, observe that the slice category $\mathbb{C}/_c$ is the category of elements of the representable functor $\mathsf{y}c$,

$$\textstyle\int_{\mathbb{C}} \mathsf{y}c \;\cong\; \mathbb{C}/_c \,.$$

Thus for representables $\mathsf{y}c$, we indeed have the required natural isomorphism

$$\widehat{\mathbb{C}}\big(\mathsf{y}c\,,\, \nu_{\mathbb{C}}(\mathbb{A})\big) \;\cong\; \nu_{\mathbb{C}}(\mathbb{A})(c) \;=\; \mathsf{Cat}\big(\mathbb{C}/_c\,,\, \mathbb{A}\big) \;\cong\; \mathsf{Cat}\big(\textstyle\int_{\mathbb{C}} \mathsf{y}c\,,\, \mathbb{A}\big)\,.$$

For arbitrary presheaves X, one uses the presentation of X as a colimit of representables over the index category $\int_{\mathbb{C}} X$, and the easy to prove fact that $\int_{\mathbb{C}}$ itself preserves colimits. Indeed, for any category $\mathbb{D}$, we have an isomorphism in Cat,

$$\varinjlim_{d \in \mathbb{D}} \mathbb{D}/_d \;\cong\; \mathbb{D}\,.$$

□

When $\mathbb{C}$ is fixed, as here, we may omit the subscript from the expressions $\mathsf{y}_{\mathbb{C}}$ and $\int_{\mathbb{C}}$ and $\nu_{\mathbb{C}}$. The unit and counit maps of the adjunction $\int \dashv \nu$,

$$\begin{aligned} \eta &: X \longrightarrow \nu \textstyle\int X\,, \\ \epsilon &: \textstyle\int \nu \mathbb{A} \longrightarrow \mathbb{A}\,, \end{aligned}$$

are as follows. At $c \in \mathbb{C}$, for $x : \mathsf{y}c \to X$, the functor $(\eta_X)_c(x) : \mathbb{C}/_c \to \mathbb{C}/_X$ is just composition with x,

$$(\eta_X)_c(x) = \mathbb{C}/_x : \mathbb{C}/_c \longrightarrow \mathbb{C}/_X \,. \tag{4.34}$$

For $\mathbb{A} \in \mathsf{Cat}$, the functor $\epsilon : \int \nu \mathbb{A} \to \mathbb{A}$ takes a pair $(c \in \mathbb{C}, f : \mathbb{C}/_c \to \mathbb{A})$ to the object $f(1_c) \in \mathbb{A}$,

$$\epsilon(c, f) = f(1_c).$$

Lemma 4.5.2. *For any $f : Y \to X$, the naturality square below is a pullback.*

$$\begin{array}{ccc} Y & \xrightarrow{\eta_Y} & \nu \textstyle\int Y \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \nu \int f} \\ X & \xrightarrow[\eta_X]{} & \nu \textstyle\int X. \end{array} \tag{4.35}$$

Proof. It suffices to prove this for the case $f : X \to 1$. Thus consider the square

$$\begin{array}{ccc} X & \xrightarrow{\eta_X} & \nu \textstyle\int X \\ \downarrow & & \downarrow \\ 1 & \xrightarrow[\eta_1]{} & \nu \textstyle\int 1. \end{array} \tag{4.36}$$

Evaluating at $c \in \mathbb{C}$ and applying (4.34) gives the following square in Set.

$$\begin{array}{ccc} Xc & \xrightarrow{\mathbb{C}/_-} & \mathsf{Cat}\big(\mathbb{C}/_c\,,\, \mathbb{C}/_X\big) \\ \downarrow & & \downarrow \\ 1c & \xrightarrow[\mathbb{C}/_-]{} & \mathsf{Cat}\big(\mathbb{C}/_c\,,\, \mathbb{C}/_1\big) \end{array} \tag{4.37}$$

The image of $* \in 1c$ along the bottom is the forgetful functor $U_c : \mathbb{C}/_c \to \mathbb{C}$, and its fiber under the map on the right is therefore the set of functors $F : \mathbb{C}/_c \to \mathbb{C}/_X$ such that $U_X \circ F = U_c$, where $U_X : \mathbb{C}/_X \to \mathbb{C}$ is also a forgetful functor. But any such F is easily seen to be uniquely of the form $\mathbb{C}/_x$ for $x = F(1_c) : \mathsf{y}c \to X$. □

Remark 4.5.3. For the category of elements of the terminal presheaf 1 we have $\int 1 \cong \mathbb{C}$. So for every presheaf X there is a canonical projection $\int X \to \mathbb{C}$, and the functor $\int : \widehat{\mathbb{C}} \to \mathsf{Cat}$ thus factors through the slice category $\mathsf{Cat}/_{\mathbb{C}}$.

$$\begin{array}{ccc} \widehat{\mathbb{C}} & \xrightarrow{\int/_1} & \mathsf{Cat}/_{\mathbb{C}} \\ & \searrow^{\int} & \downarrow{\scriptstyle \mathbb{C}_!} \\ & & \mathsf{Cat} \end{array} \tag{4.38}$$

The adjunction $\int \dashv \nu : \mathsf{Cat} \to \widehat{\mathbb{C}}$ factors as well, but it is the unfactored adjunction that is more useful for the present purpose.

4.5.2 Classifying families

For every presheaf X the canonical projection $\int X \to \mathbb{C}$ of Remark 4.5.3 is easily seen to be a discrete fibration. It follows that for any natural transformation $Y \to X$ the associated functor $\int Y \to \int X$ is also a discrete fibration. Ignoring size issues (dealt with in [Awo24]), recall that discrete fibrations in Cat are classified by the forgetful functor $\dot{\mathsf{Set}}^{\mathsf{op}} \to \mathsf{Set}^{\mathsf{op}}$ from (the opposites of) the category of pointed sets to that of sets (cf. [Web07]). For every presheaf $X \in \widehat{\mathbb{C}}$, we therefore have a pullback diagram as follows in Cat.

$$\begin{array}{ccc} \int X & \longrightarrow & \dot{\mathsf{Set}}^{\mathsf{op}} \\ \downarrow & \lrcorner & \downarrow \\ \mathbb{C} & \xrightarrow[X]{} & \mathsf{Set}^{\mathsf{op}} \end{array} \tag{4.39}$$

Using $\int 1 \cong \mathbb{C}$ and transposing by the adjunction $\int \dashv \nu$ then gives a commutative square in $\widehat{\mathbb{C}}$,

$$\begin{array}{ccc} X & \longrightarrow & \nu\dot{\mathsf{Set}}^{\mathsf{op}} \\ \downarrow & & \downarrow \\ 1 & \xrightarrow[\tilde{X}]{} & \nu\mathsf{Set}^{\mathsf{op}}. \end{array} \tag{4.40}$$

Lemma 4.5.4. *The square* (4.40) *is a pullback in* $\widehat{\mathbb{C}}$*. More generally, for any map* $Y \to X$ *in* $\widehat{\mathbb{C}}$*, there is a pullback square*

$$\begin{array}{ccc} Y & \longrightarrow & \nu\dot{\mathsf{Set}}^{\mathsf{op}} \\ \downarrow & \lrcorner & \downarrow \\ X & \longrightarrow & \nu\mathsf{Set}^{\mathsf{op}}\,. \end{array} \tag{4.41}$$

Proof. Apply the right adjoint ν to the pullback square (4.39) and paste the naturality square (4.35) from Lemma 4.5.2 on the left, to obtain the transposed square (4.41) as a pasting of two pullbacks. □

Let us write $\dot{\mathsf{V}} \to \mathsf{V}$ for the vertical map on the right in (4.41); that is, let:

$$\begin{aligned} \dot{\mathsf{V}} &:= \nu\dot{\mathsf{Set}}^{\mathsf{op}} \\ \mathsf{V} &:= \nu\mathsf{Set}^{\mathsf{op}} \end{aligned} \tag{4.42}$$

We can then summarize our results so far as follows.

Proposition 4.5.5. *The* $\mathbb{C}$*-nerve* $\dot{\mathsf{V}} \to \mathsf{V}$ *of the classifier* $\dot{\mathsf{Set}}^{\mathsf{op}} \to \mathsf{Set}^{\mathsf{op}}$ *for discrete fibrations in* Cat*, as defined in* (4.42)*, (weakly) classifies natural transformations* $Y \to X$ *in presheaves* $\widehat{\mathbb{C}}$*, in the sense that there is always a pullback square as follows.*

$$\begin{array}{ccc} Y & \longrightarrow & \dot{\mathsf{V}} \\ \downarrow & \lrcorner & \downarrow \\ X & \xrightarrow[\tilde{Y}]{} & \mathsf{V} \end{array} \tag{4.43}$$

The classifying map $\tilde{Y} : X \to \mathsf{V}$ *is determined by the adjunction* $\int \dashv \nu$ *as the transpose of the classifying map of the discrete fibration* $\int Y \to \int X$*.*

Of course, as defined in (4.42), the classifier $\dot{\mathsf{V}} \to \mathsf{V}$ cannot be a map in $\widehat{\mathbb{C}}$, for reasons of size; this is addressed in [Awo24].

4.5.3 Type universes

A universe such as $\dot{\mathsf{V}} \to \mathsf{V}$ in a category of presheaves (or indeed a natural model $\mathsf{t} : \dot{\mathsf{T}} \to \mathsf{T}$) can be used to model the following *rules for type universes* in Martin-Löf type theory.

$$\frac{}{\Gamma \vdash \mathsf{U}\ \mathtt{type}} \qquad \frac{\Gamma \vdash a : \mathsf{U}}{\Gamma \vdash \mathsf{El}(a)\ \mathtt{type}}$$

Of course, we interpret the "decoding family" $x : \mathsf{U} \vdash \mathsf{El}(x)$ as the display map $\dot{\mathsf{V}} \to \mathsf{V}$. In practice, one often simply writes

$$\frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash A\ \mathtt{type}}$$

leaving the "decoding" El implicit. The above are the formation and elimination rules for a universe; one might also expect to see an introduction rule such as

$$\frac{\Gamma \vdash A\ \mathtt{type}}{\Gamma \vdash \mathsf{code}(A) : \mathsf{U}}$$

along with computation rules like

$$\mathsf{El}(\mathsf{code}(A)) \equiv A\,, \quad \mathsf{code}(\mathsf{El}(a)) \equiv a\,.$$

Unfortunately, this would imply $\mathsf{U} : \mathsf{U}$, which is inconsistent. Instead of the introduction and computation rules, there are other ways of populating a universe; see [AG](§2.6.2) for a good discussion.

A type universe U is convenient, because it can be used to replace a dependent family of types $\Gamma, x : X \vdash A\ \mathtt{type}$ by a term $\Gamma \vdash A : X \to \mathsf{U}$. It can also used in further constructions, such as the polynomials $\mathsf{P}_{\mathsf{t}}(\mathsf{T})$ from section 4.4.1. See [AG](§2.6.2) for some other important consequences of adding type universes to the type theory, such as proving $\mathtt{true} \neq \mathtt{false} : \mathtt{Bool}$.

Given a universe U, we may regard the types $A : \mathsf{U}$ as *small*. A universe of types interpreted as $\dot{\mathsf{V}} \to \mathsf{V}$ then acts as a "small type classifier," because a display map $A \to X$ interpreting the small type family $X \vdash A\ \mathtt{type}$, is classified by a map $\tilde{A} : X \to \mathsf{V}$, interpreting the term $\vdash A : X \to \mathsf{V}$, as in (4.43).

$$\begin{array}{ccc} A & \longrightarrow & \dot{\mathsf{V}} \\ \downarrow & \lrcorner & \downarrow \\ X & \xrightarrow[\tilde{A}]{} & \mathsf{V} \end{array} \tag{4.44}$$

Unlike with subobject classifiers, however, the classifying map $\tilde{A} : X \to \mathsf{V}$ of a given family $A \to X$ cannot be expected to be unique, essentially because pullbacks are determined only up to isomorphism. That such maps can be chosen uniquely "up to homotopy," in a certain precise sense, is the content of the celebrated *univalence axiom*, to which we now turn.

4.6 Univalence

Let $\dot{\mathsf{U}} \to \mathsf{U}$ be a universe in presheaves, e.g., a natural model as in Section 4.4, or a Hofmann-Streicher universe as in Section 4.5, and suppose that $\dot{\mathsf{U}} \to \mathsf{U}$ supports the structures for $1, \Sigma, \Pi, \mathtt{Id}$ in the sense of Sections 4.4.1–4.4.2. It follows that the maps $A \to X$ that are classified by $\dot{\mathsf{U}} \to \mathsf{U}$, in the sense of (4.44), are closed under those same type formers, in the sense of the "incoherent" interpretation in LCCCs, wfs's, or categories with display maps, regarding objects X as contexts and classified maps $A \to X$ (together with a chosen classifier $\tilde{A} : X \to \mathsf{U}$) as families of types in context X.

An *equivalence* between types $A \simeq B$ (possibly over X) is like an isomorphism of sets, but formulated in a way that is more suitable for types with "higher structure" arising from non-degenerate $\mathtt{Id}$-types. Formally:

Definition 4.6.1. A map $e : A \to B$ is a (type) *equivalence* if it has both right and left inverses, in the sense that there are maps $f, g : B \rightrightarrows A$ such that

$$e \circ f =_{B^B} 1_B \qquad \text{and} \qquad g \circ e =_{A^A} 1_A \,.$$

Formally, given $e : A \to B$ we define

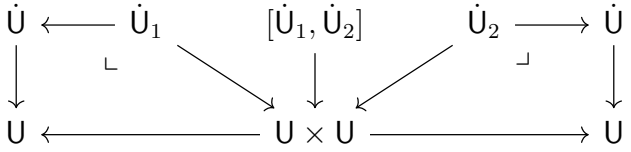
$$\mathsf{isEquiv}(e) \;:\equiv\; \Sigma_{f,g:B\to A}\, \mathtt{Id}_{B^B}(e \circ f, 1_B) \times \mathtt{Id}_{A^A}(g \circ e, 1_A) \,.$$

and then for $A, B : \mathsf{U}$, we define the notation $A \simeq B$ by

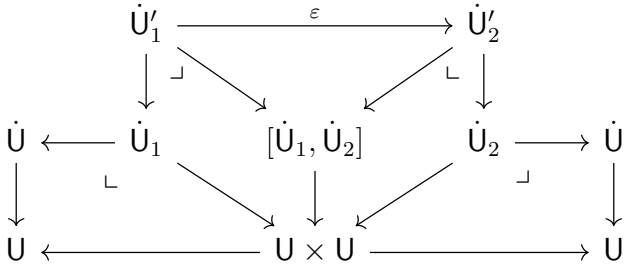
$$A \simeq B \;=\; \mathsf{Equiv}(A, B) \;:\equiv\; \Sigma_{e:A\to B}\, \mathsf{isEquiv}(e) \,.$$

We construct the interpretation of the type family $A, B : \mathsf{U} \vdash A \simeq B$ as a map $\mathsf{Equiv} \to \mathsf{U} \times \mathsf{U}$ as follows.

First, pull $\dot{\mathsf{U}} \to \mathsf{U}$ back along the two different projections $\mathsf{U} \times \mathsf{U} \rightrightarrows \mathsf{U}$ to obtain two different objects $\dot{\mathsf{U}}_1, \dot{\mathsf{U}}_2$ over $\mathsf{U} \times \mathsf{U}$, and then take their exponential $[\dot{\mathsf{U}}_1, \dot{\mathsf{U}}_2]$ in the slice category over $\mathsf{U} \times \mathsf{U}$, which interprets the type family $A, B : \mathsf{U} \vdash A \to B$.



Indeed, after pulling $\dot{\mathsf{U}}_1, \dot{\mathsf{U}}_2$ back to $[\dot{\mathsf{U}}_1, \dot{\mathsf{U}}_2]$ there is a (universal) arrow $\varepsilon : \dot{\mathsf{U}}'_1 \to \dot{\mathsf{U}}'_2$ over $[\dot{\mathsf{U}}_1, \dot{\mathsf{U}}_2]$, as indicated below.

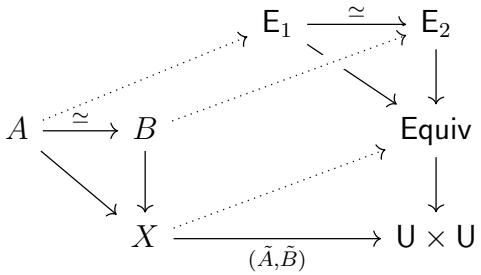


(4.45)

As $\mathsf{Equiv} \to \mathsf{U} \times \mathsf{U}$ we then take the composite of the canonical projection $\mathsf{isEquiv}(\varepsilon) \to [\dot{\mathsf{U}}_1, \dot{\mathsf{U}}_2]$ with the map $[\dot{\mathsf{U}}_1, \dot{\mathsf{U}}_2] \to \mathsf{U} \times \mathsf{U}$.

$$\begin{array}{ccc} \Sigma_{e:A\to B}\mathsf{isEquiv}(e) & \longrightarrow & [\dot{\mathsf{U}}_1, \dot{\mathsf{U}}_2] \\ \| & & \downarrow \\ \mathsf{Equiv} & \longrightarrow & \mathsf{U} \times \mathsf{U} \end{array}$$

Given $\tilde{A}, \tilde{B} : X \to \mathsf{U}$ classifying types $A \to X$ and $B \to X$, factorizations of the map $(\tilde{A}, \tilde{B}) : X \to \mathsf{U} \times \mathsf{U}$ through $\mathsf{Equiv} \to \mathsf{U} \times \mathsf{U}$ can then be seen using (4.45) to correspond to equivalences $A \simeq B$ over X, as indicated in the following, in which the indicated map $\mathsf{E}_1 \simeq \mathsf{E}_2$ is the pullback of $\varepsilon : \dot{\mathsf{U}}_1' \to \dot{\mathsf{U}}_2'$ along $\mathsf{Equiv} \to [\dot{\mathsf{U}}_1, \dot{\mathsf{U}}_2]$.



The correspondence is natural in X, because it is mediated by pulling back a universal instance.

If U is itself a type in a "larger" model $\dot{\mathsf{U}}_1 \to \mathsf{U}_1$ with $\mathtt{Id}$-types, then there is also the type family $A, B : \mathsf{U} \vdash \mathtt{Id}_{\mathsf{U}}(A, B)$, which is interpreted as a map $\mathtt{Id}_{\mathsf{U}} \to \mathsf{U} \times \mathsf{U}$. Since for every $A : \mathsf{U}$ the map $1_A : A \to A$ is an equivalence, by $\mathtt{Id}_{\mathsf{U}}$-elim we obtain a distinguished map $\mathtt{Id}_{\mathsf{U}}(A, B) \to \mathsf{Equiv}(A, B)$ over $\mathsf{U} \times \mathsf{U}$. The *univalence axiom* is the statement that this map is itself an equivalence:

$$\begin{array}{ccc} \mathtt{Id}_{\mathsf{U}} & \xrightarrow{\simeq} & \mathsf{Equiv} \\ \downarrow & & \downarrow \\ \mathsf{U} \times \mathsf{U} & = & \mathsf{U} \times \mathsf{U} \end{array} \tag{4.46}$$

Pulling the equivalence back along any $(A, B) : X \to \mathsf{U} \times \mathsf{U}$ results in the more familiar formulation:

$$(A =_{\mathsf{U}} B) \simeq (A \simeq B)\,. \tag{UA}$$

The interpretation of univalence thus requires at least a universe U with $1, \Sigma, \Pi, \mathtt{Id}$ (in order to define $A \simeq B$), inside a larger universe $\mathsf{U} : \mathsf{U}_1$ also with $1, \Sigma, \Pi, \mathtt{Id}$ (in order to have the family $A =_{\mathsf{U}} B$ and to state the central equivalence in UA). The operations on the larger universe are usually required to restrict to the corresponding ones on the smaller universe. See [AG] for a detailed presentation.

Example 4.6.2 (Univalence in the groupoid model [HS98])**.** We have a univalent universe $\dot{\mathsf{U}} \to \mathsf{U}$ in the groupoid model of section 4.2. Namely, the groupoid set of *small, discrete groupoids*, where we identify sets with their discrete groupoids, and by *small* we mean sets of size $\leq \kappa$ for a sufficiently large cardinal bound κ. Thus we interpret:

$$\mathsf{U} = \mathsf{set}^{\mathsf{core}}$$
$$\dot{\mathsf{U}} = \dot{\mathsf{set}}^{\mathsf{core}}$$

where $\mathsf{set}^{\mathsf{core}}$ is the *groupoid core* of the category of small sets (i.e. the category of small sets and bijections between them), and $\dot{\mathsf{set}}^{\mathsf{core}}$ is the groupoid of small pointed sets.

The interpretation of the $\mathtt{Id}$-type of the universe U in Gpd, like that of any type, is the path groupoid

$$\mathtt{Id}_{\mathsf{U}} = \mathsf{U}^{\mathsf{I}} \to \mathsf{U} \times \mathsf{U}\,,$$

which in this case is the small discrete groupoid of isos of small sets, and therefore corresponds to the functor $\mathtt{Id}_{\mathsf{U}} : \mathsf{U} \times \mathsf{U} \to \mathsf{Gpd}$, with discrete groupoids as its values, namely for small sets A, B, the small discrete groupoid of isos $i : A \cong B$,

$$\mathtt{Id}_{\mathsf{U}}(A, B) = \{i : A \cong B\}\,. \tag{4.47}$$

Now consider the interpretation of the type $\mathsf{Equiv} \to \mathsf{U} \times \mathsf{U}$. Unpacking the definition 4.6.1 in this case, for small sets A, B, we have the groupoid:

$$\begin{aligned} \mathsf{Equiv}(A, B) &= \Sigma_{e:A\to B}\mathsf{isEquiv}(e) \\ &= \Sigma_{e:A\to B}\,\Sigma_{f,g:B\to A}\,\mathtt{Id}_{B^B}(e \circ f, 1_B) \times \mathtt{Id}_{A^A}(g \circ e, 1_A)\,. \end{aligned}$$

The comparison map $j : \mathtt{Id}_{\mathsf{U}} \to \mathsf{Equiv}$ (sometimes called "IdtoEq") over $\mathsf{U} \times \mathsf{U}$ results as the $\mathtt{J}$-term in the following $\mathtt{Id}$-elim

$$\begin{array}{ccc} \mathsf{U} & \xrightarrow{\;e\;} & \mathsf{Equiv} \\ {\scriptstyle \mathtt{refl}}\downarrow & \overset{\mathtt{J}}{\nearrow} & \downarrow \\ \mathtt{Id}_U & \longrightarrow & \mathsf{U} \times \mathsf{U} \end{array}$$

where $e : \mathsf{U} \to \mathsf{Equiv}$ is the map that takes a small set A to its identity map $1_A : A \to A$, along with 1_A itself as left- and right-inverses. Note that the map $\mathtt{refl} : \mathsf{U} \to \mathtt{Id}_U = \mathsf{U}^{\mathsf{I}}$ is indeed an injective-on-objects equivalence and $\mathsf{Equiv} \to \mathsf{U} \times \mathsf{U}$ is a fibration of groupoids.

Exercise 4.6.3. Prove that $\mathsf{U} \to \mathsf{U}^{\mathsf{I}}$ is an injective-on-objects equivalence, and the projection $\mathsf{Equiv} \to \mathsf{U} \times \mathsf{U}$ is a (split) fibration of groupoids, as just claimed.

Finally, to verify the univalence axiom, we must prove that $j : \mathtt{Id}_U \to \mathsf{Equiv}$ is an equivalence of groupoids. It suffices to show this fiberwise $j_{(A,B)} : \mathtt{Id}_U(A, B) \to \mathsf{Equiv}(A, B)$, and since A, B are both discrete groupoids, we have $\mathsf{Equiv}(A, B) \simeq \mathsf{Iso}(A, B)$. But $\mathsf{Iso}(A, B)$ is exactly the interpretation of $\mathtt{Id}_{\mathsf{U}}(A, B)$, by (4.47), and $j(i : A \cong B)$ is just (i, i^{-1}, i^{-1}), so we do indeed have

$$(A =_{\mathsf{U}} B) \;=\; \mathtt{Id}_U(A, B) \;\simeq\; \mathsf{Equiv}(A, B) \;=\; (A \simeq B)\,.$$

Bibliography

[ADHS01] T. Altenkirch, P. Dybjer, M. Hofmann, and P. Scott. Normalization by evaluation for typed lambda calculus with coproducts. In *Proceedings 16th Annual IEEE Symposium on Logic in Computer Science (LICS 2001)*, pages 303–310, 2001.

[AG] C. Angiuli and D. Gratzer. Principles of dependent type theory. Online at `https://carloangiuli.com/courses/b619-sp24/notes.pdf`. Version 2024-11-26.

[AGH24] S. Awodey, N. Gambino, and S. Hazratpour. Kripke-Joyal forcing for type theory and uniform fibrations, 2024. Preprint available as `https://arxiv.org/abs/2110.14576`.

[AGS17] Steve Awodey, Nicola Gambino, and Kristina Sojakova. Homotopy-initial algebras in type theory. 2017.

[AR94] Jiri Adamek and Jiri Rosicky. *Locally Presentable and Accessible Categories*. Number 189 in London Mathematical Society Lecture Notes. Cambridge University Press, 1994.

[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.

[AW09] Steve Awodey and Michael A. Warren. Homotopy theoretic models of identity types. *Math. Proc. Cambridge Philos. Soc.*, 146(1):45–55, 2009.

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

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

[Awo10] Steve Awodey. *Category Theory*. Number 52 in Oxford Logic Guides. Oxford University Press, 2010.

[Awo12] Steve Awodey. Type theory and homotopy. In Peter Dybjer, Sten Lindström, Erik Palmgren, and Göran Sundholm, editors, *Epistemology Versus Ontology: Essays on the Philosophy and Foundations of Mathematics in Honour of Per Martin-Löf*, pages 183–201. Springer, 2012. arXiv:1010.1810.

[Awo16] Steve Awodey. Natural models of homotopy type theory. *Mathematical Structures in Computer Science*, 28:1–46, 11 2016.

[Awo24] Steve Awodey. On Hofmann-Streicher universes. *Mathematical Structures in Computer Science*, 34:894–910, 2024.

[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`.

[Coq22] Thierry Coquand. Type theory, 2022. The Stanford Encyclopedia of Philosophy, `https://plato.stanford.edu/archives/fall2022/entries/type-theory`.

[Ded88] R. Dedekind. *Was sind und was sollen die Zahlen?* Vieweg, 1888.

[Dyb96] P. Dybjer. Internal type theory. *LNCS*, 1158:120–134, 1996.

[FDCB02] M. Fiore, R. Di Cosmo, and V. Balat. Remarks on isomorphisms in typed lambda calculi with empty and sum types. In *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science (LICS 2002)*, pages 147–156, 2002.

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

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

[GG08] Nicola Gambino and Richard Garner. The identity type weak factorisation system. *Theoretical Computer Science*, 409(1):94–109, 2008.

[GK13] Nicola Gambino and Joachim Kock. Polynomial functors and polynomial monads. *Mathematical Proceedings of the Cambridge Philosophical Society*, 154(1):153–192, 2013.

[GL23] N. Gambino and M.F. Larrea. Models of Martin-Löf type theory from algebraic weak factorisation systems. *The Journal of Symbolic Logic*, 88(1):242–289, March 2023.

[Gro83] Alexander Grothendieck. Pursuing stacks. unpublished, 1983.

[Hof94] Martin Hofmann. On the interpretation of type theory in locally cartesian closed categories. *Lecture Notes in Computer Science*, 933:427–441, 1994.

[Hof95] Martin Hofmann. Syntax and semantics of dependent types. In *Semantics and logics of computation*, volume 14 of *Publ. Newton Inst.*, pages 79–130. Cambridge University Press, Cambridge, 1995.

[How80] William A. Howard. The formulae-as-types notion of construction. In Jonathan P. Seldin and J. Roger Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. 1980. Reprinted from 1969 manuscript.

[HS] Martin Hofmann and Thomas Streicher. Lifting Grothendieck universes. unpublished, dated Spring 1997.

[HS98] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 83–111. Oxford Univ. Press, New York, 1998.

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

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

[Joy17] André Joyal. Notes on clans and tribes, 2017. unpublished `https://arxiv.org/abs/1710.10238`.

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

[Law63] F. W. Lawvere. Functorial semantics of algebraic theories. Ph.D. thesis, Columbia University, 1963.

[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).

[Law70] F.W. Lawvere. Equality in hyperdoctrines and comprehension schema as an adjoint functor. *Proceedings of the AMS Symposium on Pure Mathematics XVII*, pages 1–14, 1970.

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

[LW15] Peter Lefanu Lumsdaine and Michael A. Warren. The local universes model: An overlooked coherence construction for dependent type theories. *ACM Trans. Comput. Logic*, 16(3), 2015.

[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.

[MM92] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic. A First Introduction to Topos Theory*. Springer-Verlag, New York, 1992.

[MP00] Ieke Moerdijk and Erik Palmgren. Wellfounded trees in categories. *Annals of Pure and Applied Logic*, 104(1):189–218, 2000.

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

[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.

[Rij25] Egbert Rijke. *Introduction to Homotopy Type Theory*. Cambridge University Press, 2025.

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

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

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

[See84] R. A. G. Seely. Locally cartesian closed categories and type theory. *Mathematical Proceedings of the Cambridge Philosophical Society*, 95(1):33, 1984.

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

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

[Tay99] Paul Taylor. *Practical Foundations of Mathematics*. Cambridge University Press, 1999.

[Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.

[Web07] Mark Weber. Yoneda structures from 2-toposes. *Applied Categorical Structures*, 15, 2007.