Notes on Type Theory

[DRAFT: January 16, 2025]

Steve Awodey

with contributions from Andrej Bauer

Contents

Chapter 1

Introduction

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 [Coq].

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.

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

This deduction shows that

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

But so does the following:

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

As does:

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

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

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

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

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

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

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:

$$\frac{\begin{array}{c}\Phi, [\phi] \\ \vdots \\ \psi\end{array}}{\phi \Rightarrow \psi}$$

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.

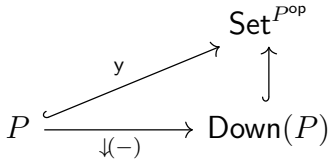
Proof. The total downset P is obviously terminal, and for any downsets $S, T \in \mathsf{Down}(P)$, the intersection $S \cap T$ is also closed down, so we have the products $S \wedge T = S \cap T$. For the exponential, let

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

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 **??**).

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. Of course, one can 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; but then the ordering on $\mathsf{Pos}(P, \mathbb{2})$ will be the *reverse pointwise ordering* of maps $f : P \to \mathbb{2}$.

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.*

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 S to the free Heyting algebra $H(S)$ generated by it. As for all algebraic structures, the construction of $H(S)$ can be performed in two steps: first, define a set $H[S]$ 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 $H[S]$ be the set of formal expressions generated inductively by the following rules:

1. Generators: if $x \in S$ then $x \in H[S]$.
2. Constants: $\bot, \top \in H[S]$.
3. Connectives: if $\phi, \psi \in H[S]$ then $(\phi \wedge \psi), (\phi \vee \psi), (\phi \Rightarrow \psi) \in H[S]$.

We then impose an equivalence relation $\sim$ on $H[S]$, 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

$$H(S) = H[S]/_{\sim}$$

to be a Heyting algebra, as is easily checked.

We define the action of the functor H on morphisms as usual: a function $f : S \to T$ is mapped to the Heyting algebra homomorphism $H(f) : H(S) \to H(T)$ (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 $\eta_S : S \to UH(S)$ into the underlying set of the free Heyting algebra $H(S)$ is then the component at S 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 $f : S \to U\mathcal{K}$. Then the Heyting algebra homomorphism $\overline{f} : H(S) \to \mathcal{K}$ 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:

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

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

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, ..., 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 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 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* can 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 1.5.14. 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 1.5.15. 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 1.5.16. Prove 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 1.5.17. 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:

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

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* 1.5.18. 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 1.5.19. 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:

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

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 1.5.20 (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 **??** 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. Transposing the composite of this iso with the Stone representation $\eta : \mathcal{B} \rightarrowtail \mathcal{P}(\mathsf{BA}(\mathcal{B}, \mathbb{2}))$ in Pos,

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

we arrive at the (monotone) evaluation map

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

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 1.5.21 (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,

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

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

Lemma 1.5.22 (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 $I_0 \subseteq I_1 \subseteq \ldots$ 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 1.5.21. 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),

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

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 1.5.22). Thus it only remains to show that

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

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

$$f(p \Rightarrow q) = 1 \quad \text{iff} \quad \text{for all } g \geq f,\ g(p) = 1 \text{ implies } g(q) = 1. \tag{1.17}$$

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

$$p \Rightarrow q \in F \quad \text{iff} \quad \text{for all prime } G \supseteq F,\ p \in G \text{ implies } q \in G. \tag{1.18}$$

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 classical case of "truth tables" results by considering *arbitrary* assignments from $\mathsf{Var} = \{p_0, p_1, \ldots\}$ to truth values $\mathbb{2}$, which correspond to Boolean homomorphisms from the free Boolean algebra $\mathcal{B}(p_0, p_1, \ldots)$ and therefore to maximal filters in $\mathcal{B}(p_0, p_1, \ldots)$. Joyal's representation theorem then agrees with that of Stone, and the resulting completeness theorem is then the classical one for CPC.

Corollary 1.5.23 (Completeness for classical PC)**.** *The classical propositional calculus is deductively complete with respect to truth-valued semantics ("truth tables").*

Exercise 1.5.24. Give a Kripke countermodel to show that the Law of Excluded Middle $\phi \vee \neg\phi$ 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

Appendix A

Category Theory

A.1 Categories

Definition A.1.1. A *category* $\mathcal{C}$ consists of classes

$\mathcal{C}_0$ of objects A, B, C, ...
$\mathcal{C}_1$ of morphisms f, g, h, ...

such that:

- Each morphism f has uniquely determined *domain* $\mathsf{dom}\, f$ and *codomain* $\mathsf{cod}\, f$, which are objects. This is written:
$$f : \mathsf{dom}\, f \to \mathsf{cod}\, f$$
- For any morphisms $f : A \to B$ and $g : B \to C$ there exists a uniquely determined *composition* $g \circ f : A \to C$. Composition is associative:
$$h \circ (g \circ f) = (h \circ g) \circ f \ ,$$
where domains are codomains are as follows:
$$A \xrightarrow{\ f\ } B \xrightarrow{\ g\ } C \xrightarrow{\ h\ } D$$
- For every object A there exists the *identity* morphism $1_A : A \to A$ which is a unit for composition,
$$1_A \circ f = f \ , \qquad\qquad g \circ 1_A = g \ ,$$
where $f : B \to A$ and $g : A \to C$.

Morphisms are also called *arrows* or *maps*. Note that morphisms do not actually have to be functions, and objects need not be sets or spaces of any sort. We often write $\mathcal{C}$ instead of $\mathcal{C}_0$.

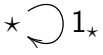
Definition A.1.2. A category $\mathcal{C}$ is *small* when the objects $\mathcal{C}_0$ and the morphisms $\mathcal{C}_1$ are sets (as opposed to proper classes). A category is *locally small* when for all objects $A, B \in \mathcal{C}_0$ the class of morphisms with domain A and codomain B, written $\mathsf{Hom}(A, B)$ or $\mathcal{C}_0(A, B)$, is a set.

We normally restrict attention to locally small categories, so unless we specify otherwise all categories are taken to be locally small. Next we consider several examples of categories.

A.1.1 Examples

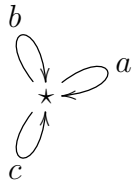
The empty category $\mathsf{0}$ The empty category has no objects and no arrows.

The unit category $\mathsf{1}$ The unit category, also called the terminal category, has one object $\star$ and one arrow $1_\star$:



Other finite categories There are other finite categories, for example the category with two objects and one (non-identity) arrow, and the category with two parallel arrows:

Groups as categories Every group $(G, \cdot)$, is a category with a single object $\star$ and each element of G as a morphism:



$$a, b, c, \ldots \in G$$

The composition of arrows is given by the group operation:

$$a \circ b = a \cdot b$$

The identity arrow is the group unit e. This is indeed a category because the group operation is associative and the group unit is the unit for the composition. In order to get a category, we do not actually need to know that every element in G has an inverse. It suffices to take a *monoid*, also known as *semigroup*, which is an algebraic structure with an associative operation and a unit.

We can turn things around and *define* a monoid to be a category with a single object. A group is then a category with a single object in which every arrow is an *isomorphism* (in the sense of definition A.1.5 below).

Posets as categories Recall that a *partially ordered set*, or *poset* $(P, \leq)$, is a set with a reflexive, transitive, and antisymmetric relation:

$$\begin{aligned} x &\leq x && \text{(reflexive)} \\ x \leq y \;\&\; y \leq z &\;\Rightarrow\; x \leq z && \text{(transitive)} \\ x \leq y \;\&\; y \leq x &\;\Rightarrow\; x = y && \text{(antisymmetric)} \end{aligned}$$

Each poset is a category whose objects are the elements of P, and there is a single arrow $p \to q$ between $p, q \in P$ if, and only if, $p \leq q$. Composition of $p \to q$ and $q \to r$ is the unique arrow $p \to r$, which exists by transitivity of $\leq$. The identity arrow on p is the unique arrow $p \to p$, which exists by reflexivity of $\leq$.

Antisymmetry tells us that any two isomorphic objects in P are equal.[1] We do not need antisymmetry in order to obtain a category, i.e., a *preorder* would suffice.

Again, we may *define* a preorder to be a category in which there is at most one arrow between any two objects. A poset is a skeletal preorder, i.e. one in which the only isomorphisms are the identity arrows. We allow for the possibility that a preorder or a poset is a proper class rather than a set.

A particularly important example of a poset category is the poset of open sets $\mathcal{O}X$ of a topological space X, ordered by inclusion.

Sets as categories Any set S is a category whose objects are the elements of S and whose only arrows are identity arrows. Such a category, in which the only arrows are the identity arrows, is called a *discrete category*.

A.1.2 Categories of structures

In general, structures like groups, topological spaces, posets, etc., determine categories in which the maps are structure-preserving functions, composition is composition of functions, and identity morphisms are identity functions:

- Group is the category whose objects are groups and whose morphisms are group homomorphisms.
- Top is the category whose objects are topological spaces and whose morphisms are continuous maps.
- Set is the category whose objects are sets and whose morphisms are functions.[2]
- Graph is the category of (directed) graphs an graph homomorphisms.
- Poset is the category of posets and monotone maps.

[1] A category in which isomorphic object are equal is a *skeletal* category.

[2] A function between sets A and B is a relation $f \subseteq A \times B$ such that for every $x \in A$ there exists a unique $y \in B$ for which $\langle x, y \rangle \in f$. A morphism in Set is a triple $\langle A, f, B \rangle$ such that $f \subseteq A \times B$ is a function.

Such categories of structures are generally *large*, but locally small. Note that it is not necessary to check the associative and unit laws for such categories of functions (why?), unlike the following example.

Exercise A.1.3. The *category of relations* Rel has as objects all sets $A, B, C, \ldots$ and as arrows $A \to B$ the relations $R \subseteq A \times B$. The composite of $R \subseteq A \times B$ and $S \subseteq B \times C$, and the identity arrow on A, are defined by:

$$S \circ R = \left\{ \langle x, z \rangle \in A \times C \;\middle|\; \exists\, y \in B \,.\, xRy \;\&\; ySz \right\},$$
$$1_A = \left\{ \langle x, x \rangle \;\middle|\; x \in A \right\}.$$

Show that this is indeed a category!

A.1.3 Basic notions

We recall some further basic notions from category theory.

Definition A.1.4. A *subcategory* $\mathcal{C}'$ of a category $\mathcal{C}$ is given by a subclass of objects $\mathcal{C}'_0 \subseteq \mathcal{C}_0$ and a subclass of morphisms $\mathcal{C}'_1 \subseteq \mathcal{C}_1$ such that $f \in \mathcal{C}'_1$ implies $\mathsf{dom}\, f, \mathsf{cod}\, f \in \mathcal{C}'_0$, $1_A \in \mathcal{C}'_1$ for every $A \in \mathcal{C}'_0$, and $g \circ f \in \mathcal{C}'_1$ whenever $f, g \in \mathcal{C}'_1$ are composable.

A subcategory $\mathcal{C}'$ of $\mathcal{C}$ is *full* if for all $A, B \in \mathcal{C}'_0$, we have $\mathcal{C}'(A, B) = \mathcal{C}(A, B)$, i.e. every $f : A \to B$ in $\mathcal{C}_1$ is also in $\mathcal{C}'_1$.

Definition A.1.5. An *inverse* of a morphism $f : A \to B$ is a morphism $f^{-1} : B \to A$ such that

$$f \circ f^{-1} = 1_B \qquad \text{and} \qquad f^{-1} \circ f = 1_A \,.$$

A morphism that has an inverse is an *isomorphism*, or *iso*. If there exists a pair of mutually inverse morphisms $f : A \to B$ and $f^{-1} : B \to A$ we say that the objects A and B are *isomorphic*, written $A \cong B$.

The notation f^{-1} is justified because an inverse, if it exists, is unique. A *left inverse* is a morphism $g : B \to A$ such that $g \circ f = 1_A$, and a *right inverse* is a morphism $g : B \to A$ such that $f \circ g = 1_B$. A left inverse is also called a *retraction*, whereas a right inverse is called a *section*.

Definition A.1.6. A *monomorphism*, or *mono*, is a morphism $f : A \to B$ that can be cancelled on the left: for all $g : C \to A$, $h : C \to A$,

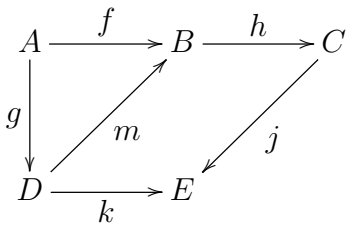
$$f \circ g = f \circ h \Rightarrow g = h \,.$$

An *epimorphism*, or *epi*, is a morphism $f : A \to B$ that can be cancelled on the right: for all $g : B \to C$, $h : B \to A$,

$$g \circ f = h \circ f \Rightarrow g = h \,.$$

In Set monomorphisms are the injective functions and epimorphisms are the surjective functions. Isomorphisms in Set are the bijective functions. Thus, in Set a morphism is iso if, and only if, it is both mono and epi. However, this example is misleading! In general, a morphism can be mono and epi without being an iso. For example, the non-identity morphism in the category consisting of two objects and one morphism between them is both epi and mono, but it has no inverse. A more interesting example of morphisms that are both epi and mono but are not iso occurs in the category Top of topological spaces and continuous maps, where not every continuous bijection is a homeomorphism.

A *diagram* of objects and morphisms is a directed graph whose vertices are objects of a category and edges are morphisms between them, for example:



Such a diagram is said to *commute* when the composition of morphisms along any two paths with the same beginning and end gives equal morphisms. Commutativity of the above diagram is equivalent to the following two equations:

$$f = m \circ g \ , \qquad\qquad k = j \circ h \circ m \ .$$

From these we can derive $k \circ g = j \circ h \circ f$ by a *diagram chase*.

A.2 Functors

Definition A.2.1. A *functor* $F : \mathcal{C} \to \mathcal{D}$ from a category $\mathcal{C}$ to a category $\mathcal{D}$ consists of functions

$$F_0 : \mathcal{C}_0 \to \mathcal{D}_0 \qquad \text{and} \qquad F_1 : \mathcal{C}_1 \to \mathcal{D}_1$$

such that, for all $f : A \to B$ and $g : B \to C$ in $\mathcal{C}$:

$$\begin{aligned} F_1 f : F_0 A \to F_0 B \ , \\ F_1(g \circ f) = (F_1 g) \circ (F_1 f) \ , \\ F_1(1_A) = 1_{F_0 A} \ . \end{aligned}$$

We usually write F for both F_0 and F_1.

A functor is thus a homomorphism of the category structure; note that it maps commutative diagrams to commutative diagrams because it preserves composition.

We may form the "category of categories" Cat whose objects are small categories and whose morphisms are functors. Composition of functors is composition of the corresponding functions, and the identity functor is one that is identity on objects and on morphisms. The category Cat is large but locally small.

Definition A.2.2. A functor $F : \mathcal{C} \to \mathcal{D}$ is *faithful* when it is "locally injective on morphisms", in the sense that for all $f, g : A \to B$, if $Ff = Fg$ then $f = g$.

A functor $F : \mathcal{C} \to \mathcal{D}$ is *full* when it is "locally surjective on morphisms": for every $g : FA \to FB$ there exists $f : A \to B$ such that $g = Ff$.

We consider several examples of functors.

A.2.1 Functors between sets, monoids and posets

When sets, monoids, groups, and posets are regarded as categories, the functors turn out to be the *usual morphisms*, for example:

- A functor between sets S and T is a function from S to T.
- A functor between groups G and H is a group homomorphism from G to H.
- A functor between posets P and Q is a monotone function from P to Q.

Exercise A.2.3. Verify that the above claims are correct.

A.2.2 Forgetful functors

For categories of structures Group, Top, Graph, Poset, ..., there is a *forgetful* functor U which maps an object to the underlying set and a morphism to the underlying function. For example, the forgetful functor $U : \mathsf{Group} \to \mathsf{Set}$ maps a group $(G, \cdot)$ to the set G and a group homomorphism $f : (G, \cdot) \to (H, \star)$ to the function $f : G \to H$.

There are also forgetful functors that forget only part of the structure, for example the forgetful functor $U : \mathsf{Ring} \to \mathsf{Group}$ which maps a ring $(R, +, \times)$ to the additive group $(R, +)$ and a ring homomorphism $f : (R, +_R, \cdot_S) \to (S, +_S, \cdot_S)$ to the group homomorphism $f : (R, +_R) \to (S, +_S)$. Note that there is another forgetful functor $U' : \mathsf{Ring} \to \mathsf{Mon}$ from rings to monoids.

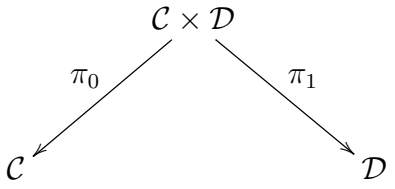
Exercise A.2.4. Show that taking the graph $\Gamma(f) = \{\langle x, f(x)\rangle \mid x \in A\}$ of a function $f : A \to B$ determines a functor $\Gamma : \mathsf{Set} \to \mathsf{Rel}$, from sets and functions to sets and relations, which is the identity on objects. Is this a forgetful functor?

A.3 Constructions of Categories and Functors

A.3.1 Product of categories

Given categories $\mathcal{C}$ and $\mathcal{D}$, we form the *product category* $\mathcal{C} \times \mathcal{D}$ whose objects are pairs of objects $\langle C, D\rangle$ with $C \in \mathcal{C}$ and $D \in \mathcal{D}$, and whose morphisms are pairs of morphisms $\langle f, g\rangle : \langle C, D\rangle \to \langle C', D'\rangle$ with $f : C \to C'$ in $\mathcal{C}$ and $g : D \to D'$ in $\mathcal{D}$. Composition is given by $\langle f, g\rangle \circ \langle f', g'\rangle = \langle f \circ f', g \circ g'\rangle$.

There are evident *projection* functors



which act as indicated in the following diagrams:

Exercise A.3.1. Show that, for any categories $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$, there are distinguished isos:

$$\mathbf{1} \times \mathbb{C} \cong \mathbb{C}$$
$$\mathbb{B} \times \mathbb{C} \cong \mathbb{C} \times \mathbb{B}$$
$$\mathbb{A} \times (\mathbb{B} \times \mathbb{C}) \cong (\mathbb{A} \times \mathbb{B}) \times \mathbb{C}$$

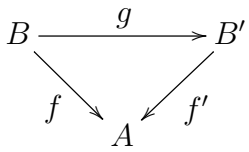
Does this make Cat a (commutative) monoid?

A.3.2 Slice categories

Given a category $\mathcal{C}$ and an object $A \in \mathcal{C}$, the *slice* category $\mathcal{C}/A$ has as objects, morphisms into A,

$$\begin{array}{c} B \\ \downarrow f \\ A \end{array} \tag{A.1}$$

and as morphisms, commutative diagrams over A:



(A.2)

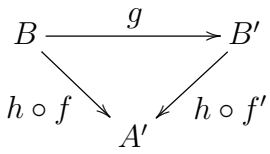
That is, a morphism from $f : B \to A$ to $f' : B' \to A$ is a morphism $g : B \to B'$ such that $f = f' \circ g$. Composition of morphisms in $\mathcal{C}/A$ is composition of morphisms in $\mathcal{C}$.

There is a forgetful functor $U_A : \mathcal{C}/A \to \mathcal{C}$ which maps an object (A.1) to its domain B, and a morphism (A.2) to the morphism $g : B \to B'$.

Furthermore, for each morphism $h : A \to A'$ in $\mathcal{C}$ there is a functor "composition by h",

$$\mathcal{C}/h : \mathcal{C}/A \to \mathcal{C}/A'$$

which maps an object (A.1) to the object $h \circ f : B \to A'$ and a morphisms (A.2) to the morphism

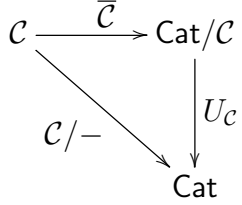


The construction of slice categories is itself a functor

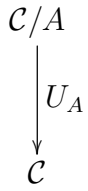
$$\mathcal{C}/- : \mathcal{C} \to \mathsf{Cat}$$

provided that $\mathcal{C}$ is small. This functor maps each $A \in \mathcal{C}$ to the category $\mathcal{C}/A$ and each morphism $h : A \to A'$ to the composition functor $\mathcal{C}/h : \mathcal{C}/A \to \mathcal{C}/A'$.

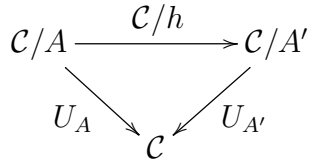
Since Cat is itself a category, we may form the slice category $\mathsf{Cat}/\mathcal{C}$ for any small category $\mathcal{C}$. The slice functor $\mathcal{C}/-$ then factors through the forgetful functor $U_{\mathcal{C}} : \mathsf{Cat}/\mathcal{C} \to \mathsf{Cat}$ via a functor $\overline{\mathcal{C}} : \mathcal{C} \to \mathsf{Cat}/\mathcal{C}$,



where for $A \in \mathcal{C}$, the object part $\overline{\mathcal{C}}A$ is

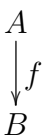


and for $h : A \to A'$ in $\mathcal{C}$, the morphism part $\overline{\mathcal{C}}h$ is



A.3.3 Arrow categories

Similar to the slice categories, an arrow category has arrows as objects, but without a fixed codomain. Given a category $\mathcal{C}$, the *arrow* category $\mathcal{C}^{\to}$ has as objects the morphisms of $\mathcal{C}$,



(A.3)

and as morphisms $f \to f'$ the commutative squares,

$$\begin{array}{ccc} A & \xrightarrow{g} & A' \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f'} \\ B & \xrightarrow[g']{} & B'. \end{array} \tag{A.4}$$

That is, a morphism from $f : A \to B$ to $f' : A' \to B'$ is a pair of morphisms $g : A \to A'$ and $g' : B \to B'$ such that $g' \circ f = f' \circ g$. Composition of morphisms in $\mathcal{C}^{\to}$ is just componentwise composition of morphisms in $\mathcal{C}$.

There are two evident forgetful functors $U_1, U_2 : \mathcal{C}^{\to} \to \mathcal{C}$, given by the domain and codomain operations. (Can you find a common section for these?)

A.3.4 Opposite categories

For a category $\mathcal{C}$ the *opposite category* $\mathcal{C}^{\mathsf{op}}$ has the same objects as $\mathcal{C}$, but all the morphisms are turned around, that is, a morphism $f : A \to B$ in $\mathcal{C}^{\mathsf{op}}$ is a morphism $f : B \to A$ in $\mathcal{C}$. The identity arrows in $\mathcal{C}^{\mathsf{op}}$ are the same as in $\mathcal{C}$, but the order of composition is reversed. The opposite of the opposite of a category is clearly the original category.

A functor $F : \mathcal{C}^{\mathsf{op}} \to \mathcal{D}$ is sometimes called a *contravariant functor* (from $\mathcal{C}$ to $\mathcal{D}$), and a functor $F : \mathcal{C} \to \mathcal{D}$ is a *covariant* functor.

For example, the opposite category of a preorder $(P, \leq)$ is the preorder P turned upside down, $(P, \geq)$.

Exercise A.3.2. Given a functor $F : \mathcal{C} \to \mathcal{D}$, can you define a functor $F^{\mathsf{op}} : \mathcal{C}^{\mathsf{op}} \to \mathcal{D}^{\mathsf{op}}$ in such a way that $-^{\mathsf{op}}$ itself becomes a functor? On what category is it a functor?

A.3.5 Representable functors

Let $\mathcal{C}$ be a locally small category. Then for each pair of objects $A, B \in \mathcal{C}$ the collection of all morphisms $A \to B$ forms a set, written $\mathsf{Hom}_{\mathcal{C}}(A, B)$, $\mathsf{Hom}(A, B)$ or $\mathcal{C}(A, B)$. For every $A \in \mathcal{C}$ there is a functor

$$\mathcal{C}(A, -) : \mathcal{C} \to \mathsf{Set}$$

defined by

$$\begin{aligned} \mathcal{C}(A, B) &= \big\{ f \in \mathcal{C}_1 \;\big|\; f : A \to B \big\} \\ \mathcal{C}(A, g) &: f \mapsto g \circ f \end{aligned}$$

where $B \in \mathcal{C}$ and $g : B \to C$. In words, $\mathcal{C}(A, g)$ is composition by g. This is indeed a functor because, for any morphisms

$$A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D \tag{A.5}$$

we have

$$\mathcal{C}(A, h \circ g)f = (h \circ g) \circ f = h \circ (g \circ f) = \mathcal{C}(A, h)(\mathcal{C}(A, g)f) \ ,$$

and $\mathcal{C}(A, 1_B)f = 1_A \circ f = f = 1_{\mathcal{C}(A,B)}f$.

We may also ask whether $\mathcal{C}(-, B)$ is a functor. If we define its action on morphisms to be precomposition,

$$\mathcal{C}(f, B) : g \mapsto g \circ f \ ,$$

it becomes a *contravariant* functor,

$$\mathcal{C}(-, B) : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set} \ .$$

The contravariance is a consequence of precomposition; for morphisms (A.5) we have

$$\mathcal{C}(g \circ f, D)h = h \circ (g \circ f) = (h \circ g) \circ f = \mathcal{C}(f, D)(\mathcal{C}(g, D)h) \ .$$

A functor of the form $\mathcal{C}(A, -)$ is a *(covariant) representable functor*, and a functor of the form $\mathcal{C}(-, B)$ is a *(contravariant) representable functor*.

It follows that the hom-set is a functor

$$\mathcal{C}(-, -) : \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathsf{Set}$$

which maps a pair of objects $A, B \in \mathcal{C}$ to the set $\mathcal{C}(A, B)$ of morphisms from A to B, and it maps a pair of morphisms $f : A' \to A$, $g : B \to B'$ in $\mathcal{C}$ to the function

$$\mathcal{C}(f, g) : \mathcal{C}(A, B) \to \mathcal{C}(A', B')$$

defined by

$$\mathcal{C}(f, g) : h \mapsto g \circ h \circ f \ .$$

(Why does it follow that this is a functor?)

A.3.6 Group actions

A group $(G, \cdot)$ is a category with one object $\star$ and elements of G as the morphisms. Thus, a functor $F : G \to \mathsf{Set}$ is given by a set $F\star = S$ and for each $a \in G$ a function $Fa : S \to S$ such that, for all $x \in S$, $a, b \in G$,

$$(Fe)x = x \ , \qquad\qquad (F(a \cdot b))x = (Fa)((Fb)x) \ .$$

Here e is the unit element of G. If we write $a \cdot x$ instead of $(Fa)x$, the above two equations become the familiar laws for a *left group action on the set* S:

$$e \cdot x = x \ , \qquad\qquad (a \cdot b) \cdot x = a \cdot (b \cdot x) \ .$$

Exercise A.3.3. A *right group action* by a group $(G, \cdot)$ on a set S is an operation $\cdot : S \times G \to S$ that satisfies, for all $x \in S$, $a, b \in G$,

$$x \cdot e = x \ , \qquad\qquad x \cdot (a \cdot b) = (x \cdot a) \cdot b \ .$$

Exhibit right group actions as functors.

A.4 Natural Transformations and Functor Categories

Definition A.4.1. Let $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{C} \to \mathcal{D}$ be functors. A *natural transformation* $\eta : F \Longrightarrow G$ from F to G is a map $\eta : \mathcal{C}_0 \to \mathcal{D}_1$ which assigns to every object $A \in \mathcal{C}$ a morphism $\eta_A : FA \to GA$, called the *component of* η *at* A, such that for every $f : A \to B$ in $\mathcal{C}$ we have $\eta_B \circ Ff = Gf \circ \eta_A$, i.e., the following diagram in $\mathcal{D}$ commutes:

$$\begin{array}{ccc} FA & \xrightarrow{\eta_A} & GA \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FB & \xrightarrow[\eta_B]{} & GB \end{array}$$

A simple example is given by the "twist" isomorphism $t : A \times B \to B \times A$ (in Set). Given any maps $f : A \to A'$ and $g : B \to B'$, there is a commutative square:

$$\begin{array}{ccc} A \times B & \xrightarrow{t_{A,B}} & B \times A \\ {\scriptstyle f \times g}\downarrow & & \downarrow{\scriptstyle g \times f} \\ A' \times B' & \xrightarrow[t_{A',B'}]{} & B' \times A' \end{array}$$

Thus naturality means that the two *functors* $F(X, Y) = X \times Y$ and $G(X, Y) = Y \times X$ are related to each other (by $t : F \to G$), and not simply their individual values $A \times B$ and $B \times A$. As a further example of a natural transformation, consider groups G and H as categories and two homomorphisms $f, g : G \to H$ as functors between them. A natural transformation $\eta : f \Longrightarrow g$ is given by a single element $\eta_\star = b \in H$ such that, for every $a \in G$, the following diagram commutes:

$$\begin{array}{ccc} \star & \xrightarrow{b} & \star \\ {\scriptstyle fa}\downarrow & & \downarrow{\scriptstyle ga} \\ \star & \xrightarrow[b]{} & \star \end{array}$$

This means that $b \cdot fa = (ga) \cdot b$, that is $ga = b \cdot (fa) \cdot b^{-1}$. In other words, a natural transformation $f \Longrightarrow g$ is a *conjugation* operation $b^{-1} \cdot - \cdot b$ which transforms f into g.

For every functor $F : \mathcal{C} \to \mathcal{D}$ there exists the *identity transformation* $\mathbf{1}_F : F \Longrightarrow F$ defined by $(\mathbf{1}_F)_A = \mathbf{1}_A$. If $\eta : F \Longrightarrow G$ and $\theta : G \Longrightarrow H$ are natural transformations, then their composition $\theta \circ \eta : F \Longrightarrow H$, defined by $(\theta \circ \eta)_A = \theta_A \circ \eta_A$ is also a natural transformation. Composition of natural transformations is associative because it is composition in the codomain category $\mathcal{D}$. This leads to the definition of functor categories.

Definition A.4.2. Let $\mathcal{C}$ and $\mathcal{D}$ be categories. The *functor category* $\mathcal{D}^{\mathcal{C}}$ is the category whose objects are functors from $\mathcal{C}$ to $\mathcal{D}$ and whose morphisms are natural transformations between them.

A functor category may be quite large, too large in fact. In order to avoid problems with size we normally require $\mathcal{C}$ to be a locally small category. The "hom-class" of all natural transformations $F \Longrightarrow G$ is usually written as

$$\mathsf{Nat}(F, G)$$

instead of the more awkward $\mathsf{Hom}_{\mathcal{D}^{\mathcal{C}}}(F, G)$.

Suppose we have functors F, G, and H with a natural transformation $\theta : G \Longrightarrow H$, as in the following diagram:

$$\mathcal{C} \xrightarrow{F} \mathcal{D} \underset{H}{\overset{G}{\Downarrow \theta}} \mathbb{E}$$

Then we can form a natural transformation $\theta \circ F : G \circ F \Longrightarrow H \circ F$ whose component at $A \in \mathcal{C}$ is $(\theta \circ F)_A = \theta_{FA}$.

Similarly, if we have functors and a natural transformation

$$\mathcal{C} \underset{H}{\overset{G}{\Downarrow \theta}} \mathcal{D} \xrightarrow{F} \mathbb{E}$$

we can form a natural transformation $(F \circ \theta) : F \circ G \Longrightarrow F \circ H$ whose component at $A \in \mathcal{C}$ is $(F \circ \theta)_A = F\theta_A$. These operations are known as *whiskering*.

A *natural isomorphism* is an isomorphism in a functor category. Thus, if $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{C} \to \mathcal{D}$ are two functors, a natural isomorphism between them is a natural transformation $\eta : F \Longrightarrow G$ whose components are isomorphisms. In this case, the inverse natural transformation $\eta^{-1} : G \Longrightarrow F$ is given by $(\eta^{-1})_A = (\eta_A)^{-1}$. We write $F \cong G$ when F and G are naturally isomorphic.

The definition of natural transformations is motivated in part by the fact that, for any small categories $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$, we have

$$\mathsf{Cat}(\mathbb{A} \times \mathbb{B}, \mathbb{C}) \cong \mathsf{Cat}(\mathbb{A}, \mathbb{C}^{\mathbb{B}}) \; . \tag{A.6}$$

The isomorphism takes a functor $F : \mathbb{A} \times \mathbb{B} \to \mathbb{C}$ to the functor $\widetilde{F} : \mathbb{A} \to \mathbb{C}^{\mathbb{B}}$ defined on objects $A \in \mathbb{A}$, $B \in \mathbb{B}$ by

$$(\widetilde{F}A)B = F\langle A, B\rangle$$

and on a morphism $f : A \to A'$ by

$$(\widetilde{F}f)_B = F\langle f, 1_B\rangle \; .$$

The functor $\widetilde{F}$ is called the *transpose* of F.

The inverse isomorphism takes a functor $G : \mathbb{A} \to \mathbb{C}^{\mathbb{B}}$ to the functor $\widetilde{G} : \mathbb{A} \times \mathbb{B} \to \mathbb{C}$, defined on objects by

$$\widetilde{G}\langle A, B\rangle = (GA)B$$

and on a morphism $\langle f, g\rangle : A \times B \to A' \times B'$ by

$$\widetilde{G}\langle f, g\rangle = (Gf)_{B'} \circ (GA)g = (GA')g \circ (Gf)_B \ ,$$

where the last equation holds by naturality of Gf:

$$\begin{array}{ccc} (GA)B & \xrightarrow{(Gf)_B} & (GA')B \\ {\scriptstyle (GA)g}\downarrow & & \downarrow{\scriptstyle (GA')g} \\ (GA)B' & \xrightarrow[(Gf)_{B'}]{} & (GA')B' \end{array}$$

A.4.1 Directed graphs as a functor category

Recall that a *directed graph* G is given by a set of vertices G_V and a set of edges G_E. Each edge $e \in G_E$ has a uniquely determined *source* $\mathsf{src}_G\, e \in G_V$ and *target* $\mathsf{trg}_G\, e \in G_V$. We write $e : a \to b$ when a is the source and b is the target of e. A *graph homomorphism* $\phi : G \to H$ is a pair of functions $\phi_0 : G_V \to H_V$ and $\phi_1 : G_E \to H_E$, where we usually write ϕ for both ϕ_0 and ϕ_1, such that whenever $e : a \to b$ then $\phi_1 e : \phi_0 a \to \phi_0 b$. The category of directed graphs and graph homomorphisms is denoted by Graph.

Now let $\cdot \rightrightarrows \cdot$ be the category with two objects and two parallel morphisms, depicted by the following "sketch":

$$E \underset{t}{\overset{s}{\rightrightarrows}} V$$

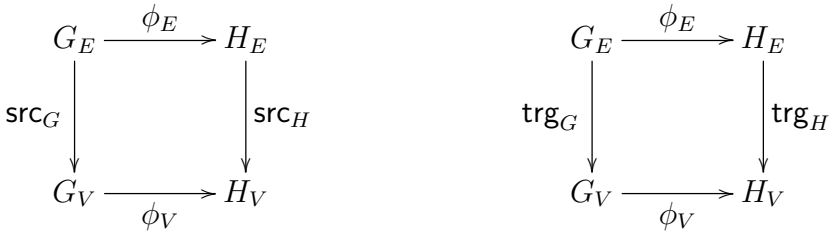
An object of the functor category $\mathsf{Set}^{\cdot\rightrightarrows\cdot}$ is a functor $G : (\cdot \rightrightarrows \cdot) \to \mathsf{Set}$, which consists of two sets GE and GV and two functions $Gs : GE \to GV$ and $Gt : GE \to GV$. But this is precisely a directed graph whose vertices are GV, the edges are GE, the source of $e \in GE$ is $(Gs)e$ and the target is $(Gt)e$. Conversely, any directed graph G is a functor $G : (\cdot \rightrightarrows \cdot) \to \mathsf{Set}$, defined by

$$GE = G_E \ , \qquad GV = G_V \ , \qquad Gs = \mathsf{src}_G \ , \qquad Gt = \mathsf{trg}_G \ .$$

Now category theory begins to show its worth, for the morphisms in $\mathsf{Set}^{\cdot\rightrightarrows\cdot}$ are precisely the graph homomorphisms. Indeed, a natural transformation $\phi : G \Longrightarrow H$ between graphs is a pair of functions,

$$\phi_E : G_E \to H_E \qquad \text{and} \qquad \phi_V : G_V \to H_V$$

whose naturality is expressed by the commutativity of the following two diagrams:



This is precisely the requirement that $e : a \to b$ implies $\phi_E e : \phi_V a \to \phi_V b$. Thus, in sum, we have,

$$\mathsf{Graph} = \mathsf{Set}^{\cdot \rightrightarrows \cdot}.$$

Exercise A.4.3. Exhibit the arrow category $\mathcal{C}^{\to}$ and the category of group actions $\mathsf{Set}(G)$ as functor categories.

A.4.2 The Yoneda embedding

The example $\mathsf{Graph} = \mathsf{Set}^{\cdot \rightrightarrows \cdot}$ leads one to wonder which categories $\mathcal{C}$ can be represented as functor categories $\mathsf{Set}^{\mathcal{D}}$ for a suitably chosen $\mathcal{D}$ or, when that is not possible, at least as full subcategories of $\mathsf{Set}^{\mathcal{D}}$.

For a locally small category $\mathcal{C}$, there is the hom-functor

$$\mathcal{C}(-,-) : \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathsf{Set}\ .$$

By transposing as in (A.6) we obtain the functor

$$\mathsf{y} : \mathcal{C} \to \mathsf{Set}^{\mathcal{C}^{\mathsf{op}}}$$

which maps an object $A \in \mathcal{C}$ to the representable functor

$$\mathsf{y}A = \mathcal{C}(-, A) : B \mapsto \mathcal{C}(B, A)$$

and a morphism $f : A \to A'$ in $\mathcal{C}$ to the natural transformation $\mathsf{y}f : \mathsf{y}A \Longrightarrow \mathsf{y}A'$ whose component at B is

$$(\mathsf{y}f)_B = \mathcal{C}(B, f) : g \mapsto f \circ g\ .$$

This functor y is called the *Yoneda embedding*.

Exercise A.4.4. Show that this *is* a functor.

Theorem A.4.5 (Yoneda embedding)**.** *For any locally small category $\mathcal{C}$ the Yoneda embedding*

$$\mathsf{y} : \mathcal{C} \to \mathsf{Set}^{\mathcal{C}^{\mathsf{op}}}$$

is full and faithful and injective on objects. Therefore, $\mathcal{C}$ is a full subcategory of $\mathsf{Set}^{\mathcal{C}^{\mathsf{op}}}$.

The proof of the theorem uses the famous Yoneda Lemma.

Lemma A.4.6 (Yoneda)**.** *Every functor $F : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$ is naturally isomorphic to the functor $\mathsf{Nat}(\mathsf{y}-, F)$. That is, for every $A \in \mathcal{C}$,*

$$\mathsf{Nat}(\mathsf{y}A, F) \cong FA \ ,$$

and this isomorphism is natural in A.

Indeed, the displayed isomorphism is also natural in F.

Proof. The desired natural isomorphism θ_A maps a natural transformation $\eta \in \mathsf{Nat}(\mathsf{y}A, F)$ to $\eta_A \mathsf{1}_A$. The inverse ${\theta_A}^{-1}$ maps an element $x \in FA$ to the natural transformation $({\theta_A}^{-1}x)$ whose component at B maps $f \in \mathcal{C}(B, A)$ to $(Ff)x$. To summarize, for $\eta : \mathcal{C}(-, A) \Longrightarrow F$, $x \in FA$ and $f \in \mathcal{C}(B, A)$, we have

$$\begin{aligned} &\theta_A : \mathsf{Nat}(\mathsf{y}A, F) \to FA \ , && {\theta_A}^{-1} : FA \to \mathsf{Nat}(\mathsf{y}A, F) \ , \\ &\theta_A \eta = \eta_A \mathsf{1}_A \ , && ({\theta_A}^{-1}x)_B f = (Ff)x \ . \end{aligned}$$

To see that θ_A and ${\theta_A}^{-1}$ really are inverses of each other, observe that

$$\theta_A({\theta_A}^{-1}x) = ({\theta_A}^{-1}x)_A \mathsf{1}_A = (F\mathsf{1}_A)x = \mathsf{1}_{FA}x = x \ ,$$

and also

$$({\theta_A}^{-1}(\theta_A \eta))_B f = (Ff)(\theta_A \eta) = (Ff)(\eta_A \mathsf{1}_A) = \eta_B(\mathsf{1}_A \circ f) = \eta_B f \ ,$$

where the third equality holds by the following naturality square for η:

$$\begin{array}{ccc} \mathcal{C}(A, A) & \xrightarrow{\eta_A} & FA \\ {\scriptstyle \mathcal{C}(f, A)}\downarrow & & \downarrow{\scriptstyle Ff} \\ \mathcal{C}(B, A) & \xrightarrow[\eta_B]{} & FB \end{array}$$

It remains to check that θ is natural, which amounts to establishing the commutativity of the following diagram, with $g : A \to A'$:

$$\begin{array}{ccc} \mathsf{Nat}(\mathsf{y}A, F) & \xrightarrow{\theta_A} & FA \\ {\scriptstyle \mathsf{Nat}(\mathsf{y}g, F)}\uparrow & & \uparrow{\scriptstyle Fg} \\ \mathsf{Nat}(\mathsf{y}A', F) & \xrightarrow[\theta_{A'}]{} & FA' \end{array}$$

The diagram is commutative because, for any $\eta : \mathsf{y}A' \Longrightarrow F$,

$$(Fg)(\theta_{A'}\eta) = (Fg)(\eta_{A'}1_{A'}) = \eta_A(1_{A'} \circ g) = \\ \eta_A(g \circ 1_A) = (\mathsf{Nat}(\mathsf{y}g, F)\eta)_A 1_A = \theta_A(\mathsf{Nat}(\mathsf{y}g, F)\eta) \ ,$$

where the second equality is justified by naturality of η. □

Proof of Theorem A.4.5. That the Yoneda embedding is full and faithful means that for all $A, B \in \mathcal{C}$ the map

$$\mathsf{y} : \mathcal{C}(A, B) \to \mathsf{Nat}(\mathsf{y}A, \mathsf{y}B)$$

which maps $f : A \to B$ to $\mathsf{y}f : \mathsf{y}A \Longrightarrow \mathsf{y}B$ is an isomorphism. But this is just the Yoneda Lemma applied to the case $F = \mathsf{y}B$. Indeed, with notation as in the proof of the Yoneda Lemma and $g : C \to A$, we see that the isomorphism

$$\theta_A^{-1} : \mathcal{C}(A, B) = (\mathsf{y}B)A \to \mathsf{Nat}(\mathsf{y}A, \mathsf{y}B)$$

is in fact y:

$$(\theta_A{}^{-1} f)_C g = ((\mathsf{y}A)g)f = f \circ g = (\mathsf{y}f)_C g \ .$$

Furthermore, if $\mathsf{y}A = \mathsf{y}B$ then $1_A \in \mathcal{C}(A, A) = (\mathsf{y}A)A = (\mathsf{y}B)A = \mathcal{C}(B, A)$ which can only happen if $A = B$. Therefore, y is injective on objects. □

The following corollary is often useful.

Corollary A.4.7. *For $A, B \in \mathcal{C}$, $A \cong B$ if, and only if, $\mathsf{y}A \cong \mathsf{y}B$ in $\mathsf{Set}^{\mathcal{C}^{\mathsf{op}}}$.*

Proof. Every functor preserves isomorphisms, and a full and faithful one also reflects them. (A functor $F : \mathcal{C} \to \mathcal{D}$ is said to *reflect* isomorphisms when $Ff : FA \to FB$ being an isomorphisms implies that $f : A \to B$ is an isomorphism.) □

Exercise A.4.8. Prove that a full and faithful functor reflects isomorphisms.

Functor categories $\mathsf{Set}^{\mathcal{C}^{\mathsf{op}}}$ are important enough to deserve a name. They are called *presheaf categories*, and a functor $F : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$ is called a *presheaf* on $\mathcal{C}$. We also use the notation $\widehat{\mathcal{C}} = \mathsf{Set}^{\mathcal{C}^{\mathsf{op}}}$.

A.4.3 Equivalence of categories

An isomorphism of categories $\mathcal{C}$ and $\mathcal{D}$ in Cat consists of functors

$$\mathcal{C} \underset{G}{\overset{F}{\rightleftarrows}} \mathcal{D}$$

such that $G \circ F = 1_{\mathcal{C}}$ and $F \circ G = 1_{\mathcal{D}}$. This is often too restrictive a notion. A more general notion which replaces the above identities with natural isomorphisms is more useful.

Definition A.4.9. An *equivalence of categories* is a pair of functors

$$\mathcal{C} \underset{G}{\overset{F}{\rightleftarrows}} \mathcal{D}$$

such that there are natural isomorphisms

$$G \circ F \cong \mathsf{1}_{\mathcal{C}} \qquad \text{and} \qquad F \circ G \cong \mathsf{1}_{\mathcal{D}} \,.$$

We say that $\mathcal{C}$ and $\mathcal{D}$ are *equivalent categories* and write $\mathcal{C} \simeq \mathcal{D}$.

A functor $F : \mathcal{C} \to \mathcal{D}$ is called an *equivalence functor* if there exists $G : \mathcal{D} \to \mathcal{C}$ such that F and G form an equivalence.

The point of equivalence of categories is that it preserves almost all categorical properties, but ignores those concepts that are not of interest from a categorical point of view, such as identity of objects.

The following proposition requires the Axiom of Choice as stated. However, in many specific cases a canonical choice can be made without appeal to that axiom.

Proposition A.4.10. *A functor $F : \mathcal{C} \to \mathcal{D}$ is an equivalence functor if, and only if, F is full and faithful, and* essentially surjective *on objects, meaning that for every $B \in \mathcal{D}$ there exists $A \in \mathcal{C}$ such that $FA \cong B$.*

Proof. It is easily seen that the conditions are necessary, so we only show they are sufficient. Suppose $F : \mathcal{C} \to \mathcal{D}$ is full and faithful, and essentially surjective on objects. For each $B \in \mathcal{D}$, *choose* an object $GB \in \mathcal{C}$ and an isomorphism $\eta_B : F(GB) \to B$. If $f : B \to C$ is a morphism in $\mathcal{D}$, let $Gf : GB \to GC$ be the unique morphism in $\mathcal{C}$ for which

$$F(Gf) = {\eta_C}^{-1} \circ f \circ \eta_B \,. \tag{A.7}$$

Such a unique morphism exists because F is full and faithful. This defines a functor $G : \mathcal{D} \to \mathcal{C}$, as can be easily checked. In addition, (A.7) ensures that η is a natural isomorphism $F \circ G \Longrightarrow \mathsf{1}_{\mathcal{D}}$.

It remains to show that $G \circ F \cong \mathsf{1}_{\mathcal{C}}$. For $A \in \mathcal{C}$, let $\theta_A : G(FA) \to A$ be the unique morphism such that $F\theta_A = \eta_{FA}$. Naturality of θ_A follows from functoriality of F and naturality of η. Because F reflects isomorphisms, θ_A is an isomorphism for every A. □

Example A.4.11. As an example of equivalence of categories we consider the category of sets and partial functions and the category of pointed sets.

A *partial function* $f : A \rightharpoonup B$ is a function defined on a subset $\mathsf{supp}\, f \subseteq A$, called the *support*[3] of f, and taking values in B. Composition of partial functions $f : A \rightharpoonup B$ and $g : B \rightharpoonup C$ is the partial function $g \circ f : A \rightharpoonup C$ defined by

$$\mathsf{supp}\,(g \circ f) = \left\{ x \in A \;\middle|\; x \in \mathsf{supp}\, f \wedge fx \in \mathsf{supp}\, g \right\}$$
$$(g \circ f)x = g(fx) \quad \text{for } x \in \mathsf{supp}\,(g \circ f)$$

[3]The support of a partial function $f : A \rightharpoonup B$ is usually called its *domain*, but this terminology conflicts with A being the domain of f as a morphism.

Composition of partial functions is associative. This way we obtain a category Par of sets and partial functions.

A *pointed set* (A, a) is a set A together with an element $a \in A$. A *pointed function* $f : (A, a) \to (B, b)$ between pointed sets is a function $f : A \to B$ such that $fa = b$. The category $\mathsf{Set}_\bullet$ consists of pointed sets and pointed functions.

The categories Par and $\mathsf{Set}_\bullet$ are equivalent. The equivalence functor $F : \mathsf{Set}_\bullet \to \mathsf{Par}$ maps a pointed set (A, a) to the set $F(A, a) = A \setminus \{a\}$, and a pointed function $f : (A, a) \to (B, b)$ to the partial function $Ff : F(A, a) \rightharpoonup F(B, b)$ defined by

$$\mathsf{supp}\,(Ff) = \left\{x \in A \;\middle|\; fx \neq b\right\} , \qquad\qquad (Ff)x = fx .$$

The inverse equivalence functor $G : \mathsf{Par} \to \mathsf{Set}_\bullet$ maps a set $A \in \mathsf{Par}$ to the pointed set $GA = (A + \{\bot_A\}, \bot_A)$, where $\bot_A$ is an element that does not belong to A. A partial function $f : A \rightharpoonup B$ is mapped to the pointed function $Gf : GA \to GB$ defined by

$$(Gf)x = \begin{cases} fx & \text{if } x \in \mathsf{supp}\, f \\ \bot_B & \text{otherwise} . \end{cases}$$

A good way to think about the "bottom" point $\bot_A$ is as a special "undefined value". Let us look at the composition of F and G on objects:

$$\begin{aligned} G(F(A, a)) = G(A \setminus \{a\}) &= ((A \setminus \{a\}) + \bot_A, \bot_A) \cong (A, a) . \\ F(GA) = F(A + \{\bot_A\}, \bot_A) &= (A + \{\bot_A\}) \setminus \{\bot_A\} = A . \end{aligned}$$

The isomorphism $G(F(A, a)) \cong (A, a)$ is easily seen to be natural.

Example A.4.12. Another example of an equivalence of categories arises when we take the poset reflection of a preorder. Let $(P, \leq)$ be a preorder, If we think of P as a category, then $a, b \in P$ are isomorphic, when $a \leq b$ and $b \leq a$. Isomorphism $\cong$ is an equivalence relation, therefore we may form the quotient set $P/\!\cong$. The set $P/\!\cong$ is a poset for the order relation $\sqsubseteq$ defined by

$$[a] \sqsubseteq [b] \iff a \leq b .$$

Here $[a]$ denotes the equivalence class of a. We call $(P/\!\cong, \sqsubseteq)$ the *poset reflection* of P. The quotient map $q : P \to P/\!\cong$ is a functor when P and $P/\!\cong$ are viewed as categories. By Proposition A.4.10, q is an equivalence functor. Trivially, it is faithful and surjective on objects. It is also full because $qa \sqsubseteq qb$ in $P/\!\cong$ implies $a \leq b$ in P.

A.5 Adjoint Functors

The notion of adjunction is perhaps the most important concept revealed by category theory. It is a fundamental logical and mathematical concept that occurs everywhere and often marks an important and interesting connection between two constructions of interest. In logic, adjoint functors are pervasive, although this is only recognizable through the lens of category theory.

A.5.1 Adjoint maps between preorders

Let us begin with a simple situation. We have already seen that a preorder $(P, \leq)$ is a category in which there is at most one morphism between any two objects. A functor between preorders is a monotone map. Suppose we have preorders P and Q with monotone maps back and forth,

$$P \underset{g}{\overset{f}{\rightleftarrows}} Q\,.$$

We say that f and g are *adjoint*, and write $f \dashv g$, when for all $x \in P$, $y \in Q$,

$$fx \leq y \iff x \leq gy\,. \tag{A.8}$$

Note that adjointness is *not* a symmetric relation. The map f is the *left adjoint* and g is the *right adjoint* (note their positions with respect to $\leq$).

Equivalence (A.8) is more conveniently displayed as

$$\frac{fx \leq y}{\;\;x \leq gy\;\;}$$

The double line indicates the fact that this is a two-way rule: the top line implies the bottom line, and vice versa.

Let us consider two examples.

Conjunction is adjoint to implication Consider a propositional calculus with logical operations of conjunction $\wedge$ and implication $\Rightarrow$ (perhaps among others). The formulas of this calculus are built from variables x_0, x_1, x_2, ..., the truth values $\bot$ and $\top$, and the logical connectives $\wedge, \Rightarrow, \ldots$. The logical rules are given in natural deduction style:

$$\frac{}{\top} \qquad \frac{\bot}{A} \qquad \frac{A \qquad B}{A \wedge B} \qquad \frac{A \wedge B}{A} \qquad \frac{A \wedge B}{B}$$

$$\frac{A \Rightarrow B \qquad A}{B} \qquad\qquad \frac{\begin{array}{c}[u : A]\\ \vdots \\ B\end{array}}{A \Rightarrow B}\, u$$

For example, we read the inference rules for $\Rightarrow$ as, respectively, "from $A \Rightarrow B$ and A we infer B" and "if from assumption A we infer B, then (without any assumptions) we infer $A \Rightarrow B$". Discharged assumptions are indicated by enclosing them in brackets, along with a label $[u : A]$ for the assumption, which is recorded along with the rule that discharges it, as above.

Logical entailment $\vdash$ between formulas of the propositional calculus is the relation $A \vdash B$ which holds if, and only if, from assuming A we can infer B (by using only the inference rules of the calculus). It is trivially the case that $A \vdash A$, and also

$$\text{if } A \vdash B \text{ and } B \vdash C \text{ then } A \vdash C \ .$$

In other words, $\vdash$ is a reflexive and transitive relation on the set P of all propositional formulas, so that $(\mathsf{P}, \vdash)$ is a preorder.

Let A be a propositional formula. Define $f : \mathsf{P} \to \mathsf{P}$ and $g : \mathsf{P} \to \mathsf{P}$ to be the maps

$$fB = (A \wedge B) \ , \qquad\qquad gB = (A \Rightarrow B) \ .$$

To see that the maps f and g are functors we need to show they respect entailment. Indeed, if $B \vdash B'$ then $A \wedge B \vdash A \wedge B'$ and $A \Rightarrow B \vdash A \Rightarrow B'$ by the following two derivations.

$$\dfrac{\dfrac{A \wedge B}{A} \qquad \begin{array}{c} \dfrac{A \wedge B}{B} \\ \vdots \\ B' \end{array}}{A \wedge B'} \qquad\qquad \dfrac{\dfrac{A \Rightarrow B \qquad [u : A]}{B} \\ \vdots \\ B'}{A \Rightarrow B'} \, u$$

We claim that $f \dashv g$. For this we need to prove that $A \wedge B \vdash C$ if, and only if, $B \vdash A \Rightarrow C$. The following two derivations establish the required equivalence.

$$\dfrac{\dfrac{[u : A] \qquad B}{A \wedge B} \\ \vdots \\ C}{A \Rightarrow C} \, u \qquad\qquad \dfrac{\dfrac{A \wedge B}{B} \\ \vdots \\ A \Rightarrow C \qquad \dfrac{A \wedge B}{A}}{C}$$

Therefore, *conjunction is left adjoint to implication.*

Topological interior as an adjoint Recall that a *topological space* $(X, \mathcal{O}X)$ is a set X together with a family $\mathcal{O}X \subseteq \mathcal{P}X$ of subsets of X which contains $\emptyset$ and X, and is closed under finite intersections and arbitrary unions. The elements of $\mathcal{O}X$ are called the *open sets*.

The *topological interior* of a subset $S \subseteq X$ is the largest open set contained in S, namely,

$$\mathsf{int}\, S = \bigcup \big\{ U \in \mathcal{O}X \;\big|\; U \subseteq S \big\} \ .$$

Both $\mathcal{O}X$ and $\mathcal{P}X$ are posets ordered by subset inclusion. The inclusion $i : \mathcal{O}X \to \mathcal{P}X$ is thus a monotone map, and so indeed is the interior $\mathsf{int} : \mathcal{P}X \to \mathcal{O}X$, as follows immediately from its construction. So we have:

$$\mathcal{O}X \underset{\mathsf{int}}{\overset{i}{\rightleftarrows}} \mathcal{P}X$$

Moreover, for $U \in \mathcal{O}X$ and $S \in \mathcal{P}X$ we plainly also have

$$\frac{iU \subseteq S}{U \subseteq \mathsf{int}\, S}$$

since $\mathsf{int}\, S$ is the largest open set contained in S. Thus *topological interior is right adjoint* to the inclusion of $\mathcal{O}X$ into $\mathcal{P}X$.

A.5.2 Adjoint functors

Let us now generalize the notion of adjoint monotone maps from posets to the situation

$$\mathcal{C} \underset{G}{\overset{F}{\rightleftarrows}} \mathcal{D}$$

with arbitrary categories and functors. For monotone maps $f \dashv g$, the adjunction condition is a bijection

$$\frac{fx \to y}{x \to gy}$$

between morphisms of the form $fx \to y$ and morphisms of the form $x \to gy$. This is the notion that generalizes the special case; for any $A \in \mathcal{C}$, $B \in \mathcal{D}$ we require a bijection between the sets $\mathcal{D}(FA, B)$ and $\mathcal{C}(A, GB)$:

$$\frac{FA \to B}{A \to GB}$$

Definition A.5.1. An *adjunction* $F \dashv G$ between the functors

$$\mathcal{C} \underset{G}{\overset{F}{\rightleftarrows}} \mathcal{D}$$

is a natural isomorphism θ between functors

$$\mathcal{D}(F-,-) : \mathcal{C}^{\mathsf{op}} \times \mathcal{D} \to \mathsf{Set} \qquad \text{and} \qquad \mathcal{C}(-,G-) : \mathcal{C}^{\mathsf{op}} \times \mathcal{D} \to \mathsf{Set}\,.$$

This means that for every $A \in \mathcal{C}$ and $B \in \mathcal{D}$ there is a bijection

$$\theta_{A,B} : \mathcal{D}(FA, B) \cong \mathcal{C}(A, GB)\,,$$

and naturality of θ means that for $f : A' \to A$ in $\mathcal{C}$ and $g : B \to B'$ in $\mathcal{D}$ the following diagram commutes:

$$\begin{array}{ccc} \mathcal{D}(FA, B) & \xrightarrow{\theta_{A,B}} & \mathcal{D}(A, GB) \\ \Big\downarrow{\scriptstyle \mathcal{D}(Ff, g)} & & \Big\downarrow{\scriptstyle \mathcal{C}(f, Gg)} \\ \mathcal{D}(FA', B') & \xrightarrow[\theta_{A',B'}]{} & \mathcal{C}(A', GB') \end{array}$$

Equivalently, for every $h : FA \to B$ in $\mathcal{D}$,

$$Gg \circ (\theta_{A,B} h) \circ f = \theta_{A',B'}(g \circ h \circ Ff) \ .$$

We say that F is the *left adjoint* and G is the *right adjoint*.

We have already seen examples of adjoint functors. For any category $\mathbb{B}$ we have functors $(-) \times \mathbb{B}$ and $(-)^{\mathbb{B}}$ from Cat to Cat. Recall the isomorphism (A.6),

$$\mathsf{Cat}(\mathbb{A} \times \mathbb{B}, \mathbb{C}) \cong \mathsf{Cat}(\mathbb{A}, \mathbb{C}^{\mathbb{B}}) \ .$$

This isomorphism is in fact natural in $\mathbb{A}$ and $\mathbb{C}$, so that

$$(-) \times \mathbb{B} \dashv (-)^{\mathbb{B}} \ .$$

Similarly, for any set $B \in \mathsf{Set}$ there are functors

$$(-) \times B : \mathsf{Set} \to \mathsf{Set} \ , \qquad\qquad (-)^B : \mathsf{Set} \to \mathsf{Set} \ ,$$

where $A \times B$ is the cartesian product of A and B, and C^B is the set of all functions from B to C. For morphisms, $f \times B = f \times 1_B$ and $f^B = f \circ (-)$. We then indeed have a natural isomorphism, for all $A, C \in \mathsf{Set}$,

$$\mathsf{Set}(A \times B, C) \cong \mathsf{Set}(A, C^B) \ ,$$

which maps a function $f : A \times B \to C$ to the function $(\widetilde{f}x)y = f\langle x, y\rangle$. Therefore,

$$(-) \times B \dashv (-)^B \ .$$

Exercise A.5.2. Verify that the definition (A.8) of adjoint monotone maps between preorders is a special case of Definition A.5.1. What happened to the naturality condition?

For another example, consider the forgetful functor

$$U : \mathsf{Cat} \to \mathsf{Graph} \ ,$$

which maps a category to the underlying directed graph. It has a left adjoint $P \dashv U$. The functor P is the *free* construction of a category from a graph; it maps a graph G to the *category of paths* $P(G)$. The objects of $P(G)$ are the vertices of G. The morphisms of $P(G)$ are the finite paths

$$v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} v_n$$

of edges in G, composition is concatenation of paths, and the identity morphism on a vertex v is the empty path starting and ending at v.

By using the Yoneda Lemma we can easily prove that adjoints are unique up to natural isomorphism.

Proposition A.5.3. *Let $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$ be adjoint functors, with $F \dashv G$. If also $G' : \mathcal{D} \to \mathcal{C}$ with $F \dashv G'$, then $G \cong G'$.*

Proof. Since the Yoneda embedding is full and faithful, we have $GB \cong G'B$ if, and only if, $\mathcal{C}(-, GB) \cong \mathcal{C}(-, G'B)$. But this indeed holds, because, for any $A \in \mathcal{C}$, we have

$$\mathcal{C}(A, GB) \cong \mathcal{D}(FA, B) \cong \mathcal{C}(A, G'B) \ ,$$

naturally in A. □

Left adjoints are of course also unique up to isomorphism, by duality.

A.5.3 The unit of an adjunction

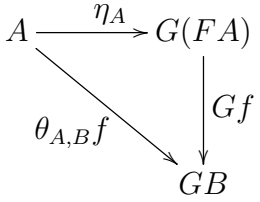
Let $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$ be adjoint functors, $F \dashv G$, and let $\theta : \mathcal{D}(F-, -) \to \mathcal{C}(-, G-)$ be the natural isomorphism witnessing the adjunction. For any object $A \in \mathcal{C}$ there is a distinguished morphism $\eta_A = \theta_{A,FA} 1_{FA} : A \to G(FA)$,

$$\frac{1_{FA} : FA \to FA}{\eta_A : A \to G(FA)}$$

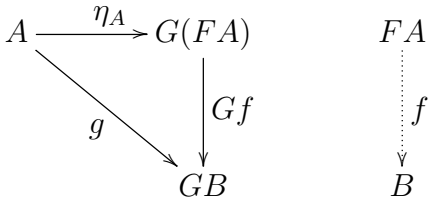
Since θ is natural in A, we have a natural transformation $\eta : 1_{\mathcal{C}} \Longrightarrow G \circ F$, which is called the *unit of the adjunction* $F \dashv G$. In fact, we can recover θ from η as follows. For $f : FA \to B$, we have

$$\theta_{A,B} f = \theta_{A,B}(f \circ 1_{FA}) = Gf \circ \theta_{A,FA}(1_{FA}) = Gf \circ \eta_A \ ,$$

where we used naturality of θ in the second step. Schematically, given any $f : FA \to B$, the following diagram commutes:



Since $\theta_{A,B}$ is a bijection, it follows that *every* morphism $g : A \to GB$ has the form $g = Gf \circ \eta_A$ for a *unique* $f : FA \to B$. We say that $\eta_A : A \to G(FA)$ is a *universal* morphism to G, or that η has the following *universal mapping property*: for every $A \in \mathcal{C}$, $B \in \mathcal{D}$, and $g : A \to GB$, there exists a *unique* $f : FA \to B$ such that $g = Gf \circ \eta_A$:



This means that an adjunction can be given in terms of its unit. The isomorphism $\theta : \mathcal{D}(F-,-) \to \mathcal{C}(-,G-)$ is then recovered by

$$\theta_{A,B} f = Gf \circ \eta_A \ .$$

Proposition A.5.4. *A functor $F : \mathcal{C} \to \mathcal{D}$ is left adjoint to a functor $G : \mathcal{D} \to \mathcal{C}$ if, and only if, there exists a natural transformation*

$$\eta : 1_{\mathcal{C}} \Longrightarrow G \circ F \ ,$$

called the unit *of the adjunction, such that, for all $A \in \mathcal{C}$ and $B \in \mathcal{D}$ the map $\theta_{A,B} : \mathcal{D}(FA, B) \to \mathcal{C}(A, GB)$, defined by*

$$\theta_{A,B} f = Gf \circ \eta_A \ ,$$

is an isomorphism.

Let us demonstrate how the universal mapping property of the unit of an adjunction appears as a well known construction in algebra. Consider the forgetful functor from monoids to sets,

$$U : \mathsf{Mon} \to \mathsf{Set} \ .$$

Does it have a left adjoint $F : \mathsf{Set} \to \mathsf{Mon}$? In order to obtain one, we need a "most economical" way of making a monoid FX from a given set X. Such a construction readily suggests itself, namely the *free monoid* on X, consisting of finite sequences of elements of X,

$$FX = \left\{ x_1 \dots x_n \;\middle|\; n \geq 0 \ \&\ x_1, \dots, x_n \in X \right\} \ .$$

The monoid operation is concatenation of sequences

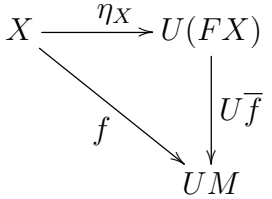
$$x_1 \dots x_m \cdot y_1 \dots y_n = x_1 \dots x_m y_1 \dots y_n \ ,$$

and the empty sequence is the unit of the monoid. In order for F to be a functor, it should also map morphisms to morphisms. If $f : X \to Y$ is a function, define $Ff : FX \to FY$ by

$$Ff : x_1 \dots x_n \mapsto (fx_1) \dots (fx_n) \ .$$

There is an inclusion $\eta_X : X \to U(FX)$ which maps every element $x \in X$ to the singleton sequence x. This gives a natural transformation $\eta : 1_{\mathsf{Set}} \Longrightarrow U \circ F$.

The monoid FX is "free" in the sense that it "satisfies only the equations required by the monoid laws"; we make this precise as follows. For every monoid M and function $f : X \to UM$ there exists a unique monoid homomorphism $\overline{f} : FX \to M$ such that the following diagram commutes:



This is precisely the condition required by Proposition A.5.4 for η to be the unit of the adjunction $F \dashv U$. In this case, the universal mapping property of η is just the usual characterization of the free monoid FX generated by the set X: a homomorphism from FX is uniquely determined by its values on the generators.

A.5.4 The counit of an adjunction

Let $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$ be adjoint functors with $F \dashv G$, and let $\theta : \mathcal{D}(F-,-) \to \mathcal{C}(-,G-)$ be the natural isomorphism witnessing the adjunction. For any object $B \in \mathcal{D}$ we have a distinguished morphism $\varepsilon_B = \theta^{-1}_{GB,B} 1_{GB} : F(GB) \to B$ by:

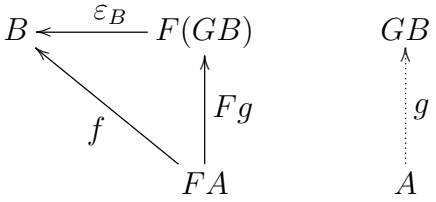
$$\frac{1_{GB} : GB \to GB}{\varepsilon_B : F(GB) \to B}$$

The natural transformation $\varepsilon : F \circ G \Longrightarrow 1_{\mathcal{D}}$ is called the *counit* of the adjunction $F \dashv G$. It is the dual notion to the unit of an adjunction. We state briefly the basic properties of the counit, which are easily obtained by "turning around" all the morphisms in the previous section and exchanging the roles of the left and right adjoints.

The bijection $\theta^{-1}_{A,B}$ can be recovered from the counit. For $g : A \to GB$ in $\mathcal{C}$, we have

$$\theta^{-1}_{A,B} g = \theta^{-1}_{A,B}(1_{GB} \circ g) = \theta^{-1}_{A,B} 1_{GB} \circ Fg = \varepsilon_B \circ Fg \ .$$

The universal mapping property of the counit is this: for every $A \in \mathcal{C}$, $B \in \mathcal{D}$, and $f : FA \to B$, there exists a *unique* $g : A \to GB$ such that $f = \varepsilon_B \circ Fg$:



The following is the dual of Proposition A.5.4.

Proposition A.5.5. *A functor $F : \mathcal{C} \to \mathcal{D}$ is left adjoint to a functor $G : \mathcal{D} \to \mathcal{C}$ if, and only if, there exists a natural transformation*

$$\varepsilon : F \circ G \Longrightarrow 1_{\mathcal{D}} \ ,$$

called the counit *of the adjunction, such that, for all $A \in \mathcal{C}$ and $B \in \mathcal{D}$ the map $\theta^{-1}_{A,B} : \mathcal{C}(A, GB) \to \mathcal{D}(FA, B)$, defined by*

$$\theta^{-1}_{A,B} g = \varepsilon_B \circ Fg \ ,$$

is an isomorphism.

Let us consider again the forgetful functor $U : \mathsf{Mon} \to \mathsf{Set}$ and its left adjoint $F : \mathsf{Set} \to \mathsf{Mon}$, the free monoid construction. For a monoid $(M, \star) \in \mathsf{Mon}$, the counit of the adjunction $F \dashv U$ is a monoid homomorphism $\varepsilon_M : F(UM) \to M$, defined by

$$\varepsilon_M(x_1 x_2 \dots x_n) = x_1 \star x_2 \star \cdots \star x_n \ .$$

It has the following universal mapping property: for $X \in \mathsf{Set}$, $(M, \star) \in \mathsf{Mon}$, and a homomorphism $f : FX \to M$ there exists a unique function $\overline{f} : X \to UM$ such that $f = \varepsilon_M \circ F\overline{f}$, namely

$$\overline{f}x = fx \ ,$$

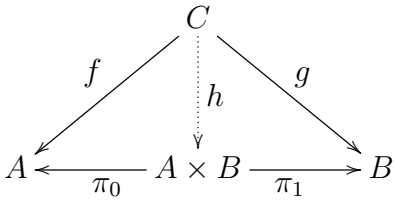
where in the above definition $x \in X$ is viewed as an element of the set X on the left-hand side, and as an element of the free monoid FX on the right-hand side. To summarize, the universal mapping property of the counit ε is the familiar piece of wisdom that a homomorphism $f : FX \to M$ from a free monoid is already determined by its values on the generators.

A.6 Limits and Colimits

The following limits and colimits are all special cases of adjoint functors, as we shall see.

A.6.1 Binary products

In a category $\mathcal{C}$, the *(binary) product* of objects A and B is an object $A \times B$ together with *projections* $\pi_0 : A \times B \to A$ and $\pi_1 : A \times B \to B$ such that, for every object $C \in \mathcal{C}$ and every pair of morphisms $f : C \to A$, $g : C \to B$ there exists a *unique* morphism $h : C \to A \times B$ for which the following diagram commutes:



We normally refer to the product $(A \times B, \pi_0, \pi_1)$ just by its object $A \times B$, but you should keep in mind that a product is given by an object *and* two projections. The arrow $h : C \to A \times B$ is denoted by $\langle f, g \rangle$. The property

$$\begin{aligned} &\text{for all } C, \text{ for all } f : C \to A, \text{ for all } g : C \to B, \\ &\text{there is a unique } h : C \to A \times B, \\ &\text{with } \pi_0 \circ h = f \ \& \ \pi_1 \circ h = g \end{aligned}$$

is the *universal mapping property* of the product $A \times B$. It characterizes the product of A and B uniquely up to isomorphism in the sense that if $(P, p_0 : P \to A, p_1 : P \to B)$ is

another product of A and B, then there is a unique isomorphism $r : P \xrightarrow{\sim} A \times B$ such that $p_0 = \pi_0 \circ r$ and $p_1 = \pi_1 \circ r$.

If in a category $\mathcal{C}$ every two objects have a product, we can turn binary products into an operation[4] by *choosing* a product $A \times B$ for each pair of objects $A, B \in \mathcal{C}$. In general this requires the Axiom of Choice, but in many specific cases a particular choice of products can be made without appeal to that axiom. When we view binary products as an operation, we say that "$\mathcal{C}$ has chosen products". The same holds for other instances of limits and colimits.

For example, in Set the usual cartesian product of sets is a product. In categories of structures, products are the usual construction: the product of topological spaces in Top is their topological product, the product of directed graphs in Graph is their cartesian product, the product of categories in Cat is their product category, and so on.

A.6.2 Terminal objects

A *terminal object* in a category $\mathcal{C}$ is an object $1 \in \mathcal{C}$ such that for every $A \in \mathcal{C}$ there exists a *unique* morphism $!_A : A \to 1$.

For example, in Set an object is terminal if, and only if, it is a singleton. The terminal object in Cat is the unit category $\mathsf{1}$ consisting of one object and one morphism.

Exercise A.6.1. Prove that if 1 and $1'$ are terminal objects in a category then they are isomorphic.

Exercise A.6.2. Let Field be the category whose objects are fields and morphisms are field homomorphisms.[5] Does Field have a terminal object? What about the category Ring of rings?

A.6.3 Equalizers

Given objects and morphisms

$$E \xrightarrow{\ e\ } A \underset{g}{\overset{f}{\rightrightarrows}} B$$

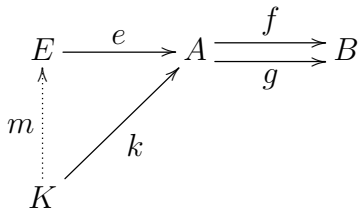
we say that e *equalizes* f and g when $f \circ e = g \circ e$.[6] An *equalizer* of f and g is a *universal* equalizing morphism; thus $e : E \to A$ is an equalizer of f and g when it equalizes them and, for all $k : K \to A$, if $f \circ k = g \circ k$ then there exists a unique morphism $m : K \to E$

[4] More precisely, binary product is a functor from $\mathcal{C} \times \mathcal{C}$ to $\mathcal{C}$, cf. Section A.6.11.

[5] A field $(F, +, \cdot, ^{-1}, 0, 1)$ is a ring with a unit in which all non-zero elements have inverses. We also require that $0 \neq 1$. A homomorphism of fields preserves addition and multiplication, and consequently also 0, 1 and inverses.

[6] Note that this does *not* mean the diagram involving f, g and e is commutative!

such that $k = e \circ m$:



In Set the equalizer of parallel functions $f : A \to B$ and $g : A \to B$ is the set

$$E = \{x \in A \mid fx = gx\}$$

with $e : E \to A$ being the subset inclusion $E \subseteq A$, $ex = x$. In general, equalizers can be thought of as those subobjects (subsets, subgroups, subspaces, ...) that can be defined by an equation.

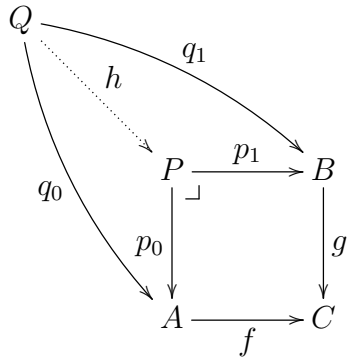
Exercise A.6.3. Show that an equalizer is a monomorphism, i.e., if $e : E \to A$ is an equalizer of f and g, then, for all $r, s : C \to E$, $e \circ r = e \circ s$ implies $r = s$.

Definition A.6.4. A morphism is a *regular mono* if it is an equalizer.

The difference between monos and regular monos is best illustrated in the category Top: a continuous map $f : X \to Y$ is mono when it is injective, whereas it is a regular mono when it is a topological embedding.[7]

A.6.4 Pullbacks

A *pullback* of $f : A \to C$ and $g : B \to C$ is an object P with morphisms $p_0 : P \to A$ and $p_1 : P \to B$ such that $f \circ p_0 = g \circ p_1$, and whenever Q, $q_0 : Q \to A$, and $q_1 : Q \to B$ are such that $f \circ q_0 = g \circ q_1$, there then exists a unique $h : Q \to P$ such that $q_0 = p_0 \circ h$ and $q_1 = p_1 \circ h$:



We indicate that P is a pullback by drawing a square corner next to it, as in the above diagram. The pullback is sometimes written $A \times_C B$, since it is indeed a product in the slice category over C.

[7] A continuous map $f : X \to Y$ is a topological embedding when, for every $U \in \mathcal{O}X$, the image $f[U]$ is an open subset of the image $\mathsf{im}(f)$; this means that there exists $V \in \mathcal{O}Y$ such that $f[U] = V \cap \mathsf{im}(f)$.

In Set, the pullback of $f : A \to C$ and $g : B \to C$ is the set

$$P = \big\{ \langle x, y \rangle \in A \times B \;\big|\; fx = gy \big\}$$

and the functions $p_0 : P \to A$, $p_1 : P \to B$ are the projections, $p_0\langle x, y\rangle = x$, $p_1\langle x, y\rangle = y$.

When we form the pullback of $f : A \to C$ and $g : B \to C$ we may also say that we *pull g back along f* and draw the diagram

$$\begin{array}{ccc} f^*B & \longrightarrow & B \\ {\scriptstyle f^*g}\downarrow & \lrcorner & \downarrow{\scriptstyle g} \\ A & \xrightarrow[f]{} & C \end{array}$$

We think of $f^*g : f^*B \to A$ as the inverse image of B along f. This terminology is explained by looking at the pullback of a subset inclusion $u : U \hookrightarrow C$ along a function $f : A \to C$ in the category Set:

$$\begin{array}{ccc} f^*U & \longrightarrow & U \\ \downarrow & \lrcorner & \downarrow{\scriptstyle u} \\ A & \xrightarrow[f]{} & C \end{array}$$

In this case the pullback is $\big\{ \langle x, y\rangle \in A \times U \;\big|\; fx = y \big\} \cong \big\{ x \in A \;\big|\; fx \in U \big\} = f^*U$, the inverse image of U along f.

Exercise A.6.5. Prove that in a category $\mathcal{C}$, a morphism $f : A \to B$ is mono if, and only if, the following diagram is a pullback:

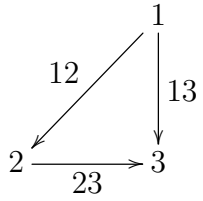
$$\begin{array}{ccc} A & \xrightarrow{1_A} & A \\ {\scriptstyle 1_A}\downarrow & & \downarrow{\scriptstyle f} \\ A & \xrightarrow[f]{} & B \end{array}$$

A.6.5 Limits

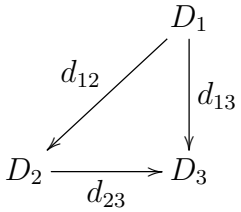
Let us now define the general notion of a limit.

A *diagram of shape $\mathcal{I}$* in a category $\mathcal{C}$ is a functor $D : \mathcal{I} \to \mathcal{C}$, where the category $\mathcal{I}$ is called the *index category*. We use letters i, j, k, ... for objects of an index category $\mathcal{I}$, call them *indices*, and write D_i, D_j, D_k, ... instead of Di, Dj, Dk, ...

For example, if $\mathcal{I}$ is the category with three objects and three morphisms



where $13 = 23 \circ 12$ then a diagram of shape $\mathcal{I}$ is a commutative diagram



(A.9)

For each object $A \in \mathcal{C}$, the *constant A-valued diagram* of shape $\mathcal{I}$ is given by the constant functor $\Delta_A : \mathcal{I} \to \mathcal{C}$, which maps every object to A and every morphism to $\mathsf{1}_A$.

Let $D : \mathcal{I} \to \mathcal{C}$ be a diagram of shape $\mathcal{I}$. A *cone* on D from an object $A \in \mathcal{C}$ is a natural transformation $\alpha : \Delta_A \Longrightarrow D$. This means that for every index $i \in \mathcal{I}$ there is a morphism $\alpha_i : A \to D_i$ such that whenever $u : i \to j$ in $\mathcal{I}$ then $\alpha_j = Du \circ \alpha_i$.

For a given diagram $D : \mathcal{I} \to \mathcal{C}$, we can collect all cones on D into a category $\mathsf{Cone}(D)$ whose objects are cones on D. A morphism between cones $f : (A, \alpha) \to (B, \beta)$ is a morphism $f : A \to B$ in $\mathcal{C}$ such that $\alpha_i = \beta_i \circ f$ for all $i \in \mathcal{I}$. Morphisms in $\mathsf{Cone}(D)$ are composed as morphisms in $\mathcal{C}$. A morphism $f : (A, \alpha) \to (B, \beta)$ is also called a factorization of the cone (A, α) through the cone (B, β).

A *limit* of a diagram $D : \mathcal{I} \to \mathcal{C}$ is a terminal object in $\mathsf{Cone}(D)$. Explicitly, a limit of D is given by a cone (L, λ) such that for every other cone (A, α) there exists a *unique* morphism $f : A \to L$ such that $\alpha_i = \lambda_i \circ f$ for all $i \in \mathcal{I}$. We denote (the object part of) a limit of D by one of the following:

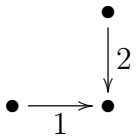
$$\lim D \qquad \lim_{i \in \mathcal{I}} D_i \qquad \varprojlim_{i \in \mathcal{I}} D_i \ .$$

Limits are also called *projective limits*. We say that a category *has limits of shape $\mathcal{I}$* when every diagram of shape $\mathcal{I}$ in $\mathcal{C}$ has a limit.

Products, terminal objects, equalizers, and pullbacks are all special cases of limits:

- a product $A \times B$ is the limit of the functor $D : \mathsf{2} \to \mathcal{C}$ where $\mathsf{2}$ is the discrete category on two objects 0 and 1, and $D_0 = A$, $D_1 = B$.
- a terminal object $\mathsf{1}$ is the limit of the (unique) functor $D : \mathsf{0} \to \mathcal{C}$ from the empty category.
- an equalizer of $f, g : A \to B$ is the limit of the functor $D : (\cdot \rightrightarrows \cdot) \to \mathcal{C}$ which maps one morphism to f and the other one to g.

- the pullback of $f : A \to C$ and $g : B \to C$ is the limit of the functor $D : \mathcal{I} \to \mathcal{C}$ where $\mathcal{I}$ is the category



  with $D1 = f$ and $D2 = g$.

It is clear how to define the product of an arbitrary family of objects

$$\left\{ A_i \in \mathcal{C} \;\middle|\; i \in I \right\} .$$

Such a family is a diagram of shape I, where I is viewed as a discrete category. A *product* $\prod_{i \in I} A_i$ is then given by an object $P \in \mathcal{C}$ and morphisms $\pi_i : P \to A_i$ such that, whenever we have a family of morphisms $\left\{ f_i : B \to A_i \;\middle|\; i \in I \right\}$ there exists a *unique* morphism $\langle f_i \rangle_{i \in I} : B \to P$ such that $f_i = \pi_i \circ f$ for all $i \in I$.

A *finite product* is a product of a finite family. As a special case we see that a terminal object is the product of an empty family. It is not hard to show that a category has finite products precisely when it has a terminal object and binary products.

A diagram $D : \mathcal{I} \to \mathcal{C}$ is *small* when $\mathcal{I}$ is a small category. A *small limit* is a limit of a small diagram. A *finite limit* is a limit of a diagram whose index category is finite.

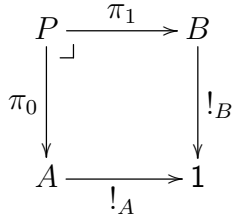
Exercise A.6.6. Prove that a limit, when it exists, is unique up to isomorphism.

The following proposition and its proof tell us how to compute arbitrary limits from simpler ones. We omit detailed proofs as they can be found in any standard textbook on category theory.

Proposition A.6.7. *The following are equivalent for a category $\mathcal{C}$:*

1. *$\mathcal{C}$ has a terminal object and all pullbacks.*
2. *$\mathcal{C}$ has equalizers and all finite products.*
3. *$\mathcal{C}$ has all finite limits.*

Proof. We only show how to get binary products from pullbacks and a terminal object. For objects A and B, let P be the pullback of $!_A$ and $!_B$:



Then (P, π_0, π_1) is a product of A and B because, for all $f : X \to A$ and $g : X \to B$, it is trivially the case that $!_A \circ f = !_B \circ g$. □

Proposition A.6.8. *The following are equivalent for a category $\mathcal{C}$:*

1. *$\mathcal{C}$ has equalizers and all small products.*
2. *$\mathcal{C}$ has all small limits.*

Proof. We indicate how to construct an arbitrary limit from a product and an equalizer. Let $D : \mathcal{I} \to \mathcal{C}$ be a small diagram of an arbitrary shape $\mathcal{I}$. First form an $\mathcal{I}_0$-indexed product P and an $\mathcal{I}_1$-indexed product Q

$$P = \prod_{i \in \mathcal{I}_0} D_i \ , \qquad\qquad Q = \prod_{u \in \mathcal{I}_1} D_{\mathsf{cod}\, u} \ .$$

By the universal property of products, there are unique morphisms $f : P \to Q$ and $g : P \to Q$ such that, for all morphisms $u \in \mathcal{I}_1$,

$$\pi^Q_u \circ f = Du \circ \pi^P_{\mathsf{dom}\, u} \ , \qquad\qquad \pi^Q_u \circ g = \pi^P_{\mathsf{cod}\, u} \ .$$

Let E be the equalizer of f and g,

$$E \xrightarrow{\ e\ } P \underset{g}{\overset{f}{\rightrightarrows}} Q$$

For every $i \in \mathcal{I}$ there is a morphism $\varepsilon_i : E \to D_i$, namely $\varepsilon_i = \pi^P_i \circ e$. We claim that (E, ε) is a limit of D. First, (E, ε) is a cone on D because, for all $u : i \to j$ in $\mathcal{I}$,

$$Du \circ \varepsilon_i = Du \circ \pi^P_i \circ e = \pi^Q_u \circ f \circ e = \pi^Q_u \circ g \circ e = \pi^P_j \circ e = \varepsilon_j \ .$$

If (A, α) is any cone on D there exists a unique $t : A \to P$ such that $\alpha_i = \pi^P_i \circ t$ for all $i \in \mathcal{I}$. For every $u : i \to j$ in $\mathcal{I}$ we have

$$\pi^Q_u \circ g \circ t = \pi^P_j \circ t = t_j = Du \circ t_i = Du \circ \pi^P_i \circ t = \pi^Q_u \circ f \circ t \ ,$$

therefore $g \circ t = f \circ t$. This implies that there is a unique factorization $k : A \to E$ such that $t = e \circ k$. Now for every $i \in \mathcal{I}$

$$\varepsilon_i \circ k = \pi^P_i \circ e \circ k = \pi^P_i \circ t = \alpha_i$$

so that $k : A \to E$ is the required factorization of the cone (A, α) through the cone (E, ε). To see that k is unique, suppose $m : A \to E$ is another factorization such that $\alpha_i = \varepsilon_i \circ m$ for all $i \in \mathcal{I}$. Since e is mono it suffices to show that $e \circ m = e \circ k$, which is equivalent to proving $\pi^P_i \circ e \circ m = \pi^P_i \circ e \circ k$ for all $i \in \mathcal{I}$. This last equality holds because

$$\pi^P_i \circ e \circ k = \pi^P_i \circ t = \alpha_i = \varepsilon_i \circ m = \pi^P_i \circ e \circ m \ .$$

□

A category is *(small) complete* when it has all small limits, and it is *finitely complete* (or *left exact*, briefly *lex*) when it has finite limits.

Limits of presheaves Let $\mathcal{C}$ be a locally small category. Then the presheaf category $\widehat{\mathcal{C}} = \mathsf{Set}^{\mathcal{C}^{\mathsf{op}}}$ has all small limits and they are computed pointwise, e.g., $(P \times Q)A = PA \times QA$ for $P, Q \in \widehat{\mathcal{C}}$, $A \in \mathcal{C}$. To see that this is really so, let $\mathcal{I}$ be a small index category and $D : \mathcal{I} \to \widehat{\mathcal{C}}$ a diagram of presheaves. Then for every $A \in \mathcal{C}$ the diagram D can be instantiated at A to give a diagram $DA : \mathcal{I} \to \mathsf{Set}$, $(DA)_i = D_i A$. Because Set is small complete, we can define a presheaf L by computing the limit of DA:

$$LA = \lim DA = \varprojlim_{i \in \mathcal{I}} D_i A \ .$$

We should keep in mind that $\lim DA$ is actually given by an object $(\lim DA)$ and a natural transformation $\delta A : \Delta_{(\lim DA)} \Longrightarrow DA$. The value of LA is supposed to be just the object part of $\lim DA$. From a morphism $f : A \to B$ we obtain for each $i \in \mathcal{I}$ a function $D_i f \circ (\delta A)_i : LA \to D_i B$, and thus a cone $(LA, Df \circ \delta A)$ on DB. Presheaf L maps the morphism $f : A \to B$ to the unique factorization $Lf : LA \Longrightarrow LB$ of the cone $(LA, Df \circ \delta A)$ on DB through the limit cone LB on DB.

For every $i \in \mathcal{I}$, there is a function $\Lambda_i = (\delta A)_i : LA \to D_i A$. The family $\{\Lambda_i\}_{i \in \mathcal{I}}$ is a natural transformation from Δ_{LA} to DA. This gives us a cone (L, Λ) on D, which is in fact a limit cone. Indeed, if (S, Σ) is another cone on D then for every $A \in \mathcal{C}$ there exists a unique function $\phi_A : SA \to LA$ because SA is a cone on DA and LA is a limit cone on DA. The family $\{\phi_A\}_{A \in \mathcal{C}}$ is the unique natural transformation $\phi : S \Longrightarrow L$ for which $\Sigma = \phi \circ \Lambda$.

A.6.6 Colimits

Colimits are the dual notion of limits. Thus, a *colimit* of a diagram $D : \mathcal{I} \to \mathcal{C}$ is a limit of the dual diagram $D^{\mathsf{op}} : \mathcal{I}^{\mathsf{op}} \to \mathcal{C}^{\mathsf{op}}$ in the dual (i.e., opposite) category $\mathcal{C}^{\mathsf{op}}$:

$$\operatorname{colim}(D : \mathcal{I} \to \mathcal{C}) = \lim(D^{\mathsf{op}} : \mathcal{I}^{\mathsf{op}} \to \mathcal{C}^{\mathsf{op}}) \ .$$

Explicitly, the colimit of a diagram $D : \mathcal{I} \to \mathcal{C}$ is the initial object in the category of *cocones* $\mathsf{Cocone}(D)$ on D. A cocone (A, α) on D is a natural transformation $\alpha : D \Longrightarrow \Delta_A$. It is given by an object $A \in \mathcal{C}$ and, for each $i \in \mathcal{I}$, a morphism $\alpha_i : D_i \to A$, such that $\alpha_i = \alpha_j \circ Du$ whenever $u : i \to j$ in $\mathcal{I}$. A morphism between cocones $f : (A, \alpha) \to (B, \beta)$ is a morphism $f : A \to B$ in $\mathcal{C}$ such that $\beta_i = f \circ \alpha_i$ for all $i \in \mathcal{I}$.

A colimit of $D : \mathcal{I} \to \mathcal{C}$ is then given by a cocone (C, ζ) on D such that, for every cocone (A, α) on D there exists a unique morphism $f : C \to A$ such that $\alpha_i = f \circ \zeta_i$ for all $i \in D$. We denote a colimit of D by one of the following:

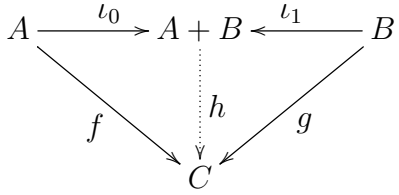
$$\operatorname{colim} D \qquad \operatorname{colim}_{i \in \mathcal{I}} D_i \qquad \varinjlim_{i \in \mathcal{I}} D_i \ .$$

Colimits are also called *inductive limits*.

Exercise A.6.9. Formulate the dual of Proposition A.6.7 and Proposition A.6.8 for colimits (coequalizers are defined in Section A.6.9).

A.6.7 Binary coproducts

In a category $\mathcal{C}$, the *(binary) coproduct* of objects A and B is an object $A + B$ together with *injections* $\iota_0 : A \to A + B$ and $\iota_1 : B \to A + B$ such that, for every object $C \in \mathcal{C}$ and all morphisms $f : A \to C$, $g : B \to C$ there exists a *unique* morphism $h : A + B \to C$ for which the following diagram commutes:



The arrow $h : A + B \to C$ is denoted by $[f, g]$.

The coproduct $A + B$ is the colimit of the diagram $D : 2 \to \mathcal{C}$, where $\mathcal{I}$ is the discrete category on two objects 0 and 1, and $D_0 = A$, $D_1 = B$.

In Set the coproduct is the disjoint union, defined by

$$X + Y = \big\{ \langle 0, x\rangle \;\big|\; x \in X \big\} \cup \big\{ \langle 1, y\rangle \;\big|\; x \in Y \big\} \ ,$$

where 0 and 1 are distinct sets, for example $\emptyset$ and $\{\emptyset\}$. Given functions $f : X \to Z$ and $g : Y \to Z$, the unique function $[f, g] : X + Y \to Z$ is the usual *definition by cases*:

$$[f, g]u = \begin{cases} fx & \text{if } u = \langle 0, x\rangle \\ gx & \text{if } u = \langle 1, x\rangle \ . \end{cases}$$

Exercise A.6.10. Show that the categories of posets and of topological spaces both have coproducts.

A.6.8 Initial objects

An *initial object* in a category $\mathcal{C}$ is an object $\mathsf{0} \in \mathcal{C}$ such that for every $A \in \mathcal{C}$ there exists a *unique* morphism $\mathsf{o}_A : \mathsf{0} \to A$.

An initial object is the colimit of the empty diagram.

In Set, the initial object is the empty set.

Exercise A.6.11. What is the initial and what is the terminal object in the category of groups?

A *zero object* is an object that is both initial and terminal.

Exercise A.6.12. Show that in the category of Abelian[8] groups finite products and coproducts agree, that is $\mathsf{0} \cong \mathsf{1}$ and $A \times B \cong A + B$.

Exercise A.6.13. Suppose A and B are Abelian groups. Is there a difference between their coproduct in the category Group of groups, and their coproduct in the category $\mathsf{AbGroup}$ of Abelian groups?

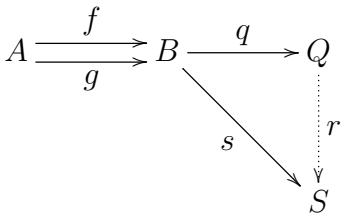
[8] An Abelian group is one that satisfies the commutative law $x \cdot y = y \cdot x$.

A.6.9 Coequalizers

Given objects and morphisms

$$A \underset{g}{\overset{f}{\rightrightarrows}} B \xrightarrow{q} Q$$

we say that q *coequalizes* f and g when $e \circ f = e \circ g$. A *coequalizer* of f and g is a *universal* coequalizing morphism; thus $q : B \to Q$ is a coequalizer of f and g when it coequalizes them and, for all $s : B \to S$, if $s \circ f = s \circ g$ then there exists a *unique* morphism $r : Q \to S$ such that $s = r \circ q$:



In Set the coequalizer of parallel functions $f : A \to B$ and $g : A \to B$ is the quotient set $Q = B/\!\sim$ where $\sim$ is the least equivalence relation on B satisfying

$$fx = gy \Rightarrow x \sim y \ .$$

The function $q : B \to Q$ is the canonical quotient map which assigns to each element $x \in B$ its equivalence class $[x] \in B/\!\sim$. In general, a coequalizer can be thought of as the quotient by the equivalence relation generated by the corresponding equation.

Exercise A.6.14. Show that a coequalizer is an epimorphism, i.e., if $q : B \to Q$ is a coequalizer of f and g, then, for all $u, v : Q \to T$, $u \circ q = v \circ q$ implies $u = v$. [Hint: use the duality between limits and colimits and Exercise A.6.3.]

Definition A.6.15. A morphism is a *regular epi* if it is a coequalizer.

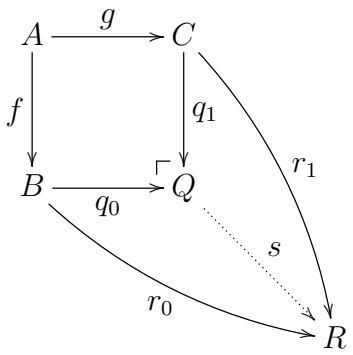
The difference between epis and regular epis is also illustrated in the category Top: a continuous map $f : X \to Y$ is epi when it is surjective, whereas it is a regular epi when it is a topological quotient map.[9]

A.6.10 Pushouts

A *pushout* of $f : A \to B$ and $g : A \to C$ is an object Q with morphisms $q_0 : B \to Q$ and $q_1 : C \to Q$ such that $q_0 \circ f = q_1 \circ g$, and whenever $r_0 : B \to R$, $r_1 : C \to R$ are such that

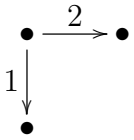
[9] A continuous map $f : X \to Y$ is a topological quotient map when it is surjective and, for every $U \subseteq Y$, U is open if, and only if, f^*U is open.

$r_0 \circ f = r_1 \circ g$, then there exists a unique $s : Q \to R$ such that $r_0 = s \circ q_0$ and $r_1 = s \circ q_1$:



We indicate that Q is a pushout by drawing a square corner next to it, as in the above diagram. The above pushout Q is sometimes denoted by $B +_A C$.

A pushout as above is a colimit of the diagram $D : \mathcal{I} \to \mathcal{C}$ where the index category $\mathcal{I}$ is



and $D1 = f$, $D2 = g$.

In Set, the pushout of $f : A \to C$ and $g : B \to C$ is the quotient set

$$Q = (B + C)/\!\sim$$

where $B + C$ is the disjoint union of B and C, and $\sim$ is the least equivalence relation on $B + C$ such that, for all $x \in A$,

$$fx \sim gx \ .$$

The functions $q_0 : B \to Q$, $q_1 : C \to Q$ are the injections, $q_0 x = [x]$, $q_1 y = [y]$, where $[x]$ is the equivalence class of x.

A.6.11 Limits as adjoints

Limits and colimits can be defined as adjoints to certain very simple functors.

First, observe that an object $A \in \mathcal{C}$ can be viewed as a functor from the terminal category $\mathsf{1}$ to $\mathcal{C}$, namely the functor which maps the only object $\star$ of $\mathsf{1}$ to A. Since $\mathsf{1}$ is the terminal object in Cat, there exists a unique functor $!_{\mathcal{C}} : \mathcal{C} \to \mathsf{1}$, which maps every object of $\mathcal{C}$ to $\star$.

Now we can ask whether this simple functor $!_{\mathcal{C}} : \mathcal{C} \to \mathsf{1}$ has any adjoints. Indeed, it has a right adjoint just if $\mathcal{C}$ has a terminal object $\mathsf{1}_{\mathcal{C}}$, for the corresponding functor $\mathsf{1}_{\mathcal{C}} : \mathsf{1} \to \mathcal{C}$ has the property that, for every $A \in \mathcal{C}$ we have a (trivially natural) bijective correspondence:

$$\frac{!_A : A \to \mathsf{1}_{\mathcal{C}}}{\mathsf{1}_\star : !_{\mathcal{C}} A \to \star}$$

Similarly, an initial object is a left adjoint to $!_{\mathcal{C}}$:

$$0_{\mathcal{C}} \dashv !_{\mathcal{C}} \dashv 1_{\mathcal{C}} \ .$$

Now consider the diagonal functor,

$$\Delta : \mathcal{C} \to \mathcal{C} \times \mathcal{C},$$

defined by $\Delta A = \langle A, A \rangle$, $\Delta f = \langle f, f \rangle$. When does this have adjoints?

If $\mathcal{C}$ has all binary products, then they determine a functor

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

which maps $\langle A, B \rangle$ to $A \times B$ and a pair of morphisms $\langle f : A \to A', g : B \to B' \rangle$ to the unique morphism $f \times g : A \times B \to A' \times B'$ for which $\pi_0 \circ (f \times g) = f \circ \pi_0$ and $\pi_1 \circ (f \times g) = g \circ \pi_1$,

$$\begin{array}{ccccc} A & \xleftarrow{\pi_0} & A \times B & \xrightarrow{\pi_1} & B \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle f \times g} & & \downarrow{\scriptstyle g} \\ A' & \xleftarrow[\pi_0]{} & A' \times B' & \xrightarrow[\pi_1]{} & B' \end{array}$$

The product functor $\times$ is right adjoint to the diagonal functor Δ. Indeed, there is a natural bijective correspondence:

$$\frac{\langle f, g \rangle : \langle A, A \rangle \to \langle B, C \rangle}{f \times g : A \to B \times C}$$

Similarly, binary coproducts are easily seen to be left adjoint to the diagonal functor,

$$+ \dashv \Delta \dashv \times \ .$$

Now in general, consider limits of shape $\mathcal{I}$ in a category $\mathcal{C}$. There is the constant diagram functor

$$\Delta : \mathcal{C} \to \mathcal{C}^{\mathcal{I}}$$

that maps $A \in \mathcal{C}$ to the constant diagram $\Delta_A : \mathcal{I} \to \mathcal{C}$. The limit construction is a functor

$$\varprojlim : \mathcal{C}^{\mathcal{I}} \to \mathcal{C}$$

that maps each diagram $D \in \mathcal{C}^{\mathcal{I}}$ to its limit $\varprojlim D$. These two are adjoint, $\Delta \dashv \varprojlim$, because there is a natural bijective correspondence between cones $\alpha : \Delta_A \Longrightarrow D$ on D, and their factorizations through the limit of D,

$$\frac{\alpha : \Delta_A \Longrightarrow D}{A \to \varprojlim D}$$

An analogous correspondence holds for colimits, so that we obtain a pair of adjunctions,

$$\varinjlim \dashv \Delta \dashv \varprojlim \ ,$$

which, of course, subsume all the previously mentioned cases.

Exercise A.6.16. How are the functors $\Delta : \mathcal{C} \to \mathcal{C}^{\mathcal{I}}$, $\varinjlim : \mathcal{C}^{\mathcal{I}} \to \mathcal{C}$, and $\varprojlim : \mathcal{C}^{\mathcal{I}} \to \mathcal{C}$ defined on morphisms?

A.6.12 Preservation of limits

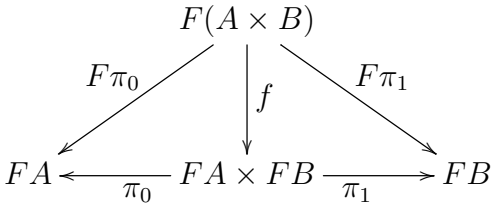
We say that a functor $F : \mathcal{C} \to \mathcal{D}$ *preserves products* when, given a product

$$A \xleftarrow{\pi_0} A \times B \xrightarrow{\pi_1} B$$

its image in $\mathcal{D}$,

$$FA \xleftarrow{F\pi_0} F(A \times B) \xrightarrow{F\pi_1} FB$$

is a product of FA and FB. If $\mathcal{D}$ has chosen binary products, F preserves binary products if, and only if, the unique morphism $f : F(A \times B) \to FA \times FB$ which makes the following diagram commutative is an isomorphism: [10]



In general, a functor $F : \mathcal{C} \to \mathcal{D}$ is said to *preserve limits* of shape $\mathcal{I}$ when it maps limit cones to limit cones: if (L, λ) is a limit of $D : \mathcal{I} \to \mathcal{C}$ then $(FL, F \circ \lambda)$ is a limit of $F \circ D : \mathcal{I} \to \mathcal{D}$.

Analogously, a functor $F : \mathcal{C} \to \mathcal{D}$ is said to *preserve colimits* of shape $\mathcal{I}$ when it maps colimit cocones to colimit cocones: if (C, ζ) is a colimit of $D : \mathcal{I} \to \mathcal{C}$ then $(FC, F \circ \zeta)$ is a colimit of $F \circ D : \mathcal{I} \to \mathcal{D}$.

Proposition A.6.17. *(a) A functor preserves finite (small) limits if, and only if, it preserves equalizers and finite (small) products. (b) A functor preserves finite (small) colimits if, and only if, it preserves coequalizers and finite (small) coproducts.*

Proof. This follows from the fact that limits are constructed from equalizers and products, cf. Proposition A.6.8, and that colimits are constructed from coequalizers and coproducts, cf. Exercise A.6.9. □

Proposition A.6.18. *For a locally small category $\mathcal{C}$, the Yoneda embedding $\mathsf{y} : \mathcal{C} \to \widehat{\mathcal{C}}$ preserves all limits that exist in $\mathcal{C}$.*

[10] Products are determined up to isomorphism only, so it would be too restrictive to require $F(A \times B) = FA \times FB$. When that is the case, however, we say that the functor F *strictly* preserves products.

Proof. Suppose (L, λ) is a limit of $D : \mathcal{I} \to \mathcal{C}$. The Yoneda embedding maps D to the diagram $\mathsf{y} \circ D : \mathcal{I} \to \widehat{\mathcal{C}}$, defined by

$$(\mathsf{y} \circ D)_i = \mathsf{y}D_i = \mathcal{C}(-, D_i) \ .$$

and it maps the limit cone (L, λ) to the cone $(\mathsf{y}L, \mathsf{y} \circ \lambda)$ on $\mathsf{y} \circ D$, defined by

$$(\mathsf{y} \circ \lambda)_i = \mathsf{y}\lambda_i = \mathcal{C}(-, \lambda_i) \ .$$

To see that $(\mathsf{y}L, \mathsf{y} \circ \lambda)$ is a limit cone on $\mathsf{y} \circ D$, consider a cone (M, μ) on $\mathsf{y} \circ D$. Then $\mu : \Delta_M \Longrightarrow D$ consists of a family of functions, one for each $i \in \mathcal{I}$ and $A \in \mathcal{C}$,

$$(\mu_i)_A : MA \to \mathcal{C}(A, D_i) \ .$$

For every $A \in \mathcal{C}$ and $m \in MA$ we get a cone on D consisting of morphisms

$$(\mu_i)_A m : A \to D_i \ . \qquad (i \in \mathcal{I})$$

There exists a unique morphism $\phi_A m : A \to L$ such that $(\mu_i)_A m = \lambda_i \circ \phi_A m$. The family of functions

$$\phi_A : MA \to \mathcal{C}(A, L) = (\mathsf{y} \circ L)A \qquad (A \in \mathcal{C})$$

forms a factorization $\phi : M \Longrightarrow \mathsf{y}L$ of the cone (M, μ) through the cone (L, λ). This factorization is unique because each $\phi_A m$ is unique. □

In effect we showed that a covariant representable functor $\mathcal{C}(A, -) : \mathcal{C} \to \mathsf{Set}$ preserves existing limits,

$$\mathcal{C}(A, \varprojlim_{i \in \mathcal{I}} D_i) \cong \varprojlim_{i \in \mathcal{I}} \mathcal{C}(A, D_i) \ .$$

By duality, the contravariant representable functor $\mathcal{C}(-, A) : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$ maps existing colimits to limits,

$$\mathcal{C}(\varinjlim_{i \in \mathcal{I}} D_i, A) \cong \varprojlim_{i \in \mathcal{I}} \mathcal{C}(D_i, A) \ .$$

Exercise A.6.19. Prove the above claim that a contravariant representable functor $\mathcal{C}(-, A) : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$ maps existing colimits to limits. Use duality between limits and colimits. Does it also follow *by a simple duality argument* that a contravariant representable functor $\mathcal{C}(-, A)$ maps existing limits to colimits? How about a covariant representable functor $\mathcal{C}(A, -)$ mapping existing colimits to limits?

Exercise A.6.20. Prove that a functor $F : \mathcal{C} \to \mathcal{D}$ preserves monos if it preserves limits. In particular, the Yoneda embedding preserves monos. Hint: Exercise A.6.5.

Proposition A.6.21. *Right adjoints preserve limits, and left adjoints preserve colimits.*

Proof. Suppose we have adjoint functors

$$\mathcal{C} \underset{G}{\overset{F}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathcal{D}$$

and a diagram $D : \mathcal{I} \to \mathcal{D}$ whose limit exists in $\mathcal{D}$. We would like to use the following slick application of Yoneda Lemma to show that G preserves limits: for every $A \in \mathcal{C}$,

$$\begin{aligned}\mathcal{C}(A, G(\varprojlim D)) \cong \mathcal{D}(FA, \varprojlim D) &\cong \varprojlim_{i \in \mathcal{I}} \mathcal{D}(FA, D_i) \\ &\cong \varprojlim_{i \in \mathcal{I}} \mathcal{C}(A, GD_i) \cong \mathcal{C}(A, \varprojlim (G \circ D)) \ .\end{aligned}$$

Therefore $G(\lim D) \cong \lim(G \circ D)$. However, this argument only works if we already know that the limit of $G \circ D$ exists.

We can also prove the stronger claim that whenever the limit of $D : \mathcal{I} \to \mathcal{D}$ exists then the limit of $G \circ D$ exists in $\mathcal{C}$ and its limit is $G(\lim D)$. So suppose (L, λ) is a limit cone of D. Then $(GL, G \circ \lambda)$ is a cone on $G \circ D$. If (A, α) is another cone on $G \circ D$, we have by adjunction a cone (FA, γ) on D,

$$\frac{\alpha_i : A \to GD_i}{\gamma_i : FA \to D_i}$$

There exists a unique factorization $f : FA \to L$ of this cone through (L, λ). Again by adjunction, we obtain a unique factorization $g : A \to GL$ of the cone (A, α) through the cone $(GL, G \circ \lambda)$:

$$\frac{f : FA \to L}{g : A \to GL}$$

The factorization g is unique because γ is uniquely determined from α, f uniquely from α, and g uniquely from f.

By a dual argument, a left adjoint preserves colimits. □

Appendix B

Logic

B.1 Concrete and abstract syntax

By *syntax* we generally mean manipulation of finite strings of symbols according to given *grammatical rules*. For instance, the strings "$7)6+/(8$" and "$(6+8)/7$" both consist of the same symbols but you will recognize one as junk and the other as *well formed* because you have (implicitly) applied the grammatical rules for arithmetical expressions.

Grammatical rules are usually quite complicated, as they need to prescribe associativity of operators (does "$5+6+7$" mean "$(5+6)+7$" or "$5+(6+7)$"?) and their precedence (does "$6+8/7$" mean "$(6+8)/7$" or "$6+(8/7)$"?), the role of *white space* (empty space between symbols and line breaks), rules for nesting and balancing parentheses, etc. It is not our intention to dwell on such details, but rather to focus on the mathematical nature of well-formed expressions, namely that they represent inductively generated finite trees.[1] Under this view the string "$(6+8)/7$" is just a concrete representation of the tree depicted in Figure B.1.

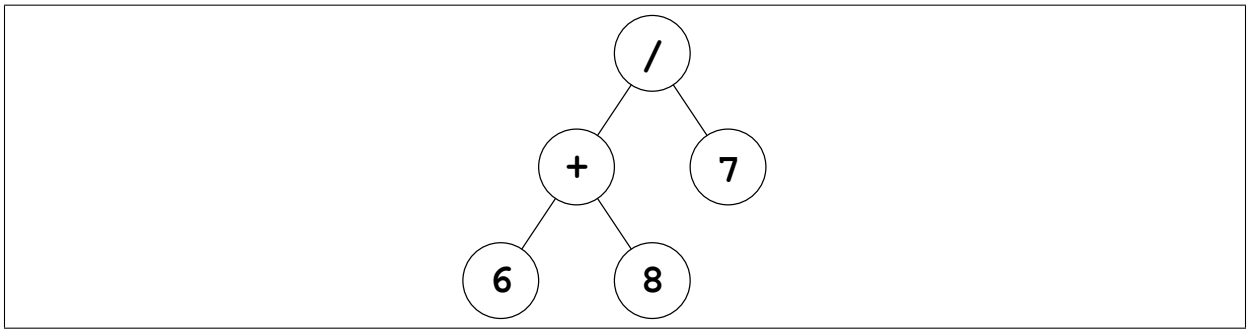


Figure B.1: The tree represented by $(6+8)/7$

Concrete representation of expressions as finite strings of symbols is called *concrete syntax*, while in *abstract syntax* we view expressions as finite trees. The passage from the

[1] We are limiting attention to the so-called *context-free* grammar, which are sufficient for our purposes. More complicated grammars are rarely used to describe formal languages in logic and computer science.

former to the latter is called *parsing* and is beyond the scope of this book. We will always specify only abstract syntax and assume that the corresponding concrete syntax follows the customary rules for parentheses, associativity and precedence of operators.

As an illustration we give rules for the (abstract) syntax of propositional calculus in *Backus-Naur* form:

$$\text{Propositional variable } p ::= \mathtt{p}_1 \mid \mathtt{p}_2 \mid \mathtt{p}_3 \mid \cdots$$
$$\text{Propositional formula } \phi ::= p \mid \bot \mid \top \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \Rightarrow \phi_2 \mid \neg\phi$$

The vertical bars should be read as "or". The first rule says that a propositional variable is the constant $\mathtt{p}_1$, or the constant $\mathtt{p}_2$, or the constant $\mathtt{p}_3$, etc.[2] The second rule tells us that there are seven inductive rules for building a propositional formula:

- a propositional variable is a formula,
- the constants $\bot$ and $\top$ are formulas,
- if ϕ_1, ϕ_2, and ϕ are formulas, then so are $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\phi_1 \Rightarrow \phi_2$, and $\neg\phi$.

Even though abstract syntax rules say nothing about parentheses or operator associativity and precedence, we shall rely on established conventions for mathematical notation and write down concrete representations of propositional formulas, e.g., $\mathtt{p}_4 \wedge (\mathtt{p}_1 \vee \neg \mathtt{p}_1) \wedge \mathtt{p}_4 \vee \mathtt{p}_2$.

A word of warning: operator associativity in syntax is not to be confused with the usual notion of associativity in mathematics. We say that an operator $\star$ is *left associative* when an expression $x \star y \star z$ represents the left-hand tree in Figure B.2, and *right associative* when it represents the right-hand tree. Thus the usual operation of subtraction $-$ is left

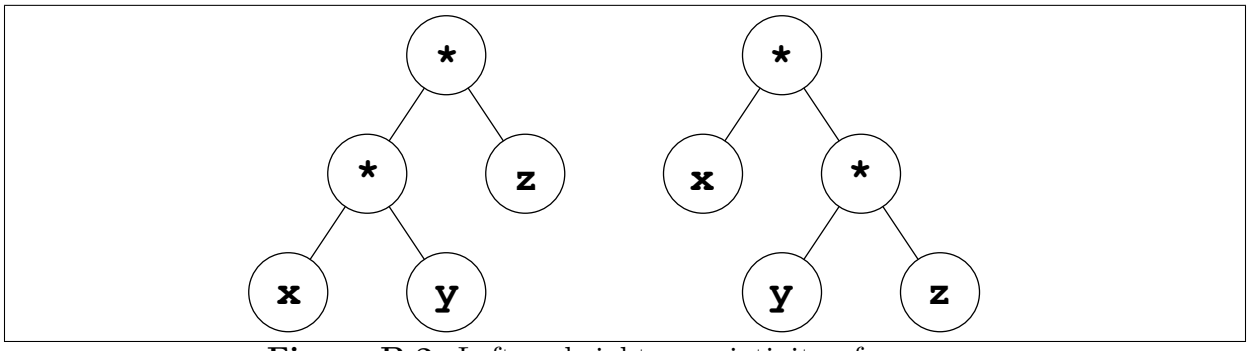


Figure B.2: Left and right associativity of $x \star y \star z$

associative, but is not associative in the usual mathematical sense.

[2] In an actual computer implementation we would allow arbitrary finite strings of letters as propositional variables. In logic we only care about the fact that we can never run out of fresh variables, i.e., that there are countably infinitely many of them.

B.2 Free and bound variables

Variables appearing in an expression may be *free* or *bound*. For example, in expressions

$$\int_0^1 \sin(a \cdot x)\, dx, \qquad x \mapsto ax^2 + bx + c, \qquad \forall\, x\,.\,(x < a \;\vee\; b < x)$$

the variables a, b and c are free, while x is bound by the integral operator $\int$, the function formation $\mapsto$, and the universal quantifier $\forall$, respectively. To be quite precise, it is an *occurrence* of a variable that is free or bound. For example, in expression $\phi(x) \vee \exists\, x\,.\, A\psi(x, x)$ the first occurrence of x is free and the remaining ones are bound.

In this book the following operators bind variables:

- quantifiers $\exists$ and $\forall$, cf. **??**,
- λ-abstraction, cf. **??**,
- search for others **??**.

When a variable is bound we may always rename it, provided the renaming does not confuse it with another variable. In the integral above we could rename x to y, but not to a because the binding operation would *capture* the free variable a to produce the unintended $\int_0^1 \sin(a^2)\, da$. Renaming of bound variables is called *α-renaming*.

We consider two expressions *equal* if they only differ in the names of bound variables, i.e., if one can be obtained from the other by α-renaming. Furthermore, we adhere to *Barendregt's variable convention [**?**, p. 2]*, which says that bound variables are always chosen so as to differ from free variables. Thus we would never write $\phi(x) \vee \exists\, x\,.\, A\psi(x, x)$ but rather $\phi(x) \vee \exists\, y\,.\, A\psi(y, y)$. By doing so we need not worry about capturing or otherwise confusing free and bound variables.

In logic we need to be more careful about variables than is customary in traditional mathematics. Specifically, we always specify which free variables may appear in an expression.[3] We write

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

to indicate that expression t may contain only free variables $x_1, \ldots, x_n$ of types $A_1, \ldots, A_n$. The list

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

is called a *context* in which t appears. To see why this is important consider the different meaning that the expression $x^2 + y^2 \leq 1$ recevieves in different contexts:

- $x : \mathbb{Z}, y : \mathbb{Z} \mid x^2 + y^2 \leq 1$ denotes the set of tuples $\{(-1, 0), (0, 1), (1, 0), (0, -1)\}$,
- $x : \mathbb{R}, y : \mathbb{R} \mid x^2 + y^2 \leq 1$ denotes the closed unit disc in the plane, and

[3] This is akin to one of the guiding principles of good programming language design, namely, that all variables should be *declared* before they are used.

- $x : \mathbb{R}, y : \mathbb{R}, z : \mathbb{R} \mid x^2 + y^2 \leq 1$ denotes the infinite cylinder in space whose base is the closed unit disc.

In single-sorted theories there is only one type or sort A. In this case we abbreviate a context by listing just the variables, $x_1, \ldots, x_n$.

B.3 Substitution

Substitution is a basic syntactic operation which replaces (free occurrences of) distinct variables $x_1, \ldots, x_n$ in an expression t with expressions $t_1, \ldots, t_n$, which is written as

$$t[t_1/x_1, \ldots, t_n/x_n].$$

We sometimes abbreviate this as $t[\vec{t}/\vec{x}]$ where $\vec{x} = (x_1, \ldots, x_n)$ and $\vec{t} = (t_1, \ldots, t_n)$. Here are several examples:

$$\begin{aligned} (x^2 + x + y)[(2+3)/x] &= (2+3)^2 + (2+3) + y \\ (x^2 + y)[y/x, x/y] &= y^2 + x \\ \left(\forall\, x\,.\, \left(x^2 < y + x^3\right)\right)[x + y/y] &= \forall\, z\,.\, \left(z^2 < (x+y) + z^3\right). \end{aligned}$$

Notice that in the third example we first renamed the bound variable x to z in order to avoid a capture by $\forall$.

Substitution is simple to explain in terms of trees. Assuming Barendregt's convention, the substitution $t[u/x]$ means that in the tree t we replace the leaves labeled x by copies of the tree u. Thus a substitution never changes the structure of the tree–it only "grows" new subtrees in places where the substituted variables occur as leaves.

Substitution satisfies the distributive law

$$(t[u/x])[v/y] = (t[v/y])[u[v/y]/x],$$

provided x and y are distinct variables. There is also a corresponding multivariate version which is written the same way with a slight abuse of vector notation:

$$(t[\vec{u}/\vec{x}])[\vec{v}/\vec{y}] = (t[\vec{v}/\vec{y}])[\vec{u}[\vec{v}/\vec{y}]/\vec{x}].$$

B.4 Judgments and deductive systems

A formal system, such as first-order logic or type theory, concerns itself with *judgments*. There are many kinds of judgments, such as:

- The most common judgments are equations and other logical statements. We distinguish a formula ϕ and the judgment "ϕ holds" by writing the latter as

  $$\vdash \phi\,.$$

  The symbol $\vdash$ is generally used to indicate judgments.

- Typing judgments
$$\vdash t : A$$
expressing the fact that a term t has type A. This is not to be confused with the set-theoretic statement $t \in u$ which says that individuals t and u (of type "set") are in relation "element of" $\in$.

- Judgments expressing the fact that a certain entity is well formed. A typical example is a judgment
$$\vdash x_1 : A_1, \ldots, x_n : A_n \quad \texttt{ctx}$$
which states that $x_1 : A_1, \ldots, x_n : A_n$ is a well-formed context. This means that $x_1, \ldots, x_n$ are distinct variables and that $A_1, \ldots, A_n$ are well-formed types. This kind of judgement is often omitted and it is tacitly assumed that whatever entities we deal with are in fact well-formed.

A *hypothetical judgement* has the form

$$H_1, \ldots, H_n \vdash C$$

and means that hypotheses $H_1, \ldots, H_n$ entail consequence C (with respect to a given decuctive system). We may also add a typing context to get a general form of judgment

$$x_1 : A_1, \ldots, x_n : A_n \mid H_1, \ldots, H_m \vdash C.$$

This should be read as: "if $x_1, \ldots, x_n$ are variables of types $A_1, \ldots, A_n$, respectively, then hypotheses $H_1, \ldots, H_m$ entail conclusion C." For our purposes such contexts will suffice, but you should not be surprised to see other kinds of judgments in logic.

A *deductive system* is a set of inference rules for deriving judgments. A typical inference rule has the form

$$\frac{J_1 \quad J_2 \quad \cdots \quad J_n}{J}\, C$$

This means that we can infer judgment J if we have already derived judgments $J_1, \ldots, J_n$, provided that the optional side-condition C is satisfied. An *axiom* is an inference rule of the form

$$\frac{}{J}$$

A *two-way rule*

$$\frac{J_1 \quad J_2 \quad \cdots \quad J_n}{K_1 \quad K_2 \quad \cdots \quad K_m}$$

is a combination of $n+m$ inference rules stating that we may infer each K_i from $J_1, \ldots, J_n$ and each J_i from $K_1, \ldots, K_m$.

A *derivation* of a judgment J is a finite tree whose root is J, the nodes are inference rules, and the leaves are axioms. An example is presented in the next subsection.

The set of all judgments that hold in a given deductive system is generated inductively by starting with the axioms and applying inference rules.

B.5 Example: Equational reasoning

Equational reasoning is so straightforward that one almost doesn't notice it, consisting mainly, as it does, of "substituting equals for equals". The only judgements are equations between terms, $s = t$, which consist of function symbols, constants, and variables. The inference rules are just the usual ones making $s = t$ a congruence relation on the terms. More formally, we have the following specification of what may be called the *equational calculus*.

$$\begin{aligned} \text{Variable } v &::= x \mid y \mid z \mid \cdots \\ \text{Constant symbol } c &::= \mathtt{c}_1 \mid \mathtt{c}_2 \mid \cdots \\ \text{Function symbol } f^k &::= \mathtt{f}_1^{k_1} \mid \mathtt{f}_2^{k_2} \mid \cdots \\ \text{Term } t &::= v \mid c \mid f^k(t_1, \ldots, t_k) \end{aligned}$$

The superscript on the function symbol f^k indicates the arity.

The equational calculus has just one form of judgement

$$x_1, \ldots, x_n \mid t_1 = t_2$$

where $x_1, \ldots, x_n$ is a *context* consisting of distinct variables, and the variables in the equation must occur among the ones listed in the context.

There are four inference rules for the equational calculus. They may be assumed to leave the contexts unchanged, which may therefore be omitted.

$$\frac{}{t = t} \qquad \frac{t_1 = t_2}{t_2 = t_1} \qquad \frac{t_1 = t_2,\ t_2 = t_3}{t_1 = t_3} \qquad \frac{t_1 = t_2,\ t_3 = t_4}{t_1[t_3/x] = t_2[t_4/x]}$$

An *equational theory* $\mathbb{T}$ consists of a set of constant and function symbols (with arities), and a set of equations, called *axioms*. We write

$$\mathbb{T} \vdash t_1 = t_2$$

to mean that the equation $t_1 = t_2$ has a derivation from the axioms of $\mathbb{T}$ using the equational calculus.

B.6 Example: Predicate calculus

We spell out the details of single-sorted predicate calculus and first-order theories. This is the most common deductive system taught in classical courses on logic.

The predicate calculus has the following syntax:

$$\begin{aligned}
\text{Variable } v &::= x \mid y \mid z \mid \cdots \\
\text{Constant symbol } c &::= \mathtt{c}_1 \mid \mathtt{c}_2 \mid \cdots \\
\text{Function symbol}^4\ f^k &::= \mathtt{f}_1^{k_1} \mid \mathtt{f}_2^{k_2} \mid \cdots \\
\text{Term } t &::= v \mid c \mid f^k(t_1, \ldots, t_k) \\
\text{Relation symbol } R^m &::= \mathtt{R}_1^{m_1} \mid \mathtt{R}_2^{m_2} \mid \cdots \\
\text{Formula } \phi &::= \bot \mid \top \mid R^m(t_1, \ldots, t_m) \mid t_1 = t_2 \mid \\
&\qquad \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \Rightarrow \phi_2 \mid \neg\phi \mid \forall\, x \,.\, \phi \mid \exists\, x \,.\, \phi.
\end{aligned}$$

The variable x is bound in $\forall\, x \,.\, \phi$ and $\exists\, x \,.\, \phi$.

The predicate calculus has one form of judgement

$$x_1, \ldots, x_n \mid \phi_1, \ldots, \phi_m \vdash \phi$$

where $x_1, \ldots, x_n$ is a *context* consisting of distinct variables, $\phi_1, \ldots, \phi_m$ are *hypotheses* and ϕ is the *conclusion*. The free variables in the hypotheses and the conclusion must occur among the ones listed in the context. We abbreviate the context with Γ and Φ with hypotheses. Because most rules leave the context unchanged, we omit the context unless something interesting happens with it.

The following inference rules are given in the form of adjunctions. See Appendix **??** for the more usual formulation in terms of introduction an elimination rules.

$$\frac{}{\phi_1, \ldots, \phi_m \vdash \phi_i} \qquad \frac{}{\Phi \vdash \top} \qquad \frac{}{\Phi, \bot \vdash \phi}$$

$$\frac{\Phi \vdash \phi_1 \quad \Phi \vdash \phi_2}{\Phi \vdash \phi_1 \wedge \phi_2} \qquad \frac{\Phi, \phi_1 \vdash \psi \quad \Phi, \phi_2 \vdash \psi}{\Phi, \phi_1 \vee \phi_2 \vdash \psi} \qquad \frac{\Phi, \phi_1 \vdash \phi_2}{\Phi \vdash \phi_1 \Rightarrow \phi_2}$$

$$\frac{\Gamma, x, y \mid \Phi, x = y \vdash \phi}{\Gamma, x \mid \Phi \vdash \phi[x/y]} \qquad \frac{\Gamma, x \mid \Phi, \phi \vdash \psi}{\Gamma \mid \Phi, \exists\, x \,.\, \phi \vdash \psi} \qquad \frac{\Gamma, x \mid \Phi \vdash \phi}{\Gamma \mid \Phi \vdash \forall\, x \,.\, \phi}$$

The equality rule implicitly requires that y does not appear in Φ, and the quantifier rules implicitly require that x does not occur freely in Φ and ψ because the judgments below the lines are supposed to be well formed.

Negation $\neg\phi$ is defined to be $\phi \Rightarrow \bot$. To obtain *classical* logic we also need the law of excluded middle,

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

Comment on the fact that contraction and weakening are admissible.

Give an example of a derivation.

A *first-order theory* $\mathbb{T}$ consists of a set of constant, function and relation symbols with corresponding arities, and a set of formulas, called *axioms*.

Give examples of a first-order theories.

Bibliography

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

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

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

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

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

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

[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.

[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.