Categorical logic from a categorical point of view

Michael Shulman

DRAFT for AARMS Summer School 2016
Version of October 14, 2020

Preface

These notes were written partly as a supplement to part of a course on higher categories and categorical logic at the AARMS Summer School 2016; but also with a goal of clarify and setting down some ideas that had only recently begun coalescing in my own head. They are currently somewhat rough, with promised sections or chapters missing and others in an imperfectly consistent state; no guarantee of correctness is provided. Nevertheless I have some hope they may be useful to the category theorist who needs some help, as I did, to figure out what this "type theory" is all about.

In my own case, whatever correct understanding I have come to (and attempted to share in this book) is due largely to the generous help and explanations provided by others, particularly Dan Licata and Peter LeFanu Lumsdaine, but also (in no particular order) Todd Trimble, Toby Bartels, Steve Awodey, Bob Harper, Vladimir Voevodsky, Thorsten Altenkirch, Martín Escardó, Andrej Bauer, and many others all of whom it would be impossible to list. Of course, all the errors are my own.

I am also very grateful to Dorette Pronk and the other organizers of the 2016 AARMS Summer School for the invitation to co-teach a course on categorical logic; and to Peter Lumsdaine for teaching the other half of the course and helping with the planning of my half. Finally, I would like to thank all the students in that course for the lively classroom discussion; frequently their questions made me realize that something was unclear or missing from the notes.

Contents

Chapter 0

Introduction

In this optional chapter I will attempt to motivate and describe the discipline of categorical logic for the newcomer, and also locate this book within the ecosystem thereof for the expert. Nothing herein is required for reading the rest of the book; but I hope that it may serve some purpose nevertheless.

0.1 Appetizer: inverses in group objects

In this section we consider an extended example. We do not expect the reader to understand it very deeply, but we hope it will give some motivation for what follows, as well as a taste of the power and flexibility of categorical logic as a tool for category theory.

Our example will consist of several varations on the following theorem:

Theorem 0.1.1. *If a monoid has inverses (hence is a group), then those inverses are unique.*

When "monoid" and "group" have their usual meaning, namely sets equipped with structure, the proof is easy. For any x, if y and z are both two-sided inverse of x, then we have

$$y = y \cdot e = y \cdot (x \cdot z) = (y \cdot x) \cdot z = e \cdot z = z \tag{0.1.2}$$

However, the theorem is true much more generally than this. We consider first the case of monoid/group objects in a category with products. A *monoid object* is an object A together with maps $m : A \times A \to A$ and $e : 1 \to A$ satisfying associativity and unitality axioms:

$$\begin{array}{ccc} A \times A \times A & \xrightarrow{1\times m} & A \times A \\ \downarrow{\scriptstyle m\times 1} & & \downarrow{\scriptstyle m} \\ A \times A & \xrightarrow[m]{} & A \end{array} \qquad \begin{array}{ccccc} A & \xrightarrow{(1,e)} & A \times A & \xleftarrow{(e,1)} & A \\ & {\scriptstyle 1}\searrow & \downarrow{\scriptstyle m} & \swarrow{\scriptstyle 1} & \\ & & A & & \end{array} \tag{0.1.3}$$

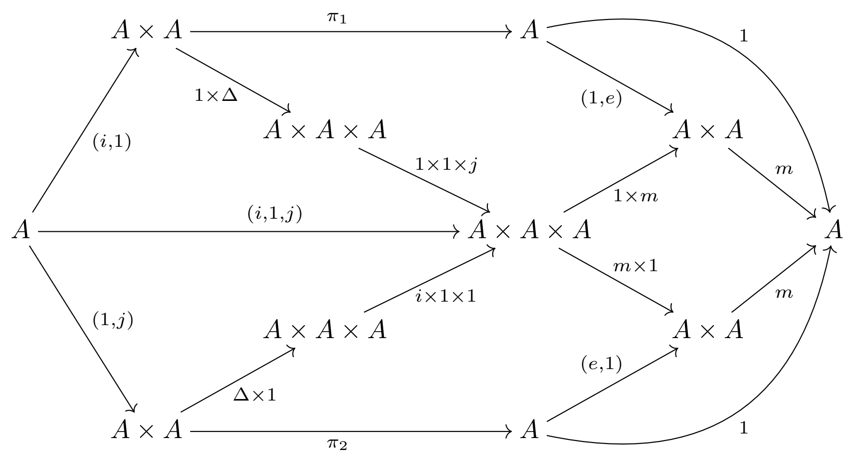
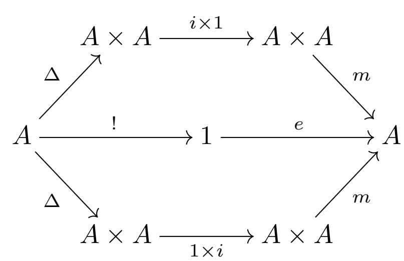


Figure 1: Uniqueness of inverses by diagram chasing

An *inverse operator* for a monoid object is a map $i : A \to A$ such that the following diagrams commute:



(0.1.4)

The internalized claim, then, is that *any two inverse operators for a monoid object are equal*. A standard category-theoretic proof would be to suppose i and j are both inverse operators and draw a large commutative diagram such as that shown in Figure 1. Here the composite around the top is equal to i, the composite around the bottom is equal to j, and all the internal polygons commute either by one of the monoid axioms, the inverse axiom for i or j, or the universal property of products. (We encourage the reader to verify this.)

While there is a certain beauty to Figure 1, it takes considerable effort to write it down and arrange it in such a pleasing form (as opposed to a horrid mess on scratch paper), let alone typeset it prettily. And this is really a fairly simple fact about monoids; for more complicated theorems, the complexity of the resulting diagrams grows accordingly (see Exercises 0.1.4 and 0.1.5).

Nevertheless, there is a sense in which Figure 1 is obtained *algorithmically* from the simple proof (0.1.2). Specifically, each expression in (0.1.2) corresponds to one or more paths through Figure 1, and each equality in (0.1.2) corresponds to a commuting polygon in Figure 1.[1] With experience, one can learn to do such translations without much effort, at least in simple cases. However, if it really is an algorithm, we shouldn't have to re-do it on a case-by-case basis at all; we should be able to prove a single general "meta-theorem" and then appeal to it whenever we want to. This is the goal of categorical logic.

Specifically, the *type theory for categories with products* allows us to replace Figure 1 by an argument that looks almost the same as (0.1.2). The morphisms m and e are represented in this logic by the notations

$$x : A, y : A \vdash x \cdot y : A \qquad\qquad \vdash e : A.$$

Don't worry if this notation doesn't make a whole lot of sense yet. The symbol $\vdash$ (called a "turnstile") is the logic version of a morphism arrow $\to$, and the entire notation is called a *sequent* or a *judgment*. The fact that m is a morphism $A \times A \to A$ is indicated by the fact that A appears twice to the left of $\vdash$ and once to the right; the comma "," in between $x : A$ and $y : A$ represents the product $\times$, and the variables x, y are there so that we have a good notation "$x \cdot y$" for the morphism m. In particular, the notation $x : A, y : A \vdash x \cdot y : A$ should be bracketed as

$$((x : A), (y : A)) \vdash ((x \cdot y) : A).$$

Similarly, the associativity, unit, and inverse axioms are indicated by the notations

$$x : A, y : A, z : A \vdash (x \cdot y) \cdot z = x \cdot (y \cdot z) : A$$

$$x : A \vdash x \cdot e = x : A \qquad\qquad x : A \vdash e \cdot x = x : A$$

$$x : A \vdash x \cdot i(x) = e : A \qquad\qquad x : A \vdash i(x) \cdot x = e : A$$

Now (0.1.2) can be essentially copied in this notation:

$$x : A \vdash i(x) = i(x) \cdot e = i(x) \cdot (x \cdot j(x)) = (i(x) \cdot x) \cdot j(x) = e \cdot j(x) = j(x) : A.$$

The essential point is that the notation *looks set-theoretic*, with "variables" representing "elements", and yet (as we will see) its formal structure is such that it can be interpreted into *any* category with products. Therefore, writing the proof in this way yields automatically a proof of the general theorem that any two inverse *operators* for a monoid *object* in a category with products are equal.

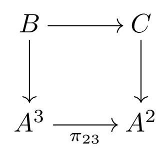
Before leaving this appetizer section, we mention some further generalizations of this result. While type theory allows us to use set-like notation to prove facts about any category with finite products, the allowable notation is fairly limited, essentially restricting us to algebraic calculations with variables. However, if our category has more structure, then we can "internalize" more set-theoretic arguments.

[1] Not every polygon in Figure 1 corresponds to anything in (0.1.2), though: the "universal property" quadrilaterals on the left are "invisible" algebraically. This is why we said each expression corresponds to "one or more" paths: $y \cdot (x \cdot z)$ and $(y \cdot x) \cdot z$ don't care which route we take from A to $A \times A \times A$.

As an example, note that for ordinary monoids in sets, the uniqueness of inverses (0.1.2) is expressed "pointwise" rather than in terms of inverse-assigning operators. In other words, for each element $x \in A$, if x has two two-sided inverses y and z, then $y = z$, regardless of whether any other elements of A have inverses. If we think hard enough, we can express this diagrammatically in terms of the category **Set**. Consider the following two sets:

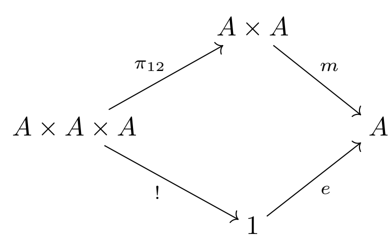
$$B = \{ (x, y, z) \in A^3 \mid xy = e, yx = e, xz = e, zx = e \}$$
$$C = \{ (y, z) \in A^2 \mid y = z \}$$

In other words, B is the set of elements x equipped with two inverses, and C is the set of pairs of equal elements. Then the uniqueness of pointwise inverses can be expressed by saying there is a commutative diagram



where the vertical arrows are inclusions and the lower horizontal arrow projects to the second and third components.

This is a statement that makes sense for a monoid object A in any category with finite *limits*. The object C can be constructed categorically as the equalizer of the two projections $A \times A \rightrightarrows A$ (which is in fact isomorphic to A itself), while the object B is a "joint equalizer" of four parallel pairs, one of which is



and the others are similar. We can then try to *prove*, in this generality, that there is a commutative square as above. We can do this by manipulating arrows, or by appealing to the Yoneda lemma, but we can also use a *type theory for categories with finite limits*. This is a syntax like the type theory for categories with finite products, but which also allows us to *hypothesize equalities*. The judgment in question is

$$x : A, y : A, z : A, (x \cdot y = e), (y \cdot x = e), (x \cdot z = e), (z \cdot x = e) \vdash (y = z). \quad (0.1.5)$$

As before, the comma binds the most loosely, so this should be read as

$$((x : A), (y : A), (z : A), (x \cdot y = e), (y \cdot x = e), (x \cdot z = e), (z \cdot x = e)) \vdash (y = z).$$

We can prove this by set-like equational reasoning, essentially just as before. The "interpretation machine" then produces from this a morphism $B \to C$, for the objects B and C constructed above.

Next, note that in the category **Set**, the uniqueness of inverses ensures that if every element $x \in A$ has an inverse, then there is a *function* $i : A \to A$ assigning inverses — even without using the axiom of choice. (If we define functions as sets of ordered pairs, as is usual in set-theoretic foundations, we could take $i = \{ (x, y) \mid xy = e \}$; the pointwise uniqueness ensures that this is indeed a function.) This fact can be expressed in the type theory of an *elementary topos*. We postpone the definition of a topos until later; for now we just remark that its structure allows both sides of the turnstile $\vdash$ to contain *logical formulas* such as $\exists x.\forall y.\phi(x, y)$ rather than just elements and equalities. In this language we can state and prove the following:

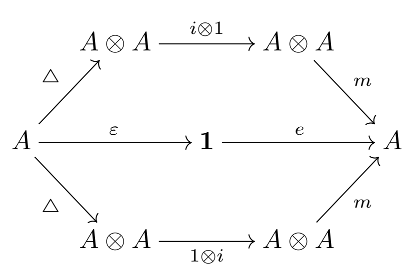
$$\forall x{:}A.\, \exists y{:}A.\, ((x \cdot y = e) \wedge (y \cdot x = e)) \vdash \exists i{:}A^A.\, \forall x{:}A.\, ((x \cdot i(x) = e) \wedge (i(x) \cdot x = e))$$

As before, the proof is essentially exactly like the usual set-theoretic one. Moreover, the interpretation machine allows us to actually extract an "inverse operator" morphism in the topos from this proof. Such a result can also be stated and proved using arrows and commutative diagrams, but as the theorems get more complicated, the translation gets more tedious to do by hand, and the advantage of type-theoretic notation becomes greater.

So much for adding extra structure. In fact, we can also take structure away! A monoid object can be defined internal to any *monoidal* category, not just a cartesian monoidal one; now the structure maps are $m : A \otimes A \to A$ and $e : \mathbf{1} \to A$, and the commutative diagrams are essentially the same.

To define an inverse operator in this case, however, we need some sort of "diagonal" $\triangle : A \to A \otimes A$ and also a "projection" or "augmentation" $\varepsilon : A \to \mathbf{1}$. The most natural hypothesis is that these maps make A into a *comonoid* object, i.e. a monoid in the opposite monoidal category, and that the monoid and comonoid structures preserve each other; this is the notion of a *bimonoid* (or "bialgebra"). (Exercise 0.1.1: in a cartesian monoidal category, every object is a bimonoid in a unique way.)

Now given a bimonoid A, we can define an "inverse operator" — which in this context is usually called an *antipode* — to be a map $i : A \to A$ such that



(0.1.6)

commutes, where now $\triangle$ and ε are the comonoid structure of A rather than the diagonal and projection of a cartesian product. A bimonoid equipped with an

$$
\begin{aligned}
x : A, y : A &\vdash x \cdot y : A \\
&\vdash e : A \\
x : A, y : A, z : A &\vdash (x \cdot y) \cdot z = x \cdot (y \cdot z) : A \\
x : A &\vdash x \cdot e = x : A \\
x : A &\vdash e \cdot x = x : A
\end{aligned}
$$

$$
\begin{aligned}
x : A &\vdash (x_{(1)}, x_{(2)}) : (A, A) \\
x : A &\vdash (\mid \not{x}) : () \\
x : A &\vdash (x_{(1)(1)}, x_{(1)(2)}, x_{(2)}) = (x_{(1)}, x_{(2)(1)}, x_{(2)(2)}) : (A, A, A) \\
x : A &\vdash (x_{(1)} \mid \cancel{x_{(2)}}) = x : A \\
x : A &\vdash (x_{(2)} \mid \cancel{x_{(1)}}) = x : A
\end{aligned}
$$

Figure 2: Monoids and comonoids

antipode is called a *Hopf monoid* (or "Hopf algebra"). The obvious question then is, if a bimonoid has two antipodes, are they equal?

In some cases it is possible to apply the previous results directly. For instance, the category of *(co)commutative* comonoids in a symmetric monoidal category inherits a monoidal structure that turns out to be *cartesian* (Exercise 0.1.2), so a cocommutative bimonoid is actually a monoid in a cartesian monoidal category, and we can apply the first version of our result. Similarly, the category of commutative monoids is cocartesian, so a commutative bimonoid is a comonoid in a cocartesian monoidal category, so we can apply the dual of the first version of our result. But what if neither the multiplication nor the comultiplication is commutative?

Internal logic is up to this task. In a monoidal category we can consider judgments with multiple outputs as well as multiple inputs.[2] This allows us to describe monoids and comonoids in a roughly "dual" way; compare the first and second groups of axioms in Figure 2.

Don't worry about the precise syntax being used in the comonoid axioms; it will be explained in §3.1. (But if you are familiar with the informal "Sweedler notation" for coalgebras, it should look familiar.) The main point to make about this syntax is that on the right we have lists such as $(M, N \mid P) : (A, B)$, where M and N correspond in some way to A and B respectively, while P and anything else on the right of the divider has "no type" (we call it a *scalar*; categorically it represents a morphism to the unit object).

[2]For the benefit of readers who are already experts, I should mention that this is *not* ordinary "classical linear logic": the comma represents the same monoidal structure $\otimes$ on both sides of the turnstile, rather than $\otimes$ on the left and $\invamp$ on the right.

In this syntax, the bimonoid axioms are

$$\begin{aligned}
x : A, y : A &\vdash (x_{(1)} \cdot y_{(1)}, x_{(2)} \cdot y_{(2)}) = ((x \cdot y)_{(1)}, (x \cdot y)_{(2)}) : (A, A)\\
&\vdash (e_{(1)}, e_{(2)}) = (e, e) : (A, A)\\
x : A, y : A &\vdash (\,|\, \cancel{x \cdot y}) = (\,|\, \cancel{x}, \cancel{y}) : ()\\
&\vdash (\,|\, \cancel{e}) = () : ()
\end{aligned}$$

And an antipode is a map $x : A \vdash i(x) : A$ such that

$$\begin{aligned}
x : A &\vdash (x_{(1)} \cdot i(x_{(2)})) = (e \mid \cancel{x}) : A\\
x : A &\vdash (i(x_{(1)}) \cdot x_{(2)}) = (e \mid \cancel{x}) : A
\end{aligned}$$

Now if we have another antipode j, we can compute

$$\begin{aligned}
x : A \vdash (i(x)) &= (i(x) \cdot e)\\
&= (i(x_{(1)}) \cdot e \mid \cancel{x_{(2)}})\\
&= i(x_{(1)}) \cdot (x_{(2)(1)} \cdot j(x_{(2)(2)}))\\
&= (i(x_{(1)}) \cdot x_{(2)(1)}) \cdot j(x_{(2)(2)})\\
&= (i(x_{(1)(1)}) \cdot x_{(1)(2)}) \cdot j(x_{(2)})\\
&= (e \cdot j(x_{(2)}) \mid \cancel{x_{(1)}})\\
&= (e \cdot j(x))\\
&= (j(x)) \qquad : A
\end{aligned}$$

yielding the same result $i = j$. (In fact, as in traditional Sweedler notation, we can also incorporate coassociativity directly into the notation, writing $(x_{(1)}, x_{(2)}, x_{(3)})$ instead of $(x_{(1)(1)}, x_{(1)(1)}, x_{(2)})$ or $(x_{(1)}, x_{(2)(1)}, x_{(2)(2)})$; this shortens the above proof by one line.)

So even in a non-cartesian situation, we can use a very similar set-like argument, as long as we keep track of where elements get "duplicated and discarded".

This concludes our "appetizer"; I hope it has given you a taste of what categorical logic looks like, and what it can do for category theory. In chapter 1 we will rewind back to the beginning and start with very simple type theories (even simpler than the ones we used in this section). Before we actually start doing type theory, however, let me prepare the ground a little by explaining how, in principle, the sort of "interpretation machine" mentioned above can work.

Exercises

Exercise 0.1.1. Prove that in a cartesian monoidal category, every object is a bimonoid in a unique way.

Exercise 0.1.2. Show that the category of cocommutative comonoids in a symmetric monoidal category inherits a monoidal structure, and that this monoidal structure is cartesian.

Exercise 0.1.3. Prove, using arrows and commutative diagrams, that any two antipodes for a bimonoid (not necessarily commutative or cocommutative) are equal.

Exercise 0.1.4. Suppose A is a set with two monoid structures (m_1, e) and (m_2, e) having the same unit element e, and satisfying the "interchange law" $m_1(m_2(x, y), m_2(z, w)) = m_2(m_1(x, z), m_1(y, w))$. Then we have

$$m_1(x, y) = m_1(m_2(x, e), m_2(e, y)) = m_2(m_1(x, e), m_1(e, y)) = m_2(x, y)$$

and also

$$m_1(x, y) = m_1(m_2(e, x), m_2(y, e)) = m_2(m_1(e, y), m_1(x, e)) = m_2(y, x)$$

so that $m_1 = m_2$ and both are commutative. This is called the *Eckmann–Hilton argument*. State and prove an analogous fact about objects in any category with finite products having two monoid structures satisfying an "interchange law". *(In Exercises 1.7.3 and 2.9.1 you will re-do this proof using internal logic for comparison.)*

Exercise 0.1.5. A "distributive near-ring" is like a ring but without the assumption that addition is commutative; thus we have a monoid structure $(\cdot, 1)$ and a group structure $(+, 0)$ such that $\cdot$ distributes over $+$ on both sides.

(a) Prove that every distributive near-ring is actually a ring. *(For this reason, in an unqualified "near-ring" only one side of distributivity is assumed.)*

(b) Define a "distributive near-ring object" in a category with finite products. Try for a little while to prove that any such is actually a "ring object", at least until you can see how much work it would be. In Exercises 1.7.4 and 2.9.1 you will prove this using type theory for comparison.

0.2 On syntax and free objects

The way that type theory allows us to prove things about categorical structures is by providing a *syntactic presentation of free objects*. To explain what this means, let's consider an example that (apparently) has very little to do with type theory or category theory. The following is a standard result from elementary group theory.

Theorem 0.2.1. *Recall that for elements g, h of a group G, the* ***conjugation*** *of h by g is defined by $h^g = ghg^{-1}$. For any g, h, k we have $h^g\, k^g = (hk)^g$.*

Proof.

$$h^g\, k^g = (ghg^{-1})(gkg^{-1}) = ghkg^{-1} = (hk)^g$$ □

Now this "proof" is not, technically, a complete proof from the usual axioms of a group. In fact, even the definition of conjugation is not, technically, meaningful, because the usual axioms of a group only involve a way to multiply *two* elements, not three. Technically, we should choose a parenthesization and write, say, $h^g = (gh)g^{-1}$; and then use the associativity and unit axioms explicitly throughout the above proof:

$$\begin{aligned} h^g k^g &= ((gh)g^{-1})((gk)g^{-1}) = (g(hg^{-1}))((gk)g^{-1}) = ((g(hg^{-1}))(gk))g^{-1} \\ &= (g((hg^{-1})(gk)))g^{-1} = (g(h(g^{-1}(gk))))g^{-1} = (g(h((g^{-1}g)k)))g^{-1} \\ &= (g(h(ek)))g^{-1} = (g(hk))g^{-1} = (hk)^g \end{aligned}$$

Of course, this would be horrific, so no one ever does it. But what justifies *not* doing it?

Normally, if mathematicians think about this sort of question at all, they would probably say that technically the extra steps have to be there, but we omit them because the reader could fill them in him- or herself. There's nothing intrinsically wrong with this (although it does start to become problematic when formalizing mathematics in a computer, since the computer can't fill in the steps itself unless some programmer takes the time to teach it exactly how).

Interestingly, however, there *is* a way to make the nice short proof of Theorem 0.2.1 completely rigorous on its own. Consider the *free group* F_3 generated by three elements g, h, k. Then the elements of F_3 can be represented by finite strings composed of the letters g, h, k and their formal inverses, in which no letter is ever adjacent to its inverse (we call these **reduced words**). In particular, ghg^{-1} and gkg^{-1} and $ghkg^{-1}$ are all elements of F_3, and the product $(ghg^{-1})(gkg^{-1})$ is equal to $ghkg^{-1}$ by definition of multiplication in F_3 (concatenate strings and cancel any elements adjacent to their inverses). Thus, the calculation in the proof of Theorem 0.2.1 makes literal sense as a statement *about elements of* F_3.

Of course, we want the theorem to be true about *any* three elements of *any* group, not just the generators of F_3. But if we have three such elements $g', h', k' \in G$, the freeness of F_3 means there is a unique group homomorphism $\overline{\omega} : F_3 \to G$ such that $\overline{\omega}(g) = g'$, $\overline{\omega}(h) = h'$, and $\overline{\omega}(k) = k'$. Since $\overline{\omega}$ is a group homomorphism, it preserves conjugation and multiplication. Thus, since Theorem 0.2.1 is true about $g, h, k \in F_3$, it must also be true about $g', h', k' \in G$.

This is the basic method of categorical logic: we do a calculation in a free structure, then map it everywhere else using the unique morphisms determined by the universal property of that free structure. Of course, not much is gained by this in our current fairly trivial example; in particular, no one would ever consider teaching undergraduates group theory that way! But as we will see, the same principle applies in much more complicated situations, and tends to get more and more useful the more complicated the structures and proofs are.

It's natural, however, to wonder why such an approach gains us *anything at all*! Why would it be any easier to prove something in a free group than in an arbitrary group? It almost seems as if it must be false by definition: anything that's true in a free group must be true in all groups, precisely by freeness, so any proof that works in a free group must work in any group.

This "false by definition" argument is almost valid. It *is* valid if the only thing we know about free groups is their universal property. The crucial ingredient in our simplified proof of Theorem 0.2.1, however, is that we knew *more* about free groups than their universal property: we had an explicit description of their elements as reduced words. Thus, we were able to make use of this knowledge to give a proof in a free group that wouldn't work in an arbitrary group.

I want to emphasize that this explicit description of a free group is *not* a trivial consequence of its universal property. There is a "tautological" way to construct free groups, but it produces a quite different description:

(a) Start with the generators.

(b) Successively apply the operations appearing in the definition (binary multiplication, the unary operation of inversion, and the nullary operation of the identity element) without reference to the axioms. This yields expressions such as $(gh)^{-1}(k^{-1}(kg))$.

(c) Define an equivalence relation on these expressions to be the smallest one that forces all the axioms to hold and is respected by all the operations. Thus, for instance, $(gh)^{-1}(k^{-1}(kg))$ would be identified with $(h^{-1}((g^{-1}k^{-1})k))g$, and also $(gh)^{-1}g$, and also h^{-1}.

(d) The quotient of this equivalence relation is the free group on our chosen generators.

This sort of method works to construct free algebras for any "algebraic theory"; but it would not help us justify the short proof of Theorem 0.2.1. The tautological construction produces a free group whose elements are equivalence classes, without any way to choose canonical representatives; in contrast to our explicit description with words, which involved no equivalence classes at all. Moreover, there are other algebraic theories, such as *abelian* groups, for which there *are* no canonical representatives for elements of free algebras; so something relatively special to do with groups in particular is happening here.

Roughly the same thing is also true for categorical logic: for many kinds of categorical structure, "something special" happens, enabling us to give an explicit description of free structures, and thereby simplify many proofs. Moreover, the "something special" that happens is more or less the same thing that happens in the case of groups; so it is worth explaining the latter a bit more.

How do we prove that the free group on a set X can be presented using reduced words, given that it is not the tautological construction? The very first thing we need to do is prove that the set of reduced words, call it $\mathfrak{F}X$, *is* a group. To multiply two reduced words w_1 and w_2, we have to concatenate them, but then "cancel all the element-inverse pairs that appear in the middle". A very formal way to describe this process is by induction.

Consider the second word w_2. If it has length 0 (i.e. it is the empty word), then we can define the product $w_1 \cdot w_2$ to be just w_1. Otherwise, w_2 must end with either a generator or its inverse. Suppose it ends with a generator, so that $w_2 = w_2' g$ for some $g \in X$ (we leave the other case to the reader). Then w_2' is shorter than w_2, so by induction on its length, we may suppose that we have defined how to multiply w_1 by w_2', obtaining a new word $w_1 \cdot w_2'$.

Now, since we hope multiplication will be associative, and we expect w_2 to actually be the product $w_2' \cdot g$ (not just the concatenation), we should have $w_1 \cdot w_2 = w_1 \cdot (w_2' \cdot g) = (w_1 \cdot w_2') \cdot g$. Thus, since we have inductively defined $w_1 \cdot w_2'$, we only need to multiply it on the right by g. We would like to just concatenate g on the end, but this might not result in a reduced word, if it happens that $w_1 \cdot w_2'$ ends with g^{-1}. (How could this happen, given that w_2' doesn't end with g (since $w_2 = w_2'g$ is reduced)? The simplest case is if w_1 ends with g^{-1} and w_2' is empty. More generally, all of w_2' could get canceled by part of w_1 to expose a g^{-1} inside w_1.)

Thus, we have to inspect $w_1 \cdot w_2'$. If it ends with g^{-1}, say $w_1 \cdot w_2' = w_3 g^{-1}$, then we define the product $w_1 \cdot w_2$ to be w_3. Otherwise, the concatenated word $(w_1 \cdot w_2')g$ is reduced, so we can define it to be $w_1 \cdot w_2$.

This completes our formal definition of multiplication of reduced words. The reason for writing out the proof in such a pedantic way is that essentially the same method works for the rest of the argument. For instance, how do we prove that multiplication is associative? Given three reduced words w_1, w_2, and w_3, we induct on the length of w_3. If it is empty, then $(w_1 \cdot w_2) \cdot w_3$ and $w_1 \cdot (w_2 \cdot w_3)$ are both $w_1 \cdot w_2$ by definition. Otherwise, $w_3 = w_3'g$ (or $w_3 = w_3'g^{-1}$), and we can use the definitions of multiplication and an inductive hypothesis that $(w_1 \cdot w_2) \cdot w_3' = w_1 \cdot (w_2 \cdot w_3')$.

Similarly, how do we extend a function $\omega : X \to G$ to a group homomorphism $\overline{\omega} : \mathfrak{F}X \to G$? Each reduced word is either empty, in which case it must go to the identity of G, or of the form wg for some $g \in X$, in which case it must go to the product $\overline{\omega}(w) \cdot \omega(g)$ in G. And we prove that $\overline{\omega}(w_1 \cdot w_2) = \overline{\omega}(w_1) \cdot \overline{\omega}(w_2)$ by — you guessed it — induction on the length of w_2.

This fairly simple proof actually displays many of the characteristics of analogous proofs about type theories that we will encounter throughout the book. The construction of multiplication in $\mathfrak{F}X$ is a simple form of **cut admissibility**, and the proof that $\mathfrak{F}X$ is the free group on X provides a prototype for the "initiality theorems" that we will prove for all of our type theories. (This is the "something special" I referred to earlier that happens for groups, and also for categories, but not for abelian groups.) But the most important thing to take away is the overall picture: we gave a concrete description of a free structure that was not obvious from its universal property, enabling us to write proofs in the free structure that would not make sense in an arbitrary structure, but nevertheless imply conclusions about arbitrary structures by the universal property.

Now you may be able to look back at §0.1 and have a slightly better idea of what is happening. The funny type-theoretic syntax such as $x : A, y : A \vdash x \cdot y : A$ is a particular explicit presentation of (in this case) the category with products "freely generated by a monoid with two inverse operators". This is a category with products, say $\mathfrak{F}\mathcal{I}$, containing a monoid object A with two inverse operators, with the property that given any other category with products $\mathcal{M}$ and a monoid object B therein with two inverse operators, there is a unique functor $\mathfrak{F}\mathcal{I} \to \mathcal{M}$ mapping A and its inverse operators to B and its inverse operators. Our calculation in this type theory showed that the two inverse operators of A are equal; therefore, so must those of B, for any $\mathcal{M}$ and B.

I emphasize again that we will make all of this more precise later on, so it is not necessary to understand deeply right now. But the idea is unfamiliar enough to many mathematicians that you may need to let it wash over you for a while before coming to understand it. (Certainly that was my own experience with learning type theory.) I also hope that seeing this glimpse of the bigger picture will motivate the reader to make it through the (sometimes rather technical) details of subsequent chapters.

Remark 0.2.2. It is worth noting that type theory is not the only syntax for free structures in category theory; probably the best-known alternative syntax is *string diagrams* (see e.g. [**?**, **?**]). Type theory and string diagrams use the same idea of giving a concrete presentation of a free object that is easier to reason about, but the particular presentations used are quite different. Type-theoretic presentations are typically characterized by cut-admissibility and similar theorems, whereas string diagrams use topological structures and deformations. Each has strengths and weaknesses; some proofs are easy in type theory and difficult with string diagrams, while for other proofs the opposite is true.

In fact, the *usual* way of reasoning in category theory (or any other subject), in which we speak explicitly about objects, arrows, and so on, can be interpreted to be simply making use of the "tautological" presentation of a free structure rather than some fancier one. In this case, of course, there is nothing practical to be gained by such a viewpoint; but it may help to "relate the new with the old".

Exercises

Exercise 0.2.1. Write out the remaining details in the proof that $\mathfrak{F}X$ is the free group generated by the set X.

0.3 On type theory and category theory

So much for the big picture of categorical logic. However, since there are many other introductions to categorical logic (a non-exhaustive list could include [**?**, **?**, **?**, **?**, **?**]), it seems appropriate to say a few words about what distinguishes this one. These words may not make very much sense to the beginner who doesn't yet know what we are talking about, but they may help to orient the expert, and as the beginner becomes more expert he or she can return to them later on.

Our perspective is very much that of the category theorist: our primary goal is to use type theory as a convenient syntax to prove things about categories, by presenting free structures in a particular way. (It can be tempting for the category theorist to want to generalize away from free structures to arbitrary ones, but this temptation should be resisted; see Remark 1.2.14.) In particular, this means that we are not interested in aspects of type theory such as computability, canonicity, proof search, cut-elimination, focusing, and so on *for their own sake*. However, at the same time we recognize their importance for type theory as a subject in its own right, which suggests that they should not be ignored by the category theorist. If nothing else, the category theorist will encounter these words when speaking to type theorists, and so it is advantageous to have at least a passing familiarity with them.

In fact, our perspective is that the esoteric-sounding notion of *cut admissibility* (and its various other incarnations such as *cut-elimination* or *admissibility of substitution*) essentially *defines* what we mean by a "type theory" (as opposed to some other syntax for free structures, such as string diagrams or the tautological one). Of course this is not literally true; a more careful statement would be that type theories with cut elimination are those that exhibit the most behavior most characteristic of type theories. (Jean-Yves Girard remarked that "a logic without cut-elimination is like a car without an engine.") A "type theory without cut elimination" can still yield explicit presentations of free structures, but will tend to lack some of the characteristic features of categorical logic.

So what is this mysterious cut-admissibility, from a categorical perspective? We saw a simple example of it for groups in §0.2. In general, cut admissibility says that the morphisms in a free categorical structure can be presented *without explicit reference to composition*. This is a bit of a cheat, because as we will see, in fact what we do is to build just enough "implicit" reference to composition into our rules to ensure that we no longer need to talk about composition explicitly. However, this does not make the process trivial, and it can still yield valuable results.

As a simple example of nontriviality, if an arrow is constructed by applying a universal property, then that property automatically determines some of the composites of that arrow. For instance, a pairing $\langle f, g\rangle : X \to A \times B$ must compose with the projections $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$ to give f and g respectively. Thus, these composites do not need to be "built in" by hand.

Another interesting fact about cut-elimination is that the composition it produces is automatically associative (and unital), despite the fact that we do not apparently put associativity in anywhere (even implicitly). Došen [**?**] uses this to "explain" or "justify" the definition of category (and other basic category-theoretic notions) in terms of cut-elimination. Of course, for our intended audience of category theorists it is cut-elimination, rather than associativity, that requires explanation and justification; but nevertheless the relationship is intriguing. There is undoubtedly a connection with the "something special" possessed by groups and categories but not by magmas or abelian groups.

Both of these facts are instances of an underlying general principle: by presenting a free categorical structure without explicit reference to composition, we are free to then *define* its composition as an operation on its already-existing morphisms, and we can choose this definition so as to ensure that various desirable properties hold automatically. This eliminates or reduces the need for quotienting by equivalence relations in the presentation of a free structure. Put differently, a type theory isolates a class of *canonical forms* for morphisms. In simple cases every morphism has exactly one canonical form, so that no equivalence relation on the canonical forms is needed. In more complicated situations we still need an equivalence relation, but the necessary equivalence relation is often simpler and/or more intuitive than that involved in more tautological presentations of free structures.

Another characteristic advantage of categorical logic is that it enables us to use "set-like" reasoning to prove things about arbitrary categories, by means of "term calculi" associated to its presentations of free structures. (This is what we exhibited several examples of in §0.1.) Such syntax is not actually a characteristic of *all* type theories, but of a large class of common ones that are sometimes known as "natural deduction" theories (although this usage of the term is much broader than its traditional denotation). Roughly speaking, natural deduction theories "build in composition" on the left side only, which from a categorical perspective suggests that they are talking about representable presheaves, i.e. describing a category by way of its Yoneda embedding. The characteristic "set-like" syntax of natural deduction theories then corresponds to the point of view that considers an arbitrary morphism $x : X \to A$ in a category to be a "generalized element" of A.

Despite the usefulness of terms, we will maintain and emphasize throughout the principle that terms should be just a convenient notation for derivation trees. This perspective has many advantages. For instance, it means that a (constructive) proof of cut-elimination *is already* a definition of substitution into terms; it is not necessary to separately define a notion of "substitution into terms" and then prove that this *separately defined* notion of substitution is admissible. It also deals quite nicely with the problems of α-equivalence and bound variable renaming: as an operation on derivations, substitution doesn't need to care about "whether a free variable is going to get captured"; the point is just that when we choose a term to represent the substituted derivation we have to accord with the general rules for how terms are assigned to derivations.

Most importantly, however, adhering to the "terms are derivations" principle greatly simplifies the proofs of the central "initiality theorems" (that the type theory really does present a free category with appropriate structure), since we can define a map out of the type theory *by induction on derivations* and deduce immediately that it is also defined on terms. If the "terms are derivations" principle is broken, then one generally ends up wanting to induct on derivations anyway, and then having to prove laboriously that the resulting "operation" on terms is independent of their derivations.

Informally, the "terms are derivations" principle means that *the meaning of a notation can be evaluated simply on the basis of the notation as written, without having to guess at the thought processes of the person who wrote it down.* That is, the meaning of "$2 + 3$" should not depend on whether we obtained it by substituting $x = 2$ into $x + 3$ or by substituting $y = 3$ into $2 + y$. This is obviously a desirable feature, and arguably even a necessary one if our "notation" is to be worthy of the name. Moreover, this "freedom from mind-reading" should hold *by definition* of the meaning of our notation: the meaning of $2+3$ should be defined on its own without reference to $x+3$ and $2+y$, with the fact that we can obtain it from the latter expressions by substitution being a later observation.

This principle demands in particular that substitution be an "admissible rule" rather than a primitive one (that is, an operation defined on terms/derivations, rather than one of the rules for producing them). For similar reasons, we present our type theories so as to ensure that as many structural rules as possible are admissible rather than primitive: not only cut/substitution, but also exchange, contraction, and weakening. The meaning of $x + y$ should not depend on which of the variables x and y happens to have been mentioned first in the course of a proof.

Many introductions to type theory are somewhat vague about exactly how these structural rules are to be imposed, especially for substructural theories such as linear logic with exchange only. However, when we try to use type theory to present a free symmetric monoidal category (as opposed to a free symmetric monoidal poset), we have to worry about the functoriality of the exchange rule, which technically requires being explicit about exactly how exchange works. If we make exchange admissible, then it is automatically functorial, just as making substitution admissible gives associativity for free; this considerably simplifies the theory. Having structural rules as primitive would also make the notation quite tedious if we continued to adhere to the principle that terms are just a notation for derivations.

In fact, it seems to me that much of the literature on categorical logic contains gaps or even errors relating to these points. It is very tempting to prove the initiality theorem by induction on derivations without realizing that by breaking the "terms are derivations" principle one thereby incurs an obligation to prove that the interpretation of a term is independent of its derivation. It is also very tempting to include too many primitive rules, perhaps based on the thought that if a rule is true anyway, it's simpler to assume it than to have to prove it. One way to break this habit is to think of primitive rules as the *operations* in an algebraic theory for which we are interested in the free algebras: clearly if there are too many operations, then the initial algebra will be too big.

Another unusual feature of our treatment is the emphasis on multicategories (of various generalized sorts, including also the still more general "polycategories" and their generalizations). We start with ordinary multicategories since they are simplest categorically, which forces us to consider "substructural" type theories such as linear logic at least briefly. But soon we move on to *cartesian multicategories*, which correspond to the more familiar kind of type theory with exchange, contraction, and weakening; these are a very natural structure, but are hard to find in the category-theoretic literature.

Although multicategories have been present in categorical logic from close to the beginnings of both (Lambek's original definition of multicategory [**?**] was motivated by logical considerations), they are rarely mentioned in introductions to the subject. One concrete advantage of using multicategories is a more direct correspondence between the type theory and the category theory: type theory distinguishes between a sequent $A, B \vdash C$ and a sequent $A \times B \vdash C$ (even though they are bijectively related), so it seems natural to work with a categorical structure that also distinguishes between morphisms $(A, B) \to C$ and $A \times B \to C$.

However, the correspondence and motivation goes deeper than that. We may ask *why* type theory distinguishes these two kinds of sequents? We will discuss this in more detail in §2.1, but the short answer is that "it makes cut-elimination work". More specifically, it enables us to formulate type theory in such a way that *each rule refers to at most one type former*, so that we can "commute these rules past each other" in the proof of cut-elimination. Moreover, including sequents such as $A, B \vdash C$ allows us to describe certain operations in a type-theoretic style that would not otherwise be possible, such as a monoidal tensor product. A type theorist speaks of this in terms of *deciding on the judgmental structure first* (including "structural rules") and then defining the connectives to "internalize" various aspects of that structure.

From a categorical point of view, the move to (generalized) multicategories has the feature that *it gives things universal properties.* For instance, the tensor product in a monoidal category has no universal property, but the tensor product in a multicategory does. In general, from a well-behaved 2-monad T we can define a notion of "T-multicategory" [**?**, **?**, **?**, **?**] in which T-algebra structure acquires a universal property (specifically, T is replaced by a lax- or colax-idempotent 2-monad with the same algebras). In type theoretic language, the move to T-multicategories corresponds to including the desired operations in the judgmental structure. The fact that the T-operations then have universal properties is what enables us to write down the usual sort of type-theoretic left/right or introduction/elimination rules for them.

Making this correspondence explicit is helpful for many reasons. Pedagogically, it can help the category theorist, who believes in universal properties, to understand why type theories are formulated the way they are. It also makes the "initiality theorems" more modular: first we model the judgmental structure with a multicategory, and then we add more type formers corresponding to objects with various universal properties. It can even be helpful from a purely type-theoretic perspective, suggesting more systematic ways to formulate cut admissibility theorems (see e.g. Theorem 2.3.5 and Lemma 2.7.2). Finally, it provides a guide for new applications of categorical logic: when seeking a categorical structure to model a given type theory, we should look for a kind of multicategory corresponding to its judgments; while when seeking an internal logic for a categorical structure, we should represent it using universal properties in some kind of multicategory, from which we can extract an appropriate judgmental structure.

These facts about cut-elimination and multicategories have surely been known in some form to experts for a long time, but I am not aware of a clear presentation of them for the beginner coming from a category-theoretic background. They are not strictly necessary if one wants simply to use type theory for internal reasoning about categories, and there are plenty of good introductions that take a geodesic route to that application. However, I believe that they yield a deeper understanding of the type/category correspondence; and they are especially valuable when it comes to designing type theories that correspond to new categorical structures (or vice versa).

0.4 Expectations of the reader

I will not assume that the reader has any acquaintance with type theory, or any interest in it apart from its uses for category theory. However, because one of my goals is to help the reader become familiar with the lingo and concerns of type theorists, I will sometimes include a little more detail than is strictly necessary for categorical applications. The reader should feel free to skip over these brief digressions.

It is possible that my zeal to explain all the aspects of type theory that can be puzzling to the category theorist has made this book a little more "encyclopedic" than would be ideal for a first introduction to the subject. I hope that it will still be of use to the newcomer; but if you haven't had any prior exposure to type theory or categorical logic, you may want to supplement it with other readings as well.

Another reason for supplementing is that I have intentionally minimized the space devoted to category theory. Of course, we are interested in type theory as a language for categories, and I have tried to include enough examples to illustrate its usefulness. But our focus will be on type theories and how they correspond to categories, not on the categorical structures themselves. For basic notions of category theory, see e.g. [**?**, **?**, **?**, **?**]. To learn more about the multicategories that appear in chapter 2, I recommend [**?**, **?**]. And for the indexed categories and "logical" types of categories appearing in chapters 4 to 6 one can consult [**?**, **?**, **?**, **?**].

I have endeavored to include a reasonable number of exercises of varying difficulty; these are placed at the end of most sections, and then compiled again for ease of reference at the end of each chapter. As always, doing exercises is important (perhaps even essential) for coming to understand a subject. But whether or not you plan to *do* the exercises, I highly recommend at least *reading* all the exercises as part of each section, and spending at least a few seconds thinking about how one might do them. A number of ideas are introduced in exercises and then come back again later in the text.

Chapter 1

Unary type theories

We begin our study of type theories and their categorical counterparts with a class of very simple cases that we will call *unary type theories*. (This terminology is not standard in the literature.) On the type-theoretic side the word "unary" indicates that there is only one type on each side of a sequent $A \vdash B$. On the categorical side it means, roughly, that we deal with categories rather than any kind of multicategory. In later chapters we will generalize away from this in various ways.

In some ways the unary case is fairly trivial, but for that very reason it serves as a good place to become familiar with basic notions of type theory and how they correspond to category theory.[1] Some of these notions and remarks may seem very pedantic in the unary case, but will become more important later on. I encourage the reader new to type theory to skim over any such parts of this chapter, and then return to it after some acquaintance with later chapters.

1.1 Posets

We start with the simplest sort of categories: those in which each hom-set has at most one element. These are well-known to be equivalent to *preordered sets*, where the existence of an arrow $A \to B$ is regarded as the assertion that $A \le B$. I will abusively call them *posets*, although traditionally posets (partially ordered sets) also satisfy the antisymmetry axiom (if $A \le B$ and $B \le A$ then $A = B$). From a category-theoretic perspective, antisymmetry means asking a category to be skeletal, which is both unnatural and pointless. Conveniently, posets also correspond to the simplest version of logic, namely *propositional* logic, as we will see in §2.7.

From a category-theoretic perspective, the question we are concerned with is the following. Suppose we have some objects in a poset, and some ordering

[1] I am indebted to Dan Licata [**?**] for the insight that unary type theories can be easier but still interesting.

relations between them. For instance, we might have

$$A \leq B \qquad A \leq C \qquad D \leq A \qquad B \leq E \qquad D \leq C$$

Now we ask, given two of these objects — say, D and E — is it necessarily the case that $D \leq E$? In other words, is it the case in *any* poset containing objects A, B, C, D, E satisfying the given relations that $D \leq E$? In this example, the answer is yes, because we have $D \leq A$ and $A \leq B$ and $B \leq E$, so by transitivity $D \leq E$. More generally, we would like a method to answer all possible questions of this sort.

There is an elegant categorical way to do this based on the notion of *free structure* (analogously to the situation for free groups we considered in §0.2). Namely, consider the category **Poset** of posets, and also the category **RelGr** of *relational graphs*, by which I mean sets equipped with an arbitrary binary relation. There is a forgetful functor $U : \mathbf{Poset} \to \mathbf{RelGr}$, which has a left adjoint $\mathfrak{F}_{\mathbf{Poset}}$.

Now, the abstract information about "five objects A, B, C, D, E satisfying five given relations" can be regarded as an object $\mathcal{G}$ of **RelGr**, and to give five such objects satisfying those relations in a poset $\mathcal{P}$ is to give a map $\mathcal{G} \to U\mathcal{P}$ in **RelGr**. By the adjunction, therefore, this is equivalent to giving a map $\mathfrak{F}_{\mathbf{Poset}}\mathcal{G} \to \mathcal{P}$ in **Poset**. Therefore, a given inequality such as $A \leq E$ will hold in *all* posets if and only if it holds in the *particular, universal* poset $\mathfrak{F}_{\mathbf{Poset}}\mathcal{G}$ freely generated by the assumed data.

Thus, to answer all such questions at once, it suffices to give a concrete presentation of the free poset $\mathfrak{F}_{\mathbf{Poset}}\mathcal{G}$ generated by a relational graph $\mathcal{G}$. In this simple case, it is easy to give an explicit description of $\mathfrak{F}_{\mathbf{Poset}}$: it is the reflexive-transitive closure. But since soon we will be trying to generalize vastly, we want instead a general method to describe free objects. From our current perspective, this is the role of type theory.

As noted in §0.1, when we move into type theory we use the symbol $\vdash$ instead of $\to$ or $\leq$. Type theory is concerned with *(hypothetical) judgments*, which (roughly speaking) are syntactic gizmos of the form "$\Gamma \vdash \Delta$", where Γ and Δ are syntactic gadgets whose specific nature is determined by the specific type theory under consideration (and, thus, by the particular kind of categories we care about). We call Γ the *antecedent* or *context*, and Δ the *consequent* or *co-context*. In our simple case of posets, the judgments are simply

$$A \vdash B$$

where A and B are objects of our (putative) poset; such a judgment represents the relation $A \leq B$. In general, the categorical view is that a hypothetical judgment represents a sort of *morphism* (or, as we will see later, a sort of *object*) in some sort of categorical structure.

In addition to a class of judgments, a type theory consists of a collection of *rules* by which we can operate on such judgments. Each rule can be thought of as a partial n-ary operation on the set of possible judgments for some n (usually a finite natural number), taking in n judgments (its *premises*) that satisfy some compatibility conditions and producing an output judgment (its *conclusion*). We generally write a rule in the form

$$\frac{\mathcal{J}_1 \qquad \mathcal{J}_2 \qquad \cdots \qquad \mathcal{J}_n}{\mathcal{J}}$$

with the premises above the line and the conclusion below. A rule with $n = 0$ is sometimes called an *axiom*. The categorical view is that we have a given "starting" set of judgments representing some objects and putative morphisms in the "underlying data" of a categorical structure, and the closure of this set under application of the rules yields the objects and morphisms in the *free* structure it generates.

We will attempt to make all of this precise in Appendix A, which the reader is free to consult now. However, it is probably more illuminating at the moment to bring it back down to earth in our very simple example. Since the properties distinguishing a poset are reflexivity and transitivity, we have two rules:

$$\frac{}{A \vdash A} \qquad\qquad \frac{A \vdash B \qquad B \vdash C}{A \vdash C}$$

in which A, B, C represent arbitrary objects. In other words, the first says that for any object A we have a 0-ary rule whose conclusion is $A \vdash A$, while the second says that for any objects A, B, C we have a 2-ary rule whose premises are $A \vdash B$ and $B \vdash C$ (that is, any two judgments of which the consequent of the first is the antecedent of the second) and whose conclusion is $A \vdash C$. We will refer to the pair of these two rules as the **free type theory of posets**.

Hopefully it makes sense that we can construct the reflexive-transitive closure of a relational graph by expressing its relations in this funny syntax and then closing up under these two rules, since they are exactly reflexivity and transitivity. Categorically, of course, that means identities and composition. In type theory the composition/transitivity rule is often called **cut**, and plays a unique role, as we will see later.

In the example we started from,

$$A \le B \qquad A \le C \qquad D \le A \qquad B \le E \qquad D \le C$$

we have the two instances of the transitivity rule

$$\frac{D \vdash A \qquad A \vdash B}{D \vdash B} \qquad\qquad \frac{D \vdash B \qquad B \vdash E}{D \vdash E}$$

allowing us to conclude $D \vdash E$. When applying multiple rules in sequence to reach a conclusion, it is customary to write them in a "tree" structure like so:

$$\frac{\dfrac{D \vdash A \qquad A \vdash B}{D \vdash B} \qquad B \vdash E}{D \vdash E}$$

Such a tree is called a *derivation*. The way to typeset rules and derivations in LaTeX is with the `mathpartir` package; the above diagram was produced with

```
\inferrule*{
  \inferrule*{D\types A \\ A\types B}{D\types B} \\
  B\types E
}{
  D\types E
}
```

Note that `mathpartir` has only recently made it into standard distributions of LaTeX, so if you have an older system you may need to download it manually.

Formally speaking, what we have observed is the following *initiality theorem*.

Theorem 1.1.1. *For any relational graph $\mathcal{G}$, the free poset $\mathfrak{F}_{\mathbf{Poset}}\mathcal{G}$ that it generates is has the same objects and its morphisms are the judgments that are derivable from $\mathcal{G}$ in free type theory of posets.*

Proof. In the preceding discussion we assumed it as known that the free poset on a relational graph is its reflexive-transitive closure, which makes this theorem more or less obvious. However, it is worth also presenting an explicit proof that does not assume this, since same pattern of proof will reappear many times for more complicated type theories where we don't know the answer in advance.

Thus, let us define $\mathfrak{F}_{\mathbf{Poset}}\mathcal{G}$ as stated in the theorem. The reflexivity and transitivity rules imply that $\mathfrak{F}_{\mathbf{Poset}}\mathcal{G}$ is in fact a poset. Now suppose $\mathcal{A}$ is any other poset and $P : \mathcal{G} \to \mathcal{A}$ is a map of relational graphs. The objects of $\mathfrak{F}_{\mathbf{Poset}}\mathcal{G}$ are the same as those of $\mathcal{G}$, so P extends uniquely to a map on underlying sets $\mathfrak{F}_{\mathbf{Poset}}\mathcal{G} \to \mathcal{A}$. Thus it suffices to show that this map is order-preserving, i.e. that if $A \vdash B$ is derivable from $\mathcal{G}$ in the free type theory of posets, then $P(A) \le P(B)$.

For this purpose we *induct on the derivation of* $A \vdash B$. There are multiple ways to phrase such an induction. One is to define the *height* of a derivation to be the number of rules appearing in it, and then induct on the height of the derivation of $A \vdash B$.

(a) If there are no rules at all, then $A \vdash B$ must come from a relation $A \le B$ in $\mathcal{G}$; hence $P(A) \le P(B)$ since P is a map of relational graphs.

(b) If there are $n > 0$ rules, then consider the last rule.

   (i) If it is the identity rule $A \vdash A$, then $P(A) \le P(A)$ in $\mathcal{A}$ since $\mathcal{A}$ is a poset and hence reflexive.

   (ii) Finally, if it is the transitivity rule, then each of its premises $A \vdash B$ and $B \vdash C$ must have a derivation with strictly smaller height, so by the (strong) inductive hypothesis we have $P(A) \le P(B)$ and $P(B) \le P(C)$. Since $\mathcal{A}$ is a poset and hence transitive, we have $P(A) \le P(C)$. □

A different way to phrase such an induction, which is more flexible and more type-theoretic in character, uses what is called *structural induction*. This means that rather than introduce the auxiliary notion of "height" of a derivation, we apply a general principle that *to prove that a property P holds of all derivations,* *it suffices to show for each rule that if P holds of the premises then it holds of the conclusion.* We can also define operations on derivations by *structural recursion*, meaning that it suffices to define what happens to the conclusion of each rule assuming that we have already defined what happens to the premises. Structural induction and recursion can be justified formally by set-theoretic arguments — see Appendix A for some general statements. However, intuitively they are implicit in what is meant by saying that "derivations are what we obtain by applying rules one by one," just as ordinary mathematical induction is implicit in saying that "the natural numbers are what we obtain by starting with zero and constructing successors one by one", and constructive type-theoretic foundations for mathematics often take them as axiomatic. From now on we will use structural induction and recursion on derivations in all type theories without further comment.

However, it is proved, Theorem 1.1.1 enables us to reach conclusions about arbitrary posets by deriving judgments in type theory. In our present trivial case this is not very useful, but as we will see it becomes more useful for more complicated structures.

Another way to express the initiality theorem is to incorporate $\mathcal{G}$ into the rules. Given a relational graph $\mathcal{G}$, we define the **type theory of posets under** $\mathcal{G}$ to be the free type theory of posets together with a 0-ary rule

$$\frac{}{A \vdash B}$$

for any relation $A \le B$ in $\mathcal{G}$. Now a derivation can be written without any "leaves" at the top, such as

$$\dfrac{\dfrac{\dfrac{}{D \vdash A} \quad \dfrac{}{A \vdash B}}{D \vdash B} \qquad \dfrac{}{B \vdash E}}{D \vdash E}$$

Clearly this produces the same judgments; thus the initiality theorem can also be expressed as follows.

Theorem 1.1.2. *For any relational graph $\mathcal{G}$, the free poset $\mathfrak{F}_{\mathbf{Poset}}\mathcal{G}$ that it generates has the same objects and its morphisms are the derivable judgments in the type theory of posets under $\mathcal{G}$.* □

We can extract from this our first general statement about categorical logic: it is *a syntax for generating free categorical structures using derivations from rules.* The reader may be forgiven at this time for wondering what the point is; but bear with us and things will get less trivial.

1.2 Categories

Let's now generalize from posets to categories. The relevant adjunction is now between categories **Cat** and *directed graphs* **Gr**; the latter are sets $\mathcal{G}$ of "vertices" equipped with a set $\mathcal{G}(A, B)$ of "edges" for each $A, B \in \mathcal{G}$. Thus, we hope to generate the free category $\mathfrak{F}_{\mathbf{Cat}}\mathcal{G}$ on a directed graph $\mathcal{G}$ type-theoretically.

Our judgments $A \vdash B$ will still represent morphisms from A to B, but now of course there can be more than one such morphism. Thus, to specify a particular morphism, we need more information than the simple *derivability* of a judgment $A \vdash B$. Naïvely, the first thing we might try is to identify this extra information with the *derivation* of such a judgment, i.e. with the tree of rules that were applied to reach it. This makes the most sense if we take the approach of Theorem 1.1.2 rather than Theorem 1.1.1, so that distinct edges $f, g \in \mathcal{G}(A, B)$ can be regarded as distinct *rules*

$$\frac{}{A \vdash B}\, f \qquad\qquad \frac{}{A \vdash B}\, g$$

Thus, for instance, if we have also $h \in \mathcal{G}(B, C)$, the distinct composites $h \circ g$ and $h \circ f$ will be represented by the distinct derivations

$$\frac{\dfrac{}{B \vdash C}\, h \qquad \dfrac{}{A \vdash B}\, g}{A \vdash C}\, \circ \qquad\qquad \frac{\dfrac{}{B \vdash C}\, h \qquad \dfrac{}{A \vdash B}\, f}{A \vdash C}\, \circ$$

Note that when we have distinct rules with the same premises and conclusion, we have to label them so that we can tell which is being applied. For consistency, we also begin labeling the composition and identity rules, with $\circ$ and id respectively.

Of course, this naïve approach founders on the fact that composition in a category is supposed to be associative and unital, since the two composites $h \circ (g \circ f)$ and $(h \circ g) \circ f$, which ought to be equal, nevertheless correspond to distinct derivations:

$$\frac{\dfrac{}{C \vdash D}\, h \qquad \dfrac{\dfrac{}{B \vdash C}\, g \qquad \dfrac{}{A \vdash B}\, f}{A \vdash C}\, \circ}{A \vdash D}\, \circ$$
$$\frac{\dfrac{\dfrac{}{C \vdash D}\, h \qquad \dfrac{}{B \vdash C}\, g}{B \vdash D}\, \circ \qquad \dfrac{}{A \vdash B}\, f}{A \vdash D}\, \circ \tag{1.2.1}$$

Thus, with this type theory we don't get the free category on $\mathcal{G}$, but rather some free category-like structure that lacks associativity and unitality. There are two ways to deal with this problem; we consider them in turn.

1.2.1 Primitive cuts

The first solution is to simply quotient by an equivalence relation. Our equivalence relation will have to identify the two derivations in (1.2.1), and also the similar pairs for identities:

$$\dfrac{\dfrac{}{A \vdash A}\,\mathsf{id} \qquad A \vdash B}{A \vdash B}\,\circ \qquad \equiv \qquad A \vdash B$$

$$\dfrac{A \vdash B \qquad \dfrac{}{B \vdash B}\,\mathsf{id}}{A \vdash B}\,\circ \qquad \equiv \qquad A \vdash B$$

Our equivalence relation must also be a "congruence for the tree-construction of derivations", meaning that these identifications can be made anywhere in the middle of a long derivation, such as:

$$\dfrac{\begin{matrix}\mathscr{D}_1 \\ \vdots\end{matrix} \qquad \dfrac{\dfrac{}{A \vdash A}\,\mathsf{id} \qquad \dfrac{\begin{matrix}\mathscr{D}_2 \\ \vdots\end{matrix}}{A \vdash B}}{A \vdash B}\,\circ}{\begin{matrix}\vdots \\ \mathscr{D}_3\end{matrix}} \qquad \equiv \qquad \dfrac{\begin{matrix}\mathscr{D}_1 \\ \vdots\end{matrix} \qquad \dfrac{\begin{matrix}\mathscr{D}_2 \\ \vdots\end{matrix}}{A \vdash B}}{\begin{matrix}\vdots \\ \mathscr{D}_3\end{matrix}}$$

We will also have to close it up under reflexivity, symmetry, and transitivity to make an equivalence relation.

Of course, it quickly becomes tedious to draw such derivations, so it is convenient to adopt a more succinct syntax for them. We begin by labeling each judgment with a one-dimensional syntactic representation of its derivation tree, such as:

$$\dfrac{\dfrac{}{g : (B \vdash C)}\,g \qquad \dfrac{\dfrac{}{\mathsf{id}_B : (B \vdash B)}\,\mathsf{id} \qquad \dfrac{}{f : (A \vdash B)}\,f}{(\mathsf{id}_B \circ_B f) : (A \vdash B)}\,\circ}{(g \circ_B (\mathsf{id}_B \circ_B f)) : (A \vdash C)}\,\circ$$

These labels are called *terms*. Of course, in this case they are none other than the usual notation for composition and identities. Formally, this means the rules are now:

$$\dfrac{f \in \mathcal{G}(A, B)}{f : (A \vdash B)} \qquad \dfrac{A \in \mathcal{G}}{\mathsf{id}_A : (A \vdash A)} \qquad \dfrac{\phi : (A \vdash B) \qquad \psi : (B \vdash C)}{\psi \circ_B \phi : (A \vdash C)}$$

Here ϕ, ψ denote arbitrary terms, and if they contain $\circ$'s themselves then we put parentheses around them, as in the example above. Now the generators of our equivalence relation look even more familiar:

$$\begin{aligned} \chi \circ_C (\psi \circ_B \phi) &\equiv (\chi \circ_C \psi) \circ_B \phi \\ \phi \circ_A \mathsf{id}_A &\equiv \phi \\ \mathsf{id}_B \circ_B \phi &\equiv \phi \end{aligned}$$

Again ϕ, ψ, χ denote arbitrary terms, corresponding to the fact that arbitrary derivations can appear at the top of our identified trees; and similarly these identifications can also happen anywhere inside another term, so that for instance

$$k \circ_C (h \circ_B (g \circ_A f)) \equiv k \circ_C ((h \circ_B g) \circ_A f).$$

Of course, we only impose these relations when they make sense. We can describe the conditions under which this happens using rules for a secondary judgment $\phi \equiv \psi : (A \vdash B)$. The rules for our generating equalities are

$$\frac{\phi : (A \vdash B) \qquad \psi : (B \vdash C) \qquad \chi : (C \vdash D)}{(\chi \circ_C (\psi \circ_B \phi) \equiv (\chi \circ_C \psi) \circ_B \phi) : (A \vdash D)}$$

$$\frac{\phi : (A \vdash B)}{(\phi \circ \mathsf{id}_A \equiv \phi) : (A \vdash B)} \qquad \frac{\phi : (A \vdash B)}{(\mathsf{id}_B \circ \phi \equiv \phi) : (A \vdash B)}$$

and we must also have rules ensuring that we have an equivalence relation and a congruence:

$$\frac{\phi : (A \vdash B)}{(\phi \equiv \phi) : (A \vdash B)} \qquad \frac{(\phi \equiv \psi) : (A \vdash B)}{(\psi \equiv \phi) : (A \vdash B)}$$

$$\frac{(\phi \equiv \psi) : (A \vdash B) \qquad (\psi \equiv \chi) : (A \vdash B)}{(\phi \equiv \chi) : (A \vdash B)}$$

$$\frac{(\phi_1 \equiv \psi_1) : (A \vdash B) \qquad (\phi_2 \equiv \psi_2) : (B \vdash C)}{(\phi_2 \circ_B \phi_1 \equiv \psi_2 \circ_B \psi_1) : (A \vdash C)}$$

The last of these is sufficient, in our simple case, to ensure we have a congruence; in general we would have to have one such equality rule for each basic rule of the theory (except for those with no premises, like id).

Many of our type theories will involve such an equality judgment, for which we always use the notation $\equiv$, and the need for the equivalence relation and congruence rules is always the same. Thus, we generally decline to mention them, stating only the "interesting" generating equalities for the theory. A general framework for such equality judgments is described in §A.3.

In our case, when the rules for $\circ$ and id are augmented by these rules for $\equiv$, and we also add axioms for the edges of a given directed graph $\mathcal{G}$, we call the result the **cut-ful type theory for categories under** $\mathcal{G}$. It may seem obvious that this produces the free category on $\mathcal{G}$, but again we write it out carefully to help ourselves get used to the patterns. In particular, we want to emphasize the role played by the following lemma:

Lemma 1.2.2. *If $\phi : (A \vdash B)$ is derivable in the cut-ful type theory for categories under $\mathcal{G}$, then it has a unique derivation.*

Proof. The point is that the terms produced by all the rules have disjoint forms. If ϕ is of the form "f" for some $f \in \mathcal{G}(A, B)$, then it can only be derived by the first rule applied to f. If it is of the form "id_A", then it can only be derived by the identity rule applied to A. Finally, if it is of the form "$\psi \circ_C \phi$" it can only be derived by the composition rule applied to $\phi : (A \vdash C)$ and $\psi : (C \vdash B)$, and by induction the latter judgments also have unique derivations. □

In other words, the terms (before we impose the relation $\equiv$ on them) really are simply one-dimensional representations of derivations, as we intended. Not everything that "looks like a term" represents a derivation, but if it does, it represents a unique one. (We have not precisely defined exactly what "looks like a term", but it should make intuitive sense; a formal definition is given in Appendix A.) It is easy to see that conversely every derivation is represented by a unique term, since the above rules for annotating derivations by terms are deterministic.

The above simple inductive proof of Lemma 1.2.2 depends in particular on the presence of the subscript on the symbol $\circ$. Similar annotations will reappear in many subsequent theories. In the present case we could omit these annotations and still reconstruct a unique derivation, because we know the domain and codomain of all the generating morphisms in $\mathcal{G}$. However, this would require a more "global" analysis of the term; whereas a clean inductive proof such as the above has the advantage that it can be regarded as a *recursively defined algorithm*.

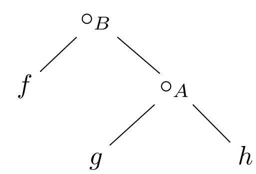
We call this algorithm *type-checking*: it starts with a putative sequent with term $\phi : (A \vdash B)$ and, by following the algorithm of Lemma 1.2.2 until it terminates or encounters a contradiction, either produces a derivation of that sequent or decides that it has no such derivation. This algorithm can be programmed into a computer, and arguably represents reasonably faithfully what human mathematicians do when reading syntax. With that said, when writing for a human reader (and even an electronic reader whose programmer has been clever enough) it is often possible to leave off annotations of this sort without fear of ambiguity, and we will frequently do so.

Not all type theories have the property that terms uniquely determine their derivations by a direct inductive algorithm; but those that don't tend to be much more complicated to analyze and prove the initiality theorem for. We will call this property **terms are derivations** or **type-checking is possible**, and we will always attempt to construct our type theories so that it holds.

Remark 1.2.3. Technically, there is either more or less happening here than may appear (depending on your point of view). A term as we write it on the page is really just a string of symbols, whereas in the proof of Lemma 1.2.2 we have assumed that a term such as "$f \circ_B (g \circ_A h)$" can uniquely be read as $\circ_B$ applied to "f" and "$g \circ_A h$". This simple string of symbols could technically be regarded as $\circ_A$ applied to "$f \circ_B (g$" and "$h)$", but of course that would make no sense because those are not meaningful terms in their own right (in particular, they contain unbalanced parentheses).

Thus, something *more* must be happening, and that something else is called *parsing* a term. Human mathematicians do it instinctively without thinking; electronic mathematicians have to be programmed to do it. In either case, the result of parsing a string of symbols is an "internal" representation (a mental idea for humans, a data structure for computers) that generally has the form of a tree, indicating the "outermost" operation as the root with its operands as branches, and so on, for instance:

$$f \circ_B (g \circ_A h) \qquad \rightsquigarrow$$



Of course, this "internal" tree representation of a term is nothing but the corresponding derivation flipped upside-down. So in that sense Lemma 1.2.2 is actually saying *less* than one might think: the derivation tree is actually being constructed by the silent step of parsing, while the type-checking algorithm consists only of *labeling* the nodes of this tree by rules in a consistent manner. We will not say much more about parsing, however; we trust the human reader to do it on their own, and we trust programmers to have good algorithms for it.

Now we can prove the initiality theorem.

Theorem 1.2.4. *The free category on a directed graph $\mathcal{G}$ has the same objects as $\mathcal{G}$, and its morphisms $A \to B$ are the derivations of $A \vdash B$ (or equivalently, the terms ϕ such that $\phi : (A \vdash B)$ is derivable) in the cut-ful type theory for categories under $\mathcal{G}$, modulo the equivalence relation $\phi \equiv \psi : (A \vdash B)$.*

Proof. Let $\mathfrak{F}_{\mathbf{Cat}}\mathcal{G}$ be defined as described in the theorem; the identity and composition rules give it the structure necessary to be a category, and the transitivity and unitality relations make it a category.

Now suppose $\mathcal{A}$ is any category and $\omega : \mathcal{G} \to \mathcal{A}$ is a map of directed graphs. Then ω extends uniquely to the objects of $\mathfrak{F}_{\mathbf{Cat}}\mathcal{G}$, since they are the same as those of $\mathcal{G}$. But unlike the case of posets, we have to define our desired extension $\overline{\omega}$ on the morphisms of $\mathfrak{F}_{\mathbf{Cat}}\mathcal{G}$ as well.

If $\phi : (A \vdash B)$ is derivable, then by Lemma 1.2.2 it has a unique derivation; thus we can define $\overline{\omega}(\phi)$ by recursion on the derivation of ϕ. Of course, if the derivation of ϕ ends with $f \in \mathcal{G}(A, B)$, then we define $\overline{\omega}(\phi) = \omega(f)$; if it ends with id_A we define $\overline{\omega}(\phi) = \mathsf{id}_{P(A)}$; and if it ends with $\psi \circ_C \chi$ we define $\overline{\omega}(\phi) = \overline{\omega}(\psi) \circ \omega(\chi)$.

We also have to show that this definition respects the equivalence relation $\equiv$. This is clear since $\mathcal{A}$ is a category; formally it would be another induction on the derivations of $\equiv$ judgments.

Finally, we have to show that this $\overline{\omega} : \mathfrak{F}_{\mathbf{Cat}}\mathcal{G} \to \mathcal{A}$ is a functor. This follows by definition of the category structure of $\mathfrak{F}_{\mathbf{Cat}}\mathcal{G}$ and the action of $\overline{\omega}$ on its arrows. $\square$

Of course, once again very little seems to be happening; we are just using a complicated funny syntax to build a free algebraic structure. (In fact, what we are doing now is analogous to the "tautological construction" of free groups from

§0.2.) Therefore, it is the second way to deal with the problem of associativity that is more interesting.

1.2.2 Cut admissibility

In this case what we do is *remove the composition rule* $\circ$ *entirely*; instead we "build (post)composition into the axioms". That is, the only rule independent of $\mathcal{G}$ is identities:

$$\frac{}{A \vdash A}\ \mathsf{id}$$

while for every edge $f \in \mathcal{G}(A, B)$ we take the following rule:

$$\frac{X \vdash A}{X \vdash B}\ f$$

for any X. Informally, one might say that we represent f by its "image under the Yoneda embedding".

Note that we have made a choice to build in *postcomposition*; we could also have chosen to build in precomposition. In the current context, either choice would work just as well; but later on we will see that there were reasons to choose postcomposition here. We will call this the **cut-free type theory for categories under $\mathcal{G}$**.

In this theory, if we have $f \in \mathcal{G}(A, B)$, $g \in \mathcal{G}(B, C)$, and $h \in \mathcal{G}(C, D)$ there is *only one way* to derive $A \vdash D$:

$$\cfrac{\cfrac{\cfrac{\cfrac{}{A \vdash A}\ \mathsf{id}}{A \vdash B}\ f}{A \vdash C}\ g}{A \vdash D}\ h$$

Thus, we no longer have to worry about distinguishing between $h \circ (g \circ f)$ and $(h \circ g) \circ f$. Of course, we have a new problem: if we are trying to build a category, then we *do* need to be able to compose arrows! So we need the following theorem:

Theorem 1.2.5. *If we have derivations of $A \vdash B$ and $B \vdash C$ in the cut-free type theory for categories under $\mathcal{G}$, then we can construct a derivation of $A \vdash C$.*

Proof. We induct on the derivation of $B \vdash C$. If it ends with id, then it must be that $B = C$; so our given derivation of $A \vdash B$ is also a derivation of $A \vdash C$. Otherwise, we must have some $f \in \mathcal{G}(D, C)$ and our derivation of $B \vdash C$ ends like this:

$$\cfrac{\cfrac{\begin{matrix}\mathscr{D}\\ \vdots\end{matrix}}{B \vdash D}}{B \vdash C}\ f$$

In particular, it contains a derivation $\mathscr{D}$ of $B \vdash D$. Thus, by the inductive hypothesis we have a derivation, say $\mathscr{D}'$, of $A \vdash D$. Now we can simply follow this with the rule for f:

$$\dfrac{\dfrac{\begin{array}{c}\mathscr{D}'\\ \vdots\end{array}}{A \vdash D}}{A \vdash C}\, f$$

□

In type-theoretic lingo, Theorem 1.2.5 says that **the cut rule is admissible** in the cut-free type theory for categories under $\mathcal{G}$. In other words, although the cut/composition rule

$$\frac{A \vdash B \qquad B \vdash C}{A \vdash C}\,\circ$$

is not *part of the type theory* as defined, it is nevertheless true that whenever we have derivations of the premises of this rule, we can construct a derivation of its conclusion.

Remark 1.2.6. This is what it means in general for a rule to be **admissible**: it is not part of the theory as defined (that is, it is not one of the **primitive rules**), but nevertheless if it were added to the theory it would not change the set of derivable sequents.[2] In between primitive and admissible rules there are **derivable rules**: those that can be expanded out directly into a fragment of a derivation in terms of the primitive rules. For instance, if we have $f \in \mathcal{G}(A, B)$ and $g \in \mathcal{G}(B, C)$, then the left-hand rule below is derivable:

$$\frac{X \vdash A}{X \vdash C} \qquad\qquad \dfrac{\dfrac{X \vdash A}{X \vdash B}\, f}{X \vdash C}\, g$$

because we can expand it out into the right-hand derivation in terms of the primitive rules. Any derivable rule is admissible: if we have a derivation of $X \vdash A$ we can follow it with the f and g rules to obtain a derivation of $X \vdash C$. Note the difference with the proof of cut-admissibility: here we do not need to modify the given derivation, we only apply further primitive rules to its conclusion. (The reader should beware, however, that the words "derivable" and "admissible" are frequently misused.) We will return to this distinction in Remark 1.2.14.

Closely related to cut-admissibility is **cut-elimination**, which in our theory takes the following form.

Theorem 1.2.7. *Consider the cut-free type theory for categories under $\mathcal{G}$ with the cut rule* added *as primitive. If $A \vdash B$ has a derivation in this new theory, then it also has a derivation in the cut-free theory.*

[2]This terminology comes from the posetal case, where "derivability" is the important concept. If we care about distinguishing between different derivations of the same sequent (to represent multiple parallel morphsims in a category), then an admissibility theorem is better regarded as an *operation* on derivations. We will return to this later on.

Proof. We induct on the derivation of $A \vdash B$. If it ends with id, it is already cut-free. If it ends like this for some $f \in \mathcal{G}(C, B)$:

$$\begin{array}{c} \mathscr{D} \\ \vdots \\ \hline A \vdash C \\ \hline A \vdash B \end{array}\; f$$

then by induction, $A \vdash C$ has a cut-free derivation, to which we can apply the f rule to obtain a cut-free derivation of $A \vdash B$. Finally, if it ends with the cut rule:

$$\begin{array}{c} \begin{array}{c} \mathscr{D}_1 \\ \vdots \\ \hline A \vdash C \end{array} \qquad \begin{array}{c} \mathscr{D}_2 \\ \vdots \\ \hline C \vdash B \end{array} \\ \hline A \vdash B \end{array}\; \textsc{cut}$$

then by induction $A \vdash C$ and $C \vdash B$ have cut-free derivations, and thus by Theorem 1.2.5 so does $A \vdash B$. □

Note that cut-elimination is a fairly straightforward consequence of cut-admissibility: the latter allows us to eliminate each cut one by one. This will nearly always be true for our type theories, so we will usually just prove cut admissibility and rarely remark on the cut-elimination theorem that follows from it. On the other hand, cut admissibility is a special case of cut-elimination, and sometimes people prove cut-elimination directly without explicitly using cut-elimination as a lemma. Under this approach, the inductive step in cut-admissibility is viewed instead as a step of "pushing cuts upwards" through a derivation: given a derivation as on the left below in the theory with cut, we transform it into the derivation on the right in which the cut is higher up.

$$\begin{array}{c} \begin{array}{c} \mathscr{D}_1 \\ \vdots \\ \hline A \vdash B \end{array} \qquad \begin{array}{c} \mathscr{D}_2 \\ \vdots \\ \hline B \vdash C \\ \hline B \vdash D \end{array}\, f \\ \hline A \vdash D \end{array}\; \textsc{cut} \quad \rightsquigarrow \quad \begin{array}{c} \begin{array}{c} \begin{array}{c} \mathscr{D}_1 \\ \vdots \\ \hline A \vdash B \end{array} \qquad \begin{array}{c} \mathscr{D}_2 \\ \vdots \\ \hline B \vdash C \end{array} \\ \hline A \vdash C \end{array}\; \textsc{cut} \\ \hline A \vdash D \end{array}\; f$$

Because our derivation trees are finite (or, more generally, well-founded) this process must eventually terminate with all the cuts eliminated.

A more category-theoretic way to say what is going on is that the morphisms in the free category on a directed graph $\mathcal{G}$ have an explicit description as *finite strings of composable edges* in $\mathcal{G}$. (This is analogous to the description of free groups using reduced words in §0.2.) We have just given an inductive definition of "finite string of composable edges": there is a finite string (of length 0) from A to A; and if we have such a string from X to A and an edge $f \in \mathcal{G}(A, B)$, we can construct a string from X to B.

We could prove the initiality theorem by appealing to this known fact about free categories, but as before, we prefer to give a more explicit proof to illustrate the patterns of type theory. For this purpose, it is convenient to first introduce terms, as we did in the previous section for the cut-ful theory. We can do this with terms directly constructed so that their parse tree will mirror the derivation tree, for instance writing the rules as

$$\dfrac{}{\mathsf{id}_A : (A \vdash A)}\ \mathsf{id} \qquad\qquad \dfrac{\phi : (X \vdash A)}{f\circ(\phi) : (X \vdash B)}\ f$$

Then a term derivation and corresponding parse tree would look like

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{id}_A : (A\vdash A)}\,\mathsf{id}}{f\circ(\mathsf{id}_A) : (A\vdash B)}\,f}{g\circ(f\circ(\mathsf{id}_A)) : (A\vdash C)}\,g}{h\circ(g\circ(f\circ(\mathsf{id}_A))) : (A\vdash D)}\,h \qquad\rightsquigarrow\qquad \begin{array}{c} h\circ \\ | \\ g\circ \\ | \\ f\circ \\ | \\ \mathsf{id}_A \end{array}$$

However, now there is another option available to us, which begins to show more of the characteristic behavior of type-theoretic terms. Rather than describing the entire judgment $A \vdash B$ with a term, the way we did for the cut-ful theory, we assign a *formal variable* such as x to the domain A, and then an expression containing x to the codomain B. For the theory of plain categories that we are working with here, the only possible expressions are repeated applications of function symbols to the variable, such as $h(g(f(x)))$. We write this as

$$x : A \vdash h(g(f(x))) : B$$

The identity and generator rules can now be written as

$$\dfrac{}{x : A \vdash x : A}\ \mathsf{id} \qquad\qquad \dfrac{x : X \vdash M : A \qquad f \in \mathcal{G}(A, B)}{x : X \vdash f(M) : B}\ f$$

Here M denotes an arbitrary term, which will of course involve the variable x. Thus, for instance, the composite of h, g, and f would be written like so:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : A \vdash x : A}\,\mathsf{id}}{x : A \vdash f(x) : B}\,f}{x : A \vdash g(f(x)) : C}\,g}{x : A \vdash h(g(f(x))) : D}\,h$$

Of course, the term $h(g(f(x)))$ has essentially the same parse tree as the term $h\circ(g\circ(f\circ(\mathsf{id}_A)))$ shown above, so it can clearly represent the same derivation. The main difference is that instead of id_A we have the variable x representing the identity rule.

This is our first encounter with how type theory permits a "set-like" syntax when reasoning about arbitrary categorical structures. It is also one reason why we chose to build in postcomposition rather than precomposition. If we used precomposition instead, then the analogous syntax would be backwards: we would have to represent $f : A \to B$ as $f(u) : A \vdash u : B$ rather than $x : A \vdash f(x) : B$. At a formal level, there would be little difference, but it feels much more familiar to apply functions to variables than to co-apply functions to co-variables. (We can still dualize at the level of the categorical models; we already mentioned in §0.1 that we could apply the type theory of categories with finite products to the opposite of the category of commutative rings.)

Now we observe that terms are still derivations in this theory.

Lemma 1.2.8. *If $x : X \vdash M : B$ is derivable in the cut-free type theory for categories under $\mathcal{G}$, then it has a unique derivation.*

Proof. If M is the variable x, then the only possible derivation is id. And if $M = f(N)$, where $f \in \mathcal{G}(A, B)$, then it can only be obtained from the generator rule for f applied to $x : X \vdash N : A$. □

Note that the terms in this theory are simpler than those in the cut-ful theory in that we don't need the type subscripts on the composition operation $\circ_A$. This is because each rule composes with only one generator f, and each such generator "knows" its domain, so the premise of the rule is determined by the conclusion.

Another difference between the two theories that instead of attaching a term to the entire derivation such as $(f \circ g) : (A \vdash C)$, we now attach a variable to the antecedent and a more complex term to the consequent. Really it is the pair of both of these that plays the role played by the terms in §1.2.1; that is, we may regard $x : A \vdash M : B$ as a notational variation of something like[3] $x.M : (A \vdash B)$, and regard $x.M$ as the real "term". However, everyone always refers to the non-variable part M as the *term*, and the separation into variable (or, later, variables) and term is responsible for much of the characteristic behavior of terms in type theory.

In particular, unlike in the cut-ful theory, it is no longer true that each derivation determines a *unique* term (or more precisely, variable-term pair), because we have to choose a name for the variable. As written on the page, the judgments $x : A \vdash f(x) : B$ and $y : A \vdash f(y) : B$ are distinct; but they represent the same derivation (if we remove the term annotations) and the same morphism:

$$\dfrac{\dfrac{}{x : A \vdash x : A}\,\mathsf{id}}{x : A \vdash f(x) : B}\,f \qquad\qquad \dfrac{\dfrac{}{y : A \vdash y : A}\,\mathsf{id}}{y : A \vdash f(y) : B}\,f$$

This should not really be overly worrisome. Recall that we regard terms as merely *notation* for derivations, which we introduced in order to talk about

[3] The period used for the pairing here is a "variable binder"; we will return to it later on.

derivations (and, in particular, to describe an equivalence relation $\equiv$ on them) in a more concise and readable way. Thus, we are really just saying that we have more than one notation for the same thing, which is of course commonplace in mathematics. For instance, saying "let $f(x) = x^2$" and "let $f(t) = t^2$" are two notationally different ways to define exactly the same function $\mathbb{R} \to \mathbb{R}$.

To be sure, there is a different viewpoint on type theory that takes *terms* as primary objects rather than derivations, regarding the derivability of a judgment such as $x : X \vdash M : B$ as a *property* of the term M, rather than regarding (as we do) the term M as a notation for a particular derivation of $X \vdash B$. One reason for this is that terms are (by design) much more concise than derivations, and so if we want to represent type theory in a computer then it is attractive to use terms as the basic objects rather than derivations.

We will not follow this route. However, even though we maintain the viewpoint that derivations are primary, there are reasons to think a bit more carefully about the issue of variable names. Most of these reasons will not arise until chapter 2, so we will not say very much about the issue here; but we will at least introduce in our present simple context the two basic ways of dealing with the ambiguity in variable names.

The first method is to decide, once and for all, on a single variable name (say, x) to use for *all* our derivations. Then we cannot write $y : A \vdash f(y) : B$ at all, and so every derivation does determine a unique term. We call this the **de Bruijn method**. (In theories with multiple variables this method becomes more complicated; we will return to this in chapter 2.)

The second method is to allow arbitrary choices of variable names (from some standard alphabet), but be aware of the operation of variable renaming. We say that two terms are **α-equivalent** if they differ by renaming the variable; thus we can say that a derivation determines a unique α-equivalence class of terms. (In theories with "variable binding", the definition of α-equivalence is likewise more complicated; we will return to this in §1.5 and discuss it formally in Appendix A.)

Of these two methods, the de Bruijn method is theoretically cleaner, and better for implementation in a computer, but tends to detract from readability for human mathematicians. We will return to discuss these two methods when we have more complicated theories where there is more interesting to say about them. For now, we continue to use arbitrary variables, remembering that the particular choice of variable name is irrelevant, that derivations are primary, and that terms are just a convenient notation for derivations.

Now that we have such a convenient notation, we can observe that Theorem 1.2.5 is not just a statement about derivability. Indeed, the proof that we gave is "constructive", in the strong sense that it actually determines an *algorithm* for transforming a pair of derivations of $A \vdash B$ and $B \vdash C$ into a derivation of $A \vdash C$. The inductive nature of the proof means that this algorithm is recursive. And because terms uniquely represent derivations (modulo α-equivalence), it can equivalently be considered an operation on derivable term judgments.

For instance, suppose we start with $x : A \vdash f(x) : B$ and $y : B \vdash h(g(y)) : C$;

then the construction proceeds in the following steps.

- The second derivation ends with an application of h, so we apply the inductive hypothesis to $x : A \vdash f(x) : B$ and $y : B \vdash g(y) : D$.
- Now the second derivation begins with an application of g, so we recurse again on $x : A \vdash f(x) : B$ and and $y : B \vdash y : B$.
- This time the second derivation is just the identity rule, so the result is the first given derivation $x : A \vdash f(x) : B$.
- Backing out of the induction one step, we apply g to this result to get $x : A \vdash g(f(x)) : D$.
- Finally, backing out one more time, we apply h to the previous result to get $x : A \vdash h(g(f(x))) : C$.

Intuitively, the result $h(g(f(x)))$ has been obtained by *substituting* the term $f(x)$ for the variable y in the term $h(g(y))$. Thus, we refer to the operation defined by Theorem 1.2.5 as **substitution**, and sometimes state Theorem 1.2.5 and its analogues as **substitution is admissible**. In general, given $x : A \vdash M : B$ and $y : B \vdash N : C$ we denote the substitution of M for y in N by $N[M/y]$ (although unfortunately one also finds other notations in the literature; including, quite confusingly, $[M/y]N$ and $N[y/M]$).

The operation $N[M/y]$ this is "meta-notation": the square brackets are not part of the syntax of terms, instead they denote an operation *on* terms. The proof of Theorem 1.2.5 *defines* the notion of substitution recursively in the following way:

$$y[M/y] = M \tag{1.2.9}$$
$$f(N)[M/y] = f(N[M/y]) \tag{1.2.10}$$

When terms are regarded as objects of study in their own right, rather than just as notations for derivations, it is common to define substitution as an operation on terms first, and then to state Theorem 1.2.5 as "if $x : A \vdash M : B$ and $y : B \vdash N : C$ are derivable, then so is $x : A \vdash N[M/y] : C$". We instead consider Theorem 1.2.5 as fundamentally an operation on derivations, which we call "substitution" especially when representing it using term notation.

Note, though, that because a derivation is represented by a term together with a variable for the antecedent (that is, $x : X \vdash M : B$ is a notational variant of $x.M : (X \vdash B)$), technically this operation on derivations has to specify the variables too. The notation $N[M/y]$ represents only the term part; so the definitions (1.2.9) and (1.2.10) are only complete when combined with the statement that the variable of $N[M/y]$ is the same as that of M.

Remark 1.2.11. Substitution is already a place where the use of distinct named variables (and hence α-equivalence) makes the exposition substantially clearer for a human reader. We even teach our calculus students (or, at least, the author does) that when composing functions f and g, it is clearer to use different variables for the two functions, writing $y = f(x)$ but $z = g(y)$ and then plugging $f(x)$ in place of y in the second equation to get $z = g(f(x))$. It is possible to get away with using the same variable for the inputs of all functions, as we do in de Bruijn style, but it is much easier to get confused that way.

Before proving the initiality theorem, let us first observe that substitution does, in fact, define a category:

Lemma 1.2.12. *Substitution is associative: given $x : A \vdash M : B$ and $y : B \vdash N : C$ and $z : C \vdash P : D$, we have $P[N/z][M/y] = P[N[M/y]/z]$. (This is a literal equality of derivations, or equivalently of terms modulo α-equivalence.)*

Proof. By induction on the derivation of P. If it ends with the identity, so that $P = z$, then

$$P[N/z][M/y] = z[N/z][M/y] = N[M/y] = z[N[M/y]/z] = P[N[M/y]/z]$$

If it ends with an application of a morphism f, so that $P = f(Q)$, then

$$\begin{aligned} f(Q)[N/z][M/y] = f(Q[N/z])[M/y] &= f(Q[N/z][M/y]) \\ &= f(Q[N[M/y]/z]) = f(Q)[N[M/y]/z] \end{aligned}$$

using the inductive hypothesis for Q in the third step. □

Theorem 1.2.13. *The free category on a directed graph $\mathcal{G}$ has the same objects as $\mathcal{G}$, and its morphisms are the derivations $A \vdash B$ in the cut-free type theory for categories under $\mathcal{G}$ (or, equivalently, the derivable term judgments $x : A \vdash M : B$, modulo α-equivalence).*

Proof. Let $\mathfrak{F}_{\mathbf{Cat}}\mathcal{G}$ be defined as in the statement, with composition given by substitution constructed as in Theorem 1.2.5. By Lemma 1.2.12, composition is associative. For unitality, we have $y[M/y] = y$ by definition, while $N[x/x] = N$ is another easy induction on the structure of N. Thus, $\mathfrak{F}_{\mathbf{Cat}}\mathcal{G}$ is a category.

Now suppose $\mathcal{A}$ is any category and $\omega : \mathcal{G} \to \mathcal{A}$ is a map of directed graphs. We define $\overline{\omega} : \mathfrak{F}_{\mathbf{Cat}}\mathcal{G} \to \mathcal{A}$ by recursion on the rules of the type theory: the identity $x : A \vdash x : A$ goes to $\mathsf{id}_{P(A)}$, while $x : A \vdash f(M) : B$ goes to $\omega(f) \circ \overline{\omega}(M)$, with $\overline{\omega}(M)$ defined recursively. Since $x : A \vdash f(M) : B$ is the composite of $x : A \vdash M : C$ and $y : C \vdash f(y) : B$ in $\mathfrak{F}_{\mathbf{Cat}}\mathcal{G}$, this is the only possible definition that could make $\overline{\omega}$ a functor. It remains to check that it actually *is* a functor, i.e. that it preserves *all* composites; that is, we must show that $\overline{\omega}(N[M/y]) = \overline{\omega}(N) \circ \overline{\omega}(M)$. This follows by yet another induction on the derivation of N. □

Note that we did not have to impose any equivalence relation on the derivations in this theory. This suggests a second, more interesting, general statement about categorical logic: it is a syntax for generating free categorical structures using derivations from rules *that yield elements in canonical form*, eliminating the need for quotients. This statement is actually too narrow; as we will see later on, type theory is not *just* about canonical forms. However, canonical forms do play a very important role.

From the perspective of category theory, the reason for the importance of canonical forms is that we can easily decide whether two canonical forms are equal. In the cut-free type theory for categories, two terms present the same morphism in a free category just when they are literally equal (modulo α-equivalence); whereas to check whether two terms are equal in the cut-ful theory we have to remove the identities and reassociate them all to the left or the right.

In fact, a good algorithm for checking equality of terms in the cut-ful theory is to *interpret them into the cut-free theory*! That is, we note that every rule of the cut-ful theory is admissible in the cut-free theory, and hence eliminable; so any term (i.e. derivation) in the cut-ful theory yields a derivation in the cut-free theory. For instance, to translate the cut-ful term $h \circ_C ((\mathrm{id}_C \circ_C g) \circ_B f)$ into the cut-free theory, we first write it as a derivation

$$\dfrac{h : (C \vdash D) \qquad \dfrac{\dfrac{\mathrm{id}_C : (C \vdash C) \qquad g : (B \vdash C)}{(\mathrm{id}_C \circ_C g) : (B \vdash C)}\circ \qquad f : (A \vdash B)}{((\mathrm{id}_C \circ_C g) \circ_B f) : (A \vdash C)}\circ}{(h \circ_C ((\mathrm{id}_C \circ_C g) \circ_B f)) : (A \vdash D)}\circ$$

and then annotate the same derivation by cut-free terms, using substitution for composition:

$$\dfrac{z : C \vdash h(z) : D \qquad \dfrac{\dfrac{z : C \vdash z : C \qquad y : B \vdash g(y) : C}{y : B \vdash g(y) : C}\circ \qquad x : A \vdash f(x) : B}{x : A \vdash g(f(x)) : C}\circ}{x : A \vdash h(g(f(x))) : D}\circ$$

Since, as we have proven, both the cut-ful and the cut-free theory present the same free structure, it follows that *two terms in the cut-ful theory are equal modulo* $\equiv$ *exactly when their images in the cut-free theory are identical*. Informally, we are just comparing two terms by “removing all the identities and the parentheses”; but in a more complicated theory much more can be going on.

In this sense, type theory can be considered to be about solving *coherence problems* in category theory. In general, the coherence problem for a categorical structure is to decide when two morphisms “constructed from its basic data” are equal (or isomorphic, etc.) For instance, the classical coherence theorem of MacLane for monoidal categories says, informally, that two parallel morphisms constructed from the basic constraint isomorphisms of a monoidal category are *always* equal; whereas the analogous theorem for braided monoidal categories says that they are equal if and only if they have the same underlying braid. A type-theoretic calculus of canonical forms gives a way to answer this question, by translating a cut-ful theory into a cut-free one, and cut-elimination methods have frequently been used in the proof of coherence theorems. We will not explore this aspect here, however.

A related remark is that categorical logic is about *showing that two different categories have the same*[4] *initial object*. The primitive rules of a type theory can be regarded as the "operations" of a certain algebraic theory, and the judgments that can be derived from these rules form the initial algebra for this theory, i.e. the initial object in a certain category. (See Appendix A for a precise statement along these lines.) The initiality theorems we care about, however, show that these initial objects are *also* initial in some other, quite different, category that is of more intrinsic categorical interest.

Remark 1.2.14. This point of view sheds further light on the distinction between derivable and admissible rules mentioned in Remark 1.2.6. A derivable rule automatically holds in any model of the "algebraic theory" version of a type theory, whereas an admissible rules holds *only in the initial algebra* (or, more generally, free algebras) for this algebraic theory. In particular, an arbitrary model of the algebraic rules of the cut-free type theory for categories is not even a category, e.g. it may not satisfy the cut rule.

It can be tempting for a category theorist, upon learning that type theory is a presentation of a certain free structure, to conclude that the emphasis on *free* structures is myopic or of only historical interest, and attempt to generalize to not-necessarily-free algebras over the same theory. This temptation should be resisted. At best, it leads to neglect of some of the most important and interesting features of type theory, such as cut-elimination, which holds only in free structures. At worst, it leads to nonsense, for central type-theoretic notions such as "bound variable" (see §1.5) *only make sense* in free structures. We will see in §1.7 that we can still use type theory to "present" categorical objects that are not themselves free (at least, not in the usual sense); but the syntax of types and terms/derivations must still itself be freely generated.

Exercises

Exercise 1.2.1. Let $\mathscr{M}$ be a fixed category; then we have an induced adjunction between $\mathbf{Cat}/\mathscr{M}$ and $\mathbf{Gr}/\mathscr{M}$. Describe a cut-free type theory for presenting the free category-over-$\mathscr{M}$ on a directed-graph-over-$\mathscr{M}$, and prove the initiality theorem (the analogue of Theorem 1.2.13). Note that you will have to prove that cut is admissible first. *(Hint: index the judgments by arrows in $\mathscr{M}$, so that for instance $A \vdash_\alpha B$ represents an arrow lying over a given arrow α in $\mathscr{M}$.)*

Exercise 1.2.2. Category theorists are accustomed to consider **Cat** as a 2-category, but our free category $\mathfrak{F}_{\mathbf{Cat}}\mathcal{G}$ only has a 1-categorical universal property, expressed by the 1-categorical adjunction between **Cat** and **Gr**. It is not immediately obvious how it could be otherwise, since unlike **Cat**, **Gr** is only a 1-category; but there is something along these lines that we can say.

(a) Suppose $\mathcal{G}$ is a directed graph and $\mathcal{C}$ a category; define a category $\mathbf{Gr}(\mathcal{G}, \mathcal{C})$ whose objects are graph morphisms $\mathcal{G} \to \mathcal{C}$ and whose morphisms are an

[4] Of course, technically, an object of one category is not generally also an object of another one. So what we mean is that there is an easy way to transform the initial object of one category into the initial object of another.

appropriate kind of "natural transformation".

(b) Prove that $\mathbf{Gr}(\mathcal{G}, -)$ is a 2-functor $\mathbf{Cat} \to \mathbf{Cat}$.

(c) Using the cut-free presentation of $\mathfrak{F}_{\mathbf{Cat}}\mathcal{G}$, prove that it is a representing object for this 2-functor.

Exercise 1.2.3. Regarding the cut-free type theory for categories as describing a multi-sorted algebraic theory, define a particular algebra for this theory that does not satisfy the cut rule. Then define another algebra that does admit a "cut rule", but in which the resulting "composition" is not associative.

1.3 Meet-semilattices

Moving gradually up the ladder of nontriviality, we now consider categories with finite products, or more precisely binary products and a terminal object. In fact, let us revert back to the posetal world first and consider posets with binary meets and a top element, i.e. meet-semilattices. We will make all this structure algebraic, so that our meet-semilattices are posets (which, recall, is not necessarily skeletal) *equipped with* a chosen top element and an operation assigning to each pair of objects a meet. We then have an adjunction relating the category **mSLat** of such meet-semilattices (and morphisms preserving all the structure strictly) with the category **RelGr** of relational graphs, and we want to describe the free meet-semilattice on a relational graph $\mathcal{G}$.

One new feature this introduces is that the objects of $\mathfrak{F}_{\mathbf{mSLat}}\mathcal{G}$ will no longer be the same as those of $\mathcal{G}$: we need to add a top element and freely apply the meet operation. In order to describe this type-theoretically, we introduce a new judgment "$\vdash A\ \mathsf{type}$", meaning that A will be one of the objects of the poset we are generating. The rules for this judgment are

$$\frac{}{\vdash \top\ \mathsf{type}} \qquad\qquad \frac{\vdash A\ \mathsf{type} \qquad \vdash B\ \mathsf{type}}{\vdash A \wedge B\ \mathsf{type}}$$

When talking about type theory under $\mathcal{G}$, we additionally include "axiom" rules saying that each object of $\mathcal{G}$ is a type:

$$\frac{A \in \mathcal{G}}{\vdash A\ \mathsf{type}}$$

Note that the premise $A \in \mathcal{G}$ here is not a judgment; rather it is an "external" fact that serves as a precondition for application of this rule. Thus it would be more correct to write this rule as

$$\frac{}{\vdash A\ \mathsf{type}}\ (\text{if } A \in \mathcal{G})$$

but we will generally write such conditions as premises, since otherwise the notation can get rather unwieldy.

As an example of the application of these rules, if $A, B \in \mathcal{G}$ we have a derivation

$$\dfrac{\dfrac{A \in \mathcal{G}}{\vdash A\ \mathsf{type}} \qquad \dfrac{\dfrac{}{\vdash \top\ \mathsf{type}} \qquad \dfrac{B \in \mathcal{G}}{\vdash B\ \mathsf{type}}}{\vdash \top \wedge B\ \mathsf{type}}}{\vdash (A \wedge (\top \wedge B))\ \mathsf{type}}$$

so that $A \wedge (\top \wedge B)$ will be one of the objects of $\mathfrak{F}_{\mathbf{mSLat}}\mathcal{G}$.

Now we need to describe the morphisms, i.e. the relation $\leq$ in $\mathfrak{F}_{\mathbf{mSLat}}\mathcal{G}$. The obvious thing to do is to assert the universal property of the meet and the top element:

$$\dfrac{}{A \vdash \top} \qquad \dfrac{}{A \wedge B \vdash A} \qquad \dfrac{}{A \wedge B \vdash B} \qquad \dfrac{A \vdash B \qquad A \vdash C}{A \vdash B \wedge C}$$

This works, but it forces us to go back to asserting transitivity/cut. For instance, if $A, B, C \in \mathcal{G}$ we have the following derivation:

$$\dfrac{\dfrac{}{(A \wedge B) \wedge C \vdash A \wedge B} \qquad \dfrac{}{A \wedge B \vdash A}}{(A \wedge B) \wedge C \vdash A}$$

but there is no way to deduce this without using the cut rule. Thus, this "cut-ful type theory for meet-semilattices under $\mathcal{G}$" works, but to have a better class of "canonical forms" for its relations we would also like a cut-free version.

What we need to do is to treat the "projections" $A \wedge B \to A$ and $A \wedge B \to B$ similarly to how we treated the edges of $\mathcal{G}$ in §1.2. However, at this point we have to make a choice of whether to build in postcomposition or precomposition:

$$\dfrac{A \vdash C}{A \wedge B \vdash C} \qquad \text{or} \qquad \dfrac{C \vdash A \wedge B}{C \vdash A} \qquad ?$$

Both choices work (that is, they make cut admissible), and lead to different kinds of type theories with different properties. The first leads to a kind of type theory called **sequent calculus**, and the second to a kind of type theory called **natural deduction**. We consider each in turn.

1.3.1 Sequent calculus for meet-semilattices

To be precise, for a relational graph $\mathcal{G}$, the **unary sequent calculus for meet-semilattices under $\mathcal{G}$** has the following rules (in addition to the rules for the judgment $\vdash A\ \mathsf{type}$ mentioned above). We label each rule on the right to make them easier to refer to later on.

$$\dfrac{A \in \mathcal{G}}{A \vdash A}\ \mathsf{id} \qquad \dfrac{f \in \mathcal{G}(A, B) \qquad X \vdash A}{X \vdash B}\ fR \qquad \dfrac{\vdash A\ \mathsf{type}}{A \vdash \top}\ \top R$$

$$\dfrac{A \vdash C \qquad \vdash B\ \mathsf{type}}{A \wedge B \vdash C}\ \wedge L1 \qquad \dfrac{B \vdash C \qquad \vdash A\ \mathsf{type}}{A \wedge B \vdash C}\ \wedge L2 \qquad \dfrac{A \vdash B \qquad A \vdash C}{A \vdash B \wedge C}\ \wedge R$$

There are several things to note about this. The first is that we have included in the premises some judgments of the form $\vdash A\ \mathsf{type}$. This ensures that whenever we can derive a sequent $A \vdash B$, that A and B are well-formed as types. However, we don't need to assume explicitly as premises that *all* types appearing in any sequent are well-formed, only those that are introduced without belonging to any previous sequents; this is sufficient for the following inductive proof.

Theorem 1.3.1. *In the unary sequent calculus for meet-semilattices under $\mathcal{G}$, if $A \vdash B$ is derivable, then so are $\vdash A\ \mathsf{type}$ and $\vdash B\ \mathsf{type}$.*

Proof. By induction on the derivation of $A \vdash B$.

- If it is the id rule, then $A \in \mathcal{G}$ and so $\vdash A\ \mathsf{type}$.
- If it ends with the rule fR for some $f \in \mathcal{G}(A, B)$, then $B \in \mathcal{G}$ and so $\vdash B\ \mathsf{type}$, while $X \vdash A$ and so $\vdash X\ \mathsf{type}$ by the inductive hypothesis.
- If it ends with the rule $\top R$, then $\vdash A\ \mathsf{type}$ by assumption.
- If it ends with the rule $\wedge L1$, then $\vdash B\ \mathsf{type}$ by assumption, while $\vdash A\ \mathsf{type}$ and $\vdash C\ \mathsf{type}$ by the inductive hypothesis; thus also $\vdash A \wedge B\ \mathsf{type}$.
- The cases for $\wedge L2$ and $\wedge R$ are similar. □

We will generally formulate our type theories with just enough premises to make theorems such as Theorem 1.3.1 true. Essentially this means that if some type appears in the conclusion but not in any of the premises, we have to add its "type-ness" judgment as an additional premise. We will not usually state and prove theorems analogous to Theorem 1.3.1 explicitly, but the reader can verify that they will always be true.

The second thing to note about our current type theory is that we only assert the identity rule $A \vdash A$ when A is a *generating object* (also called a *base type*), i.e. an object of $\mathcal{G}$. This is sufficient because in the sequent calculus, we can derive the identity rule for any type:

Theorem 1.3.2. *In the unary sequent calculus for meet-semilattices under $\mathcal{G}$, if A is a type (that is, if $\vdash A\ \mathsf{type}$ is derivable), then $A \vdash A$ is derivable.*

Proof. We induct on the derivation of $\vdash A\ \mathsf{type}$. There are three cases:

(a) A is in $\mathcal{G}$. In this case $A \vdash A$ is an axiom.

(b) $A = \top$. In this case $\top \vdash \top$ is a special case of the rule that anything $\vdash \top$.

(c) $A = B \wedge C$ and we have derivations $\mathscr{D}_B$ and $\mathscr{D}_C$ of $\vdash B\ \mathsf{type}$ and $\vdash C\ \mathsf{type}$ respectively. Therefore we have, inductively, derivations $\mathscr{D}_1$ and $\mathscr{D}_2$ of $B \vdash B$ and $C \vdash C$, and we can put them together like this:

$$\dfrac{\dfrac{\dfrac{\mathscr{D}_1\ \vdots}{B \vdash B} \quad \dfrac{\mathscr{D}_C\ \vdots}{\vdash C\ \mathsf{type}}}{B \wedge C \vdash B} \qquad \dfrac{\dfrac{\mathscr{D}_2\ \vdots}{C \vdash C} \quad \dfrac{\mathscr{D}_B\ \vdots}{\vdash B\ \mathsf{type}}}{B \wedge C \vdash C}}{B \wedge C \vdash B \wedge C}$$

□

In other words, the general identity rule

$$\frac{\vdash A \ \mathsf{type}}{A \vdash A}$$

is also *admissible*. This is a general characteristic of sequent calculi.

Next we prove that the cut rule is admissible for this sequent calculus too.

Theorem 1.3.3. *In the unary sequent calculus for meet-semilattices under $\mathcal{G}$, if $A \vdash B$ and $B \vdash C$ are derivable, then so is $A \vdash C$.*

Proof. By induction on the derivation of $B \vdash C$.

(a) If it is id, then $B = C$. Now $A \vdash C$ is just $A \vdash B$ and we are done.

(b) If it is $f \in \mathcal{G}(C', C)$, then we have a derivation of $B \vdash C'$. So by the inductive hypothesis we can derive $A \vdash C'$, whence also $A \vdash C$ by the rule for f.

(c) If it ends with $\top R$, then $C = \top$. Since $A \vdash B$ is derivable, by Theorem 1.3.1 $\vdash A \ \mathsf{type}$ is also derivable; thus by $\top R$ we have $A \vdash \top$.

(d) If it ends with $\wedge R$, then $C = C_1 \wedge C_2$ and we have derivations of $B \vdash C_1$ and $B \vdash C_2$. By the inductive hypothesis we can derive both $A \vdash C_1$ and $A \vdash C_2$, to which we can apply $\wedge R$ to get $A \vdash C_1 \wedge C_2$.

(e) If it ends with $\wedge L1$, then $B = B_1 \wedge B_2$ and we can derive $B_1 \vdash C$. We now do a secondary induction on the derivation of $A \vdash B$.

   (i) It cannot end with id or f or $\top R$, since $B = B_1 \wedge B_2$ is not in $\mathcal{G}$ and not equal to $\top$.

   (ii) If it ends with $\wedge L1$, then $A = A_1 \wedge A_2$ and we can derive $A_1 \vdash B$. By the inductive hypothesis, we can derive $A_1 \vdash C$, and hence by $\wedge L1$ also $A \vdash C$. The case of $\wedge L2$ is similar.

   (iii) If it ends with $\wedge R$, then we can derive $A \vdash B_1$ and $A \vdash B_2$. Recall that we are also assuming a derivation of $B_1 \vdash C$. Thus, by the inductive hypothesis on $A \vdash B_1$ and $B_1 \vdash C$, we can derive $A \vdash C$.

(f) The case when it ends with $\wedge L2$ is similar. □

This simple proof already displays many of the characteristic features of a cut-admissibility argument. The final case (e)(iii) is called the **principal case** for the operation $\wedge$, when the type B we are composing over (also called the **cut formula**) is obtained from $\wedge$ and both sequents are also obtained from the $\wedge$ rules. In a direct argument for cut-elimination such as that sketched after

Theorem 1.2.7, this case becomes the following transformation on derivations:

$$\dfrac{\dfrac{\dfrac{\mathscr{D}_1}{\vdots}}{A \vdash B_1} \quad \dfrac{\dfrac{\mathscr{D}_2}{\vdots}}{A \vdash B_2}}{A \vdash B_1 \wedge B_2}\wedge R \qquad \dfrac{\dfrac{\dfrac{\mathscr{D}_3}{\vdots}}{B_1 \vdash C}}{B_1 \wedge B_2 \vdash C}\wedge L1 \over A \vdash C}\text{CUT} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\mathscr{D}_1}{\vdots}}{A \vdash B_1} \quad \dfrac{\dfrac{\mathscr{D}_3}{\vdots}}{B_1 \vdash C}}{A \vdash C}\text{CUT}$$

Remark 1.3.4. It may seem somewhat odd that we can prove the admissibility of all cuts (compositions), but we have to assert identities as a primitive rule for base/generating types. This is essentially because we chose to "build a cut" into the rule fR that represents the generating arrows. If we had not, then we would have to assert "cuts over base types" (that is, where the cut formula is an object of $\mathcal{G}$) as primitive rules, the way we did in the cut-ful theory of §1.2.1. Put differently, building a cut into fR is essentially the "morphism version" of asserting identities primitively for base types.

Finally, we have the initiality theorem:

Theorem 1.3.5. *For any relational graph $\mathcal{G}$, the free meet-semilattice $\mathfrak{F}_{\mathbf{mSLat}}\mathcal{G}$ it generates is described by the unary sequent calculus for meet-semilattices under $\mathcal{G}$: its objects are the A such that $\vdash A$ type is derivable, with $A \le B$ just when $A \vdash B$ is derivable.*

Proof. Theorems 1.3.2 and 1.3.3 show that this defines a poset $\mathfrak{F}_{\mathbf{mSLat}}\mathcal{G}$. The rule $\top R$ implies that $\top$ is a top element, while the rules $\wedge L1$, $\wedge L2$, and $\wedge R$ imply that $A \wedge B$ is a binary meet. Therefore, we have a meet-semilattice. Moreover, the rules id and f yield a map of posets $\mathcal{G} \to \mathfrak{F}_{\mathbf{mSLat}}\mathcal{G}$.

Now suppose $\mathcal{M}$ is any other meet-semilattice with a map $\omega : \mathcal{G} \to \mathcal{M}$. Recall that a meet-semilattices is equipped with a chosen top element and meet function. We extend ω to a map from the objects of $\mathfrak{F}_{\mathbf{mSLat}}\mathcal{G}$ by recursion on the construction of the latter, sending $\top$ to the chosen top element of $\mathcal{M}$, and $A \wedge B$ to the chosen meet in $\mathcal{M}$ of the (recursively defined) images of A and B. This is clearly the only possible meet-semilattice map extending ω, and it clearly preserves the chosen meets and top element, so it suffices to check that it is a poset map. This follows by a straightforward induction over the rules for deriving the judgment $A \vdash B$. □

To finish, we observe that this sequent calculus has another important property. Inspecting the rules, we see that the operations $\wedge$ and $\top$ only ever appear in the *conclusions* of rules. Each operation $\wedge$ and $\top$ has zero or more rules allowing us to introduce it on the right of the conclusion, and likewise zero or more rules allowing us to introduce it on the left. (Specifically, $\wedge$ has two left rules and one right rule, while $\top$ has zero left rules and one right rule.) This is convenient if we are given a sequent $A \vdash B$ and want to figure out whether it is derivable: we can choose rules to apply "in reverse" by breaking down A and B according to their construction out of $\wedge$ and $\top$.

It also tells us nontrivial things about derivations. For instance, all the primitive rules have the property that every type appearing in their premises also appears as a sub-expression of some type in their conclusion. Thus, any (cut-free) *derivation* of a sequent $A \vdash B$ must involves only types appearing as sub-expressions of A and B. This is called the **subformula property**.

The phrase *sequent calculus*, like *type theory*, is difficult to define precisely, but sequent calculi generally exhibit the properties we have observed in this subsection: admissibility of the identity rule (based on an axiom applying only to base types), admissibility of cut, type operations appearing only in the conclusions of rules, and the subformula property.

1.3.2 Natural deduction for meet-semilattices

Now suppose we make the other choice about how to treat projections. We call this the **unary natural deduction for meet-semilattices under** $\mathcal{G}$; its rules (in addition to those for $\vdash A \ \mathsf{type}$) are

$$\frac{\vdash X \ \mathsf{type}}{X \vdash X}\ \mathsf{id} \qquad \frac{f \in \mathcal{G}(A,B) \qquad X \vdash A}{X \vdash B}\ fI \qquad \frac{\vdash X \ \mathsf{type}}{X \vdash \top}\ \top I$$

$$\frac{X \vdash B \wedge C}{X \vdash B}\ \wedge E1 \qquad \frac{X \vdash B \wedge C}{X \vdash C}\ \wedge E2 \qquad \frac{X \vdash B \qquad X \vdash C}{X \vdash B \wedge C}\ \wedge I$$

We observe first that this theory has the same well-formedness property as the sequent calculus:

Theorem 1.3.6. *In the unary natural deduction for meet-semilattices under $\mathcal{G}$, if $A \vdash B$ is derivable, then so are $\vdash A \ \mathsf{type}$ and $\vdash B \ \mathsf{type}$.* □

Unlike the sequent calculus, however, the general identity rule is not admissible: there is no way to derive $A \wedge B \vdash A \wedge B$ from $A \vdash A$ and $B \vdash B$ without it. Thus, we assert the id for all types, not just those coming from $\mathcal{G}$.

Cut, however, is still admissible:

Theorem 1.3.7. *In the unary natural deduction for meet-semilattices under $\mathcal{G}$, if $A \vdash B$ and $B \vdash C$ are derivable, then so is $A \vdash C$.*

Proof. We induct on the derivation of $B \vdash C$.

(a) The cases when it ends with id, f, $\top I$, and $\wedge I$ are just like those in Theorem 1.3.3 for id, f, $\top R$, and $\wedge R$.

(b) If it ends with $\wedge E1$, then we have $B \vdash C \wedge D$ for some D. Thus, $A \vdash C \wedge D$ by the inductive hypothesis, so $A \vdash C$ by $\wedge E1$. The case of $\wedge E2$ is similar. □

The proof is noticeably simpler than that of Theorem 1.3.3; we don't need the secondary inner induction. This is essentially due to the fact that all the rules of this theory involve an *arbitrary* type X on the left (rather than one built using operations such as $\wedge$). Thus, instead of the rules of sequent calculus that introduce operations like $\wedge$ and $\top$ on the left and right, we have rules like $\top I$ and $\wedge I$ that introduce them on the right, and also rules that *eliminate* them on the right like $\wedge E1$ and $\wedge E2$. These properties are characteristic of *natural deduction* theories. (Later on, in §2.7.3, we will be able to give a more convincing explanation of the origin of the phrase "natural deduction".)

Remark 1.3.8. Because the proof of cut admissibility for natural deduction theories is so much simpler than that for sequent calculus, some people say that the former is "trivial". Triviality is subjective; but what seems inarguable is that cut-admissibility for natural deduction is *saying something different* than cut-admissibility for sequent calculus. The content of cut-admissibility for sequent calculus corresponds more closely to β-conversion in natural deduction (see §1.4). Similarly, the admissibility of the identity rule for sequent calculus corresponds to the η-conversion rule in natural deduction (see §1.4 for that too).

Of course, we should also prove the initiality theorem:

Theorem 1.3.9. *For any relational graph $\mathcal{G}$, the free meet-semilattice $\mathfrak{F}_{\mathbf{mSLat}}\mathcal{G}$ it generates is described by the unary natural deduction for meet-semilattices under $\mathcal{G}$: its objects are the A such that $\vdash A\ \mathsf{type}$ is derivable, with $A \le B$ just when $A \vdash B$ is derivable.*

Proof. Almost exactly like Theorem 1.3.5. □

Exercises

Exercise 1.3.1. Using the unary sequent calculus for meet-semilattices, prove that $A \wedge A \cong A$ for any object A of any meet-semilattice. (Recall that meet-semilattices are categories with at most one morphism in each hom-set, so for two objects to be isomorphic it suffices to have a morphism in each direction.) Then prove the same thing using the natural deduction.

Exercise 1.3.2. Using either the sequent calculus or the natural deduction for meet-semilattices (your choice), prove that in any meet-semilattice we have

$$A \wedge \top \cong A \qquad \top \wedge A \cong A \qquad A \wedge B \cong B \wedge A \qquad A \wedge (B \wedge C) \cong (A \wedge B) \wedge C$$

Exercise 1.3.3. Prove that the rules $\top R$ and $\wedge R$ in the unary sequent calculus for meet-semilattices are *invertible*, in the sense that whenever we have a derivation of their conclusions, we also have a derivation of all their premises.

Exercise 1.3.4. Describe a sequent calculus for *join-semilattices* (posets with a bottom element and binary joins), and prove the admissibility and initiality theorems for it. The rules for $\bot$ and $\vee$ should be exactly dual to the rules for $\top$ and $\wedge$.

Exercise 1.3.5. By putting together the rules for meet- and join-semilattices, describe a sequent calculus for *lattices* (posets with a top and bottom element and binary meets and joins), and prove the admissibility and initiality theorems for it.

Exercise 1.3.6. Prove that in your sequent calculus for lattices from Exercise 1.3.5, the rules $\top R$, $\wedge R$, $\bot L$, and $\vee L$ are all invertible in the sense of Exercise 1.3.3.

Exercise 1.3.7. A map of posets $P : \mathscr{A} \to \mathscr{M}$ is called a *(cloven) fibration* if whenever $b \in \mathscr{A}$ and $x \le P(b)$, there is a chosen $a \in \mathscr{A}$ such that $P(a) = x$ and $a \le b$ and moreover for any $c \in \mathscr{A}$, $c \le b$ and $P(c) \le x$ together imply $c \le a$. The object a can be written as $x^*(b)$.

(a) Given a fixed poset $\mathscr{M}$, describe a sequent calculus for fibrations over $\mathscr{M}$ by adding rules governing the operations x^* to the cut-free theory of Exercise 1.2.1.

(b) Prove the initiality theorem for this sequent calculus.

(c) Use this sequent calculus to prove that in any fibration $P : \mathscr{A} \to \mathscr{M}$, if we have $b \in \mathscr{A}$ and $x \le y \le P(b)$, then $x^*(y^*(b)) \cong x^*(b)$.

Exercise 1.3.8. Now describe instead a natural deduction for fibrations over $\mathscr{M}$, prove the initiality theorem, and re-prove that $x^*(y^*(b)) \cong x^*(b)$ using this theory.

Exercise 1.3.9. Suppose we augment your sequent calculus for fibrations over $\mathscr{M}$ from Exercise 1.3.7 with the following additional rules for "fiberwise meets". Here $\vdash A\ \mathsf{type}_x$ is a judgment indicating that A will be an object of our fibration in the fiber over $x \in \mathscr{M}$.

$$\frac{}{\vdash \top_x\ \mathsf{type}_x} \qquad \frac{\vdash A\ \mathsf{type}_x \qquad \vdash B\ \mathsf{type}_x}{\vdash A \wedge_x B\ \mathsf{type}_x} \qquad \frac{\vdash A\ \mathsf{type}_x}{A \vdash_{x \le y} \top_y}$$

$$\frac{A \vdash_{x \le y} C \qquad \vdash B\ \mathsf{type}_x}{A \wedge_x B \vdash_{x \le y} C} \qquad \frac{B \vdash_{x \le y} C \qquad \vdash A\ \mathsf{type}_x}{A \wedge_x B \vdash_{x \le y} C}$$

$$\frac{A \vdash_{x \le y} B \qquad A \vdash_{x \le y} C}{A \vdash_{x \le y} B \wedge_y C}$$

Consider the sequents

$$x^*(A \wedge_y B) \vdash_{x \le x} x^*A \wedge_x x^*B$$
$$x^*A \wedge_x x^*B \vdash_{x \le x} x^*(A \wedge_y B)$$

for $x \le y$, $\vdash A\ \mathsf{type}_y$, and $\vdash B\ \mathsf{type}_y$.

(a) Construct derivations of these sequents in the above sequent calculus.

(b) Write down an analoguous natural deduction and derive the above sequents therein.

(c) What categorical structure do you think these type theories construct the initial one of? If you feel energetic, prove the initiality theorem.

Exercise 1.3.10. A map of posets $P : \mathscr{A} \to \mathscr{M}$ is called an *opfibration* if $P^{\mathrm{op}} : \mathscr{A}^{\mathrm{op}} \to \mathscr{M}^{\mathrm{op}}$ is a fibration. The analogous operation takes $a \in \mathscr{A}$ and $P(a) \le y$ to a $b \in \mathscr{A}$ with $P(b) = y$ and $a \le b$ and a universal property; we write this b as $y_!(a)$. We say P is a *bifibration* if it is both a fibration and an opfibration. Describe a sequent calculus for bifibrations over a fixed $\mathscr{M}$, and prove the initiality theorem.

Exercise 1.3.11. Use your sequent calculus from Exercise 1.3.10 to prove that in a bifibration of posets, if $x \le y$ in $\mathscr{M}$, we have an adjunction $y_! \dashv x^*$.

Exercise 1.3.12. Use your sequent calculus from Exercise 1.3.10 to prove that in a bifibration of posets, if $x \cong y$ in $\mathscr{M}$, we have an isomorphism $x_! \cong x^*$ (that is, for any a in the fiber over y, we have $x_!(a) \cong x^*(a)$).

1.4 Categories with products

Now we move back from posets to categories. For brevity, by a **category with products** we will mean a category with specified binary products and a specified terminal object. Let **PrCat** be the category of such categories with products, and functors preserving them strictly. Then we have an adjunction relating **PrCat** to **Gr**, and we want to describe the left adjoint with a type theory.

As in §1.3.2, we could take either the sequent calculus route or the natural deduction route. Unfortunately, even if we build in enough composition to make cut admissible, in *both* cases we need to impose a further equivalence relation on the derivations, as there are single morphisms that can be derived in multiple ways. However, the ways in which this happens in the two cases are different.

On one hand, if we have an arrow $f : A \to C$ in a directed graph $\mathcal{G}$, then there is a morphism $A \times B \to A \to A \times C$ in the free category-with-products on $\mathcal{G}$. In a sequent calculus, there are two distinct derivations of this morphism:

$$
\dfrac{\dfrac{A \vdash A}{A \times B \vdash A} \qquad \dfrac{\dfrac{A \vdash A}{A \vdash C}\,f}{A \times B \vdash C}}{A \times B \vdash A \times C}
\qquad\qquad
\dfrac{\dfrac{A \vdash A \qquad \dfrac{A \vdash A}{A \vdash C}\,f}{A \vdash A \times C}}{A \times B \vdash A \times C}
$$

whereas in a natural deduction there will be only one:

$$
\dfrac{\dfrac{A \times B \vdash A \times B}{A \times B \vdash A} \qquad \dfrac{\dfrac{A \times B \vdash A \times B}{A \times B \vdash A}}{A \times B \vdash C}\,f}{A \times B \vdash A \times C}
$$

This sort of thing is true quite generally. A sequent calculus includes both left and right rules, so to derive a given sequent we must choose whether a left or a right rule is to be applied last. By contrast, both kinds of rules in a natural deduction (introduction and elimination) act on the right, so there is less choice about what rule to apply last.

On the other hand, if we have an arrow $A \to B$ in $\mathcal{G}$, then in a natural deduction there are (at least) two derivations of the identity $A \to A$:

$$\dfrac{\dfrac{A \vdash A \qquad \dfrac{A \vdash A}{A \vdash B}}{A \vdash A \times B}}{A \vdash A} \qquad \text{and} \qquad \dfrac{}{A \vdash A} \tag{1.4.1}$$

while in a sequent calculus there is only one:

$$\dfrac{}{A \vdash A}$$

This is also true quite generally. A natural deduction includes both introduction and elimination rules, so we will always be able to introduce a type and then eliminate it, essentially "doing nothing". By contrast, in a sequent calculus we have "only introduction rules" (of both left and right sorts), so this cannot happen.

Remark 1.4.2. In §0.1 we mentioned that by presenting free categorical structures without explicit reference to composition, type theory enables us to *define* composition in such a way as to ensure that various desirable properties hold as exact equalities. The point here is just that sequent calculus and natural deduction make *different* choices of which properties to ensure by their definitions of composition. Roughly speaking, sequent calculus chooses to make the *defining equations* of universal properties hold exactly (e.g. the composites of a paired morphism $\langle f, g \rangle : X \to A \times B$ with π_1 and π_2 are exactly f and g respectively); this is what the "principal case" of its cut-admissibility proof does. On the other hand, natural deduction chooses to make the *naturality*[5] of universal properties hold exactly; the equality between the two distinct sequent calculus derivations above expresses the naturality of pairing $\langle f, g \rangle : X \to A \times B$ with respect to precomposition by π_1.

In fact, in the simple case of a unary type theory for categories with products, there are tricks enabling us to eliminate both kinds of redundancy, and thereby do without any "$\equiv$" for both the sequent calculus and the natural deduction. (For sequent calculus the trick is called "focusing", and for natural deduction the trick is called "canonical/atomic terms".) However, in even slightly more complicated theories the analogous tricks do not eliminate $\equiv$ completely (though they do reduce its complexity). Moreover, our focus here is on type theory as a notation for category-theoretic arguments, not as a tool for isolating canonical forms and proving coherence theorems. Thus, we will not spend any time on these tricks, but instead bite the bullet and deal with $\equiv$.

As was the case in §1.2.1, it is easier to describe the rules for $\equiv$ if we first introduce terms for derivations. Thus, we generalize the abstract-variable term syntax of §1.2.2 with terms for the $\times$ and $\mathbb{1}$ rules, as shown in Figure 1.1.

[5] This is not actually the origin of the term "natural deduction" — see §2.7.3 for that — but it serves as a useful mnemonic.

$$\frac{\vdash X \ \mathsf{type}}{x : X \vdash x : X}\ \mathsf{id} \qquad \frac{f \in \mathcal{G}(A,B) \qquad x : X \vdash M : A}{x : X \vdash f(M) : B}\ fI \qquad \frac{\vdash X \ \mathsf{type}}{x : X \vdash * : \mathbb{1}}\ \mathbb{1}I$$

$$\frac{x : X \vdash M : B \times C}{x : X \vdash \pi_1^{B,C}(M) : B}\ \times E1 \qquad \frac{x : X \vdash M : B \times C}{x : X \vdash \pi_2^{B,C}(M) : C}\ \times E2$$

$$\frac{x : X \vdash M : B \qquad x : X \vdash N : C}{x : X \vdash \langle M, N\rangle : B \times C}\ \times I$$

Figure 1.1: Terms for categories with products

Since terms are just a notation for derivations, the general principle for naming derivations by terms is that each rule (being an operation on derivations) should correspond to a "term operation" that indicates unambiguously what rule is being applied, what the premises are (by including terms for them), and how they are combined. This is no different from any other mathematical notation for any other operation. For instance, in §1.2.1 the composition rule was represented by a (subscripted) binary operation $\circ_B$ combining a pair of terms for the premises.

However, the use of abstract variables complicates matters a bit, because we are only free to describe a notation for the term part (attached to the consequent), whereas the term part only describes a derivation when combined with a variable (attached to the antecedent). For instance, the two premises $x : X \vdash M : A$ and $x : X \vdash N : B$ of the rule $\times I$ are really $x.M : (X \vdash A)$ and $x.N \vdash (X \vdash B)$, and so the term for the induced derivation $X \vdash A \times B$ ought to "pair up" $x.M$ and $x.N$ somehow; but how would it then be associated to a variable in its own antecedent?

In our present case, we can solve this by using the fact that both premises have the same antecedent *and the same variable*, so we can "pull that variable out" of the pair and write $x.\langle M, N\rangle : (X \vdash A \times B)$, or $x : X \vdash \langle M, N\rangle : A \times B$. (In other cases, such as §1.5, a more complicated solution is needed.) But why *do* they have the same variable? Of course, if we use the de Bruijn method, then they must always have the same variable because all judgments have the same variable. But otherwise, they might in principle have different variables, and so we might have to rename the variable in one of them (that is, apply an α-equivalence) before we can use this notation for $\times I$. This is an important general principle.

Principle 1.4.3. *A rule applies equally to any derivations of its premises, but the abstract-variable term notation for that rule may require certain compatibilities between the variables occurring in the premises, which can always be ensured by α-equivalence.*

The terms for the other rules $\times E1, \times E2, \mathbb{1}I$ are relatively straightforward.

The superscript type annotations on the projections $\pi_1^{B,C}$ and $\pi_2^{B,C}$ are necessary to make type-checking possible, since otherwise it would not be clear from a term such as $x : A \vdash \pi_1(M) : B$ what the type of M should be in the premise. However, in practice we often omit them. It is then straightforward to prove the analogue of Lemma 1.2.2.

Note how well the natural-deduction choice of "all rules acting on the right" matches the use of abstract variables: in all cases we can think of "applying functions to arguments" in a familiar way. It is possible to describe sequent calculus derivations using terms as well, but they are less pretty. For this reason, *we will henceforth use sequent calculus only for posetal theories.* (I emphasize that this is a choice of focus and exposition for the present notes only; sequent calculus has successfully been used to answer many coherence questions about non-posetal categorical structures.) The need to impose the identity rule for all types (not just those coming from $\mathcal{G}$) also makes perfect sense from the abstract variable standpoint: a variable in any type is also a term of that type.

Now that we have our terms, the desired equivalence between the two derivations (1.4.1) of the identity $A \to A$ can be written as

$$\pi_1^{A,B}(\langle M, N\rangle) \equiv M \tag{1.4.4}$$

and of course we should also have

$$\pi_2^{A,B}(\langle M, N\rangle) \equiv N \tag{1.4.5}$$

Note that in these equalities we allow M and N to be arbitrary terms. Categorically speaking, therefore, we are asserting that the maps $X \to A \times B$ induced by the universal property of the product (the $\times I$ rule) do in fact have the desired composites with the projections.

The other half of the universal property is the uniqueness of maps into a product. This corresponds to a dual family of simplifications: we want to identify the following derivations of $A \times B \to A \times B$.

$$\dfrac{\dfrac{\overline{A \times B \vdash A \times B}}{A \times B \vdash A} \qquad \dfrac{\overline{A \times B \vdash A \times B}}{A \times B \vdash B}}{A \times B \vdash A \times B} \qquad\qquad \overline{A \times B \vdash A \times B}$$

In term syntax, this means that

$$\langle \pi_1^{A,B}(M), \pi_2^{A,B}(M)\rangle \equiv M \tag{1.4.6}$$

In type-theoretic lingo, the equalities (1.4.4) and (1.4.5) are called β**-conversion**[6] while the equality (1.4.6) is called an η**-conversion**.

[6] Presumably β-conversion is so named because it is the "second most basic" equivalence relation on terms, with α-equivalence (renaming of variables) being the first. However, there is a significant difference between the two: α-equivalent terms represent the *same* derivation, while β-conversion relates *distinct derivations* (though we generally notate them with terms).

$$\frac{x : X \vdash M : A \qquad x : X \vdash N : B}{x : X \vdash \pi_1^{A,B}(\langle M, N\rangle) \equiv M : A} \qquad \frac{x : X \vdash M : A \qquad x : X \vdash N : B}{x : X \vdash \pi_2^{A,B}(\langle M, N\rangle) \equiv N : B}$$

$$\frac{x : X \vdash M : A \times B}{x : X \vdash \langle \pi_1^{A,B}(M), \pi_2^{A,B}(M)\rangle \equiv M : A \times B} \qquad \frac{x : X \vdash M : \mathbb{1}}{x : X \vdash * \equiv M : \mathbb{1}}$$

$$\frac{x : X \vdash M : A}{x : X \vdash M \equiv M : A} \qquad \frac{x : X \vdash M \equiv N : A}{x : X \vdash N \equiv M : A}$$

$$\frac{x : X \vdash M \equiv N : A \qquad x : X \vdash N \equiv P : A}{x : X \vdash M \equiv P : A}$$

$$\frac{f \in \mathcal{G}(A, B) \qquad x : X \vdash M \equiv N : A}{x : X \vdash f(M) \equiv f(N) : B}$$

$$\frac{x : X \vdash M \equiv N : B \times C}{x : X \vdash \pi_1^{B,C}(M) \equiv \pi_1^{B,C}(N) : B} \qquad \frac{x : X \vdash M \equiv N : B \times C}{x : X \vdash \pi_2^{B,C}(M) \equiv \pi_2^{B,C}(N) : B}$$

$$\frac{x : X \vdash M \equiv M' : B \qquad x : X \vdash N \equiv N' : C}{x : X \vdash \langle M, N\rangle \equiv \langle M', N'\rangle : B \times C}$$

Figure 1.2: Equality rules for categories with products

For $\mathbb{1}$ there is no β-conversion rule, while the η-conversion rule is

$$* \equiv M \tag{1.4.7}$$

for any term $M : \mathbb{1}$. These conversions generate an equivalence relation on terms, which we also require to be a congruence for everything else.

We can describe this more formally with an additional judgment "$x : X \vdash M \equiv N : A$", with rules shown in Figure 1.2. Note that in addition to the β- and η-conversions, we assert reflexivity, symmetry, and transitivity, and also that all the previous rules preserve equality. As remarked in §1.2.1, all our equality judgments $\equiv$ will be equivalence relations with such a congruence property for all the primitive rules. In general we will not bother to state these "standard" rules for $\equiv$, but since this is our first encounter with such a relation involving "abstract variable" term syntax we have included them explicitly.

This completes the definition of the **unary type theory for categories with products under $\mathcal{G}$**.

Remark 1.4.8. Note that unlike the equalities $h \circ (g \circ f) \equiv (h \circ g) \circ f$ from §1.2.1, the β- and η-conversions are intutively "directional", with one side being "simpler" than the other. This suggests that we should be able to "reduce" an arbitrary term to a "simplest possible form" by successively applying β- and η-conversions. Such is indeed the case (although for technical reasons the η-conversion is usually applied in the less intuitive right-to-left direction and called an "expansion" rather than a "reduction"). This *process of reduction* (and expansion) belongs to the "computational" side of type theory, which (though of course important in its own right) is somewhat tangential to our category-theoretic emphasis, so we will not discuss it in detail.

Remark 1.4.9. Note that this type theory has three kinds of judgments (or "judgment forms"):

$$\vdash A\ \mathsf{type} \qquad\qquad x : A \vdash M : B \qquad\qquad x : A \vdash M \equiv N : B.$$

Categorically, these will represent the objects of a category, the morphisms of a category, and the equalities between those morphisms. We have discussed how in the morphism judgment $x : A \vdash M : B$, the term $x.M$ is an annotation that isomorphically represents a particular derivation of the un-annotated judgment $A \vdash B$. In the object judgment $\vdash A\ \mathsf{type}$, we can similarly regard A as a term annotation isomorphically representing a particular derivation of an un-annotatated judgment "$\vdash \mathsf{type}$". (We could thus write it as $\vdash A : \mathsf{type}$ or $A : (\vdash \mathsf{type})$, but we generally don't, to avoid confusion with elements of the "universe types" that will be introduced much later.)

By contrast, the equality judgment $x : A \vdash M \equiv N : B$ is the *un-annotated* version; we have not introduced any terms representing its derivations. This is because two morphisms in a category can't be "equal in more than one way", so there is never any reason to care which derivation of an equality judgment we used. (Of course, this perspective has to be modified when one moves on to *higher* category theory.) Note that the terms $x.M$ and $x.N$ annotating particular derivations of the morphism judgment appear in the un-annotated equality judgment, just as the terms A and B annotating particular derivations of the object judgment appear in the un-annotated morphism judgment $A \vdash B$.

Theorem 1.4.10. *Substitution is admissible in the unary type theory for categories with products under $\mathcal{G}$. That is, if we have derivations of $x : A \vdash M : B$ and $y : B \vdash N : C$, then we have a derivation of $x : A \vdash N[M/y] : C$.*

Proof. The proof is essentially the same as that of Theorem 1.3.7, but we write it out again explicitly with terms present. As always, we induct on the derivation of $y : B \vdash N : C$.

(a) If it ends with id, then we can use the given derivation M.

(b) If it ends swith fI for $f \in \mathcal{G}(C', C)$, then we have $y : B \vdash N' : C'$, so by induction we have $x : A \vdash N'[M/y] : C'$ and hence $x : A \vdash f(N'[M/y]) : C'$.

(c) If it ends with $\mathbb{1}I$, then by $\mathbb{1}I$ we have $x : A \vdash * : \mathbb{1}$ as well.

(d) If it ends with $\times E1$, then we have $y : B \vdash N' : C \times C'$, so by induction we have $x : A \vdash N'[M/y] : C \times C'$, hence $x : A \vdash \pi_1(N'[M/y]) : C$ by $\times E1$. The case for $\times E2$ is similar.

(e) Finally, if it ends with $\times I$, we have $y : B \vdash N_1 : C_1$ and $y : B \vdash N_2 : C_2$, so by induction we have $x : A \vdash N_1[M/y] : C_1$ and $x : A \vdash N_2[M/y] : C_2$, hence $x : A \vdash \langle N_1[M/y], N_2[M/y]\rangle : C_1 \times C_2$. □

As with Theorem 1.2.5, this proof can be regarded as *defining* recursively what it means to "substitute" M for y in N. The defining clauses are

$$\begin{aligned} N[M/y] &= M \\ (f(N))[M/y] &= f(N[M/y]) \\ *[M/y] &= * \\ (\pi_1(N))[M/y] &= \pi_1(N[M/y]) \\ (\pi_2(N))[M/y] &= \pi_2(N[M/y]) \\ \langle N_1, N_2\rangle[M/y] &= \langle N_1[M/y], N_2[M/y]\rangle \end{aligned}$$

We leave it to the reader to similarly prove the following (Exercise 1.4.3):

Lemma 1.4.11. *The relation $\equiv$ is a congruence for substitution in the unary type theory for categories with products under $\mathcal{G}$. In other words, if we have derivations of $x : X \vdash M \equiv M' : B$ and $y : B \vdash N \equiv N' : C$, then we can derive $x : A \vdash N[M/y] \equiv N'[M'/y] : C$.* □

Lemma 1.4.12. *Substitution is associative in the unary type theory for categories with products under $\mathcal{G}$: we have $P[N/z][M/y] = P[N[M/y]/z]$.* □

The proof of the initiality theorem is also similar, but we write out some of the details for later reference.

Theorem 1.4.13. *For any directed graph $\mathcal{G}$, the free category-with-products $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ it generates is described by the unary type theory for categories with products under $\mathcal{G}$: its objects are the A such that $\vdash A$ type is derivable, and its morphisms $A \to B$ are the terms M such that $x : A \vdash M : B$ is derivable, modulo the equivalence relation $\equiv$.*

Proof. Lemmas 1.4.11 and 1.4.12 and Theorem 1.4.10 show that we obtain a category with products in this way. Now given any other category with products $\mathcal{M}$ and a map $P : \mathcal{G} \to \mathcal{M}$, we proceed inductively:

(a) We define a map from types to the objects of $\mathcal{M}$ inductively on derivations of $\vdash A$ type, starting with P and then using the chosen products in $\mathcal{M}$.

(b) Then we define a map from terms to the morphisms of $\mathcal{M}$ inductively on derivations of $x : A \vdash M : B$, composing with the image of P for the generator rules, and using the universal properties of the finite products in $\mathcal{M}$ for the introduction and elimination rules.

(c) Then we define a map from derivations of $x : A \vdash M \equiv N : B$ to equalities of morphisms in $\mathcal{M}$ by induction on the former, using the laws of the universal properties of products in $\mathcal{M}$ and the fact that equality is an equivalence relation and a congruence for everything.

(d) Then we prove that this operation is functorial, i.e. it takes a substitution $N[M/x]$ to a composite in $\mathcal{M}$, by induction on the derivation of N (which is also, of course, how substitution is defined).

(e) Then we prove that this functor preserves (the specified) products, essentially by definition.

(f) Finally, we show that it is the unique such functor, since its definition was forced at every stage by the fact that it be a functor preserving products. □

Exercises

Exercise 1.4.1. Suppose we have

$$f \in \mathcal{G}(A,B) \qquad g \in \mathcal{G}(A,C) \qquad h \in \mathcal{G}(B,D) \qquad k \in \mathcal{G}(C,E)$$

Consider the following two derivations of $A \vdash D \times E$. Note that both use the admissible cut/substitution rule.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}}{A\vdash B}f \qquad \dfrac{\dfrac{}{B\vdash B}}{B\vdash D}h}{A\vdash D}\text{CUT} \qquad \dfrac{\dfrac{\dfrac{}{A\vdash A}}{A\vdash C}g \qquad \dfrac{\dfrac{}{C\vdash C}}{C\vdash E}k}{A\vdash E}\text{CUT}}{A\vdash D\times E}\times I$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}}{A\vdash B}f \qquad \dfrac{\dfrac{}{A\vdash A}}{A\vdash C}g}{A\vdash B\times C}\times I \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{B\times C\vdash B\times C}}{B\times C\vdash B}\times E1}{B\times C\vdash D}h \qquad \dfrac{\dfrac{\dfrac{}{B\times C\vdash B\times C}}{B\times C\vdash C}\times E2}{B\times C\vdash E}k}{B\times C\vdash D\times E}\times I}{A\vdash D\times E}\text{CUT}$$

Write down the terms corresponding to these two derivations and show directly that they are related by $\equiv$.

Exercise 1.4.2. Use the type theory for categories with products to prove that in any category with products we have

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

Note that since we are in categories now rather than posets, to show that two types A and B are isomorphic we must derive $x : A \vdash M : B$ and $y : B \vdash N : A$ and also show that their substitutions in both orders are equal (modulo $\equiv$) to identities.

Exercise 1.4.3. Prove Lemmas 1.4.11 and 1.4.12 (substitution is associative and respects $\equiv$ in the unary type theory for categories with products).

Exercise 1.4.4. A functor $P : \mathscr{A} \to \mathscr{M}$ is called a **fibration** if for any $b \in \mathscr{A}$ and $f : x \to P(b)$, there exists a morphism $\phi : a \to b$ in $\mathscr{A}$ such that $P(\phi) = f$ and ϕ is *cartesian*, meaning that for any $\psi : c \to b$ and $g : P(c) \to x$ such that $P(\psi) = fg$, there exists a unique $\chi : c \to a$ such that $P(\chi) = g$ and $\phi\chi = \psi$. The object c is denoted $f^*(b)$.

(a) Generalize your natural deduction for fibrations of posets from Exercise 1.3.8 to a type theory for fibrations of categories over a fixed base category $\mathscr{M}$, with β- and η-conversion $\equiv$ rules.

(b) Prove the initiality theorem for this type theory.

(c) Use this type theory to prove that in any fibration $P : \mathscr{A} \to \mathscr{M}$:

 (i) For any $f : x \to y$ in $\mathscr{M}$, f^* is a functor from the fiber over y to the fiber over x.

 (ii) For any $B \in \mathscr{A}$ and $x \xrightarrow{f} y \xrightarrow{g} P(B)$ in $\mathscr{M}$, we have $f^*(g^*(B)) \cong (gf)^*(M)$.

Exercise 1.4.5. Generalize Exercise 1.3.9 from posets to categories, combining your type theory from Exercise 1.4.4 with the one for categories with products from §1.4.

Exercise 1.4.6. The category **PrCat** is a 2-category whose 2-cells are arbitrary natural transformations (that is, there is no nonvacuous notion of a "product-preserving natural transformation"). Let $\mathcal{G}$ be a directed graph; as in Exercise 1.2.2, define a 2-functor $\mathbf{Gr}(\mathcal{G}, -) : \mathbf{PrCat} \to \mathbf{Cat}$, and show that $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ is a representing object for it. *(Use induction over the derivations of the judgments in its type-theoretic description.)*

Exercise 1.4.7. Exercises 1.2.2 and 1.4.6 address one worry that a category theorist might have about the strictness of our constructions. Another such worry is that the morphisms in **PrCat** preserve specified products *strictly*, while it is usually more natural in category theory to preserve products only up to isomorphism. This is not a problem if our main purpose is to have a syntax to describe objects and morphisms in particular categories with products; indeed, it is exactly what we would want. However, for abstract reasons it may be nice to also be able to say something about less strict functors.

With this in mind, prove that for any $\mathcal{G}$, the category with products $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ is *semi-flexible* in the sense of [?]: that is, if $\mathcal{M}$ has chosen products, then every functor $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G} \to \mathcal{M}$ that preserves products in the usual up-to-isomorphism sense is naturally isomorphic to a functor that preserves the chosen products strictly. *(Again, use induction over derivations in the type-theoretic description.)* Deduce that $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ satisfies a universal property relative to the 2-category of categories with products and functors that preserve them up to isomorphism.

Exercise 1.4.8. Here is another way to prove the result of Exercise 1.4.7.

(a) Use the initiality of $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ to show that if $\mathcal{M}$ has finite products and $Q : \mathcal{M} \to \mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ preserves finite products strictly, then any map of directed graphs $\mathcal{G} \to \mathcal{M}$ that lifts the inclusion $\mathcal{G} \to \mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ extends to a section $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G} \to \mathcal{M}$ of Q in **PrCat**.

(b) The results of [**?**] imply that the 2-category of categories with products and functors that preserve products in the usual up-to-isomorphism sense has 2-categorical limits called products, inserters, and equifiers, and the projections of these limits preserve products strictly. Use this, and (a), to prove that $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ satisfies a universal property relative to this 2-category.

1.5 Categories with coproducts

In Exercise 1.3.4, you obtained a sequent calculus for join-semilattices by dualizing the sequent calculus for meet-semilattices. However, natural deductions don't dualize as straightforwardly, due to the insistence that all rules act only on the right. (Of course, we could dualize them to "co-natural deductions" in which all rules act only on the *left*, but that would destroy the familiar behavior of terms on variables, as well as make it tricky to combine left and right universal properties, such as for lattices.) To describe joins in a natural deduction, we need to "build an extra cut" into their universal property:

$$\frac{X \vdash A}{X \vdash A \vee B}\ \vee I1 \qquad \frac{X \vdash B}{X \vdash A \vee B}\ \vee I2 \qquad \frac{X \vdash A \vee B \qquad A \vdash C \qquad B \vdash C}{X \vdash C}\ \vee E$$

Note that $\vee E$ is precisely the result of cutting $\vee L$ with an arbitrary sequent:

$$\frac{X \vdash A \vee B \qquad \dfrac{A \vdash C \qquad B \vdash C}{A \vee B \vdash C}\ \vee L}{X \vdash C}\ \text{CUT}$$

We treat the bottom element similarly:

$$\frac{X \vdash \bot \qquad \vdash C\ \mathsf{type}}{X \vdash C}$$

Rather than take the time to study join-semilattices, we skip directly to a **unary type theory for categories with coproducts** in which we care about distinct derivations. As usual, for this purpose we annotate the judgments with terms, as shown in Figure 1.3.

Most of the term operations are easy to guess. The two injections $A \to A+B$ and $B \to A+B$ are named inl and inr ("in-left" and "in-right"), while the unique morphism $\mathbf{0} \to C$ is called $\mathsf{match}_{\mathbf{0}}$. However, the term notation for $+E$ merits some discussion.

Recall from §1.4 that a term notation should indicate derivations of the premises of the rule by including terms for them, and that in general this can be tricky when using abstract variables because a derivation is only determined by a term together with its variable. The premises of $+E$ are, when paired explicitly

$$\frac{\vdash X\ \mathsf{type}}{x : X \vdash x : X}\ \mathsf{id} \qquad \frac{f \in \mathcal{G}(A,B) \qquad x : X \vdash M : A}{x : X \vdash f(M) : B}\ fI$$

$$\frac{x : X \vdash M : \mathbf{0} \qquad \vdash C\ \mathsf{type}}{x : X \vdash \mathsf{match}_{\mathbf{0}}(M) : C}\ \mathbf{0}E$$

$$\frac{x : X \vdash M : A}{x : X \vdash \mathsf{inl}(M) : A + B}\ {+I1} \qquad \frac{x : X \vdash N : B}{X \vdash \mathsf{inr}(N) : A + B}\ {+I2}$$

$$\frac{x : X \vdash M : A + B \qquad u : A \vdash P : C \qquad v : B \vdash Q : C}{x : X \vdash \mathsf{match}_{A+B}(M, u.P, v.Q) : C}\ {+E}$$

Figure 1.3: Unary type theory for categories with coproducts

with their variables, $x.M : (X \vdash A + B)$ and $u.P : (A \vdash C)$ and $v.Q : (B \vdash C)$, so the term notation should operate on all three of these. But unlike the case of $\times I$, now all three premises have different antecendents (and to emphasize this, we have used three different variables as well), so we cannot pull the same variable out of all of them.

However, since the antecedent of the conclusion is X, which is also the antecedent of the first premise, we can pull that variable (namely x) out to be the variable of the conclusion, leaving the other variables u and v paired with their terms. This leads us to $x.\mathsf{match}_{A+B}(M, u.P, v.Q)$. Note that the periods in $u.P$ and $v.Q$ bind more tightly than the commas, so this should be parsed as $x.\mathsf{match}_{A+B}(M, (u.P), (v.Q))$. The annotation by $A + B$ is to make type-checking possible (but often we will simplify it by writing just match_+).

The idea behind the name "match" is that to "evaluate" $\mathsf{match}_+(M, u.P, v.Q)$ the term $M : A + B$ should be compared to the "patterns" $\mathsf{inl}(u)$ and $\mathsf{inr}(v)$, and according to which one it "matches" (looks like), we branch into either P or Q. The notation $\mathsf{match}_{\mathbf{0}}(M)$ is the nullary case of this: the term $M : \mathbf{0}$ is matched against all possible ways that a term of type $\mathbf{0}$ could be constructed — of which there are none, and so there are no cases to consider.

Categorically, $+E$ expresses the universal property of the coproduct as follows. Recall that morphisms from A to B are (being derivations) represented by variable-term pairs. Thus the morphisms $A \to C$ and $B \to C$ are represented by the pairs $u.P$ and $v.Q$, while their copairing is a morphism $A + B \to C$ that should be represented by a term involving a variable $y : A + B$. This is exactly the data included in $y : A + B \vdash \mathsf{match}_+(y, u.P, v.Q) : C$; the general version with $x : X \vdash M : A + B$ just comes from building in a cut.

Note that the variables u and v, though they "appear in the term", are not the variable associated to the antecedent; we say they are **bound** variables. By contrast, the antecedent variable x is **free**. Textually each bound variable is associated to a subterm called its *scope*, delimiting the places where it can be referred to; in $+E$ the scope of u is P and the scope of v is Q. In terms of rules, this just means that the variable (or more precisely, its type) may appear as the antecedent of some, but not all, of the premises.

Bound variables are familiar in ordinary mathematics as well. For instance, the integration variable x in a definite integral $\int_0^2 x^2\,\mathrm{d}x$ is bound, because the value of the expression "doesn't depend on x"; its scope is the expression being integrated (but not the bounds of integration). Bound variables also occur in function definitions: given an expression such as x^2 depending on an unspecified variable x, we may write something like $(x \mapsto x^2)$ for "the function that squares its argument",[7] which is a fully defined object containing x as a bound variable.

It is a common convention in type theory that a prefixed variable followed by a period is bound. Our writing $x : X \vdash M : B$ as $x.M : (X \vdash B)$ follows this convention; the variable x is free in M, but bound in $x.M$, since when the term M is paired with its variable x it represents a derivation of $X \vdash B$ that does not depend on x.

If we were using de Bruijn variables, then all the variables x, u, v here would actually be the same, giving for instance $\mathsf{match}_{A+B}(M, x.f(x), x.g(x))$. Although technically fine, this can be confusing since the same unique variable x also occurs in M itself, but unrelatedly (and with a different type) than its occurrences in the case branches. In general, a bound variable "shadows" any free variable (or "outer" bound variable) of the same name, so that in $\mathsf{match}_{A+B}(M, x.f(x), x.g(x))$ the x in $f(x)$ refers to the prefix variable (called x) in $x.f(x)$, not to the outer free variable occurring in M (also called x). This makes a precise and general definition of α-equivalence in the presence of bound variables rather technically involved; one such definition is given in §A.6.

However, it is arguably bad mathematical style to use the same name for distinct variables, even if there is technically no ambiguity. For instance, we encourage calculus students to write $F(x) = \int_0^x f(t)\,\mathrm{d}t$ rather than $F(x) = \int_0^x f(x)\,\mathrm{d}x$, even though technically they mean the same thing. If we adhere to this informal convention, then we rarely need to worry about technical definitions of α-equivalence (unless we are trying to implement mathematics in a computer).

With all of this out of the way, we can now consider the appropriate β- and η-conversion rules. However, we pause first to prove admissibility of cut, i.e. to construct substitution, as in this case we will need substitution to state the β and η rules.

Lemma 1.5.1. *Substitution is admissible in the unary type theory for categories with coproducts under $\mathcal{G}$. That is, if we have derivations of $x : A \vdash M : B$ and $y : B \vdash N : C$, we can construct a derivation of $x : A \vdash N[M/y] : C$.*

Proof. By induction over derivations. As usual for natural deduction theories, there is not much happening: for each rule that might appear last in the derivation of $y : B \vdash N : C$, we apply the inductive hypothesis to its premises

[7] A more common notation for $(x \mapsto x^2)$ in type theory is $\lambda x.x^2$; see §2.8.

and then re-apply the final rule. The defining equations of the substitution operation are:

$$\begin{aligned} y[M/y] &= M \\ f(N)[M/y] &= f(N[M/y]) \\ \mathsf{match}_{\mathbf{0}}(N)[M/y] &= \mathsf{match}_{\mathbf{0}}(N[M/y]) \\ \mathsf{inl}(N)[M/y] &= \mathsf{inl}(N[M/y]) \\ \mathsf{inr}(N)[M/y] &= \mathsf{inr}(N[M/y]) \\ \mathsf{match}_{+}(N, u.P, v.Q)[M/y] &= \mathsf{match}_{+}(N[M/y], u.P, v.Q) \end{aligned}$$

□

Now we proceed to β- and η-conversion. Dualizing the β-conversion rules for products, the β-conversion rules for coproducts should say that the map $A + B \to C$ induced by $f : A \to C$ and $g : B \to C$ yields f and g when composed with the coproduct injections. Recalling that composition is given by substitution, this leads us to write down

$$\begin{aligned} \mathsf{match}_{+}(\mathsf{inl}(y), u.P, v.Q) &\equiv P[y/u] \\ \mathsf{match}_{+}(\mathsf{inr}(z), u.P, v.Q) &\equiv Q[z/v] \end{aligned}$$

Similarly, the η-conversion rule[8] should say that morphisms out of a coproduct are determined uniquely by their composites with the projections:

$$\mathsf{match}_{+}(y, u.P[\mathsf{inl}(u)/y], v.P[\mathsf{inr}(v)/y]) \equiv P \tag{1.5.2}$$

and similarly that morphisms out of the initial object are unique:

$$\mathsf{match}_{\mathbf{0}}(y) \equiv P.$$

In fact, to ensure that $\equiv$ is a congruence, we should "build in a cut" to all of these rules, so that the antecedent of the conclusion is an arbitrary type. Thus the actual generating $\equiv$ rules are those shown in Figure 1.4. As usual, we also require $\equiv$ to be an equivalence relation and a congruence for substitution. This completes the definition of our **unary type theory for categories with coproducts under $\mathcal{G}$**.

As in the dual case, by a **category with coproducts** we mean a category with specified binary coproducts and a specified initial object, and in the category of such the functors preserve the specified structure strictly.

Theorem 1.5.3. *For any directed graph $\mathcal{G}$, the free category with coproducts generated by $\mathcal{G}$ can be described by the unary type theory for categories with*

[8] In fact, for types such as coproducts with a left universal property, there is no consensus on exactly what equality "η" refers to. From a categorical point of view this equality is the most natural, since like the η-conversion rule for products it expresses the uniqueness aspect of the universal property. But sometimes η is used to refer only to the special case $\mathsf{match}_{+}(y, u.\mathsf{inl}(u), v.\mathsf{inr}(v)) \equiv y$, which is also analogous to the η rule for products in that it says that elements *of the type in question* have a canonical form (a pair in a product or a case-split in a coproduct). The stronger property is equivalent to the weaker property combined with "commuting conversions" such as $\mathsf{inl}(\mathsf{match}_{+}(M, u.P, v.Q)) \equiv \mathsf{match}_{+}(M, u.\mathsf{inl}(P), v.\mathsf{inl}(Q))$.

$$\frac{u : A \vdash P : C \qquad v : B \vdash Q : C \qquad x : X \vdash M : A}{x : X \vdash \mathsf{match}_+(\mathsf{inl}(M), u.P, v.Q) \equiv P[M/u] : C}$$

$$\frac{u : A \vdash P : C \qquad v : B \vdash Q : C \qquad x : X \vdash N : B}{x : X \vdash \mathsf{match}_+(\mathsf{inr}(N), u.P, v.Q) \equiv Q[N/v] : C}$$

$$\frac{x : X \vdash M : A + B \qquad y : A + B \vdash P : C}{x : X \vdash \mathsf{match}_+(M, u.P[\mathsf{inl}(u)/y], v.P[\mathsf{inr}(v)/y]) \equiv P[M/y] : C}$$

$$\frac{x : X \vdash M : \mathbf{0} \qquad y : \mathbf{0} \vdash P : C}{x : X \vdash \mathsf{match}_{\mathbf{0}}(M) \equiv P[M/y] : C}$$

Figure 1.4: β- and η-conversions for categories with coproducts

coproducts under $\mathcal{G}$*: its objects are the derivations of* $\vdash A$ type*, and its morphisms* $A \to B$ *are the derivations of* $A \vdash B$ *(or equivalently the derivable term judgments* $x : A \vdash M : B$ *modulo* α*-equivalence), modulo the equivalence relation* $\equiv$*.*

Proof. This is basically just like the proof of Theorem 1.4.13, with one slight twist. Lemma 1.5.1 defines substitution, and the same sort of induction proves it associative and unital; thus we have a category $\mathfrak{F}_{\mathbf{CoprCat}}\mathcal{G}$. The rules are defined just so as to give this category the structure of coproducts. Then, given any category with coproducts $\mathcal{M}$ and map of graphs $\omega : \mathcal{G} \to \mathcal{M}$, we extend ω to a unique coproduct-preserving functor $\overline{\omega} : \mathfrak{F}_{\mathbf{CoprCat}}\mathcal{G} \to \mathcal{M}$ by successive inductions over all the derivations of the type theory.

The twist is that we have to be careful about the order in which we do this. Because the rules for $\equiv$ involve the substitution *operation* on terms, to interpret these rules using the universal properties in $\mathcal{M}$ we need to know already that substitution maps to composition in $\mathcal{M}$. Thus we need to (1) extend ω to types, (2) extend ω to terms, (3) *then* prove that this extension $\overline{\omega}$ takes substitution to composition, (4) *then* inductively show that derivations of $x : A \vdash M \equiv N : B$ map to equal morphisms, and (5) then complete the proof that we have a functor preserving coproducts. □

Exercises

Exercise 1.5.1. This is the dual of Exercise 1.4.1, though of course its proof is not dual. Suppose we have

$$f \in \mathcal{G}(A, C) \qquad g \in \mathcal{G}(B, D) \qquad h \in \mathcal{G}(C, E) \qquad k \in \mathcal{G}(D, E)$$

Here is one (cut-free) derivation of $A + B \vdash E$.

$$\dfrac{\dfrac{}{A+B \vdash A+B} \qquad \dfrac{\dfrac{\dfrac{}{A \vdash A}}{A \vdash C}\,f}{A \vdash E}\,h \qquad \dfrac{\dfrac{\dfrac{}{B \vdash B}}{B \vdash D}\,g}{B \vdash E}\,k}{A+B \vdash E}\,+E$$

Write down another derivation of $A + B \vdash E$ that ends with the following cut:

$$\dfrac{\dfrac{\vdots}{A+B \vdash C+D} \qquad \dfrac{\vdots}{C+D \vdash E}}{A+B \vdash E}\,\textsc{cut}$$

Then write down the terms corresponding to the two derivations and show directly that they are related by $\equiv$.

Exercise 1.5.2. A functor $P : \mathscr{A} \to \mathscr{B}$ is called an **opfibration** if $P^{\mathrm{op}} : \mathscr{A}^{\mathrm{op}} \to \mathscr{M}^{\mathrm{op}}$ is a fibration (as in Exercise 1.4.4). The dual of $f^*(b)$ is written $f_!(a)$.

(a) Write down a type theory for opfibrations and prove the initiality theorem. (Remember that we always use natural deduction style when dealing with categories rather than posets, so you can't just dualize Exercise 1.4.4 or categorify Exercise 1.3.10. You will probably want a term syntax such as "$\mathsf{match}_!$".)

(b) Use this type theory to prove that $f_!$ is always a functor.

1.6 Universal properties and modularity

This seems an appropriate place to introduce some more type-theoretic lingo. Roughly speaking, types corresponding to objects with "mapping out" universal properties, such as $A + B$, are called **positive**, while types corresponding to objects with "mapping in" universal properties, such as $A \times B$, are called **negative**. The precise meanings of these terms relate to "focusing" and are more directly applicable to sequent calculus than to natural deduction, but they are often used informally in this broad sense, and we will sometimes do the same.

Another important thing to note is a similarity in structure between the product types of §1.4 and the coproduct types of this section. Both of them augment the basic cut-free type theory for categories from §1.2.2 with four kinds of rules:

(a) **Formation** rules for the judgment $\vdash A\ \mathsf{type}$, which tell us how to "form" the new types described by this operation (product types or coproduct types):

$$\dfrac{\vdash A\ \mathsf{type} \qquad \vdash B\ \mathsf{type}}{\vdash A \times B\ \mathsf{type}} \qquad\qquad \dfrac{\vdash A\ \mathsf{type} \qquad \vdash B\ \mathsf{type}}{\vdash A + B\ \mathsf{type}}$$

(b) **Introduction** rules for the term judgment, which tell us how to "introduce" terms belonging to such a new type:

$$\frac{x : X \vdash M : B \qquad x : X \vdash N : C}{x : X \vdash \langle M, N\rangle : B \times C}\times I$$

$$\frac{x : X \vdash M : A}{x : X \vdash \mathsf{inl}(M) : A + B} +I1 \qquad \frac{x : X \vdash N : B}{X \vdash \mathsf{inr}(N) : A + B} +I2$$

(c) **Elimination** rules for the term judgment, which tell us how to "eliminate" a term belonging to a new type to obtain terms belonging to other types:

$$\frac{x : X \vdash M : B \times C}{x : X \vdash \pi_1^{B,C}(M) : B}\times E1 \qquad \frac{x : X \vdash M : B \times C}{x : X \vdash \pi_2^{B,C}(M) : C}\times E2$$

$$\frac{x : X \vdash M : A + B \qquad u : A \vdash P : C \qquad v : B \vdash Q : C}{x : X \vdash \mathsf{match}_{A+B}(M, u.P, v.Q) : C} +E$$

(d) **Conversion** (or *computation*) rules for the equality judgment, which tell us how to "convert" or "reduce" terms built by combining the introduction and elimination rules:

$$\pi_1^{A,B}(\langle M, N\rangle) \equiv M \qquad \pi_2^{A,B}(\langle M, N\rangle) \equiv N \qquad \langle \pi_1^{A,B}(M), \pi_2^{A,B}(M)\rangle \equiv M$$

$$\mathsf{match}_+(\mathsf{inl}(y), u.P, v.Q) \equiv P[y/u] \qquad \mathsf{match}_+(\mathsf{inr}(z), u.P, v.Q) \equiv Q[z/v]$$

$$\mathsf{match}_+(y, u.P[\mathsf{inl}(u)/y], v.P[\mathsf{inr}(v)/y]) \equiv P$$

This division into four groups of rules for each type operation corresponds to the following four aspects of a category-theoretic universal property:

(a) The existence of one or more objects given certain input data (formation).

(b) Some data relating those objects to others that we already had, like the projections of a product or the injections of a coproduct (elimination for negative types, introduction for positive ones).

(c) A "factorization" operation: given data of some sort, there exist specified morphism(s) into or out of the new objects (introduction for negative types, elimination for positive ones).

(d) These specified morphism(s) satisfy some compatibility conditions with the given data that uniquely determine them (conversion).

This provides a template for representing other category-theoretic universal properties with type theory. (What if we want to represent something that doesn't have a universal property, like a monoidal structure? As we will see in chapter 2, the answer is to move into a different world where it *does* have a universal property.)

Finally, I want to emphasize the *modularity* of these rules (in the computer scientist's sense). When we added the rules for products to the cut-free type theory for categories, the proofs of the basic theorems like cut admissibility, associativity of substitution, and the initiality theorem did not change in their overall structure. They were all still proved by induction on derivations; we simply had to add new clauses to each induction step corresponding to the new rules.

The advantage of this sort of modularity means that we are free to "mix and match" rules of our type theory, corresponding to the categorical universal properties we want to have. There will be more scope for variety with this later, but at the moment there is one new combination: we can obtain a **type theory for categories with products and coproducts** by simply combining the rules from §§1.4 and 1.5. We can then prove all the same theorems, including the fact that this type theory presents the free category with products and coproducts on a directed graph, by simply combining the relevant clauses from all the proofs in §§1.4 and 1.5.

Exercises

Exercise 1.6.1. Suppose we have

$$f \in \mathcal{G}(A, C) \qquad g \in \mathcal{G}(A, D) \qquad h \in \mathcal{G}(B, C) \qquad k \in \mathcal{G}(B, D)$$

Write down two different derivations of $A + B \vdash C \times D$ in the type theory for categories with products and coproducts under $\mathcal{G}$, one that ends with $\times I$ and one that ends with $+E$. Then write down the corresponding terms and show directly that they are identified by $\equiv$.

Exercise 1.6.2. A functor $P : \mathscr{A} \to \mathscr{B}$ is called a **bifibration** if it is both a fibration and an opfibration.

(a) Combine the theories of Exercises 1.4.4 and 1.5.2 to obtain a type theory for bifibrations.

(b) If you aren't tired of proving initiality theorems yet, do it for this type theory.

(c) Use this type theory to prove that in any bifibration, $f_!$ is left adjoint to f^*.

1.7 Presentations and theories

Now that we have a type theory for categories with products, we might hope that we could express in it the proof from §0.1 of uniqueness of inverses for monoid objects. However, the theory as presented in §1.4 is inadequate even to *talk* about monoid objects!

Let us recall briefly how we use initiality theorems to deduce categorical facts. Suppose we want to prove a theorem of the form "for any objects $A, B, C, \ldots$ and morphisms $f, g, h, \ldots$ in a category with products, we have ...". Then we let $\mathcal{G}$ be a directed graph with one vertex for each object appearing in the theorem and one edge for each morphism, with appropriate source and target, and we reason in the type theory for categories with products under $\mathcal{G}$. Then whenever we have objects and morphisms of the sort described in the theorem in any other category $\mathcal{M}$ with products, we get a map $\mathcal{G} \to \mathcal{M}$, which therefore extends to the free category $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ presented by the type theory, and this extension carries our type-theoretic constructions and proofs to categorical ones.

However, although theorems about monoid objects are intuitively of the form "for any object A and morphisms $m, e, \ldots$", they do not fit into this picture, because the data cannot be described as a directed graph. The problem is that the source and target of m and e are not *single* objects mentioned in the theorem, but products of them (specifically, $m : A \times A \to A$ and $e : \mathbb{1} \to A$). Furthermore, a monoid object contains not just objects and morphisms, but *axioms* about composites of those morphisms, which the type theory of §1.4 is also unable to deal with.

We now present an extension of this theory which does suffice to discuss monoid objects. It is not the final word on the matter — as we will see, the type theories of chapter 2 can do somewhat better — but it is a first step.

The two problems mentioned above are in fact quite similar. To talk about a morphism $m : A \times A \to A$, say, using type theory, we need to replace directed graphs with some more general structure including "arrows" whose source and target can be products of "generating objects" rather than just single ones. Similarly, to talk about an axiom such as $m(x, m(y, z)) = m(m(x, y), z)$, we need to also include "generating equalities" that relate "pairs of parallel morphisms", and these morphisms could also be products and composites of generating ones. (This is an instance of the higher-categorical philosophy that equalities are just a kind of higher morphism.)

This sort of refinement can be performed for essentially any type theory; it is a sort of syntactic version of the categorical notion of "computads" and related structures [**?**, **?**, **?**]. The idea as just sketched is fairly straightforward, but the execution is a bit complicated, because we have to define the sort of "generating structure" (a generalization of a directed graph) step-by-step in parallel with the type theory that it generates. Thus, we begin with a couple of simpler cases.

1.7.1 Group presentations

Categorically, what we are about to do is talking about *presented* categories rather than *free* ones. However, the way we're doing it is perhaps not the most common way to talk about "presentations" in category theory. Consider for instance the most well-known sort of presentation, namely a presentation of a group; this consists of a set X of generating elements together with a set R

of "relations", each of which is a pair[9] of elements of the free group on X, i.e. $R \subseteq \mathfrak{F}X \times \mathfrak{F}X$. For instance, the dihedral group D_n is presented by $X = \{ x, y \}$ and $R = \{ (x^n, e), (y^2, e), (xyxy, e) \}$.

The question is, exactly what group does a group presentation present, and what is its universal property? The more common way to answer this question category-theoretically is to consider the two projections $R \rightrightarrows \mathfrak{F}X$, note that by the universal property of free groups they yield two group homomorphisms $\mathfrak{F}R \rightrightarrows \mathfrak{F}X$, and define the presented group $\langle X \mid R \rangle$ to be the coequalizer of these two morphisms in the category of groups. While this works, it requires a bit of unraveling to extract the universal property of $\langle X \mid R \rangle$.

When we come to dependent type theories, we will be forced to use a procedure like this, but until then we can package the universal property of a presented object more explicitly using an adjunction. In the case of groups, what we would do is define the *category of group presentations*, whose objects are presentations (X, R) as above, and whose morphisms $(X, R) \to (Y, S)$ are functions $g : X \to Y$ such that $R \subseteq (\mathfrak{F}g \times \mathfrak{F}g)^{-1}(S)$, i.e. each relation in R is mapped by g to a relation in S. Then we define, for each group G, its *underlying* or *canonical* presentation (G, K_G), in which the set of generators is the underlying set of G itself, and K_G is the "kernel pair" of the counit $\mathfrak{F}G \to G$ of the free-group adjunction (i.e. its pullback against itself). More explicitly, K_G is the set of *all* pairs (w_1, w_2) of elements of $\mathfrak{F}G$ that when "multiplied out" become equal in G.

Now the group presented by a group presentation is simply its image under the left adjoint to this forgetful functor from groups to presentations. This gives an explicit description of its universal property: a map of presentations $(X, R) \to (G, K_G)$ is, by definition, a function $X \to G$ such that the image of each relation in R is "true" in G, i.e. the composites $R \rightrightarrows \mathfrak{F}X \to \mathfrak{F}G \to G$ are equal.

Before we move on to categories, we note a few things about this construction. Firstly, this adjunction subsumes the adjunction between groups and sets, because any set X gives rise to a group presentation $(X, \emptyset)$ with no relations, and the corresponding presented group is just the free group on X. However, we needed to *already have* the simpler adjunction between groups and sets in order to construct this more refined adjunction between groups and presentations; we used free groups in the definition of a group presentation, and we used the universal property of free groups in defining the underlying presentation of a group.

Secondly, the underlying presentation of a group actually *presents that group*, i.e. the counit of the adjunction between presentations and groups is an isomorphism. Equivalently, by general categorical facts, the "underlying presentation" functor from groups to presentations is fully faithful. Thus, if we wanted to, we could regard groups as "being" certain group presentations whose relations are "saturated" in a suitable sense, with the adjunction exhibiting such

[9]In the case of groups, an equality $g = h$ is equivalent to an equality $gh^{-1} = e$, so it is common to take the relations to be single elements of the free group on X rather than pairs thereof. However, when we generalize to other algebraic structures this is no longer true, so we stick with the more general version even for groups.

"groups" as a reflective subcategory of group presentations.

On the other hand, this also means that the left adjoint from presentations to groups is essentially surjective. Therefore, if we define a new category whose objects are group presentations and whose morphisms are the group homomorphisms between the groups they present, we obtain a category equivalent to the category of groups. A slightly less tautological-sounding definition of the morphisms in this category uses the adjunction: a group homomorphism from $\langle X \mid R\rangle$ to $\langle Y \mid S\rangle$ is the same as a morphism of presentations from (X, R) to the underlying presentation of $\langle Y \mid S\rangle$. Such a thing can be described a bit more explicitly: it is a function $X \to \mathfrak{F}Y$ which maps each relation in R to a relation that holds in $\langle Y \mid S\rangle$.

More categorically, the observation is that any adjunction $F : \mathcal{C} \rightleftarrows \mathcal{D} : G$ whose counit is an isomorphism is "opmonadic", i.e. exhibits $\mathcal{D}$ as equivalent to the Kleisli category of the induced monad GF on $\mathcal{C}$. (The fact that $\mathcal{D}$ is equivalent to a reflective subcategory of $\mathcal{C}$ implies that the adjunction is also monadic, with the monad GF being idempotent. Moreover, every algebra for an idempotent monad is free, so the Kleisli and Eilenberg–Moore categories are equivalent.)

Most of these observations have analogues in the type-theoretic situation, to which we now turn.

1.7.2 Category presentations

We begin with presentations of categories, which are formally quite similar to presentations of groups, except for the presence of "many objects" and the absence of inverses. However, rather than taking the "free category on a directed graph" as a black box, the way we did for the free group on a set in §1.7.1, we will use its explicit presentation using the cut-free type theory of §1.2.2. This enables an analogous type-theoretic presentation of the left adjoint constructing a presented category from a presentation.

Recall from Theorem 1.2.13 that the free category $\mathfrak{F}_{\mathbf{Cat}}\mathcal{G}$ generated by a directed graph $\mathcal{G}$ has the same objects as $\mathcal{G}$ and its morphisms are the derivable term judgments $x : A \vdash M : B$ (i.e. derivations of $A \vdash B$) in the cut-free type theory for categories under $\mathcal{G}$.

Definition 1.7.1. A **category presentation** $\mathcal{P}$ consists of a directed graph $\mathcal{P}_1$ together with a set $\mathcal{P}_2$ of *generating equalities*, each of which is a pair of two derivations of $A \vdash B$ for some $A, B \in \mathcal{G}$.

Equivalently, each generating equality is two terms M, N such that $x : A \vdash M : B$ and $x : A \vdash N : B$ are both derivable, or (more category-theoretically) two parallel morphisms in the free category on $\mathcal{G}$.

Definition 1.7.2. For a category $\mathcal{C}$, its **underlying presentation** is the underlying directed graph of $\mathcal{C}$ with, as generating equalities, the set of *all* parallel pairs of morphisms in $\mathfrak{F}_{\mathbf{Cat}}\mathcal{C}$ that become equal after applying the counit functor $\mathfrak{F}_{\mathbf{Cat}}\mathcal{C} \to \mathcal{C}$.

$$\frac{x : X \vdash M : A}{x : X \vdash M \equiv M : A} \qquad \frac{x : X \vdash M \equiv N : A}{x : X \vdash N \equiv M : A}$$

$$\frac{x : X \vdash M \equiv N : A \qquad x : X \vdash N \equiv P : A}{x : X \vdash M \equiv P : A}$$

$$\frac{f \in \mathcal{P}_1(A, B) \qquad x : X \vdash M \equiv N : A}{x : X \vdash f(M) \equiv f(N) : B}$$

$$\frac{x : X \vdash M : A \qquad \mathcal{P}_2(y : A, B)(N, P)}{x : X \vdash N[M/y] \equiv P[M/y] : A}$$

Figure 1.5: Equality rules for category presentations

For a category presentation $\mathcal{P}$, we write $\mathcal{P}_2(x : A, B)(M, N)$ to mean that $(M, N) \in \mathcal{P}_2$ where both $x : A \vdash M : B$ and $x : A \vdash N : B$. Now we define the **type theory for categories under** $\mathcal{P}$ to consist of the cut-free type theory for categories under the directed graph $\mathcal{P}_1$ together with the rules for an equality judgment shown in Figure 1.5. Recall that the cut-free type theory for categories didn't need an equality judgment on its own, so the only rules other than the "generator" one (the last one) are reflexivity, symmetry, transitivity, and congruence. (As mentioned in §1.4, we often omit these rules, but we include them here because we are doing something a bit new with an equality judgment.)

The generator rule basically says that each generating equality yields an actual equality. Note that we have built in a substitution on the left. This ensures that $\equiv$ remains a congruence for substitution on both sides as in Lemma 1.4.11 (the other side is ensured by the primitive congruence rules for $\equiv$).

Theorem 1.7.3. *For any category presentation $\mathcal{P}$, the free category generated by $\mathcal{P}$ (i.e. the image of $\mathcal{P}$ under the left adjoint to the underlying-presentation functor) is described by the type theory for categories under $\mathcal{P}$: its objects are those of $\mathcal{P}$, and its morphisms are the derivations of $A \vdash B$ modulo the equivalence relation $\equiv$.*

Proof. As usual, the rules for equality ensure that it is an equivalence relation and a congruence, so that the quotient is a category. Now suppose given a category $\mathcal{C}$ and a morphism from $\mathcal{P}$ to the underlying presentation of $\mathcal{C}$. Then Theorem 1.2.13 gives a unique functor from the free category on $\mathcal{P}_1$ to $\mathcal{C}$, so it suffices to check that this functor descends to the quotient by $\equiv$. This is a straightforward induction over derivations of $\equiv$. All the basic rules of equality are always true in $\mathcal{C}$, while the generator one is true since $\mathcal{P} \to \mathcal{C}$ is a morphism of presentations. □

As in §1.7.1, the counit of this adjunction is easily shown to be an isomorphism. Thus we can regard categories as a reflective subcategory of category presentations, or we can show that the category of categories is equivalent to the Kleisli category of the induced monad on category presentations.

1.7.3 $\times$-presentations

Now we move on to the case of categories with products. Here there are three stages rather than two, since we have to add the operations on objects as well. Thus, instead of two adjunctions (with base categories of directed graphs and category presentations, respectively) we will have three. For the moment, we will call the objects of the base categories ***k*-skeletal (unary) $\times$-presentations** for $k = 0, 1, 2$; in §1.7.4 we will introduce some more standard terminology.

Definition 1.7.4. A **0-skeletal $\times$-presentation** is a set, $\mathcal{P}_0$.

The type theory of a 0-skeletal $\times$-presentation consists of the rules for type judgments from §1.4:

$$\frac{A \in \mathcal{P}_0}{\vdash A\ \mathsf{type}} \qquad \frac{}{\vdash \mathbb{1}\ \mathsf{type}} \qquad \frac{\vdash A\ \mathsf{type} \qquad \vdash B\ \mathsf{type}}{\vdash A \times B\ \mathsf{type}}$$

We write $\mathfrak{F}_0\mathcal{P}_0$ for the set of derivable types in this theory (later on we will see that it is indeed some kind of free thing generated by $\mathcal{P}_0$).

Definition 1.7.5. A **1-skeletal $\times$-presentation** consists of a 0-skeletal $\times$-presentation $\mathcal{P}_0$ (its *0-skeleton*) together with a set of *arrows* $\mathcal{P}_1$, each of which is assigned a source and a target that are types in the type theory of $\mathcal{P}_0$.

Thus, for instance, a 1-skeletal $\times$-presentation could contain objects A, B and arrows $f : A \times B \to B \times B$ and $g : \mathbb{1} \to A$.

The type theory of a 1-skeletal $\times$-presentation consists of the rules for term judgments $x : A \vdash M : B$ from §1.4. The generator rule

$$\frac{x : X \vdash M : A \qquad f \in \mathcal{P}_1(A, B)}{x : X \vdash f(M) : B}\ f$$

looks the same as before, but now it no longer implies as a side condition that A and B must be base types. We prove exactly as before that terms have unique derivations and that substitution is admissible and associative. We write $\mathfrak{F}_1\mathcal{P}_1(A, B)$ for the set of derivations/terms $x : A \vdash M : B$ in this theory.

Definition 1.7.6. A **2-skeletal $\times$-presentation** consists of a 1-skeletal $\times$-presentation $\mathcal{P}_{\le 1}$ (its *1-skeleton*) together with a set of *generating equalities* $\mathcal{P}_2$, each of which is a pair of derivations of the same judgment $A \vdash B$ in the type theory of $\mathcal{P}_{\le 1}$.

Since this is the last step, we sometimes omit the adjective "2-skeletal" and just speak of **(unary) $\times$-presentations**. We write $\mathcal{P}_2(x : A, B)(M, N)$ to mean that (M, N) is a generating equality, where $x : A \vdash M : B$ and $x : A \vdash N : B$.

The type theory of a $\times$-presentation consists of the rules for the equality judgment $\equiv$ from §1.4, together with a generator rule for equalities:

$$\frac{x : X \vdash M : A \qquad \mathcal{P}_2(y : A, B)(P, Q)}{x : X \vdash P[M/y] \equiv Q[M/y] : B} \tag{1.7.7}$$

As before, we have built in a substitution on the left.

We once again prove all the expected properties, like type-checking and cut admissibility, just as we did for the ordinary type theory for categories with products in §1.4. However, stating and proving the initiality theorem is somewhat more complicated.

Of course, generalizing from the situation of §1.7.2 we expect to have *three* adjunctions, whose base categories consist of k-skeletal $\times$-presentations for $k = 0, 1, 2$. However, unlike in §1.7.2 the other category cannot be **PrCat** for all three adjunctions, since the type theories of 0-skeletal and 1-skeletal $\times$-presentations are not yet categories with products. A 0-skeletal $\times$-presentation does not even give rise directly to a category; we could make it one by adding identity morphisms, but it still wouldn't have products. And a 1-skeletal $\times$-presentation does give a category, but without the β- and η-conversion laws (which only appear at the final stage with the equality judgment) the structure it has is weaker than having products.

To start with, in order to have adjunctions, we need categories of $\times$-presentations.

Definition 1.7.8.

(a) A morphism of 0-skeletal $\times$-presentations is just a function $K_0 : \mathcal{P}_0 \to \mathcal{Q}_0$. This induces a function $\overline{K}_0 : \mathfrak{F}_0\mathcal{P}_0 \to \mathfrak{F}_0\mathcal{Q}_0$ by a simple induction.

(b) A morphism of 1-skeletal $\times$-presentations is a morphism $K_0 : \mathcal{P}_0 \to \mathcal{Q}_0$ on 0-skeleta, together with a function $K_1 : \mathcal{P}_1 \to \mathcal{Q}_1$ respecting sources and targets (relative to $\overline{K}_0$). This induces maps $\overline{K}_1 : \mathfrak{F}_1\mathcal{P}_1(A, B) \to \mathfrak{F}_1\mathcal{Q}_1(\overline{K}_0(A), \overline{K}_0(B))$ by another simple induction.

(c) Finally, a morphism of 2-skeletal $\times$-presentations is a morphism $K_{\le 1} : \mathcal{P}_{\le 1} \to \mathcal{Q}_{\le 1}$ on 1-skeleta, together with a function $K_2 : \mathcal{P}_2 \to \mathcal{Q}_2$ respecting sources and targets (relative to $\overline{K}_1$).

We write $\mathbf{Pres}_{\times,k}$ for the category of k-skeletal $\times$-presentations.

Now, since $\times$-presentations are defined inductively on their skeleta, it is most natural to prove the initiality theorem in stages as well. For this purpose we need skeletal versions of "categories with products" as well. The idea is that "1-skeletal categories with products", for instance, should have exactly the categorical structure induced by the type and term judgments *without* any equality judgments. In particular, they have pairings and projections, but these don't satisfy the β- and η-conversion laws. The pairings do, however, have to be *natural* in the domain, since that comes from substitution which is defined and associative already on terms before we have any $\equiv$.

Definition 1.7.9.

(a) A **0-skeleton for a category with products** is a set $\mathcal{M}_0$ with a specified element $\mathbb{1} \in \mathcal{M}_0$ and a binary operation $(- \times -) : \mathcal{M}_0 \times \mathcal{M}_0 \to \mathcal{M}_0$.

(b) A **1-skeleton for a category with products** is

 (i) A 0-skeleton for a category with products, $\mathcal{M}_0$;

 (ii) A category $\mathcal{M}$ with $\mathcal{M}_0$ as its set of objects;

 (iii) For every $A, B \in \mathcal{M}_0$, morphisms $A \times B \to A$ and $A \times B \to B$;

 (iv) For every $A, B \in \mathcal{M}_0$, a natural transformation $\mathcal{M}(-, A) \times \mathcal{M}(-, B) \to \mathcal{M}(-, A \times B)$; and

 (v) A natural transformation $1 \to \mathcal{M}(-, \mathbb{1})$, where 1 denotes the terminal presheaf.

(c) A **2-skeleton for a category with products** is just a category with products.

We write $\mathbf{PrCat}_k$ for the category of k-skeleta for a category with products (in which we trust the reader to define the morphisms).

Remark 1.7.10. The morphisms $A \times B \to A$ and $A \times B \to B$ in a 1-skeleton for a category with products could also be expressed, by the Yoneda embedding, as natural transformations $\mathcal{M}(-, A \times B) \to \mathcal{M}(-, A)$ and $\mathcal{M}(-, A \times B) \to \mathcal{M}(-, B)$. This would make them look more like the other two operations in such a 1-skeleton, and moreover would correspond more directly to the type-theoretic rules for projections:

$$\frac{x : X \vdash M : A \times B}{x : X \vdash \pi_1(M) : A} \qquad \frac{x : X \vdash M : A \times B}{x : X \vdash \pi_2(M) : B}$$

Now, we will construct the following ladder of adjunctions, in which the both the left and right adjoints commute with the obvious downward-pointing forgetful functors:

$$\begin{array}{ccc}
\mathbf{Pres}_{\times,2} & \underset{\mathfrak{U}_2}{\overset{\mathfrak{F}_2}{\rightleftarrows}}\ (\bot) & \mathbf{PrCat}_2 \\
\downarrow & & \downarrow \\
\mathbf{Pres}_{\times,1} & \underset{\mathfrak{U}_1}{\overset{\mathfrak{F}_1}{\rightleftarrows}}\ (\bot) & \mathbf{PrCat}_1 \\
\downarrow & & \downarrow \\
\mathbf{Pres}_{\times,0} & \underset{\mathfrak{U}_0}{\overset{\mathfrak{F}_0}{\rightleftarrows}}\ (\bot) & \mathbf{PrCat}_0
\end{array}$$

In fact, we will define all the horizontal functors one by one from bottom to top: first $\mathfrak{U}_0$, then $\mathfrak{F}_1$, then $\mathfrak{U}_1$, etc, and show one step at a time that we get adjunctions.

Definition 1.7.11. The forgetful functor $\mathfrak{U}_0 : \mathbf{PrCat}_0 \to \mathbf{Pres}_{\times,0}$ takes a 0-skeleton for a category with products to its underlying set.

Theorem 1.7.12. *The left adjoint $\mathfrak{F}_0$ of $\mathfrak{U}_0$ takes a 0-skeletal $\times$-presentation $\mathcal{P}_0$ to the set of types $\vdash A$ type derivable in its type theory.* □

Definition 1.7.13. The forgetful functor $\mathfrak{U}_1 : \mathbf{PrCat}_1 \to \mathbf{Pres}_{\times,1}$ acts as $\mathfrak{U}_0$ on underlying 0-skeleta, and for types A, B in the type theory of $\mathfrak{U}_0\mathcal{M}$ we define $\mathfrak{U}_1(A, B)$ to be the set of morphisms in $\mathcal{M}$ between the images of A and B under the counit map $\mathfrak{F}_0\mathfrak{U}_0\mathcal{M}_0 \to \mathcal{M}_0$.

Theorem 1.7.14. *The left adjoint $\mathfrak{F}_1$ of $\mathfrak{U}_1$ acts as $\mathfrak{F}_0$ on 0-skeleta, and the morphisms in $\mathfrak{F}_1\mathcal{P}$ are the derivations $x : A \vdash M : B$ in the type theory of $\mathcal{P}$.*

Proof. We sketch the proof, with the goal of explaining the naturality requirement in the definition of 1-skeleta for categories with products. First we have to show that $\mathfrak{F}_1\mathcal{P}$ is indeed such a 1-skeleton. Note that the proofs in §1.4 that substitution/cut is associative and unital as an operation on terms/derivations *before* we impose any equality judgment; thus the present $\mathfrak{F}_1\mathcal{P}$ is a category. Moreover, the term operations π_1, π_2, $\langle -, - \rangle$, and $*$ give $\mathfrak{F}_1\mathcal{P}$ the structure of a 1-skeleton for a category with products. The *naturality* of these operations is simply the fact that they commute with substitution, e.g. $\langle P, Q\rangle[M/x] = \langle P[M/x], Q[M/x]\rangle$. Since this is true essentially *by definition of substitution*, it is a literal equality of terms/derivations.

Now let $\mathcal{C}$ be any 1-skeleton for a category with products, and let $\omega : \mathcal{P} \to \mathfrak{U}_1\mathcal{C}$ be a morphism of 1-skeletal $\times$-presentations. We extend it to a map $\overline{\omega}_0 : \mathfrak{F}_0\mathcal{P}_0 \to \mathcal{C}_0$ of 0-skeleta for categories with products using Theorem 1.7.12. Then we extend this to $\overline{\omega}_1 : \mathfrak{F}_1\mathcal{P} \to \mathcal{C}$ by induction on the term-forming operations, using the assumed structure of $\mathcal{C}$. For instance, to define $\overline{\omega}_1$ on a term $x : X \vdash \langle M, N\rangle : A \times B$, we inductively define it on the terms $x : X \vdash M : A$ and $x : X \vdash N : B$, then apply the pairing operation

$$\mathcal{C}(X, A) \times \mathcal{C}(X, B) \to \mathcal{C}(X, A \times B)$$

of $\mathcal{C}$. The other cases are similar.

We use the naturality of these operations in $\mathcal{C}$ when proving prove that this operation is a functor. For instance, the composite of $y : Y \vdash Q : X$ and $x : X \vdash \langle M, N\rangle : A \times B$ is $\langle M, N\rangle[Q/y]$, which is by definition $\langle M[Q/y], N[Q/y]\rangle$ By induction, we may assume that $\overline{\omega}(M[Q/y])$ and $\overline{\omega}(N[Q/y])$ are the composites $\overline{\omega}(Y) \xrightarrow{\overline{\omega}(Q)} \overline{\omega}(X) \xrightarrow{\overline{\omega}(M)} \overline{\omega}(A)$ and $\overline{\omega}(Y) \xrightarrow{\overline{\omega}(Q)} \overline{\omega}(X) \xrightarrow{\overline{\omega}(N)} \overline{\omega}(B)$. And by naturality, the pairing of these two composite morphisms in $\mathcal{C}$ is equal to the result of first pairing $\overline{\omega}(M)$ and $\overline{\omega}(N)$ and then composing with $\overline{\omega}Q$, which is to say

$$\overline{\omega}(\langle M[Q/y], N[Q/y]\rangle) = \overline{\omega}(\langle M, N\rangle) \circ \overline{\omega}(Q).$$

The other cases of functoriality are analogous. Finally, $\overline{\omega}$ is obviously a morphism of 1-skeleta for categories with products, and it is unique for the usual reasons. □

Definition 1.7.15. The forgetful functor $\mathfrak{U}_2 : \mathbf{PrCat}_2 \to \mathbf{Pres}_{\times,2}$ acts as $\mathfrak{U}_1$ on underlying 1-skeleta, and two parallel terms $x : A \vdash M : B$ and $x : A \vdash N : B$ in the type theory of $\mathfrak{U}_1\mathcal{M}$ are related by a generating equality if and only if their images under the counit map $\mathfrak{F}_1\mathfrak{U}_1\mathcal{M} \to \mathcal{M}$ are equal.

Theorem 1.7.16. *The left adjoint $\mathfrak{F}_2$ of $\mathfrak{U}_2$ is defined by applying $\mathfrak{F}_1$ to underlying 1-skeleta, then quotienting by the equivalence relation of derivable judgments $x : A \vdash M \equiv N : B$.*

Proof. The main point is that a category with products (in our strict sense with chosen binary and nullary products) is exactly a 1-skeleton for a category with products that satisfies categorical versions of the β- and η-conversion rules. Thus, $\mathfrak{F}_2\mathcal{P}$ is indeed a category with products, and we can extend $\overline{\omega}$ to it by induction on derivations of equality. □

This is a bit abstract, so let's consider our motivating example. The $\times$-presentation for monoid objects should have one object $A \in \mathcal{P}_0$ and two arrows, $m \in \mathcal{P}_1(A \times A, A)$ and $e \in \mathcal{P}_1(\mathbb{1}, A)$. It should have three equalities, for associativity and the two unit laws; but what terms do they relate? Consider associativity: we need two terms $x : (A \times A) \times A \vdash M : A$ expressing the two ways to multiply the three components of x. (Note that we also had to chose, arbitrarily, a particular way to associate the triple cartesian product.) First, of course we have to extract those components using π_1 and π_2. Then we have to multiply them in pairs, noting that since the source of m is $A \times A$ we have to pair things up before we can apply m to them. This leads us to the terms

$$x : (A \times A) \times A \vdash m(\langle m(\langle \pi_1(\pi_1(x)), \pi_2(\pi_1(x))\rangle), \pi_2(x)\rangle) : A$$
$$x : (A \times A) \times A \vdash m(\langle \pi_1(\pi_1(x)), m(\langle \pi_2(\pi_1(x)), \pi_2(x)\rangle)\rangle) : A$$

so one of our generating equalities will relate these two terms. The unit laws are simpler; one relates

$$x : A \vdash m(\langle x, e(*)\rangle) : A \qquad \text{and} \qquad x : A \vdash x : A$$

and the other relates

$$x : A \vdash m(\langle e(*), x\rangle) : A \qquad \text{and} \qquad x : A \vdash x : A$$

This completes the definition of the **$\times$-presentation of monoids**. For brevity we may write its generating equalities as

$$\begin{aligned} x : (A \times A) \times A \vdash m(\langle m(\langle \pi_1(\pi_1(x)), \pi_2(\pi_1(x))\rangle), \pi_2(x)\rangle) \\ \equiv m(\langle \pi_1(\pi_1(x)), m(\langle \pi_2(\pi_1(x)), \pi_2(x)\rangle)\rangle) : A \\ x : A \vdash m(\langle x, e(*)\rangle) \equiv x : A \\ x : A \vdash m(\langle e(*), x\rangle) \equiv x : A \end{aligned}$$

as long as we don't forget that the actual rule (1.7.7) builds in a substitution.

Let us write this $\times$-presentation as $\mathcal{T}$. Definition 1.7.15 tells us that a morphism of $\times$-presentations $\omega : \mathcal{T} \to \mathfrak{U}_2\mathcal{M}$, for any category with products $\mathcal{M}$, is exactly a monoid object in $\mathcal{M}$.

More explicitly, first we choose to interpret the base type A as some object $\omega(A) \in \mathcal{M}$. This means choosing a morphism on 0-skeleta $\mathcal{T}_0 \to \mathfrak{U}_0\mathcal{M}_0$.

Second, we extend this interpretation inductively to other types using the chosen products in $\mathcal{M}$, so that for instance $A \times A$ and $\mathbb{1}$ are sent to the chosen product $\omega(A) \times \omega(A)$ and terminal object 1 in $\mathcal{M}$. This means considering the adjunct morphism $\mathfrak{F}_0\mathcal{T}_0 \to \mathcal{M}_0$, which factors as the composite $\mathfrak{F}_0\mathcal{T}_0 \to \mathfrak{F}_0\mathfrak{U}_0\mathcal{M}_0 \to \mathcal{M}_0$.

Third, we choose to interpret the generating arrows m and e as morphisms $\omega(A)\times\omega(A) \to \omega(A)$ and $1 \to \omega(A)$ in $\mathcal{M}$. Technically, we are interpreting them as arrows in $\mathfrak{U}_1\mathcal{M}_1$ compatibly with the map $\mathfrak{F}_0\mathcal{T}_0 \to \mathfrak{F}_0\mathfrak{U}_0\mathcal{M}_0$ on their domains and codomains, but arrows in $\mathfrak{U}_1\mathcal{M}_1$ are induced by the counit $\mathfrak{F}_0\mathfrak{U}_0\mathcal{M}_0 \to \mathcal{M}_0$, so this really means arrows in $\mathcal{M}$ as shown. This gives a morphism $\mathcal{T}_1 \to \mathfrak{U}_1\mathcal{M}_1$.

Fourth, we extend this interpretation to other terms, such as those appearing in the above, sending them to other morphisms in $\mathcal{M}$. This means considering the adjunct morphism $\mathfrak{F}_1\mathcal{T}_1 \to \mathcal{M}_1$, which again factors through the counit $\mathfrak{F}_1\mathfrak{U}_1\mathcal{M}_1 \to \mathcal{M}_1$.

Finally, we assert that the generating equalities in $\mathcal{T}$ are sent to equalities in $\mathcal{M}$, or technically in $\mathfrak{U}_2\mathcal{M}$. Unwinding the definitions shows that the two associativity terms really are sent to the two morphisms $(\omega(A)\times\omega(A))\times\omega(A) \to \omega(A)$ that the associativity of a monoid object should equate, and similarly for the unit terms. Thus, we have precisely specified a monoid object in $\mathcal{M}$.

It now follows from Theorem 1.7.16 that $\mathfrak{F}_2\mathcal{T}$ is the free category with products generated by a monoid. Thus, a monoid object in $\mathcal{M}$ also induces a functor $\overline{\omega} : \mathfrak{F}_2\mathcal{T} \to \mathcal{M}$. Since derivations of equalities yield equal morphisms in $\mathfrak{F}_2\mathcal{T}$, it follows that any equation we can derive in this theory will be true of any monoid object in any category with products.

If we want to reproduce the uniqueness-of-inverses proof from §0.1, we need to further augment our $\times$-presentation with two inverse operations $i, j \in \mathcal{T}_1(A, A)$ and equalities such as

$$x : X \vdash m(\langle x, i(x)\rangle) \equiv e(*) : A$$

The proof of uniqueness now looks like:

$$\begin{aligned}
i(x) &\equiv m(\langle i(x), e(*)\rangle)\\
&\equiv m(\langle i(x), m(\langle x, j(x)\rangle)\rangle)\\
&\equiv m(\langle \pi_1(\pi_1(\langle\langle i(x), x\rangle, j(x)\rangle)), m(\langle \pi_2(\pi_1(\langle\langle i(x), x\rangle, j(x)\rangle)), \pi_2(\langle\langle i(x), x\rangle, j(x)\rangle)\rangle)\rangle)\\
&\equiv m(\langle m(\langle \pi_1(\pi_1(\langle\langle i(x), x\rangle, j(x)\rangle)), \pi_2(\pi_1(\langle\langle i(x), x\rangle, j(x)\rangle))\rangle), \pi_2(\langle\langle i(x), x\rangle, j(x)\rangle)\rangle)\\
&\equiv m(\langle m(\langle i(x), x\rangle), j(x)\rangle)\\
&\equiv m(\langle e(*), j(x)\rangle)\\
&\equiv j(x).
\end{aligned}$$

If it weren't for those two horrific-looking terms in the middle, the rest of the calculation looks pretty much like the argument as we gave it in §0.1. The horrific-looking terms are there because we can only apply the associativity of m to a single term belonging to $(A \times A) \times A$, so we need to tuple up the terms $i(x)$, x, and $j(x)$ and replace them by the projections from that tuple (using β-conversion).

In chapter 2 we will remedy this problem by introducing a type theory that allows us to state associativity (and m itself) without tupling — and, incidentally, remove much of the complication of this section by eliminating the need for products in the domains and codomains of generating arrows entirely.

Now, in §§1.7.1 and 1.7.2 we observed that the counit of the adjunction for presentations was an isomorphism. In the case of $\times$-presentations, this adjunction is now only an *equivalence*.

Theorem 1.7.17. *For any category with products $\mathcal{M}$, the functor $\mathfrak{F}_2\mathfrak{U}_2\mathcal{M} \to \mathcal{M}$ is an equivalence of categories.*

Proof. The definitions of $\mathfrak{U}_0$, $\mathfrak{U}_1$, and $\mathfrak{U}_2$ are cooked up precisely so as to make this functor surjective on objects, full, and faithful respectively. That is, $\mathfrak{U}_0\mathcal{M}$ contains all the objects of $\mathcal{M}$, so that the counit $\varepsilon_0 : \mathfrak{F}_0\mathfrak{U}_0\mathcal{M} \to \mathfrak{U}_0\mathcal{M}$ is surjective — though not injective, since $\mathfrak{F}_0\mathfrak{U}_0\mathcal{M}$ contains new types such as "$A \times B$" for $A, B \in \mathcal{M}$, which is distinct from the specified product of the objects A and B in $\mathcal{M}$. Then for any types A, B, $\mathfrak{U}_1\mathcal{M}(A, B)$ contains all the morphisms of $\mathcal{M}$ between their images in $\mathcal{M}$, so the counit $\varepsilon_1 : \mathfrak{F}_1\mathfrak{U}_1\mathcal{M} \to \mathfrak{U}_1\mathcal{M}$ is full — though not faithful, since $\mathfrak{F}_1\mathfrak{U}_1\mathcal{M}$ also contains new terms such as $x : A \vdash g(f(x)) : C$ for $f : A \to B$ and $g : B \to C$ in $\mathcal{M}$, which is distinct from $x : A \vdash (g \circ f)(x) : C$ for the composite $g \circ f$ of morphisms in $\mathcal{M}$. Finally, $\mathfrak{U}_2\mathcal{M}$ equates all terms whose images in $\mathcal{M}$ are equal, so that when we quotient by $\equiv$ the counit $\varepsilon_2 : \mathfrak{F}_2\mathfrak{U}_2\mathcal{M} \to \mathfrak{U}_2\mathcal{M}$ becomes faithful as well. □

In other words, the functor $\mathfrak{F}_2 : \mathbf{Pres}_{\times,2} \to \mathbf{PrCat}$ is *bicategorically* essentially surjective. Unfortunately, while $\mathbf{PrCat}$ can naturally be enhanced to a 2-category, $\mathbf{Pres}_{\times,2}$ cannot except in the trivial way with only identity 2-cells; and while we can consider $\mathfrak{F}_2$ to be a functor of bicategories with trivial domain, there is no way to extend $\mathfrak{U}_2$ to a functor of bicategories. Thus, we cannot regard categories with products as a reflective subcategory (or even reflective sub-bicategory) of $\times$-presentations. However, we can still construct a bicategory that is (bicategorically) equivalent to $\mathbf{PrCat}$ and whose objects are $\times$-presentations, by taking the hom-category from $\mathcal{P}$ to $\mathcal{Q}$ to be the hom-category $\mathbf{PrCat}(\mathfrak{F}_2(\mathcal{P}), \mathfrak{F}_2(\mathcal{Q}))$. In fact, this bicategory is a strict 2-category, and its underlying 1-category is the Kleisli category of the induced monad on $\mathbf{Pres}_{\times,2}$; we are just observing that the latter category can be enhanced to a 2-category in order to make the adjunction "bicategorically opmonadic".

Remark 1.7.18. It is worth noting that $\times$-presentations as defined here have a minor problem from a type-theoretic standpoint: if we try to formulate a sequent calculus version of their type theories with the same generator rules as before, we find that cut is no longer admissible. For instance, if $f \in \mathcal{P}_1(A, B \times C)$, there is no way to simplify a cut like the following:

$$\dfrac{\dfrac{X \vdash A}{X \vdash B \times C}\,f \qquad \dfrac{B \vdash Y}{B \times C \vdash Y}\,\times L}{X \vdash Y}\,\textsc{cut}$$

That is, if we have generating morphisms whose codomains are products, then their composites with projections cannot be "simplified". In a natural deduction theory, this problem manifests as a lack of "canonical forms" relative to β- and η-conversion.

A common way to deal with this is to simply formulate the cut-elimination theorem in a form like "all cuts except those over types appearing as the domains or codomains of generators can be eliminated". Or roughly equivalently, we can modify the generator rule to build in a cut on *both* sides:

$$\dfrac{X \vdash A \qquad B \vdash Y}{X \vdash Y}\; f \in \mathcal{T}_1(A, B)$$

With this primitive rule, the above "inadmissible" cut could be replaced by

$$\dfrac{X \vdash A \qquad \dfrac{B \vdash Y}{B \times C \vdash Y}\,\times L}{X \vdash Y}\,f$$

In any case, this is another advantage of the multiple-antecedent type theories to be presented in chapter 2: they will enable us to describe important theories (such as monoids) while restricting the generating morphisms to have only base types in their domain and codomain. (Of course we still do require generating equalities relating complex terms rather than just generators.)

The construction of $\times$-presentations, given the ordinary unary type theory for categories with products under directed graphs, while somewhat involved, is quite general and can be applied to pretty much any type theory with an initiality theorem. We will not attempt to make this generality precise, but the approach is similar to the general theory of computads in higher category theory; see [**?**, **?**, **?**].

For instance, by starting from the type theory for categories with coproducts instead, we obtain a notion of **unary +-presentation** and a corresponding tower of adjunctions. Most of the definitions and proofs are direct translations of those for $\times$-presentations, replacing $\times$ and its rules by $+$ and its rules everywhere. The only thing here that requires a little thought is the definition of a **1-skeleton for a category with coproducts**. But looking at the rules for coproduct terms from Figure 1.3, and the corresponding defining clauses of substitution from Lemma 1.5.1, leads us to define it to consist of the following.

(a) A 0-skeleton for a category with coproducts, i.e. a set $\mathcal{M}_0$ with an operation $+ : \mathcal{M}_0 \times \mathcal{M}_0 \to \mathcal{M}_0$ and an element $\mathbf{0} \in \mathcal{M}_0$.

(b) A category with $\mathcal{M}_0$ as its set of objects.

(c) For every $A, B \in \mathcal{M}_0$, morphisms $A \to A + B$ and $B \to A + B$.

(d) For any two morphisms $A \to C$ and $B \to C$, a morphism $A + B \to C$.

(e) For any $A \in \mathcal{M}_0$, a morphism $\mathbf{0} \to A$.

As noted in Remark 1.7.10, the type-theoretic rules are modeled more directly by natural transformations between representable functors. In general, the necessary naturality involves the premise(s) that share antecedents with the conclusion. But in the unary rules for coproducts, unlike those for products, there is always *exactly* one such premise, and so the natural transformations involved will always be in the image of the Yoneda embedding.

The rest of the theory of $+$-presentations is straightforward: we get a tower of adjunctions, with the top left adjoint sending a $+$-presentation to a category with coproducts generated by its type theory. We can then go on to combine these two kinds of presentation, starting from the theory of §1.6 for categories with both products and coproducts and obtain a notion of *unary* $(\times, +)$*-presentation*.

One thing worth noting is that the tower of adjunctions can further motivate the twist in the proof of Theorem 1.5.3. Recall that there we had to prove that the extension $\overline{\omega}$ to terms takes substitution to composition before we could extend it to "act on equalities". From the present perspective, this is explained by the fact that we have to construct the entire 1-skeletal adjunction $\mathfrak{F}_1 \dashv \mathfrak{U}_1$, including the functoriality of induced maps $\overline{\omega} : \mathfrak{F}_1\mathcal{P} \to \mathcal{M}$, before we can start constructing the 2-skeletal adjunction.

The category presentations studied in §1.7.2 fit into this general picture too, but a number of things are degenerate and thus not easily visible. Since there are no type operations, a "0-skeleton for a category" is (like a 0-skeletal category presentation) just a set, so the level-0 adjunction is the identity. And since there are no equality rules (other than those arising from the axioms of a presentation), 1-skeletons and 2-skeletons for categories are both just another name for categories. Thus, the level-1 adjunction is the one from Theorem 1.2.13 between directed graphs and categories, while the level-2 adjunction is the one from Theorem 1.7.3 between category presentations and categories.

In the future, whenever we introduce a new class of type theories, we will first describe the basic version that constructs free categorical structures from "graph-like" data, and then later discuss (in more or less detail, depending on the case) the corresponding kind of "presentation". As above, usually almost everything in the latter case is a straightforward generalization; the main conceptual point is a definition of the 1-skeletal version of the categorical structure with the right amount of naturality. (In some cases, such as §§2.9.2 and 2.10.2, this will require developing a bit more categorical theory.)

1.7.4 Theories

So far in this section we have stuck to categorical terminology, to avoid confusion. However, the sort of objects we have introduced are more traditionally given names arising from mathematical logic.

(a) Structures such as $\times$-presentations are usually called **theories** of an appropriate sort; our $\times$-presentations might be called something like *unary finite-product theories*. Thus, for instance, the above $\times$-presentation for monoids would be called *the unary finite-product theory of monoids*.

(b) Their 1-skeleta, consisting of generating objects and morphisms but no generating equalities, are usually called **signatures**.

(c) The generating objects are usually called **sorts** or **types**, the generating morphisms are usually called **function symbols** or **operations**, and the generating equalities are usually called **axioms**.

(d) The underlying $\times$-presentation of a category with products would usually be called its **internal logic** or **internal type theory**.

(e) A morphism from a $\times$-presentation $\mathcal{P}$ (that is, a unary finite-product theory) into the internal logic of a category with products $\mathcal{M}$ is usually called a **model** or an **interpretation** of $\mathcal{P}$ in $\mathcal{M}$.

(f) The category with products generated by a $\times$-presentation is sometimes called its **syntactic category**.

(g) When we make a (bi)category equivalent to **PrCat**, say, by taking the objects to be the $\times$-presentations, the morphisms are usually known as **translations**. Thus, a translation is a model of its domain in the syntactic category of its codomain.

Thus, the equivalence obtained by "Kleisli construction" from the category of presentations would usually be stated as something like

> Taking the internal logic of a category with products yields an equivalence between the bicategory of categories with products and the bicategory of unary finite-product theories with translations between them.

Such an equivalence is sometimes regarded as the ideal situation for categorical semantics of type theories. The author's opinion is that it is the *adjunction* between presentations and categorical structures that is more fundamental (the equivalence being obtained by a trick of redefining the morphisms in one category); but the equivalence is certainly also important.

Other descriptions of categorical logic also tend to emphasize the "internal logic" more than we have here. Note that the internal logic of $\mathcal{M}$ comes with a canonical interpretation into $\mathcal{M}$ itself, given by the counit $\mathfrak{F}_2\mathfrak{U}_2\mathcal{M} \to \mathcal{M}$; thus anything derivable in the internal logic of $\mathcal{M}$ is actually true in $\mathcal{M}$. Moreover, all the objects, morphisms, and equalities in $\mathcal{M}$ are "available" in its internal logic as generators. Thus, if we only care about semantics in one category $\mathcal{M}$, we can work in this "universal type theory" generated by $\mathcal{M}$, where we have "everything from $\mathcal{M}$" to work with. However, since in practice any actual argument only involves finitely many generators, it is generally also sufficient to work with small explicit theories, thereby making the conclusions more general.

Finally, I should point out that the word *theory* is quite overloaded, in a confusing way that mixes many levels. As stated above, one generally refers to $\times$-presentations as theories. However, we also speak of the *type theory* generated by such a presentation (e.g. what I have called the "type theory for categories with products *under* a given $\times$-presentation"), which constructs the category presented by that presentation. In some sense the original presentation/theory is present "inside" this type theory, but the latter really consists of all the rules, not just the generators.

Note also that there is a *different* "type theory" in this sense associated to each presentation/theory. On the other hand, one often speaks loosely of "the type theory for categories with products", meaning to encompass all of these type theories associated to all presentation/theories. (The word "doctrine" has also been used informally by Jon Beck at a similar level of generality; thus one would say that our above theory of monoids is a "theory in the doctrine of finite products".) And of course, at a yet higher level one speaks of "type theory" as a mathematical *subject*, like "group theory" or "category theory", encompassing *all* such "type theories" (individual collections of judgments and rules).

Finally, there is a thread in categorical logic dating back to Lawvere [**?**] that uses the word "theory" to refer to the *free category generated by* a presentation. Thus, in this usage the "theory of monoids" would be the category with products constructed from our type theory for monoids. This category does retain all the information about the *models* of a theory in categories with products, but it has lost all the information about the generating operations and axioms. For instance, the category with products corresponding to the theory of monoids treats binary multiplication $(x, y) \mapsto x \cdot y$ on an equal footing with ternary multiplication $(x, y, z) \mapsto x \cdot (y \cdot z)$ (or equivalently $(x \cdot y) \cdot z$) and n-ary multiplication for all n, as well as other weirder operations like $(x, y, z) \mapsto y \cdot (x \cdot y)$. In particular, quite different-looking presentation/theories (such as Boolean algebras and Boolean rings) can present equivalent categories, and hence have the same models everywhere; this is sometimes known as *Morita equivalence*.

The passage from presentation/theories to categories is thus undoubtedly of great importance. However, my own feeling is that using the word "theory" for the category rather than its presentation loses too much information that is traditionally included in things referred to as "theories". In mathematical practice, theories (such as "the theory of monoids") are usually specified by a small number of generators and relations; thus if nothing else it is important to understand *the process by which* these generate a category. Type theory, with its technology of cut-admissibility, gives us a concrete way to construct and understand such categories, rather than (for example) simply deducing their existence by an adjoint functor theorem.

An extended dialogue about the meaning of the word "theory" can be found at [**?**].

Exercises

Exercise 1.7.1. Write down a $+$-presentation for *comonoids* in categories with coproducts: objects A equipped with morphisms $\Delta : A \to A + A$ and $e : A \to \mathbf{0}$ satisfying coassociativity and counitality axioms.

Exercise 1.7.2. Write down a $\times$-presentation for ring objects. Then extend it to a $(\times, +)$-presentation for field objects.

Exercise 1.7.3. Write down a unary finite-product theory (that is, a $\times$-presentation) for objects having two monoid structures with the same unit and satisfying an internalized version of the "interchange law" $m_1(m_2(x, y), m_2(z, w)) = m_2(m_1(x, z), m_1(y, w))$. Prove in the resulting type theory that $m_1 = m_2$ and both are commutative. Compare this proof to Exercise 0.1.4. *(In Exercise 2.9.1 you will re-do this proof using a better internal logic for comparison.)*

Exercise 1.7.4. Recall the notion of "distributive near-ring" from Exercise 0.1.5. Write down a unary finite-product theory for internal distributive near-ring objects in a category with products. Then use the resulting type theory to prove that every distributive near-ring object is in fact a ring object; compare this proof to Exercise 0.1.5. *(In Exercise 2.9.1 you will re-do this proof using a better internal logic for comparison.)*

Exercise 1.7.5. Construct a tower of skeletal presentations and adjunctions, analogous to those constructed in this section, corresponding to the type theory for fibrations from Exercise 1.4.4.

Collected Exercises

For convenient reference, we collect the exercises from all sections in this chapter.

Exercise 0.1.1. Prove that in a cartesian monoidal category, every object is a bimonoid in a unique way.

Exercise 0.1.2. Show that the category of cocommutative comonoids in a symmetric monoidal category inherits a monoidal structure, and that this monoidal structure is cartesian.

Exercise 0.1.3. Prove, using arrows and commutative diagrams, that any two antipodes for a bimonoid (not necessarily commutative or cocommutative) are equal.

Exercise 0.1.4. Suppose A is a set with two monoid structures (m_1, e) and (m_2, e) having the same unit element e, and satisfying the "interchange law" $m_1(m_2(x, y), m_2(z, w)) = m_2(m_1(x, z), m_1(y, w))$. Then we have

$$m_1(x, y) = m_1(m_2(x, e), m_2(e, y)) = m_2(m_1(x, e), m_1(e, y)) = m_2(x, y)$$

and also

$$m_1(x, y) = m_1(m_2(e, x), m_2(y, e)) = m_2(m_1(e, y), m_1(x, e)) = m_2(y, x)$$

so that $m_1 = m_2$ and both are commutative. This is called the *Eckmann–Hilton argument*. State and prove an analogous fact about objects in any category with finite products having two monoid structures satisfying an "interchange law". *(In Exercises 1.7.3 and 2.9.1 you will re-do this proof using internal logic for comparison.)*

Exercise 0.1.5. A "distributive near-ring" is like a ring but without the assumption that addition is commutative; thus we have a monoid structure $(\cdot, 1)$ and a group structure $(+, 0)$ such that $\cdot$ distributes over $+$ on both sides.

(a) Prove that every distributive near-ring is actually a ring. *(For this reason, in an unqualified "near-ring" only one side of distributivity is assumed.)*

(b) Define a "distributive near-ring object" in a category with finite products. Try for a little while to prove that any such is actually a "ring object", at least until you can see how much work it would be. In Exercises 1.7.4 and 2.9.1 you will prove this using type theory for comparison.

Exercise 0.2.1. Write out the remaining details in the proof that $\mathfrak{F}X$ is the free group generated by the set X.

Exercise 1.2.1. Let $\mathscr{M}$ be a fixed category; then we have an induced adjunction between $\mathbf{Cat}/\mathscr{M}$ and $\mathbf{Gr}/\mathscr{M}$. Describe a cut-free type theory for presenting the free category-over-$\mathscr{M}$ on a directed-graph-over-$\mathscr{M}$, and prove the initiality theorem (the analogue of Theorem 1.2.13). Note that you will have to prove that cut is admissible first. *(Hint: index the judgments by arrows in $\mathscr{M}$, so that for instance $A \vdash_\alpha B$ represents an arrow lying over a given arrow α in $\mathscr{M}$.)*

Exercise 1.2.2. Category theorists are accustomed to consider $\mathbf{Cat}$ as a 2-category, but our free category $\mathfrak{F}_{\mathbf{Cat}}\mathcal{G}$ only has a 1-categorical universal property, expressed by the 1-categorical adjunction between $\mathbf{Cat}$ and $\mathbf{Gr}$. It is not immediately obvious how it could be otherwise, since unlike $\mathbf{Cat}$, $\mathbf{Gr}$ is only a 1-category; but there is something along these lines that we can say.

(a) Suppose $\mathcal{G}$ is a directed graph and $\mathcal{C}$ a category; define a category $\mathbf{Gr}(\mathcal{G}, \mathcal{C})$ whose objects are graph morphisms $\mathcal{G} \to \mathcal{C}$ and whose morphisms are an appropriate kind of "natural transformation".

(b) Prove that $\mathbf{Gr}(\mathcal{G}, -)$ is a 2-functor $\mathbf{Cat} \to \mathbf{Cat}$.

(c) Using the cut-free presentation of $\mathfrak{F}_{\mathbf{Cat}}\mathcal{G}$, prove that it is a representing object for this 2-functor.

Exercise 1.2.3. Regarding the cut-free type theory for categories as describing a multi-sorted algebraic theory, define a particular algebra for this theory that does not satisfy the cut rule. Then define another algebra that does admit a "cut rule", but in which the resulting "composition" is not associative.

Exercise 1.3.1. Using the unary sequent calculus for meet-semilattices, prove that $A \wedge A \cong A$ for any object A of any meet-semilattice. (Recall that meet-semilattices are categories with at most one morphism in each hom-set, so for two objects to be isomorphic it suffices to have a morphism in each direction.) Then prove the same thing using the natural deduction.

Exercise 1.3.2. Using either the sequent calculus or the natural deduction for meet-semilattices (your choice), prove that in any meet-semilattice we have

$$A \wedge \top \cong A \qquad \top \wedge A \cong A \qquad A \wedge B \cong B \wedge A \qquad A \wedge (B \wedge C) \cong (A \wedge B) \wedge C$$

Exercise 1.3.3. Prove that the rules $\top R$ and $\wedge R$ in the unary sequent calculus for meet-semilattices are *invertible*, in the sense that whenever we have a derivation of their conclusions, we also have a derivation of all their premises.

Exercise 1.3.4. Describe a sequent calculus for *join-semilattices* (posets with a bottom element and binary joins), and prove the admissibility and initiality theorems for it. The rules for $\bot$ and $\vee$ should be exactly dual to the rules for $\top$ and $\wedge$.

Exercise 1.3.5. By putting together the rules for meet- and join-semilattices, describe a sequent calculus for *lattices* (posets with a top and bottom element and binary meets and joins), and prove the admissibility and initiality theorems for it.

Exercise 1.3.6. Prove that in your sequent calculus for lattices from Exercise 1.3.5, the rules $\top R$, $\wedge R$, $\bot L$, and $\vee L$ are all invertible in the sense of Exercise 1.3.3.

Exercise 1.3.7. A map of posets $P : \mathscr{A} \to \mathscr{M}$ is called a *(cloven) fibration* if whenever $b \in \mathscr{A}$ and $x \le P(b)$, there is a chosen $a \in \mathscr{A}$ such that $P(a) = x$ and $a \le b$ and moreover for any $c \in \mathscr{A}$, $c \le b$ and $P(c) \le x$ together imply $c \le a$. The object a can be written as $x^*(b)$.

(a) Given a fixed poset $\mathscr{M}$, describe a sequent calculus for fibrations over $\mathscr{M}$ by adding rules governing the operations x^* to the cut-free theory of Exercise 1.2.1.

(b) Prove the initiality theorem for this sequent calculus.

(c) Use this sequent calculus to prove that in any fibration $P : \mathscr{A} \to \mathscr{M}$, if we have $b \in \mathscr{A}$ and $x \le y \le P(b)$, then $x^*(y^*(b)) \cong x^*(b)$.

Exercise 1.3.8. Now describe instead a natural deduction for fibrations over $\mathscr{M}$, prove the initiality theorem, and re-prove that $x^*(y^*(b)) \cong x^*(b)$ using this theory.

Exercise 1.3.9. Suppose we augment your sequent calculus for fibrations over $\mathscr{M}$ from Exercise 1.3.7 with the following additional rules for "fiberwise meets". Here $\vdash A\ \mathsf{type}_x$ is a judgment indicating that A will be an object of our fibration in the fiber over $x \in \mathscr{M}$.

$$\frac{}{\vdash \top_x \ \mathsf{type}_x} \qquad \frac{\vdash A \ \mathsf{type}_x \qquad \vdash B \ \mathsf{type}_x}{\vdash A \wedge_x B \ \mathsf{type}_x} \qquad \frac{\vdash A \ \mathsf{type}_x}{A \vdash_{x \le y} \top_y}$$

$$\frac{A \vdash_{x \le y} C \qquad \vdash B \ \mathsf{type}_x}{A \wedge_x B \vdash_{x \le y} C} \qquad \frac{B \vdash_{x \le y} C \qquad \vdash A \ \mathsf{type}_x}{A \wedge_x B \vdash_{x \le y} C}$$

$$\frac{A \vdash_{x \le y} B \qquad A \vdash_{x \le y} C}{A \vdash_{x \le y} B \wedge_y C}$$

Consider the sequents

$$x^*(A \wedge_y B) \vdash_{x \le x} x^*A \wedge_x x^*B$$
$$x^*A \wedge_x x^*B \vdash_{x \le x} x^*(A \wedge_y B)$$

for $x \le y$, $\vdash A \ \mathsf{type}_y$, and $\vdash B \ \mathsf{type}_y$.

(a) Construct derivations of these sequents in the above sequent calculus.

(b) Write down an analoguous natural deduction and derive the above sequents therein.

(c) What categorical structure do you think these type theories construct the initial one of? If you feel energetic, prove the initiality theorem.

Exercise 1.3.10. A map of posets $P : \mathscr{A} \to \mathscr{M}$ is called an *opfibration* if $P^{\mathrm{op}} : \mathscr{A}^{\mathrm{op}} \to \mathscr{M}^{\mathrm{op}}$ is a fibration. The analogous operation takes $a \in \mathscr{A}$ and $P(a) \le y$ to a $b \in \mathscr{A}$ with $P(b) = y$ and $a \le b$ and a universal property; we write this b as $y_!(a)$. We say P is a *bifibration* if it is both a fibration and an opfibration. Describe a sequent calculus for bifibrations over a fixed $\mathscr{M}$, and prove the initiality theorem.

Exercise 1.3.11. Use your sequent calculus from Exercise 1.3.10 to prove that in a bifibration of posets, if $x \le y$ in $\mathscr{M}$, we have an adjunction $y_! \dashv x^*$.

Exercise 1.3.12. Use your sequent calculus from Exercise 1.3.10 to prove that in a bifibration of posets, if $x \cong y$ in $\mathscr{M}$, we have an isomorphism $x_! \cong x^*$ (that is, for any a in the fiber over y, we have $x_!(a) \cong x^*(a)$).

Exercise 1.4.1. Suppose we have

$$f \in \mathcal{G}(A, B) \qquad g \in \mathcal{G}(A, C) \qquad h \in \mathcal{G}(B, D) \qquad k \in \mathcal{G}(C, E)$$

Consider the following two derivations of $A \vdash D \times E$. Note that both use the admissible cut/substitution rule.

$$\dfrac{\dfrac{\dfrac{\overline{A \vdash A}}{A \vdash B}\,f \qquad \dfrac{\overline{B \vdash B}}{B \vdash D}\,h}{A \vdash D}\,\textsc{cut} \qquad \dfrac{\dfrac{\overline{A \vdash A}}{A \vdash C}\,g \qquad \dfrac{\overline{C \vdash C}}{C \vdash E}\,k}{A \vdash E}\,\textsc{cut}}{A \vdash D \times E}\,\times I$$

$$\dfrac{\dfrac{\dfrac{\overline{A \vdash A}}{A \vdash B}\,f \qquad \dfrac{\overline{A \vdash A}}{A \vdash C}\,g}{A \vdash B \times C}\,\times I \qquad \dfrac{\dfrac{\dfrac{\overline{B \times C \vdash B \times C}}{B \times C \vdash B}\,\times E1}{B \times C \vdash D}\,h \qquad \dfrac{\dfrac{\overline{B \times C \vdash B \times C}}{B \times C \vdash C}\,\times E2}{B \times C \vdash E}\,k}{B \times C \vdash D \times E}\,\times I}{A \vdash D \times E}\,\textsc{cut}$$

Write down the terms corresponding to these two derivations and show directly that they are related by $\equiv$.

Exercise 1.4.2. Use the type theory for categories with products to prove that in any category with products we have

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

Note that since we are in categories now rather than posets, to show that two types A and B are isomorphic we must derive $x : A \vdash M : B$ and $y : B \vdash N : A$ and also show that their substitutions in both orders are equal (modulo $\equiv$) to identities.

Exercise 1.4.3. Prove Lemmas 1.4.11 and 1.4.12 (substitution is associative and respects $\equiv$ in the unary type theory for categories with products).

Exercise 1.4.4. A functor $P : \mathscr{A} \to \mathscr{M}$ is called a **fibration** if for any $b \in \mathscr{A}$ and $f : x \to P(b)$, there exists a morphism $\phi : a \to b$ in $\mathscr{A}$ such that $P(\phi) = f$ and ϕ is *cartesian*, meaning that for any $\psi : c \to b$ and $g : P(c) \to x$ such that $P(\psi) = fg$, there exists a unique $\chi : c \to a$ such that $P(\chi) = g$ and $\phi\chi = \psi$. The object c is denoted $f^*(b)$.

(a) Generalize your natural deduction for fibrations of posets from Exercise 1.3.8 to a type theory for fibrations of categories over a fixed base category $\mathscr{M}$, with β- and η-conversion $\equiv$ rules.

(b) Prove the initiality theorem for this type theory.

(c) Use this type theory to prove that in any fibration $P : \mathscr{A} \to \mathscr{M}$:

  (i) For any $f : x \to y$ in $\mathscr{M}$, f^* is a functor from the fiber over y to the fiber over x.

  (ii) For any $B \in \mathscr{A}$ and $x \xrightarrow{f} y \xrightarrow{g} P(B)$ in $\mathscr{M}$, we have $f^*(g^*(B)) \cong (gf)^*(M)$.

Exercise 1.4.5. Generalize Exercise 1.3.9 from posets to categories, combining your type theory from Exercise 1.4.4 with the one for categories with products from §1.4.

Exercise 1.4.6. The category **PrCat** is a 2-category whose 2-cells are arbitrary natural transformations (that is, there is no nonvacuous notion of a "product-preserving natural transformation"). Let $\mathcal{G}$ be a directed graph; as in Exercise 1.2.2, define a 2-functor $\mathbf{Gr}(\mathcal{G}, -) : \mathbf{PrCat} \to \mathbf{Cat}$, and show that $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ is a representing object for it. *(Use induction over the derivations of the judgments in its type-theoretic description.)*

Exercise 1.4.7. Exercises 1.2.2 and 1.4.6 address one worry that a category theorist might have about the strictness of our constructions. Another such worry is that the morphisms in **PrCat** preserve specified products *strictly*, while it is usually more natural in category theory to preserve products only up to isomorphism. This is not a problem if our main purpose is to have a syntax to describe objects and morphisms in particular categories with products; indeed, it is exactly what we would want. However, for abstract reasons it may be nice to also be able to say something about less strict functors.

With this in mind, prove that for any $\mathcal{G}$, the category with products $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ is *semi-flexible* in the sense of [**?**]: that is, if $\mathcal{M}$ has chosen products, then every functor $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G} \to \mathcal{M}$ that preserves products in the usual up-to-isomorphism sense is naturally isomorphic to a functor that preserves the chosen products strictly. *(Again, use induction over derivations in the type-theoretic description.)* Deduce that $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ satisfies a universal property relative to the 2-category of categories with products and functors that preserve them up to isomorphism.

Exercise 1.4.8. Here is another way to prove the result of Exercise 1.4.7.

(a) Use the initiality of $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ to show that if $\mathcal{M}$ has finite products and $Q : \mathcal{M} \to \mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ preserves finite products strictly, then any map of directed graphs $\mathcal{G} \to \mathcal{M}$ that lifts the inclusion $\mathcal{G} \to \mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ extends to a section $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G} \to \mathcal{M}$ of Q in **PrCat**.

(b) The results of [**?**] imply that the 2-category of categories with products and functors that preserve products in the usual up-to-isomorphism sense has 2-categorical limits called products, inserters, and equifiers, and the projections of these limits preserve products strictly. Use this, and (a), to prove that $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G}$ satisfies a universal property relative to this 2-category.

Exercise 1.5.1. This is the dual of Exercise 1.4.1, though of course its proof is not dual. Suppose we have

$$f \in \mathcal{G}(A, C) \qquad g \in \mathcal{G}(B, D) \qquad h \in \mathcal{G}(C, E) \qquad k \in \mathcal{G}(D, E)$$

Here is one (cut-free) derivation of $A + B \vdash E$.

$$\dfrac{\dfrac{}{A+B \vdash A+B} \qquad \dfrac{\dfrac{\dfrac{}{A \vdash A}}{A \vdash C}\,f}{A \vdash E}\,h \qquad \dfrac{\dfrac{\dfrac{}{B \vdash B}}{B \vdash D}\,g}{B \vdash E}\,k}{A+B \vdash E}\,{+E}$$

Write down another derivation of $A + B \vdash E$ that ends with the following cut:

$$\dfrac{\dfrac{\vdots}{A+B \vdash C+D} \qquad \dfrac{\vdots}{C+D \vdash E}}{A+B \vdash E}\,\textsc{cut}$$

Then write down the terms corresponding to the two derivations and show directly that they are related by $\equiv$.

Exercise 1.5.2. A functor $P : \mathscr{A} \to \mathscr{B}$ is called an **opfibration** if $P^{\mathrm{op}} : \mathscr{A}^{\mathrm{op}} \to \mathscr{M}^{\mathrm{op}}$ is a fibration (as in Exercise 1.4.4). The dual of $f^*(b)$ is written $f_!(a)$.

(a) Write down a type theory for opfibrations and prove the initiality theorem. (Remember that we always use natural deduction style when dealing with categories rather than posets, so you can't just dualize Exercise 1.4.4 or categorify Exercise 1.3.10. You will probably want a term syntax such as "$\mathsf{match}_!$".)

(b) Use this type theory to prove that $f_!$ is always a functor.

Exercise 1.6.1. Suppose we have

$$f \in \mathcal{G}(A, C) \qquad g \in \mathcal{G}(A, D) \qquad h \in \mathcal{G}(B, C) \qquad k \in \mathcal{G}(B, D)$$

Write down two different derivations of $A + B \vdash C \times D$ in the type theory for categories with products and coproducts under $\mathcal{G}$, one that ends with $\times I$ and one that ends with $+E$. Then write down the corresponding terms and show directly that they are identified by $\equiv$.

Exercise 1.6.2. A functor $P : \mathscr{A} \to \mathscr{B}$ is called a **bifibration** if it is both a fibration and an opfibration.

(a) Combine the theories of Exercises 1.4.4 and 1.5.2 to obtain a type theory for bifibrations.

(b) If you aren't tired of proving initiality theorems yet, do it for this type theory.

(c) Use this type theory to prove that in any bifibration, $f_!$ is left adjoint to f^*.

Exercise 1.7.1. Write down a $+$-presentation for *comonoids* in categories with coproducts: objects A equipped with morphisms $\Delta : A \to A + A$ and $e : A \to \mathbf{0}$ satisfying coassociativity and counitality axioms.

Exercise 1.7.2. Write down a $\times$-presentation for ring objects. Then extend it to a $(\times, +)$-presentation for field objects.

Exercise 1.7.3. Write down a unary finite-product theory (that is, a $\times$-presentation) for objects having two monoid structures with the same unit and satisfying an internalized version of the "interchange law" $m_1(m_2(x, y), m_2(z, w)) = m_2(m_1(x, z), m_1(y, w))$. Prove in the resulting type theory that $m_1 = m_2$ and both are commutative. Compare this proof to Exercise 0.1.4. *(In Exercise 2.9.1 you will re-do this proof using a better internal logic for comparison.)*

Exercise 1.7.4. Recall the notion of "distributive near-ring" from Exercise 0.1.5. Write down a unary finite-product theory for internal distributive near-ring objects in a category with products. Then use the resulting type theory to prove that every distributive near-ring object is in fact a ring object; compare this proof to Exercise 0.1.5. *(In Exercise 2.9.1 you will re-do this proof using a better internal logic for comparison.)*

Exercise 1.7.5. Construct a tower of skeletal presentations and adjunctions, analogous to those constructed in this section, corresponding to the type theory for fibrations from Exercise 1.4.4.

Chapter 2

Simple type theories

At this point we have done about all we can with *unary* type theories, where the antecedent and consequent of each sequent consist only of a single type. (In fact, most introductions to type theory skip over the unary case altogether, but I find it clarifying to start with cases that are as simple as possible.) The most common type theories allow finite *lists* of types as the antecedent. These are the object of study in this chapter; we call them *simple type theories*. This term is more common in the literature than "unary", but perhaps not with the exact meaning we are giving it; the word "simple" is primarily used to contrast with "dependent" type theories (see chapter 6).

2.1 Towards multicategories

As motivation for the generalization away from unary type theories, we consider a few problems with unary type theory, from a categorical perspective, that all turn out to have this as their solution. Let's begin by stating some general principles of type theory. Looking back at the rules of all our type theories, we see that they can be divided into two groups. On the one hand, there are rules that don't refer specifically to any operation, such as the identity rule $x : X \vdash x : X$ and the cut rule. On the other hand, there are rules that introduce or eliminate a particular operation on types, such as product, coproduct, and so on — and each such rule refers to only *one* operation (such as $\times, +, f^*$, etc.).

This "independence" between the rules for distinct operations is important for the good behavior of type theory. Many of the exercises have involved combining the rules for multiple previously-studied operations, and in §1.6 we remarked on how such operations tend to coexist "without interacting" in the metatheory: e.g. when proving the cut-admissibility theorems we essentially just commute the rules for different operations past each other. This "modularity" means that we are always free to add new structure to a theory without spoiling the structure we already had. We formulate it as a general principle:

Each rule in a type theory should refer to only one operation. $(*)$

(Like any general principle, (∗) is not always strictly adhered to. For instance, we haven't discussed the $\equiv$ relation that have to be imposed on sequent calculus derivations to present non-posetal free categories, but these turn out to involve "commutativity" relations between *different* type operations. This is arguably another advantage of natural deduction.)

Note that despite (∗), we can often obtain nontrivial results about the interaction of operations. For instance, in Exercise 1.3.2 you showed that $A \wedge \top \cong A$, even though $\wedge$ and $\top$ are distinct operations with apparently unrelated rules. Similarly, in Exercises 1.3.9 and 1.4.5 you showed that f^* preserves $\wedge$ and $\times$. In general, this tends to happen when relating operations whose universal properties all have the same "handedness". For instance, all the operations $\wedge, \top, \times, \mathbb{1}, f^*$ have "mapping in" or "from the right" universal properties. Thus, we can expect to compare two objects built using more than one of them by showing that they have the same universal property, and this is essentially what type theory does.

We also observed in §1.6 that in all cases we were able to extract the rules for a given operation from the universal property of the objects it was intended to represent in category theory. The left and right rules in a sequent calculus, or the introduction and elimination rules in a natural deduction, always expressed the "two sides" of a universal property: one of them "structures the object" and the other says that it is universal with this structure. The "principal case" of cut admissibility for a sequent calculus, and the β-conversion equality rule for a natural deduction, both express the fact that morphisms defined by the universal property "compose with the structure" to the inputs, e.g. a map $X \to A \times B$ defined from $f : X \to A$ and $g : X \to B$ gives back f and g when composed with the product projections. Similarly, the proof of identity admissibility for a sequent calculus, and the η-conversion rule for a natural deduction, express the uniqueness half of the universal property. This leads us to formulate another general principle:

The operations in a type theory should correspond categorically to objects with universal properties. (†)

The point is that from the perspective of unary type theory, these two principles *seem* overly restrictive. For instance, we remarked above that by expressing universal properties in type theory we can compare operations whose universal properties have the same handedness; but often we are interested in categorical structures satisfying nontrivial relations between objects with universal properties of different handedness. For instance, in any category with both products and coproducts, there is a canonical map $(A \times B) + (A \times C) \to A \times (B + C)$, and the category is said to be *distributive* if this map is always an isomorphism. (When the category is a poset, we call it a *distributive lattice*.) However, we saw in §1.6 that if we simply combine the unary type theoretic rules for $\times$ and $+$, we get a type theory for categories with products and coproducts, but no interaction between them. So unary type theory cannot deal with distributive categories while adhering to (∗) and (†).

Perhaps surprisingly, there *is* a way to present a type theory for distributive categories. The idea is to move into a world where the product $A \times B$ *also* has a "mapping out" universal property, so that we can compare $(A \times B) + (A \times C)$ and $A \times (B + C)$ by saying they have the same universal property. As we will see, this requires moving to a type theory with multiple antecedents.

This is one motivation. Another is that we might want to talk about operations whose universal property can't be expressed in unary type theory while adhering to $(*)$. For instance, a *cartesian closed category* has exponential objects such that morphisms $X \to Z^Y$ correspond to morphisms $X \times Y \to Z$; but how can we express this without referring to $\times$? It turns out that the solution is the same.

We might also want to talk about operations that *have* no obvious universal property, obviously violating $(\dagger)$. For instance, what about monoidal categories? In the usual presentation of a monoidal category, the tensor product $A \otimes B$ has no universal property. It turns out that there is a way to give it a universal property, and this also leads us to higher-ary antecedents.

So much for motivation. As already mentioned, on the type-theoretic side what we will do in this chapter is allow multiple types in the antecedent of a judgment (but still, for now, only one type in the consequent); we call these *simple type theories*. In a simple type theory the antecedent is often called the *context*.

On the categorical side, what we will study are *multicategories* of various sorts. An ordinary multicategory is like a category but whose morphisms can have any finite list of objects as their domain, say $f : (A_1, \ldots, A_n) \to B$, with a straightforward composition law. There are many possible variations on this definition: in a symmetric multicategory the finite lists can be permuted, in a cartesian multicategory we can add unrelated objects and collapse equal ones, and so on. All of these categorical structures are known as *generalized multicategories*. There is an abstract theory of generalized multicategories [**?**, **?**, **?**, **?**] that includes these examples and many others, but (at least in the current version of this chapter) we will simply work with concrete examples.

Our approach to the semantics of simple type theory can be summed up in the following additional principle:

> The shape of the context and the structural rules in a simple type theory should mirror the categorical structure of a generalized multicategory. $(\ddagger)$

The *structural rules* are the rules that don't refer to any operation on types, such as identity and cut. (In this chapter we will meet other structural rules, such as exchange — corresponding to permutation of domains in a symmetric multicategory – and contraction and weakening — corresponding to diagonals and projections in a cartesian multicategory.) Principle $(\dagger)$ then tells us that the *non-structural* rules (which are sometimes called *logical* rules) should all correspond to objects with universal properties in a generalized multicategory, and principle $(*)$ tells us that each non-structural rule should involve only *one* such object.

In sum, we have the following table of correspondences:

Type theory	Category theory
Structural rules	Generalized multicategory
Logical rules	Independent universal properties

Here by "independent universal properties" I mean that the universal property of each object can be defined on its own without reference to any other objects defined by universal properties (unlike, for instance, the exponential in a cartesian closed category).

We might formulate one further principle based on our experience in chapter 1:

$$\text{Insofar as possible, structural rules should be admissible rather than primitive.} \tag{§}$$

It is not generally possible to make *all* the structural rules admissible; for instance, we have seen that for sequent calculus we need a primitive identity rule at base types, while for natural deduction we need a primitive identity rule at all types. However, in chapter 1 we were always able to make the substitution/cut/composition rule admissible rather than primitive. That will continue to be the case in this chapter, and we will also strive for admissibility of the new structural rules we introduce (exchange, weakening, and contraction).

Note that together our four principles say that insofar as possible, the "algebraic operations" in a categorical structure (such as composition and identities in a category or multicategory, permutation of domains in a symmetric multicategory, and so on) are exactly what we do *not* include as primitive rules in type theory! To put this differently, recall from the end of §1.2.2 that the initiality theorems for type theory are about showing that two different categories have the same initial object; we might then say that the effect of the above principles is to ensure that these two categories are *as different as possible*. This may seem strange, but to paraphrase John Baez[1], a proof that two things are the same is more interesting (and more useful) the more different the two things appear to begin with.

Another way to say it is that in category theory we take the algebraic structure of a category as primitive, and use them to define and characterize objects with universal properties; whereas in type theory we take the universal properties as primitive and deduce that the algebraic structure is admissible. Put this way, one might say that type theory is even more category-theoretic than category theory; for what is more category-theoretic than universal properties?

This all been very abstract, so I recommend the reader come back to this section again at the end of this chapter. However, for completeness let me point out now that this general correspondence is particularly useful when designing new type theories and when looking for categorical semantics of existing type theories. On one hand, any type theory that adheres to ($*$) should have semantics in a kind of generalized multicategory that can be "read off" from the shape of its contexts and its structural rules. On the other hand, to construct a type theory for a given categorical structure, we should seek to represent that structure as a generalized multicategory in which all the relevant objects have independent

[1] "Every interesting equation is a lie." [?]

universal properties; then we can "read off" from the domains of morphisms in those multicategories the shape of the contexts and the structural rules of our desired type theory.

We will not attempt to make this correspondence precise in any general way, and in practice it has many tweaks and variations that would probably be exceptions to any putative general theorem; but it is a useful heuristic.

2.2 Introduction to multicategories

From a categorical point of view, a multicategory can be regarded as an answer to the question "in what kind of structure does a tensor product have a universal property?" The classical tensor product of vector spaces (or, more generally, modules over a commutative ring) does have a universal property: it is the target of a universal bilinear map. That is, there is a function $m : V \times W \to V \otimes W$ that is bilinear (i.e. $m(x, -)$ and $m(-, y)$ are linear maps for all $x \in V$ and $y \in W$), and any other bilinear map $V \times W \to U$ factors uniquely through m by a linear map $V \otimes W \to U$. Put differently, $V \otimes W$ is a representing object for the functor $\mathrm{Bilin}(V, W; -) : \mathbf{Vect} \to \mathbf{Set}$.

Of course, this property determines the tensor product up to isomorphism (though of course one still needs some more or less explicit construction to ensure that such a representing object exists). However, unlike many universal properties, it is not quite sufficient on its own to show that the tensor product behaves as desired. In particular, to show that the tensor product is associative, we would naturally like to show that $V \otimes (W \otimes U)$ and $(V \otimes W) \otimes U$ are both representing objects for the functor of *trilinear* maps $\mathrm{Trilin}(V, W, U; -)$, and hence isomorphic. But this is not an abstract consequence of the fact that each binary tensor product represents bilinear maps; what we need is a sort of "relative representability" such as $\mathrm{Trilin}(V, W, U; -) \cong \mathrm{Bilin}(V \otimes W, U; -)$.

Finally, when we come to prove that these associativity isomorphisms satisfy the pentagon axiom of a monoidal category, we need analogous facts about quadrilinear maps, at which point it is clear that we should be talking about n-linear maps for a general n. A multicategory is the categorical context in which to do this: in addition to ordinary morphisms like an ordinary category (e.g. linear maps) it also contains n-ary maps for all $n \in \mathbb{N}$ (e.g. multilinear maps).

Formally, just as a category is a directed graph with composition and identities, a multicategory is a *multigraph* with composition and identities.

Definition 2.2.1. A **multigraph** $\mathcal{G}$ consists of a set $\mathcal{G}_0$ of *objects*, together with for every object B and every finite list of objects $(A_1, \dots, A_n)$ a set of *arrows* $\mathcal{G}(A_1, \dots, A_n; B)$.

Note that n can be 0. We say that an arrow in $\mathcal{G}(A_1, \dots, A_n; B)$ is n**-ary**; the special cases $n = 0, 1, 2$ are *nullary*, *unary*, and *binary*.

Definition 2.2.2. A **multicategory** $\mathcal{M}$ is a multigraph together with the following structure and properties.

- For each object A, an identity arrow $\mathsf{id}_A \in \mathcal{M}(A; A)$.
- For any object C and lists of objects $(B_1, \ldots, B_m)$ and $(A_{i1}, \ldots, A_{in_i})$ for $1 \le i \le m$, a composition operation

$$\mathcal{M}(B_1, \ldots, B_m; C) \times \prod_{i=1}^{m} \mathcal{M}(A_{i1}, \ldots, A_{in_i}; B_i) \longrightarrow \mathcal{M}(A_{11}, \ldots, A_{mn_m}; C)$$
$$(g, (f_1, \ldots, f_m)) \mapsto g \circ (f_1, \ldots, f_m)$$

[TODO: Picture]

- For any $f \in \mathcal{M}(A_1, \ldots, A_n; B)$ we have

$$\mathsf{id}_B \circ (f) = f \qquad\qquad f \circ (\mathsf{id}_{A_1}, \ldots, \mathsf{id}_{A_n}) = f.$$

- For any h, g_i, f_{ij} we have

$$(h \circ (g_1, \ldots, g_m)) \circ (f_{11}, \ldots, f_{mn_m}) =$$
$$h \circ (g_1 \circ (f_{11}, \ldots, f_{1n_1}), \ldots, g_m \circ (f_{m1}, \ldots, f_{mn_m}))$$

The objects and unary arrows in a multicategory form a category; indeed, a multicategory with only unary arrows is exactly a category. Vector spaces and multilinear maps, as discussed above, are a good example to build intuition.

While the above definition is the most natural one from a certain categorical perspective, there is another equivalent way to define multicategories. If in the "multi-composition" $g \circ (f_1, \ldots, f_m)$ all the f_j's for $j \neq i$ are identities, we write it as $g \circ_i f_i$. We may also write it as $g \circ_{B_i} f_i$ if there is no danger of ambiguity (e.g. if none of the other B_j's are equal to B_i). Thus we have **one-place composition** operations

$$\circ_i : \mathcal{M}(B_1, \ldots, B_n; C) \times \mathcal{M}(A_1, \ldots, A_m; B_i)$$
$$\longrightarrow \mathcal{M}(B_1, \ldots, B_{i-1}, A_1, \ldots, A_m, B_{i+1}, \ldots, B_n; C)$$

that satisfy the following properties:

- $\mathsf{id}_B \circ_1 f = f$ (since id_B is unary, $\circ_1$ is the only possible composition here).
- $f \circ_i \mathsf{id}_{B_i} = f$ for any i.
- If h is n-ary, g is m-ary, and f is k-ary, then

$$(h \circ_i g) \circ_j f = \begin{cases} (h \circ_j f) \circ_{i+k-1} g & \text{if } j < i \\ h \circ_i (g \circ_{j-i+1} f) & \text{if } i \le j < i+m \\ (h \circ_{j-m+1} f) \circ_i g & \text{if } j \ge i+m \end{cases}$$

[TODO: Picture] We refer to the second of these equations as *associativity*, and to the first and third as *interchange*.

Conversely, given one-place composition operations satisfying these axioms, one may define

$$g \circ (f_1, \ldots, f_m) = (\cdots((g \circ_m f_m) \circ_{m-1} f_{m-1}) \cdots \circ_2 f_2) \circ_1 f_1$$

to recover the original definition of multicategory. The details can be worked out by the interested reader (Exercise 2.2.1) or looked up in a reference such as [**?**].

With multicategories in hand, we can give an abstract version of the characterization of the tensor product of vector spaces using multilinear maps.

Definition 2.2.3. Given objects $A_1, \ldots, A_n$ in a multicategory $\mathcal{M}$, a **tensor product** of them is an object $\bigotimes_i A_i$ with a morphism $\chi : (A_1, \ldots, A_n) \to \bigotimes_i A_i$ such that all the maps $(- \circ_i \chi)$ are bijections

$$\mathcal{M}(B_1, \ldots, B_k, \textstyle\bigotimes_i A_i, C_1, \ldots, C_m; D) \xrightarrow{\sim} \mathcal{M}(B_1, \ldots, B_k, A_1, \ldots, A_n, C_1, \ldots, C_m; D).$$

When $n = 2$ we write a binary tensor product as $A_1 \otimes A_2$. When $n = 0$ we call a nullary tensor product a **unit object** and write it as $\mathbf{1}$. When $n = 1$ a unary tensor product is just an object isomorphic to A.

In keeping with the usual "biased" definition of monoidal category (which has a binary tensor product and a unit object, with all other tensors built out of those), we will say that a multicategory is **representable** if it is equipped with a chosen unit object and a chosen binary tensor product for every pair of objects. Let **RepMCat** denote the category of representable multicategories and functors that preserve the chosen tensor products strictly.

Theorem 2.2.4. *The category* **RepMCat** *is equivalent to the category* **MonCat** *of monoidal categories.*

Proof. It is easy to show that $(A_1 \otimes A_2) \otimes A_3$ and $A_1 \otimes (A_2 \otimes A_3)$, if they both exist, are both a ternary tensor product $\bigotimes_{i=1}^3 A_i$, and hence canonically isomorphic. Similarly, $A \otimes \mathbf{1}$ and $\mathbf{1} \otimes A$ are unary tensor products, hence canonically isomorphic to A. The coherence axioms follow similarly; thus any representable multicategory gives rise to a monoidal category.

Conversely, any monoidal category $\mathcal{M}$ has an underlying multicategory defined by $\mathcal{M}(A_1, \ldots, A_n; B) = \mathcal{M}(\bigotimes_i A_i; B)$, where $\bigotimes_i A_i$ denotes some tensor product of the A_i's such as $(\cdots((A_1 \otimes A_2) \otimes A_3) \cdots) \otimes A_n$. The coherence theorem for monoidal categories implies that the resulting hom-sets $\mathcal{M}(A_1, \ldots, A_n; B)$ are independent, up to canonical isomorphism, of the choice of bracketing. We can similarly use the coherence theorem to define the composition of this multicategory, and to show that the given tensor product and unit make it representable. Finally, the constructions are clearly inverse up to natural isomorphism. □

There are other good references on multicategories, such as [**?**, **?**]. We end this section by discussing limits and colimits in multicategories, which are a bit less well-known.

We say that an object $\mathbb{1}$ of a multicategory is **terminal** if for any $A_1, \ldots, A_n$ there is a unique morphism $(A_1, \ldots, A_n) \to \mathbb{1}$. Similarly, a **binary product** of

A and B in a multicategory is an object $A \times B$ with projections $A \times B \to A$ and $A \times B \to B$, composing with which yields bijections

$$\mathcal{M}(C_1, \ldots, C_n; A \times B) \longrightarrow \mathcal{M}(C_1, \ldots, C_n; A) \times \mathcal{M}(C_1, \ldots, C_n; B)$$

for any $C_1, \ldots, C_n$. We will say a multicategory **has products** if it has a specified terminal object and a specified binary product for each pair of objects. The following is entirely straightforward.

Theorem 2.2.5. *A monoidal category has products (in the sense of §1.4) if and only if its underlying multicategory has products, and this yields an equivalence of categories.* □

Colimits in a multicategory are a bit more subtle. We define a **binary coproduct** in a multicategory $\mathcal{M}$ to be an object $A + B$ with injections $A \to A + B$ and $B \to A + B$ composing with which induces bijections

$$\begin{aligned}&\mathcal{M}(C_1, \ldots, C_n, A + B, D_1, \ldots, D_m; E) \xrightarrow{\sim} \\ &\qquad \mathcal{M}(C_1, \ldots, C_n, A, D_1, \ldots, D_m; E) \times \mathcal{M}(C_1, \ldots, C_n, B, D_1, \ldots, D_m; E).\end{aligned}$$

for all $C_1, \ldots, C_n$ and $D_1, \ldots, D_m$ and E. Similarly, an **initial object** is an object $\mathbf{0}$ such that for any $C_1, \ldots, C_n$ and $D_1, \ldots, D_m$ and E, there is a unique morphism $(C_1, \ldots, C_n, \mathbf{0}, D_1, \ldots, D_m) \to E$. We say a multicategory **has coproducts** if it has a specified binary coproduct for each pair of objects and a specified initial object.

By a **distributive monoidal category**, we mean a monoidal category thath has coproducts (in the sense of §1.5) and such that the canonical maps

$$(A \otimes B) + (A \otimes C) \to A \otimes (B + C) \qquad (B \otimes A) + (C \otimes A) \to (B + C) \otimes A$$

$$\mathbf{0} \to A \otimes \mathbf{0} \qquad \mathbf{0} \to \mathbf{0} \otimes A$$

are isomorphisms. (A **distributive category** is a distributive cartesian monoidal category.)

Theorem 2.2.6. *A monoidal category is distributive if and only if its underlying representable multicategory has coproducts, and this yields an equivalence of categories.*

Proof. If $\mathcal{M}$ is distributive, then by induction we have

$$\Big((\textstyle\bigotimes_i C_i) \otimes A \otimes (\bigotimes_j D_j)\Big) + \Big((\bigotimes_i C_i) \otimes B \otimes (\bigotimes_j D_j)\Big) \;\cong\; (\bigotimes_i C_i) \otimes (A + B) \otimes (\bigotimes_j D_j)$$

and similarly

$$\mathbf{0} \;\cong\; (\textstyle\bigotimes_i C_i) \otimes \mathbf{0} \otimes (\bigotimes_j D_j).$$

Since the morphisms in the underlying multicategory of $\mathcal{M}$ are maps out of iterated tensor products in $\mathcal{M}$, these isomorphisms imply that the latter has coproducts.

Conversely, if the underlying multicategory of $\mathcal{M}$ has coproducts, then taking $n = m = 0$ in their universal property we see that the ordinary category $\mathcal{M}$ has coproducts. Moreover, the universal property with $n = 1$ and $m = 0$ applied to the composites

$$(C, A) \to C \otimes A \to (C \otimes A) + (C \otimes B) \qquad (C, B) \to C \otimes B \to (C \otimes A) + (C \otimes B)$$

gives a map $(C, A + B) \to (C \otimes A) + (C \otimes B)$, and the universal property of $\otimes$ then yields a map

$$C \otimes (A + B) \to (C \otimes A) + (C \otimes B).$$

It is straightforward to show that this is an inverse to the canonical map that exists in any monoidal category with coproducts, and similarly in the other cases; thus $\mathcal{M}$ is distributive. Finally, one can check that these constructions are inverse. □

Exercises

Exercise 2.2.1. Prove that the definitions of multicategory in terms of multi-composition and one-place composition are equivalent, in the strong sense that they yield isomorphic categories of multicategories.

Exercise 2.2.2. Fill in the details in the proof of Theorem 2.2.4.

Exercise 2.2.3. Show that the category whose objects are representable multicategories but whose morphisms are *arbitrary* functors of multicategories is equivalent to the category of monoidal categories and *lax* monoidal functors.

Exercise 2.2.4. Show that the category of representable multicategories and functors that "preserve tensor products", in the sense that if $\chi : (A_1, \ldots, A_n) \to \bigotimes_i A_i$ is a tensor product then $F(\chi)$ is also *a* tensor product, is equivalent to the category of monoidal categories and *strong* monoidal functors.

Exercise 2.2.5. Fill in the details in the proof of Theorem 2.2.6.

2.3 Multiposets and monoidal posets

2.3.1 Multiposets

We begin our study of type theory for multicategories with the posetal case. A **multiposet** is a multicategory in which each set $\mathcal{M}(A_1, \ldots, A_n; B)$ has at most one element. We consider the adjunction between the category **MPos** of multiposets and the category **RelMGr** of *relational multigraphs*, i.e. sets of objects equipped with an n-ary relation "$(a_1, \ldots, a_{n-1}) \le b$" for all integers $n \ge 1$. We would like to construct the free multiposet on a relational multigraph $\mathcal{G}$ using a type theory.

Its objects, of course, will be those of $\mathcal{G}$, so we do not yet need a type judgment. We represent its relations using a judgment written

$$A_1, A_2, \ldots, A_n \vdash B.$$

As is customary, we use capital Greek letters such as Γ and Δ to stand for finite lists (possibly empty) of types; thus the above general judgment can also be written $\Gamma \vdash B$. We write "$\Gamma_1, \Gamma_2, \dots, \Gamma_n$" for the concatenation of such lists, and we also write for instance "Γ, A, Δ" to indicate a list containing the type A somewhere the middle.

At the moment, the only rules for this judgment will be identities and those coming from $\mathcal{G}$. Based on the lessons we learned from unary type theory, we represent the latter in Yoneda-style.

$$\frac{}{A \vdash A} \qquad \frac{(A_1, \dots, A_n \le B) \in \mathcal{G} \qquad \Gamma_1 \vdash A_1 \qquad \dots \qquad \Gamma_n \vdash A_n}{\Gamma_1, \dots, \Gamma_n \vdash B}$$

We call this the **cut-free type theory for multiposets under** $\mathcal{G}$. Note that we use the "multi-composition" in Yoneda-ifying the relations in $\mathcal{G}$; this is absolutely necessary for the admissibility of cut. By contrast, it is traditional to formulate the cut rule itself in terms of the one-place compositions:

Theorem 2.3.1. *In the cut-free type theory for multiposets under $\mathcal{G}$, the following cut rule is admissible: if we have derivations of $\Gamma \vdash A$ and of $\Delta, A, \Psi \vdash B$, then we can construct a derivation of $\Delta, \Gamma, \Psi \vdash B$.*

Proof. We induct on the derivation of $\Delta, A, \Psi \vdash B$. If it is the identity rule, then $A = B$ and Δ and Ψ are empty, so our given derivation of $\Gamma \vdash A$ is all we need. Otherwise, it comes from some relation $A_1, \dots, A_n \le B$ in $\mathcal{G}$, where we have derivations of $\Gamma_i \vdash A_i$. Since then $\Delta, A, \Psi = \Gamma_1, \dots, \Gamma_n$, there msut be an i such that $\Gamma_i = \Gamma_i', A, \Gamma_i''$, while $\Delta = \Gamma_1, \dots, \Gamma_{i-1}, \Gamma_i'$ and $\Psi = \Gamma_i'', \Gamma_{i+1}, \dots, \Gamma_n$. Now by the inductive hypothesis, we can construct a derivation of $\Gamma_i', \Gamma, \Gamma_i'' \vdash A_i$. Applying the rule for $A_1, \dots, A_n \le B$ again, with this derivation in place of the original $\Gamma_i \vdash A_i$, gives the desired result. ☐

However, we can also prove admissibility of "multi-cut" directly:

Theorem 2.3.2. *In the cut-free type theory for multiposets under $\mathcal{G}$, the following multi-cut rule is admissible: if we have derivations of $\Psi_i \vdash A_i$ for $1 \le i \le n$, and also $A_1, \dots, A_n \vdash B$, then we can construct a derivation of $\Psi_1, \dots, \Psi_n \vdash B$.*

Proof. If $A_1, \dots, A_n \vdash B$ ends with the identity rule, then $n = 1$ and $A_1 = B$, whence $\Psi_1 \vdash A_1$ is what we want. Otherwise, it comes from some relation $C_1, \dots, C_m \le B$, where we have a partition $A_1, \dots, A_n = \Gamma_1, \dots, \Gamma_m$ and derivations of $\Gamma_j \vdash C_j$. Let Φ_j be the concatenation of all the Ψ_i such that $A_i \in \Gamma_j$; then by the inductive hypothesis we can get $\Phi_j \vdash C_j$. Applying the generator rule again, we get $\Phi_1, \dots, \Phi_m \vdash B$, which is the desired result. ☐

The notation is certainly a bit heavier when constructing multi-cuts directly. However, as we will see later on, in more complicated situations there are definite advantages to the latter.

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \otimes B, \Delta \vdash C}\otimes L \qquad \frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\otimes R \qquad \frac{\Gamma, \Delta \vdash A}{\Gamma, \mathbf{1}, \Delta \vdash A}\mathbf{1}L \qquad \frac{}{\vdash \mathbf{1}}\mathbf{1}R$$

Figure 2.1: Sequent calculus for monoidal posets

Theorem 2.3.3. *For any relational multigraph $\mathcal{G}$, the free multiposet it generates has the same objects, and the relation $(A_1, \ldots, A_n) \leq B$ holds just when the sequent $A_1, \ldots, A_n \vdash B$ is derivable in the cut-free type theory for multiposets under $\mathcal{G}$.*

Proof. Theorem 2.3.1, together with the identity rule, tells us that this defines a multiposet $\mathfrak{F}_{\mathbf{MPos}}\mathcal{G}$. If $\mathcal{M}$ is any other multiposet with a map $P : \mathcal{G} \to \mathcal{M}$ of relational multigraphs, then since the objects of $\mathfrak{F}_{\mathbf{MPos}}\mathcal{G}$ are those of $\mathcal{G}$, there is at most one extension of P to $\mathfrak{F}_{\mathbf{MPos}}\mathcal{G}$. It suffices to check that the relations in $\mathfrak{F}_{\mathbf{MPos}}\mathcal{G}$ hold in $\mathcal{M}$; but this is clear since $\mathcal{M}$ is a multiposet and the only rules are an identity and a particular multi-transitivity. □

Now we augment the type theory for multiposets with operations representing a tensor product. Since the tensor product now has a universal property, this is essentially straightforward. First of all, we need a type judgment $\vdash A\ \mathsf{type}$, with unsurprising rules:

$$\frac{A \in \mathcal{G}}{\vdash A\ \mathsf{type}} \qquad \frac{}{\vdash \mathbf{1}\ \mathsf{type}} \qquad \frac{\vdash A\ \mathsf{type} \qquad \vdash B\ \mathsf{type}}{\vdash A \otimes B\ \mathsf{type}}$$

Second, in addition to the rules from §2.3.1, we have rules for $\otimes$. Once again we need to make a choice between sequent calculus and natural deduction; we treat these one at a time.

2.3.2 Sequent calculus for monoidal posets

The additional rules for the **sequent calculus for monoidal posets under** $\mathcal{G}$ are shown in Figure 2.1. Since $A \otimes B$ has a "mapping out" universal property like a coproduct, the left rule expresses this universal property. The right rule should be the universal relation $A, B \vdash A \otimes B$, but we have to Yoneda-ify it using the multicomposition. The rules for $\mathbf{1}$ are similar.

Note the presence of the additional contexts Γ and Δ in $\otimes L$ and $\mathbf{1}L$, which corresponds to the strong universal property of a tensor product in a multicategory referring to n-ary arrows for all n.

Theorem 2.3.4. *The general identity rule is admissible in the sequent calculus for monoidal posets under $\mathcal{G}$: if $\vdash A\ \mathsf{type}$ is derivable, then so is $A \vdash A$.*

Proof. By induction on the derivation of $\vdash A\ \mathsf{type}$. If $A \in \mathcal{G}$, then $A \vdash A$ is an axiom. If $A = \mathbf{1}$, then $\mathbf{1} \vdash \mathbf{1}$ has the following derivation:

$$\dfrac{\dfrac{}{\vdash \mathbf{1}}\ \mathbf{1}R}{\mathbf{1} \vdash \mathbf{1}}\ \mathbf{1}L$$

And if $A = B \otimes C$, by the inductive hypothesis we have derivations $\mathscr{D}_B$ and $\mathscr{D}_C$ of $B \vdash B$ and $C \vdash C$, which we can put together like so:

$$\dfrac{\dfrac{\dfrac{\mathscr{D}_B \atop \vdots}{B \vdash B} \qquad \dfrac{\mathscr{D}_C \atop \vdots}{C \vdash C}}{B, C \vdash B \otimes C}\ \otimes R}{B \otimes C \vdash B \otimes C}\ \otimes L$$

□

The proof of cut-admissibility in this case has two new features we have not seen before.

Theorem 2.3.5. *Cut is admissible in the sequent calculus for monoidal posets under $\mathcal{G}$: if we have derivations of $\Gamma \vdash A$ and of $\Delta, A, \Psi \vdash B$, then we can construct a derivation of $\Delta, \Gamma, \Psi \vdash B$.*

Proof. If the derivation of $\Delta, A, \Psi \vdash B$ ends with the identity rule or a generating relation from $\mathcal{G}$, we proceed just as in Theorem 2.3.1. It cannot end with a $\mathbf{1}R$. If it ends with a $\otimes R$, we use the inductive hypotheses on its premises and apply $\otimes R$ again.

The cases when it ends with a left rule introduce one new feature. Suppose it ends with a $\mathbf{1}L$. If A is the $\mathbf{1}$ that was introduced by this rule, then we proceed basically as before: if $\Gamma \vdash A$ is $\mathbf{1}R$, so that Γ is empty, then we are in the principal case and we can simply use the given derivation of $\Delta, \Psi \vdash B$; while if it is a left rule then we can apply a secondary induction. But it might also happen that A is a different type, with the introduced $\mathbf{1}$ appearing in Δ or Ψ. However, this case is also easily dealt with by applying the inductive hypothesis to $\Gamma \vdash A$ and the given $\Delta, \Psi \vdash B$ (with A appearing somewhere in its antecedents). In a direct argument for cut-elimination, we are transforming

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash A} \qquad \dfrac{\dfrac{\vdots}{\Delta_1, \Delta_2, A, \Psi \vdash B}}{\Delta_1, \mathbf{1}, \Delta_2, A, \Psi \vdash B}\ \mathbf{1}L}{\Delta_1, \mathbf{1}, \Delta_2, \Gamma, \Psi \vdash B}\ \textsc{cut} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash A} \qquad \dfrac{\vdots}{\Delta_1, \Delta_2, A, \Psi \vdash B}}{\Delta_1, \Delta_2, \Gamma, \Psi \vdash B}\ \textsc{cut}}{\Delta_1, \mathbf{1}, \Delta_2, \Gamma, \Psi \vdash B}\ \mathbf{1}L$$

The case when $\Delta, A, \Psi \vdash B$ ends with $\otimes L$ has a similar "commutativity" possibility. However, in this case there is also something new in the principal case, where $\Delta, A_1 \otimes A_2, \Psi \vdash B$ is derived from $\Delta, A_1, A_2, \Psi \vdash B$, while $\Gamma \vdash A_1 \otimes A_2$

is derived using $\otimes R$ from $\Gamma_1 \vdash A_1$ and $\Gamma_2 \vdash A_2$ (so that necessarily $\Gamma = \Gamma_1, \Gamma_2$). We would like to apply the inductive hypothesis twice to transform

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma_1 \vdash A_1} \quad \dfrac{\vdots}{\Gamma_2 \vdash A_2}}{\Gamma_1, \Gamma_2 \vdash A_1 \otimes A_2} \otimes R \qquad \dfrac{\dfrac{\vdots}{\Delta, A_1, A_2, \Psi \vdash B}}{\Delta, A_1 \otimes A_2, \Psi \vdash B} \otimes L}{\Delta, \Gamma_1, \Gamma_2, \Psi \vdash B} \text{CUT} \tag{2.3.6}$$

into

$$\dfrac{\dfrac{\vdots}{\Gamma_2 \vdash A_2} \qquad \dfrac{\dfrac{\vdots}{\Gamma_1 \vdash A_1} \quad \dfrac{\vdots}{\Delta, A_1, A_2, \Psi \vdash B}}{\Delta, \Gamma_1, A_2, \Psi \vdash B} \text{CUT}}{\Delta, \Gamma_1, \Gamma_2, \Psi \vdash B} \text{CUT} \tag{2.3.7}$$

However, this is a problem for our usual style of induction. We can certainly apply the inductive hypothesis to $\Gamma_1 \vdash A_1$ and $\Delta, A_1, A_2, \Psi \vdash B$ to get a derivation of $\Delta, \Gamma_1, A_2, \Psi \vdash B$. But this resulting derivation need not be "smaller" than our given derivation of $\Delta, A_1 \otimes A_2, \Psi \vdash B$, so our inductive hypothesis does not apply to it.

Probably the most common solution to this problem is to formulate the induction differently. Rather than inducting directly on the derivation of $\Delta, A, \Psi \vdash B$, we induct first on the type A (the "cut formula")), and then do an "inner" induction on the derivation. All the "commutativity" cases do not change the cut formula, so there the inner inductive hypothesis continues to apply. But in the principal case for $\otimes$, both of the cuts we want to do inductively have smaller cut formulas than the one we started with (A_1 and A_2 versus $A_1 \otimes A_2$), so they can be handled by the outer inductive hypothesis regardless of how large of derivations we need to apply them to. □

A different approach, however, is to prove the admissibility of multi-cut directly:

Theorem 2.3.8. *Multi-cut is admissible in the sequent calculus for monoidal posets under $\mathcal{G}$: if we have derivations of $\Psi_i \vdash A_i$ for $1 \le i \le n$, and also $A_1, \ldots, A_n \vdash B$, then we can construct a derivation of $\Psi_1, \ldots, \Psi_n \vdash B$.*

Proof. In this case we can return to inducting directly on the derivation of $A_1, \ldots, A_n \vdash B$. The cases of identity and generator rules are just like in Theorem 2.3.2, and $\otimes R$ is just like the generator case. Unlike in Theorem 2.3.5 it *could* end with $\mathbf{1}R$, but in this case $n = 0$ and there is nothing to be done.

If it ends with $\mathbf{1}L$, then some $A_i = \mathbf{1}$, and we can forget about the corresponding $\Psi_i \vdash A_i$ and proceed inductively with the rest of them. (Note how even this case is simpler than in Theorem 2.3.5.)

Finally, if it ends with $\otimes L$, then some $A_i = C \otimes D$, say, and we perform our secondary induction on $\Psi_i \vdash A_i$. Since $A_i = C \otimes D$ is not a base type, this derivation cannot end with the identity or generator rules, and of course it cannot end with $\mathbf{1}R$. If it ends with a left rule, we inductively cut with the premise of that rule and then apply it afterwards. The remaining case is when it ends with $\otimes R$, so that we have derivations of $\Gamma \vdash C$ and $\Delta \vdash D$ with $\Psi_i = \Gamma, \Delta$. But now we can inductively cut our given premise $A_1, \dots, A_{i-1}, C, D, A_{i+1}, \dots, A_n \vdash B$ with these two and also the given $\Psi_j \vdash A_j$ for $j \neq i$. □

That is, instead of transforming (2.3.6) into (2.3.7), where we have to feed the output of one inductive cut into another inductive cut (which is what creates the problem), we transform

$$\dfrac{\dfrac{\vdots}{\Psi_j \vdash A_j} \qquad \dfrac{\dfrac{\vdots}{\Gamma \vdash C} \quad \dfrac{\vdots}{\Gamma \vdash D}}{\Gamma, \Delta \vdash C \otimes D}\otimes R \qquad \dfrac{\dfrac{\vdots}{A_1, \dots, C, D, \dots, A_n \vdash B}}{A_1, \dots, C \otimes D, \dots, A_n \vdash B}\otimes L}{\Psi_1, \dots, \Gamma, \Delta, \dots, \Psi_n \vdash B}\,\text{CUT}$$

into

$$\dfrac{\dfrac{\vdots}{\Psi_j \vdash A_j} \qquad \dfrac{\vdots}{\Gamma \vdash C} \qquad \dfrac{\vdots}{\Gamma \vdash D} \qquad \dfrac{\vdots}{A_1, \dots, C, D, \dots, A_n \vdash B}}{\Psi_1, \dots, \Gamma, \Delta, \dots, \Psi_n \vdash B}\,\text{CUT}$$

Thus, the multicategorical perspective leads to a simpler inductive proof of cut admissibility. (Note, though, that to recover the one-place cut from the multi-cut requires composing with identities, hence invoking Theorem 2.3.4 as well.)

In any case, we are ready to prove the initiality theorem, relating to an adjunction between the categoris **RelMGr** of relational multigraphs and **MonPos** of monoidal posets. As always, the morphisms in our categories will be completely strict: so in particular the morphisms in **MonPos** are *strict* monoidal functors.

Theorem 2.3.9. *For any relational multigraph $\mathcal{G}$, the free monoidal poset generated by $\mathcal{G}$ is described by the sequent calculus for monoidal posets under $\mathcal{G}$: its objects are the A such that $\vdash A$ type is derivable, while the relation $(A_1, \dots, A_n) \leq B$ holds just when the sequent $A_1, \dots, A_n \vdash B$ is derivable.*

Proof. As before, Theorems 2.3.4 and 2.3.5 show that this defines a multiposet $\mathfrak{F}_{\mathbf{MonPos}}\mathcal{G}$. Moreover, the rules for $\otimes$ and $\mathbf{1}$ tell us that it is representable, hence monoidal.

Now suppose $P : \mathcal{G} \to \mathcal{M}$ is a map into the underlying multiposet of any other monoidal poset. We can extend P uniquely to a function from the objects of $\mathfrak{F}_{\mathbf{MonPos}}\mathcal{G}$ to those of $\mathcal{M}$ preserving $\otimes$ and $\mathbf{1}$ on objects, so it remains to check that this is a map of multiposets and preserves the universal properties of $\otimes$ and $\mathbf{1}$. However, $\otimes R$ and $\mathbf{1}R$ are preserved by the universal data of $\otimes$ and $\mathbf{1}$ in $\mathcal{M}$, while the universal properties of these data mean that $\otimes L$ and $\mathbf{1}L$ are also preserved. □

2.3.3 Natural deduction for monoidal posets

In natural deduction, the introduction rules $\otimes I$ and $\mathbf{1}I$ will coincide with the right rules $\otimes R$ and $\mathbf{1}R$ from the sequent calculus, but now we need elimination rules. Since $\otimes$ and $\mathbf{1}$ in a multicategory have a "mapping out" universal property, these elimination rules will be reminiscent of the "case analysis" rules from §1.5. Formally speaking, they can be obtained by simply cutting $\otimes L$ and $\mathbf{1}L$ with an arbitrary sequent (thereby "building in cuts" to make the cut-admissiblity theorem easier).

$$\dfrac{\Psi \vdash A \otimes B \qquad \dfrac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \otimes B, \Delta \vdash C}\otimes L}{\Gamma, \Psi, \Delta \vdash C}\text{CUT} \qquad \dfrac{\Psi \vdash \mathbf{1} \qquad \dfrac{\Gamma, \Delta \vdash A}{\Gamma, \mathbf{1}, \Delta \vdash A}\mathbf{1}L}{\Gamma, \Psi, \Delta \vdash C}\text{CUT}$$

As usual in a natural deduction, we also need to assert the identity rule for all types. Thus our complete **natural deduction for monoidal posets under** $\mathcal{G}$ consists of (the rules for $\vdash A\ \mathsf{type}$ and):

$$\dfrac{\vdash A\ \mathsf{type}}{A \vdash A} \qquad \dfrac{(A_1, \dots, A_n \le B) \in \mathcal{G} \qquad \Gamma_1 \vdash A_1 \qquad \dots \qquad \Gamma_n \vdash A_n}{\Gamma_1, \dots, \Gamma_n \vdash B}$$

$$\dfrac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\otimes I \qquad \dfrac{\Psi \vdash A \otimes B \qquad \Gamma, A, B, \Delta \vdash C}{\Gamma, \Psi, \Delta \vdash C}\otimes E$$

$$\dfrac{}{() \vdash \mathbf{1}}\mathbf{1}I \qquad \dfrac{\Psi \vdash \mathbf{1} \qquad \Gamma, \Delta \vdash C}{\Gamma, \Psi, \Delta \vdash C}\mathbf{1}E$$

We leave the metatheory of this as an exercise (Exercise 2.3.1); it is also subsumed by the categorified version discussed in more detail in the next section.

Remark 2.3.10. In §1.3.2 we remarked that in (unary) natural deductions, the conclusions (bottoms) of rules always have an arbitrary type as antecedent (left side of $\vdash$). For simple type theories, the corresponding property is that the conclusions of rules should have an arbitrary *context* on the left. This is not quite true for the above presentation of the rules, since most of their conclusions have an antecedent obtained by concatenating two or more contexts. However, such a rule is always equivalent to one whose conclusion involves an arbitrary context that is decomposed as a concatenation by an additional premise. For instance, the rule $\otimes I$ could equivalently be formulated as

$$\dfrac{\Psi = \Gamma, \Delta \qquad \Gamma \vdash A \qquad \Delta \vdash B}{\Psi \vdash A \otimes B}\otimes I$$

while $\mathbf{1}I$ could be written

$$\dfrac{\Gamma = ()}{\Gamma \vdash \mathbf{1}}\mathbf{1}I$$

This is the appropriate point of view when reading rules "bottom-up" for type-checking or proof search, as discussed at the end of §A.4: to type-check or prove $\Psi \vdash A \otimes B$ we need to find an appropriate decomposition $\Psi = \Gamma, \Delta$ for which we can type-check or prove $\Gamma \vdash A$ and $\Delta \vdash B$. However, because this transformation is so straightforward, when writing informally one generally uses the simpler form with concatenated contexts in the conclusion.

Exercises

Exercise 2.3.1. Prove the well-formedness, cut-admissibility, and initiality theorems for the natural deduction for monoidal posets.

Exercise 2.3.2. Prove that the rules $\otimes L$ and $\mathbf{1}L$ in the sequent calculus for monoidal posets are invertible in the sense of Exercise 1.3.3: whenever we have a derivation of their conclusions, we also have derivations of their premises.

Exercise 2.3.3. Write down either a sequent calculus or a natural deduction for monoidal posets that are also meet-semilattices, and prove its initiality theorem.

Exercise 2.3.4. Let us augment the sequent calculus for monoidal posets by the following versions of the rules for join-semilattices:

$$\frac{\vdash A\ \mathsf{type} \qquad \vdash B\ \mathsf{type}}{\vdash A \vee B\ \mathsf{type}} \qquad \frac{}{\vdash \bot\ \mathsf{type}} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$$

$$\frac{\Gamma, A, \Delta \vdash C \qquad \Gamma, B, \Delta \vdash C}{\Gamma, A \vee B, \Delta \vdash C} \qquad \frac{}{\Gamma, \bot, \Delta \vdash C}$$

(a) Construct derivations in this calculus of the following sequents:

$$(A \otimes B) \vee (A \otimes C) \vdash A \otimes (B \vee C)$$
$$A \otimes (B \vee C) \vdash (A \otimes B) \vee (A \otimes C)$$

(b) Prove that this sequent calculus constructs the initial distributive monoidal poset (see Theorem 2.2.6).

2.4 Multicategories and monoidal categories

Now we are ready to move back up from posets to categories; but here we encounter a bit of an expositional conundrum. We have started with ordinary (non-symmetric, non-cartesian) multicategories since they are simpler from a category-theoretic perspective; in §2.6 we will introduce symmetric and cartesian multicategories by adding extra structure. However, in type theory there are some ways in which the *cartesian* case is the simplest. This is essentially because our intuition tells us that "variables can be used anywhere", whereas in a non-cartesian type theory we have to control how many times a variable is used (and, in the non-symmetric case, what order they are used in). Nevertheless we begin in this section (and the next) with a type theory for ordinary multicategories, as it introduces several important ideas that are clearer without the symmetric and cartesian structure to worry about; but we encourage the reader not to get too bogged down in details.

2.4.1 Multicategories

Categorically, we begin with the adjunction between the category **MCat** of multicategories and the category **MGr** of multigraphs. Let $\mathcal{G}$ be a multigraph; we augment the cut-free theory of §2.3.1 with terms that represent the structure of derivations, as we did in §§1.2, 1.4 and 1.5.

Since our antecedents are now lists of formulas, we assign an abstract variable to *each* of them, and we assign a single term involving these variables to the consequent. Of course, we must assign distinct variables to distinct types in the list (or, more precisely, to distinct *occurrences* of types, since the same type might occur more than once, and these occurrences should be assigned distinct variables).

Thus, for instance, we might have a judgment such as

$$x : A, y : B, z : C \vdash f(x, g(y, z)) : E$$

where $f \in \mathcal{G}(A, D; E)$ and $g \in \mathcal{G}(B, C; D)$. Note that as always, the symbol $\vdash$ is the "outermost". Moreover, the comma between abstract variable assignments binds more loosely than the typing colons; the above judgment should be read as

$$((x : A), (y : B), (z : C)) \vdash (f(x, g(y, z)) : E).$$

As before, the derivation is actually determined by the term associated to the consequent *together with* all the free variables in the context, which we can emphasize by writing

$$xyz.f(x, g(y, z)) : (A, B, C \vdash E).$$

Since we now have multiple formal variables appearing in one sequent, it becomes important to keep track of which is which. As in unary type theory, there are two ways to name variables. In **de Bruijn style** we choose a fixed countably infinite set of variables, say $x_1, x_2, x_3, \ldots,$ and demand that any sequent with n types in its context use the first n of these variables *in order*. In fact there are two choices for this order; we might write

$$x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n \vdash M : B \quad \text{or} \quad x_n : A_n, \ldots, x_2 : A_2, x_1 : A_1 \vdash M : B$$

The first is called using **de Bruijn levels** and the second **de Bruijn indices**.

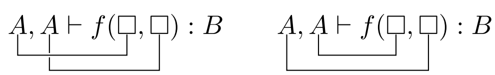
The second way to name variables is to allow arbitrary variables (perhaps taken from some fixed infinite set of variables), but keep track of α-equivalence. This now means that we can rename each variable independently, as long as we rename all of its occurrences at the same time and we don't try to rename any two variables to the same thing. For instance, if $f \in \mathcal{G}(A, A; B)$ then we can write four sequents

$$x : A, y : A \vdash f(x, y) : B \qquad x : A, y : A \vdash f(y, x) : B$$

$$y : A, x : A \vdash f(y, x) : B \qquad y : A, x : A \vdash f(x, y) : B$$

The two in the left column are the same by α-equivalence, and similarly the two in the right column are identical; but the columns are distinct from each other. (In fact, in the type theory of the present section, the sequents in the right-hand column are impossible; but in the theories to be considered in §§2.8 and 2.10 they will be possible.)

Remark 2.4.1. The intent of α-equivalence is that the names or labels of variables are themselves meaningless, but they carry the information of which variable occurrences in a term refer to which variables in the context (or, later, to which variable binding sites). Bourbaki attempted to do away with variable labels entirely, writing all variable occurrences as $\square$ and drawing connecting links to denote these references; thus the two columns above would be written



However, this notation seems unlikely to catch on.

In §2.3 we used capital Greek letters such as Γ to denote finite lists of types. As is also conventional, when we incorporate formal variables we use Γ represent a finite list of types *with variables attached* (with, of course, distinct variables attached to distinct occurrences of types), which is also called a **context**. In general, Γ represents "the sort of thing that can go on the left of $\vdash$".

Now, the rules for multiposets and monoidal posets from §2.3 involve, among other things, concatenation of such lists, which we wrote as Γ, Δ. But when Γ and Δ contain variables, simple concatenation would not preserve the invariant that distinct occurrences of types are labeled by distinct variables, so something else must be going on. If we use de Bruijn style, then the variable numbers in Γ or Δ have to be incremented; we leave the details of this to the interested reader (Exercise 2.4.4). If we instead use arbitrary named variables, as we will generally do, then we simply need to apply α-equivalences to Γ and/or Δ to make their variable names disjoint. (This is an instance of Principle 1.4.3 that term notations for rules can require applying α-equivalences to some premises for compatibility. In §1.4 "compatibility" meant using the *same* variable, while here it means using *different* variables.)

From now on we will write simply Γ, Δ (and similarly $\Gamma, x : A, \Delta$, and so on) for the concatenation of two given contexts, modified to ensure variable distinctness in whatever way is appropriate. Of course, any variable incrementing or α-equivalence that happens in Γ or Δ must also be applied to the consequents of any sequents they appear in. On the other hand, if in some situation we *assume* a sequent and write its context as Γ, Δ, then no such operation is being applied; we are simply choosing a partition of that context into two parts. When applying a rule "top-down", this applies to its premises, while when applying it "bottom-up", this applies to its conclusion (recall Remark 2.3.10).

All this futzing around with variables may seem quite tedious and uninteresting. It does matter in some situations; for instance, if mathematics is to be implemented in a computer, then all these technical issues must be dealt with carefully. However, from our point of view these are all just different tricks to ensure that the terms with formal variables (modulo α-equivalence) remain exact representations of derivation trees. The terms where we have to rename variables and so on are only a *notation* for the mathematical objects of real interest, namely derivations. Remember this if you are ever in doubt about the meaning of variables or what sorts of renamings are possible.

With all of that out of the way, we can anticlimactically state the rules for the **cut-free type theory for multicategories under** $\mathcal{G}$:

$$\frac{A \in \mathcal{G}}{x : A \vdash x : A}$$

$$\frac{f \in \mathcal{G}(A_1, \dots, A_n; B) \qquad \Gamma_1 \vdash M_1 : A_1 \qquad \dots \qquad \Gamma_n \vdash M_n : A_n}{\Gamma_1, \dots, \Gamma_n \vdash f(M_1, \dots, M_n) : B}$$

We note that this theory has the following property.

Lemma 2.4.2. *If* $\Gamma \vdash M : B$ *is derivable, then every variable in* Γ *appears exactly once in* M*.*

Proof. By induction on the derivation. The identity rule $x : A \vdash x : A$ clearly has this property. And in the conclusion of the generator rule each variable appears in exactly one Γ_i, hence can only appear in one of the M_i's, and by induction it appears exactly once there; hence it appears exactly once in $f(M_1, \dots, M_n)$. □

In type-theoretic lingo, Lemma 2.4.2 says that our current type theory is **linear** (just like a linear polynomial uses each variable exactly once, a "linear type theory" uses each variable exactly once). Note that linearity is a property of a system that we *prove*, not a requirement that we impose from outside. It is useful when proving that terms are derivations.

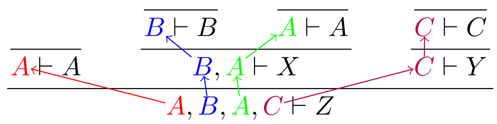
Lemma 2.4.3. *If* $\Gamma \vdash N : B$ *is derivable in the cut-free type theory for multicategories under* $\mathcal{G}$*, then it has a unique derivation.*

Proof. If $N = x$, then the derivation can only be id. And if $N = f(M_1, \dots, M_n)$, then by linearity each variable in Γ must occur in exactly one of the subterms $M_1, \dots, M_n$. If $\Gamma \vdash N : B$ is derivable, then it must be that this partition of Γ is ordered, $\Gamma = \Gamma_1, \dots, \Gamma_n$, and this (together with the known domain $(A_1, \dots, A_n)$ of f) determines the premises $\Gamma_i \vdash M_i : A_i$ that must be recursively checked (c.f. Remark 2.3.10) □

Linearity also has content as a statement about derivations rather than just their terms: it says that each occurrence of a type in the antecedent of a derivable sequent can be "traced back up" exactly one branch of the derivation tree. For instance, in the following derivation

$$\dfrac{\dfrac{}{x:A\vdash x:A} \qquad \dfrac{\dfrac{}{y:B\vdash y:B}\qquad \dfrac{}{z:A\vdash z:A}}{y:B,z:A\vdash g(y,z):X} \qquad \dfrac{\dfrac{}{w:C\vdash w:C}}{w:C\vdash h(w):Y}}{x:A,y:B,z:A,w:C\vdash f(x,g(y,z),h(w)):Z}$$

we can trace the occurrences of types in the antecedent of the conclusion as follows (omitting the variables and terms for brevity):



We now move on to the admissibility of cut/substitution. For this we may again choose between the one-place cut and the multi-cut. We choose the former, because the notation is less heavy, and because it matches the more common path taken in type theory. (The advantage of multi-cut that we saw in §2.3.2 is not relevant for natural deduction, since there are no left rules. We will see something analogous in §2.7, however.) But we encourage the interested reader to write down a multi–substitution too (Exercise 2.4.2).

Theorem 2.4.4. *Substitution is admissible in the cut-free type theory for multicategories under $\mathcal{G}$: given derivations of $\Gamma\vdash M:A$ and of $\Delta,x:A,\Psi\vdash N:B$, we can construct a derivation of $\Delta,\Gamma,\Psi\vdash M[N/x]:B$.*

Proof. This is essentially just Theorem 2.3.1, with terms carried along. There is one thing to be said: since the variables used in any context must be distinct, including the given context $\Delta,x:A,\Psi$, it must be that the variables in Δ and Ψ are pairwise distinct, and all of them are distinct from x. But the variables in Δ,Ψ may not be pairwise distinct from those in Γ, so the context of the desired conclusion $\Delta,\Gamma,\Psi\vdash M[N/x]:B$ may involve an α-equivalence. For instance, if we have $y:C\vdash f(y):A$ and $y:C,x:A,z:D\vdash g(y,x,z):B$, we cannot naively conclude $y:C,y:C,z:D\vdash g(y,f(y),z):B$; we have to rename one of the y's first and get $y:C,w:C,z:D\vdash g(y,f(w),z):B$. We emphasize, however, that this is only a point about the term notation. The proof of Theorem 2.3.1, which doesn't mention variables or terms at all, *is already* an operation on derivations, and the renaming of variables only arises when we notate those derivations in a particular way. □

As before, note that we can regard this as defining substitution; its inductive clauses are

$$\begin{aligned} x[M/x] &= M \\ f(N_1,\dots,N_n)[M/x] &= f(N_1,\dots,N_{i-1},N_i[M/x],N_{i+1},\dots,N_n) \end{aligned}$$

where i is the unique index such that x occurs in N_i (which exists by Lemma 2.4.2).

The one-place substitution operation defined in Theorem 2.4.4 will, of course, give the $\circ_i$ operations in our free multicategory. The index i is specified implicitly by the position of the variable x in the context of N. A similar thing happens with the associativity and interchange axioms.

Theorem 2.4.5. *Substitution in the cut-free type theory for multicategories satisfies the associativity/interchange rules:*

(a) If $\Gamma \vdash M : A$ *and* $\Delta, x : A, \Delta' \vdash N : B$ *and* $\Psi, y : B, \Psi' \vdash P : C$*, then*

$$P[N/y][M/x] = P[N[M/x]/y]$$

(b) If $\Gamma \vdash M : A$ *and* $\Delta \vdash N : B$ *and* $\Psi, x : A, \Psi', y : B, \Psi'' \vdash P : C$*, then*

$$P[N/y][M/x] = P[M/x][N/y]$$

Proof. In both cases we induct on the derivation of P. For (a), if $P = y$ then both sides are $N[M/x]$. If $P = f(P_1, \ldots, P_n)$, suppose y occurs in P_i. Then $P[N/y] = f(P_1, \ldots, P_i[N/y], \ldots, P_n)$ and x occurs in $P_i[N/y]$, so $P[N/y][M/x] = f(P_1, \ldots, P_i[N/y][M/x], \ldots, P_n)$ and the inductive hypothesis applies.

For (b), we can't have P being a single variable since there are two distinct variables in its context. Thus it must be $f(P_1, \ldots, P_n)$, with x and y appearing in P_i and P_j respectively. If $i = j$, then we simply apply the inductive hypothesis to P_i; while if $i \neq j$ then

$$P[N/y][M/x] = f(P_1, \ldots, P_i[M/x], \ldots, P_j[N/y], \ldots, P_n) = P[M/x][N/y] \quad \square$$

If we used de Bruijn levels instead of arbitrary named variables, then the statement of Theorem 2.4.5 would involve the same arithmetic on variable numbers that appears in the $\circ_i$ operations. It is pleasing how the use of abstract variables eliminates this tedious bookkeeping. (It is also possible to eliminate the bookkeeping at the multicategorical level by using an alternative definition of multicategories such as that of [**?**, Appendix A].)

Theorem 2.4.6. *For any multigraph* $\mathcal{G}$*, the free multicategory generated by* $\mathcal{G}$ *can be described by the cut-free type theory for multicategories under* $\mathcal{G}$*: its objects are those of* $\mathcal{G}$*, and its morphisms* $\Gamma \to B$ *are the derivations of* $\Gamma \vdash B$ *(or equivalently, the derivable term judgments* $\Gamma \vdash M : B$ *modulo* α*-equivalence).*

Proof. Theorem 2.4.4 gives us the one-place composition operations, and Theorem 2.4.5 verifies the associativity/interchange axiom for these. The two identity axioms are $x[M/x] = M$ (one of the defining clauses of substitution) and "$M[y/x] = M$", which looks false or nonsensical but is actually just an instance of α-equivalence.

Thus, we have a multicategory $\mathfrak{F}_{\mathbf{MCat}}\mathcal{G}$. Let $\mathcal{M}$ be any multicategory and $P : \mathcal{G} \to \mathcal{M}$ a map of multigraphs; as usual we extend P to the morphisms of $\mathfrak{F}_{\mathbf{MCat}}\mathcal{G}$ by induction on derivations, and such an extension is forced since the rules are all instances of functoriality. Finally we verify by induction on derivations that this extension is in fact functorial on all composites. $\square$

2.4.2 Monoidal categories

We now extend the term notation of §2.4.1 to the natural deduction for monoidal posets from §2.3.3, obtaining a **simple type theory for monoidal categories under $\mathcal{G}$** shown in Figure 2.2.

The rule $\otimes I$, like the rule $\times I$ from §1.4, "pairs up" two derivations of $\Gamma \vdash A$ and $\Delta \vdash B$, and thus must include terms notating both. In this case, however, rather than pulling out the same variable from each, we require that the variables used are disjoint, so that we can concatenate the contexts in the conclusion. Thus once again we are pairing up only the term parts (associated to the consequents), but the variables in the two contexts remain distinct; to emphasize this difference we use a different notation $\{M, N\}$ instead of $\langle M, N\rangle$.

The rule $\otimes E$, on the other hand, is more like the rule $+E$ from §1.5: it has to include terms for both premises, one of which involves variables not appearing in the conclusion. But unlike in §1.5, the term N can contain other variables that remain in the context of the conclusion (and must be disjoint from those in M, by α-equivalence if necessary). We only need to "bind" the other two variables x and y. Thus, for instance, the following application of $\otimes I$:

$$\frac{\begin{array}{c} u : C, v : D \vdash \{f(u), g(v)\} : A \otimes B \\ z : G, x : A, y : B, w : H \vdash h(z, x, y, w) : K \end{array}}{z : G, u : C, v : D, w : H \vdash \mathsf{match}_{\otimes}(\{f(u), g(v)\}, xy.h(z, x, y, w)) : K}$$

produces a term in which the variables z, w in $h(z, x, y, w)$ are free, in addition to the free variables u, v in $\{f(u), g(v)\}$.

Intuitively, because the tensor product has a "mapping out" universal property like a coproduct (that is, it is a "positive type"), its elimination rule is a sort of "case analysis" that decomposes an element of $A \otimes B$ into an element of A and an element of B, rather than a pair of projections. Just as the rule $+E$ says that "to do something with an element of $A+B$, it suffices to assume that it is either $\mathsf{inl}(u)$ or $\mathsf{inr}(v)$", the rule $\otimes E$ says that "to do something with an element of $A \otimes B$, it suffices to assume that it is $\{x, y\}$." And just as the variables u and v are "bound" in the term syntax $\mathsf{match}_{A+B}(M, u.P, v.Q)$ for coproducts, the variables x and y are bound in the term syntax $\mathsf{match}^{\Gamma|\Delta}_{A\otimes B}(M, xy.N)$. The annotations $A \otimes B$ and $\Gamma|\Delta$ are to make type-checking possible (see Lemma 2.4.8); but generally we will omit them and write simply $\mathsf{match}_{\otimes}(M, xy.N)$.

The case of $\mathbf{1}$ is similar but simpler: to do something with an element of $\mathbf{1}$, it suffices to assume that it is $\star$. This gives no extra information, so no new variables are bound. That is, the term syntax $\mathsf{match}_{\mathbf{1}}(M, N)$ binds nothing in N; it simply allows us to ignore M (while keeping the free variables of M in the context).

Like the theory of §2.4.1, this theory is linear:

Lemma 2.4.7. *If $\Gamma \vdash M : B$ is derivable in the simple type theory for monoidal categories under $\mathcal{G}$, then every variable in Γ appears exactly once free in M.*

Proof. By induction on the derivation. The cases of variables and generators are as in Lemma 2.4.2. For a pair $\{M, N\}$ coming from $\otimes I$, each variable in

$$\frac{\vdash A\ \mathsf{type}}{x : A \vdash x : A}$$

$$\frac{f \in \mathcal{G}(A_1, \ldots, A_n; B) \qquad \Gamma_1 \vdash M_1 : A_1 \qquad \ldots \qquad \Gamma_n \vdash M_n : A_n}{\Gamma_1, \ldots, \Gamma_n \vdash f(M_1, \ldots, M_n) : B}$$

$$\frac{\Gamma \vdash M : A \qquad \Delta \vdash N : B}{\Gamma, \Delta \vdash \{M, N\} : A \otimes B} \otimes I$$

$$\frac{\Psi \vdash M : A \otimes B \qquad \Gamma, x : A, y : B, \Delta \vdash N : C}{\Gamma, \Psi, \Delta \vdash \mathsf{match}^{\Gamma|\Delta}_{A\otimes B}(M, xy.N) : C} \otimes E$$

$$\frac{}{() \vdash \star : \mathbf{1}} \mathbf{1}I \qquad\qquad \frac{\Psi \vdash M : \mathbf{1} \qquad \Gamma, \Delta \vdash N : C}{\Gamma, \Psi, \Delta \vdash \mathsf{match}_{\mathbf{1}}(M, N) : C} \mathbf{1}E$$

Figure 2.2: Simple type theory for monoidal categories

Γ, Δ appears in exactly one of Γ and Δ, hence in exactly one of M and N; we then apply the inductive hypotheses to M or N respectively. Similarly, for $\mathsf{match}_{\otimes}(M, xy.N)$ coming from $\otimes E$, each variable in Γ, Ψ, Δ must appear in exactly one of Γ, Ψ, or Δ; by induction then in the first and third cases it must appear exactly once in N, and in the second case it must appear exactly once in M. The case of $\mathbf{1}E$ is similar, while there are no variables at all in $\star$. □

Lemma 2.4.8. *If $\Gamma \vdash N : B$ is derivable in the simple type theory for monoidal categories under $\mathcal{G}$, then it has a unique derivation.*

Proof. The cases of id and f are as in Lemma 2.4.3, and the case of $\otimes I$ is similar, while $\mathbf{1}I$ is trivial. For $\mathsf{match}_{\mathbf{1}}(M, N)$, by linearity each variable occurs in exactly one of M or N. If such a term is derivable, then the variables occurring in M must be contiguous in the context, thereby splitting it as Γ, Ψ, Δ and determining the premises. If it should happen that *no* variables occur in M (such as if $M = \star$), then of course $\Psi = ()$, but the splitting Γ, Δ is not uniquely determined; however since the premise has a re-joined context Γ, Δ anyway this doesn't matter.

In the case of $\otimes E$, however, this latter point makes a difference, because the premise *does* depend on which variables end up in Γ and which in Δ. This is why we have included the $\Gamma|\Delta$ annotation on $\mathsf{match}^{\Gamma|\Delta}_{\otimes}(M, xy.N)$, so that the context splitting is determined even if M contains no variables. (See Exercise 2.4.3.) □

Lemma 2.4.9. *Substitution is admissible in the simple type theory for monoidal categories under $\mathcal{G}$, in the same sense as Theorem 2.4.4. Moreover, it is associative and interchanging in the same sense as Theorem 2.4.5.*

$$\begin{aligned}
x[M/x] &= M \\
f(N_1,\ldots,N_n)[M/x] &= f(N_1,\ldots,N_i[M/x],\ldots,N_n) && \text{if } x \text{ occurs in } N_i \\
\{P,Q\}[M/x] &= \{P[M/x],Q\} && \text{if } x \text{ occurs in } P \\
\{P,Q\}[M/x] &= \{P,Q[M/x]\} && \text{if } x \text{ occurs in } Q \\
\mathsf{match}_{\otimes}(N,uv.P)[M/x] &= \mathsf{match}_{\otimes}(N[M/x],uv.P) && \text{if } x \text{ occurs in } N \\
\mathsf{match}_{\otimes}(N,uv.P)[M/x] &= \mathsf{match}_{\otimes}(N,uv.P[M/x]) && \text{if } x \text{ occurs in } P \\
\star[M/x] & && \text{cannot happen} \\
\mathsf{match}_{\mathbf{1}}(N,P)[M/x] &= \mathsf{match}_{\mathbf{1}}(N[M/x],P) && \text{if } x \text{ occurs in } N \\
\mathsf{match}_{\mathbf{1}}(N,P)[M/x] &= \mathsf{match}_{\mathbf{1}}(N,P[M/x]) && \text{if } x \text{ occurs in } P
\end{aligned}$$

Figure 2.3: Substitution in the simple type theory for monoidal categories

Proof. The method is the same as that of Theorem 2.3.1. Given judgments $\Gamma \vdash M : A$ and $\Delta, x : A, \Psi \vdash N : B$ (involving disjoint variables), we induct on the derivation of N. If the derivation is id, then Δ and Ψ are empty and $N = x$, in which case we can just use M itself. In all other cases, by Lemma 2.4.7 the variable x must appear in exactly one of the premises of the last rule applied to derive N (which is to say, in exactly one of the subterms appearing in N itself), and we inductively perform the substitution there.

Explicitly, the defining clauses of the substitution operation are shown in Figure 2.3. (Technically we also ought to indicate how the $\Gamma|\Delta$ superscripts on $\mathsf{match}_{\otimes}$ are frobnicated, but we leave that to the fastidious reader.) The proof of associativity and interchange is essentially the same as before: all the other rules behave just like the generator rules, except for $\star$ where the claim is trivial. □

There is one final point to be made here about α-equivalence: in the rule $\mathsf{match}_{\otimes}(N,uv.P)[M/x] = \mathsf{match}_{\otimes}(N,uv.P[M/x])$, we must rename variables to ensure that u and v do not appear free in M. Otherwise, such a u or v in M would after substitution be "in the scope" of the binding of u or v, whereas all the free variables of M ought to remain free in the substituted term. (This issue didn't arise in §1.5 because there it was not possible to substitute into the subterms $u.P$ and $v.Q$ of a match_{+} term containing bound variables, since they could not contain any *other* variables to be substituted for.) When we regard substitution as an operation on derivations, the point is that to eliminate a cut after $\otimes E$ of the following sort:

$$\dfrac{\Gamma \vdash M : A \qquad \dfrac{\Xi \vdash N : C \otimes D \qquad \Delta_1, x : A, \Delta_2, u : C, v : D, \Psi \vdash P : B}{\Delta_1, x : A, \Delta_2, \Xi, \Psi \vdash \mathsf{match}_{\otimes}(N, uv.P) : B}\,\otimes E}{\Delta_1, \Gamma, \Delta_2, \Xi, \Psi \vdash \mathsf{match}_{\otimes}(N, uv.P[M/x]) : B}\,\textsc{cut}$$

we have to inductively cut

$$\frac{\Gamma \vdash M : A \qquad \Delta_1, x : A, \Delta_2, u : C, v : D, \Psi \vdash P : B}{\Delta_1, \Gamma, \Delta_2, u : C, v : D, \Psi \vdash P[M/x] : B} \text{ CUT}$$

and in order for *this* cut to satisfy the variable condition explained in Theorem 2.4.4, it must be that u and v do not occur in Γ.

When one takes terms with named variables as primary, this sort of "capture-avoiding substitution" is both necessary and tedious. The de Bruijn methods avoid it, though at a fairly severe cost to readability. But with substitution treated as an operation on derivations, there are no variables to "capture" and nothing to worry about.

With substitution in hand, we can state the β- and η-conversion rules that implement the universal properties.

$$\mathsf{match}_{\otimes}(\{M, N\}, xy.P) \equiv P[M/x, N/y]$$

$$\mathsf{match}_{\otimes}(M, xy.N[\{x, y\}/u]) \equiv N[M/u]$$

$$\mathsf{match}_{\mathbf{1}}(\star, N) \equiv N \qquad\qquad \mathsf{match}_{\mathbf{1}}(M, N[\star/u]) \equiv N[M/u]$$

As before, the β-conversion rule says that the map out of $A \otimes B$ defined by its universal property has the correct composite with the universal morphism $(A, B) \to A \otimes B$, while the η-conversion rule says that any map out of $A \otimes B$ is determined by the universal property from its composite with the universal morphism. The rules for $\mathbf{1}$ are similar.

Theorem 2.4.10. *The free monoidal category generated by a multigraph $\mathcal{G}$ (or, more precisely, its underlying multicategory) can be described by the simple type theory for monoidal categories under $\mathcal{G}$: its objects are the A such that $\vdash A$ type, and its morphisms are the derivations of $\Gamma \vdash A$ (or the derivable judgments $\Gamma \vdash M : A$) modulo the congruence $\equiv$.*

Proof. Lemma 2.4.9 shows that we obtain a multicategory $\mathfrak{F}_{\mathbf{MonCat}}\mathcal{G}$ this way, just as in Theorem 2.4.6. The rules for $\otimes$ and $\mathbf{1}$, together with the β- and η-rules for $\equiv$, tell us that it is representable, and hence a monoidal category. Now if $\mathcal{M}$ is a monoidal category and $P : \mathcal{G} \to \mathcal{M}$ a map of multigraphs, we extend it to $\mathfrak{F}_{\mathbf{MonCat}}\mathcal{G}$ by induction on derivations (of objects and morphisms and equalities) using the fact that $\mathcal{M}$ is a representable multicategory, observe that this definition is forced by functoriality and (strict) preservation of the monoidal structure, and then prove by induction that it is indeed a functor. □

Note that as in Theorem 1.5.3, we have to be careful to do the induction in the right order. Since the rules for equalities refer to substitution, we have to first define the functor on types and terms, then prove that it maps substitution to composition, then define it on equalities. This will be the case for almost all type theories we consider from now on (the case of products in §1.4 is very special in that its equality rules don't need to refer to substitution), so for the most part we will no longer bother to mention it.

Remark 2.4.11. In §1.7 we mentioned that every kind of type theory can be generalized to use an appropriate kind of "presentation" (or "theory") as input. This is true for all the type theories in the current chapter; but we postpone discussion of it for a while, because the notion of 1-skeleton for non-symmetric monoidal categories entails some technical complications that would significantly derail us at this point for very little benefit. We will mention presentations for the posetal case (where the problem of 1-skeletons doesn't arise) briefly in §2.7, and then start dealing with the 1-skeletons in the cartesian case (where they are simpler) in §2.9. In §2.10 we will do the somewhat trickier symmetric case; after which the reader should be able to handle the trickiest non-symmetric case in Exercise 2.10.3.

Exercises

Exercise 2.4.1. Our proof of Theorem 2.4.10 relied on the fact that monoidal categories are equivalent to representable multicategories, which we sketched but did not prove carefully. If we don't assume this fact, then our proof of Theorem 2.4.10 is actually just about free representable multicategories. Using this version of the theorem, prove *using type theory* that any representable multicategory is monoidal: that is, its tensor product is coherently associative and unital.

Exercise 2.4.2. Formulate and prove the admissibility of a "multi-substitution" rule like Theorem 2.3.2 for the type theories considered in this section.

Exercise 2.4.3. The annotation $\Gamma|\Delta$ on $\mathsf{match}^{\Gamma|\Delta}_{A\otimes B}$ is something that appears only in the non-symmetric case, so we encourage the reader not to worry overmuch about it. However, for the reader who nevertheless insists on worrying, here is some extra reassurance.

(a) We noted in Lemma 2.4.8 that this annotation on $\mathsf{match}^{\Gamma|\Delta}_{A\otimes B}(M, xy.N)$ is only necessary if M contains no variables. To see that it can actually matter in that case, find an example of two distinct derivations whose corresponding terms differ *only* in their annotations $\Gamma|\Delta$.

(b) Prove that any two terms as in (a) are related by $\equiv$.

Exercise 2.4.4. Describe precisely what has to happen to de-Bruijn-style variables when concatenating contexts, and formulate the rules for the type theories of this section using de Bruijn variables.

2.5 Adding products and coproducts

Now that we understand the simple type theories of multicategories and monoidal categories, let's add products and coproducts as well. This is where we start to see the value of principle ($*$) from §2.1: for the most part we can just "put together" the rules from §§1.4, 1.5 and 2.4, although there is a little extra work to generalize the rules for products and coproducts to the non-unary case.

In Exercises 2.3.3 and 2.3.4 you studied sequent calculi for monoidal posets with meets and distributive monoidal posets. Now we formulate similar rules in natural deduction style, annotated with terms; the entire **simple type theory for distributive monoidal categories with products** (except for the obvious rules governing the judgment $\vdash A\ \mathsf{type}$) is shown in Figure 2.4. (To obtain theories for monoidal categories with products only, or distributive monoidal categories, or multicategories with products and coproducts, and so on, we can simply omit some of these rules and their corresponding clauses in the following proofs.)

A few things are worth remarking on. Firstly, the types $\otimes, \mathbf{1}, +, \mathbf{0}$ are "positive" (have "mapping out" universal properties), while the types $\times, \mathbb{1}$ are "negative" (have "mapping in" universal properties). All the positive types have elimination rules involving a match that binds variables (perhaps zero of them), while the negative types do not. This is a general feature of the behavior of positive and negative types with respect to abstract variables.

Secondly, as in Exercise 2.3.4, the elimination rules for $\mathbf{0}$ and $A+B$ act on a single type in the context, leaving the others untouched. This corresponds to the definition of coproducts in a multicategory from Theorem 2.2.6.

Thirdly, notice the difference between $\mathbf{1}I$ and $\mathbb{1}I$: both have no premises, but in $\mathbf{1}I$ the context of the conclusion must be empty, whereas in $\mathbb{1}I$ it can be arbitrary. Similarly, the difference between $\otimes I$ and $\times I$ is that in $\otimes I$ the contexts are concatenated in the conclusion, while in $\times I$ both premises must have the same context, which is repeated in the conclusion.

Finally, there are some curious annotations. As in §2.4.2, the superscripts $\Gamma|\Delta$ on $\mathsf{match}_{\otimes}$ and match_{+} are to ensure type-checking, and can usually be omitted; and similarly for the superscript AB on π_i as in §1.4. The superscript Γ, Δ on $\mathsf{match}_{\mathbf{0}}$, however, is there for a different purpose, which is the same purpose as the passing of all the variables in the context as arguments to $*$; it has to do with linearity.

Unlike the theory of §2.4, this type theory is not globally "linear": for instance in $x : A \vdash \langle x, x\rangle : A \times A$ the variable x appears twice. But by including the unused variables in $\mathbb{1}I$ and $\mathbf{0}E$ we can ensure the following weaker property.

Lemma 2.5.1. *In any derivable sequent $\Gamma \vdash M : A$, every variable in Γ appears at least once (free) in the term M.*

Proof. An easy induction over derivations. □

This "superlinearity" property guarantees that terms are derivations.

Lemma 2.5.2. *A derivable sequent $\Gamma \vdash M : A$ uniquely determines a derivation.*

Proof. By induction as usual. The cases involving f, $\otimes$, and $\mathbf{1}$ are essentially just like in Lemma 2.4.8; Lemma 2.5.1 ensures that each variable appears at least once in the term, and if the term is derivable then each variable must appear in only one subterm, determining the context splitting. The cases involving $\times, \mathbb{1}, +, \mathbf{0}$ are straightforward. □

$$\frac{\vdash A\ \mathsf{type}}{x : A \vdash x : A}\ \mathsf{id}$$

$$\frac{f \in \mathcal{G}(A_1, \ldots, A_n; B) \qquad \Gamma_1 \vdash M_1 : A_1 \qquad \ldots \qquad \Gamma_n \vdash M_n : A_n}{\Gamma_1, \ldots, \Gamma_n \vdash f(M_1, \ldots, M_n) : B}\ fI$$

$$\frac{\Gamma \vdash M : A \qquad \Delta \vdash N : B}{\Gamma, \Delta \vdash \{M, N\} : A \otimes B}\ \otimes I$$

$$\frac{\Psi \vdash M : A \otimes B \qquad \Gamma, x : A, y : B, \Delta \vdash N : C}{\Gamma, \Psi, \Delta \vdash \mathsf{match}^{\Gamma|\Delta}_{A \otimes B}(M, xy.N) : C}\ \otimes E$$

$$\frac{}{() \vdash \star : \mathbf{1}}\ \mathbf{1}I \qquad\qquad \frac{\Psi \vdash M : \mathbf{1} \qquad \Gamma, \Delta \vdash N : C}{\Gamma, \Psi, \Delta \vdash \mathsf{match}_{\mathbf{1}}(M, N) : C}\ \mathbf{1}E$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash \langle M, N \rangle : A \times B}\ \times I \qquad\qquad \frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \pi_1^{A,B}(M) : A}\ \times E1$$

$$\frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \pi_2^{A,B}(M) : B}\ \times E2$$

$$\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash *(x_1, \ldots, x_n) : \mathbb{1}}\ \mathbb{1}I \qquad\qquad \frac{\Psi \vdash M : \mathbf{0}}{\Gamma, \Psi, \Delta \vdash \mathsf{match}^{\Gamma,\Delta}_{\mathbf{0}}(M) : C}\ \mathbf{0}E$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \mathsf{inl}(M) : A + B}\ +I1 \qquad\qquad \frac{\Gamma \vdash N : B}{\Gamma \vdash \mathsf{inr}(N) : A + B}\ +I2$$

$$\frac{\Psi \vdash M : A + B \qquad \Gamma, u : A, \Delta \vdash P : C \qquad \Gamma, v : B, \Delta \vdash Q : C}{\Gamma, \Psi, \Delta \vdash \mathsf{match}^{\Gamma|\Delta}_{A+B}(M, u.P, v.Q) : C}\ +E$$

Figure 2.4: Distributive monoidal categories with products

A concrete example where we need the extra arguments to $*$ is:

$$\dfrac{\dfrac{}{x : A \vdash *(x) : \mathbb{1}} \qquad \dfrac{}{\cdot \vdash *() : \mathbb{1}}}{x : A \vdash \{*(x), *()\} : \mathbb{1} \otimes \mathbb{1}} \qquad \dfrac{\dfrac{}{() \vdash *() : \mathbb{1}} \qquad \dfrac{}{x : A \vdash *(x) : \mathbb{1}}}{x : A \vdash \{*(), *(x)\} : \mathbb{1} \otimes \mathbb{1}} \tag{2.5.3}$$

Unlike with the annotations $\Gamma|\Delta$ on matches (see Exercise 2.4.3), these terms really can represent distinct morphisms (see Exercise 2.5.1).

Theorem 2.5.4. *Substitution is admissible in the simple type theory for distributive monoidal categories with products: given derivations of* $\Gamma \vdash M : A$ *and* $\Delta, x : A, \Psi \vdash N : B$*, we can construct a derivation of* $\Delta, \Gamma, \Psi \vdash M[N/x] : B$*. Moreover, it is associative and interchanging.*

Proof. The defining equations are shown in Figure 2.5. They basically augment the rules from Figure 2.3 with versions of the rules from Theorem 1.4.10 and Lemma 1.5.1. Note the difference between the cases for $\{P, Q\}$ and $\langle P, Q\rangle$: in the first we recurse into only one of the subterms, while in the second we recurse into both. Also there are a couple of new rules for $\mathsf{match}_{\mathbf{0}}$ and match_{+} to deal with the fact that a free variable might occur in one of the case branches rather than the discriminee. ☐

The β- and η-conversion rules are likewise obtained by combining those of §§1.4, 1.5 and 2.4; they are shown in Figure 2.6.

Theorem 2.5.5. *The free distributive monoidal category with products generated by a multigraph* $\mathcal{G}$ *is presented by this theory in the usual way: its morphisms are the derivations of* $\Gamma \vdash M$ *(or the derivable terms* $\Gamma \vdash M : A$*) modulo* $\equiv$*.*

Proof. As usual, Theorem 2.5.4 gives us a multicategory, and the rules for the operations $\otimes, \mathbf{1}, \times, *, +, \mathbf{0}$ make it representable and give it products and coproducts. Initiality then follows by the usual induction over derivations. ☐

There are two important things to note here. Firstly, while there are a lot of rules in this type theory, each of them is essentially something we already understood from a previous section, and we were able to put them together essentially independently without worrying about how they interact. This is a good example of the "modularity" of type theory, and the value of principle ($*$) from §2.1.

Secondly, even though the rules for $\otimes$ and $+$ are completely independent, we nevertheless obtained a nontrivial interaction between them (distributivity), *because of the structure of the context* and how it mirrors the categorical notion of multicategory. This suggests that we could obtain further properties and relationships between type operations by modifying the judgmental/context structure. The categorical side of this involves moving to *generalized multicategories*.

Exercises

Exercise 2.5.1. Find an example of a distributive monoidal category with products in which the two terms in (2.5.3) represent distinct morphisms.

$$
\begin{array}{rll}
x[M/x] &= M & \\
f(N_1,\dots,N_n)[M/x] &= f(N_1,\dots,N_i[M/x],\dots,N_n) & \text{if } x \text{ occurs in } N_i \\
\{P,Q\}[M/x] &= \{P[M/x],Q\} & \text{if } x \text{ occurs in } P \\
\{P,Q\}[M/x] &= \{P,Q[M/x]\} & \text{if } x \text{ occurs in } Q \\
\mathsf{match}_{\otimes}(N,uv.P)[M/x] &= \mathsf{match}_{\otimes}(N[M/x],uv.P) & \text{if } x \text{ occurs in } N \\
\mathsf{match}_{\otimes}(N,uv.P)[M/x] &= \mathsf{match}_{\otimes}(N,uv.P[M/x]) & \text{if } x \text{ occurs in } P \\
\star[M/x] & & \text{cannot happen} \\
\mathsf{match}_{\mathbf{1}}(N,P)[M/x] &= \mathsf{match}_{\mathbf{1}}(N[M/x],P) & \text{if } x \text{ occurs in } N \\
\mathsf{match}_{\mathbf{1}}(N,P)[M/x] &= \mathsf{match}_{\mathbf{1}}(N,P[M/x]) & \text{if } x \text{ occurs in } P \\
*(\vec{y},x,\vec{z})[M/x] &= *(\vec{y},\vec{w},\vec{z}) & \vec{w} \text{ the free variables of } M \\
(\pi_1(N))[M/x] &= \pi_1(N[M/x]) & \\
(\pi_2(N))[M/x] &= \pi_2(N[M/x]) & \\
\langle P,Q\rangle[M/x] &= \langle P[M/x],Q[M/x]\rangle & \\
\mathsf{match}_{\mathbf{0}}(N)[M/x] &= \mathsf{match}_{\mathbf{0}}(N[M/x]) & \text{if } x \text{ occurs in } N \\
\mathsf{match}_{\mathbf{0}}(N)[M/x] &= \mathsf{match}_{\mathbf{0}}(N) & \text{if } x \text{ not in } N \\
\mathsf{inl}(N)[M/x] &= \mathsf{inl}(N[M/x]) & \\
\mathsf{inr}(N)[M/x] &= \mathsf{inr}(N[M/x]) & \\
\mathsf{match}_{+}(N,u.P,v.Q)[M/x] &= \mathsf{match}_{+}(N[M/x],u.P,v.Q) & \text{if } x \text{ occurs in } N \\
\mathsf{match}_{+}(N,u.P,v.Q)[M/x] &= \mathsf{match}_{+}(N,u.P[M/x],v.Q[M/x]) & \text{if } x \text{ occurs in } P,Q
\end{array}
$$

Figure 2.5: Substitution for distributive monoidal categories with products

$$\mathsf{match}_{\otimes}(\{M,N\},xy.P) \equiv P[M/x,N/y]$$

$$\mathsf{match}_{\otimes}(M,xy.N[\{x,y\}/u]) \equiv N[M/u]$$

$$\mathsf{match}_{\mathbf{1}}(\star,N) \equiv N \qquad \mathsf{match}_{\mathbf{1}}(M,N[\star/u]) \equiv N[M/u]$$

$$\pi_1(\langle M,N\rangle) \equiv M \qquad \pi_2(\langle M,N\rangle) \equiv N$$

$$\langle \pi_1(M),\pi_2(M)\rangle \equiv M \qquad *(x_1,\dots,x_n) \equiv M$$

$$\mathsf{match}_{+}(\mathsf{inl}(M),u.P,v.Q) \equiv P[M/u] \qquad \mathsf{match}_{+}(\mathsf{inr}(M),u.P,v.Q) \equiv Q[M/v]$$

$$\mathsf{match}_{+}(M,u.P[\mathsf{inl}(u)/y],v.P[\mathsf{inr}(v)/y]) \equiv P[M/y] \qquad \mathsf{match}_{\mathbf{0}}(M) \equiv P[M/y]$$

Figure 2.6: Equality rules for distributive monoidal categories with products

2.6 Some generalized multicategories

We want to consider monoidal categories with “something extra”, such as symmetric monoidal categories or cartesian monoidal categories. To describe a type theory for monoidal categories of this sort, principle (‡) from §2.1 suggests that we should ask what additional structure this “something extra” induces on their underlying multicategories. Because the morphisms $(A_1, \dots, A_n) \to B$ in the underlying multicategory of a monoidal category $\mathcal{C}$ are, by definition, the morphisms $A_1 \otimes \cdots \otimes A_n \to B$ in $\mathcal{C}$, the answer to this question depends on what morphisms between tensor products exist “generically” in monoidal categories of our desired sort. Here are some examples.

(a) If $\mathcal{C}$ is a symmetric monoidal category, we have symmetry isomorphisms $A_1 \otimes \cdots \otimes A_n \xrightarrow{\sim} A_{\sigma 1} \otimes \cdots \otimes A_{\sigma n}$ for any permutation $\sigma \in S_n$. Thus, by precomposing with these isomorphisms, we obtain functions between multicategorical hom-sets

$$\sigma^* : \mathcal{C}(A_{\sigma 1}, \dots, A_{\sigma n}; B) \to \mathcal{C}(A_1, \dots, A_n; B) \tag{2.6.1}$$

that satisfy appropriate axioms.

(b) If $\mathcal{C}$ is a cartesian monoidal category, we have symmetries but also diagonals such as $A \to A \times A$ and projections such as $A \times B \to B$. In general, for any function $\sigma : \{1, \dots, m\} \to \{1, \dots, n\}$ we have a morphism

$$A_1 \times \cdots \times A_n \longrightarrow A_{\sigma 1} \times \cdots \times A_{\sigma m}$$

whose component $A_1 \times \cdots \times A_n \to A_{\sigma k}$ is the projection onto the $(\sigma k)^{\text{th}}$ factor. Precomposition with these morphisms yields analogous functions

$$\sigma^* : \mathcal{C}(A_{\sigma 1}, \dots, A_{\sigma m}; B) \to \mathcal{C}(A_1, \dots, A_n; B). \tag{2.6.2}$$

(c) Less well-known than symmetric and cartesian monoidal categories are *semicartesian* monoidal categories, whose unit object is the terminal object, but whose tensor product is not necessarily the cartesian product. (An example familiar to higher category theorists is the category **2Cat** with its Gray tensor product.) We will always assume that semicartesian monoidal categories are additionally symmetric. The semicartesianness gives us projections but not diagonals, leading to functions (2.6.2) whenever σ is *injective*.

(d) Even less well-known are *relevance* monoidal categories, which are symmetric and equipped with a coherent system of diagonals $A \to A \otimes A$ but whose unit object is not in general terminal. A familiar example is the category of pointed sets with its smash product [**?**]. In this case we have functions (2.6.2) only when σ is *surjective*.

All of these cases can be encompassed by the following definitions.

Definition 2.6.3. Let $\mathfrak{N}$ be the full subcategory of **Set** whose objects are the sets $\{1,\ldots,n\}$ for all integers $n \geq 0$. We regard it as a *cocartesian* strict monoidal category, under the disjoint union operation $\{1,\ldots,n\} \sqcup \{1,\ldots,m\} = \{1,\ldots,n+m\}$. Moreover, for any $\sigma : \{1,\ldots,m\} \to \{1,\ldots,n\}$ and $k_1,\ldots,k_n$, let $\sigma \wr (k_1,\ldots,k_n)$ denote the composite function

$$\{1,\ldots,\textstyle\sum_{i=1}^m k_{\sigma i}\} \xrightarrow{\sim} \bigsqcup_{i=1}^m \{1,\ldots,k_{\sigma i}\} \xrightarrow{\widehat{\sigma}} \bigsqcup_{j=1}^n \{1,\ldots,k_j\} \xrightarrow{\sim} \{1,\ldots,\textstyle\sum_{j=1}^n k_j\}$$

where $\widehat{\sigma}$ acts as the identity from the i^{th} summand to the $(\sigma i)^{\text{th}}$ summand. A **faithful cartesian club** is a subcategory $\mathfrak{S} \subseteq \mathfrak{N}$ such that

(a) $\mathfrak{S}$ contains all the objects of $\mathfrak{N}$.

(b) $\mathfrak{S}$ is closed under the cocartesian monoidal structure, i.e. if σ and τ are morphisms of $\mathfrak{S}$ then so is $\sigma \sqcup \tau$.

(c) $\mathfrak{S}$ is closed under $\wr$, i.e. whenever it contains σ it also contains $\sigma \wr (k_1,\ldots,k_n)$.

The above examples are the cases when $\mathfrak{S}$ consists of the bijections, all the functions, the injections, or the surjections respectively. There is also the trivial case when $\mathfrak{S}$ contains only the identities.

Definition 2.6.4. Let $\mathfrak{S}$ be a faithful cartesian club. An **$\mathfrak{S}$-multicategory** is a multicategory $\mathcal{M}$ together with operations

$$\begin{aligned} \mathcal{M}(A_{\sigma 1},\ldots,A_{\sigma m};B) &\to \mathcal{M}(A_1,\ldots,A_n;B) \\ f &\mapsto f\sigma^* \end{aligned}$$

for all functions $\sigma : \{1,\ldots,m\} \to \{1,\ldots,n\}$ in $\mathfrak{S}$, satisfying the following axioms:

(a) $f\sigma^*\tau^* = f(\tau\sigma)^*$

(b) $f(\mathsf{id}_n)^* = f$

(c) $g \circ (f_1\sigma_1^*,\ldots,f_n\sigma_n^*) = (g \circ (f_1,\ldots,f_n))(\sigma_1 \sqcup \cdots \sqcup \sigma_n)^*$

(d) $g\sigma^* \circ (f_1,\ldots,f_n) = (g \circ (f_{\sigma 1},\ldots,f_{\sigma m}))(\sigma \wr (k_1,\ldots,k_n))^*$ where k_i is the arity of f_i.

If each hom-set $\mathcal{M}(A_1,\ldots,A_n;B)$ has at most one element, we call $\mathcal{M}$ an **$\mathfrak{S}$-multiposet**. An **$\mathfrak{S}$-multigraph** is a multigraph equipped with similar operations satisfying (a) and (b).

As special cases we have, by definition:

When $\mathfrak{S}$ =	$\mathfrak{S}$-multicategories are called
bijections	**symmetric multicategories**
all functions	**cartesian multicategories**
injections	**semicartesian (symmetric) multicategories**
surjections	**relevance multicategories**
only identities	(ordinary) multicategories

Now, recall the definition of tensor products in a multicategory from Definition 2.2.3, and the result of Theorem 2.2.4 that having all tensor products (being "representable") is equivalent to being a monoidal category. For a general faithful cartesian club $\mathfrak{S}$, we might as well *define* an **$\mathfrak{S}$-monoidal category** to be an $\mathfrak{S}$-multicategory that is representable. However, in many cases this is equivalent to a more familiar notion.

Theorem 2.6.5. *If $\mathfrak{S}$ includes all bijections, then the monoidal category obtained from any representable $\mathfrak{S}$-multicategory is symmetric. Moreover, the equivalence of Theorem 2.2.4 induces an equivalence between representable symmetric multicategories and symmetric monoidal categories.*

Proof. If $\chi : (A, B) \to A \otimes B$ is a tensor product, then by acting on it with the transposition $\sigma : \{1, 2\} \xrightarrow{\sim} \{1, 2\}$ we obtain a morphism $\chi\sigma^* : (B, A) \to A \otimes B$. Applying the universal property of the tensor product $(B, A) \to B \otimes A$, we get a map $B \otimes A \to A \otimes B$. We can similarly use the universal property to check the symmetry axioms.

Conversely, the coherence theorem for symmetric monoidal categories yields isomorphisms (2.6.1), composing with which gives its underlying multicategory a symmetric structure. It is straightforward to verify that these constructions are inverses up to isomorphism. □

Recall from §2.2 the definition of products in a multicategory.

Theorem 2.6.6. *If $\mathfrak{S}$ includes all injections, then an object $\mathbf{1}$ is terminal if and only if it is a unit object (i.e. there is a universal tensor product morphism $() \to \mathbf{1}$). Moreover, the equivalence of Theorem 2.2.4 induces an equivalence between representable semicartesian multicategories and semicartesian monoidal categories.*

Proof. If $\mathfrak{S}$ includes injections, then for any $A_1, \dots, A_n$ the injection $\emptyset \to \{1, \dots, n\}$ induces a map

$$\mathcal{M}(;B) \to \mathcal{M}(A_1, \dots, A_n; B).$$

Thus, if $\mathbf{1}$ is a unit object with universal morphism $\chi : () \to \mathbf{1}$, then this gives us induced maps $e_{A_1,\dots,A_n} : (A_1, \dots, A_n) \to \mathbf{1}$. Moreover, the fourth "equivariance" axiom of an $\mathfrak{S}$-multicategory implies that these maps are natural, in the sense that $e_{A_1,\dots,A_n} \circ (f_1, \dots, f_n) = e_{B_1,\dots,B_m}$ for any $f_1, \dots, f_n$. In particular, $e_{\mathbf{1}} \circ \chi = e_{()} = \chi$; so by the universal property of χ, we have $e_{\mathbf{1}} = \mathsf{id}_{\mathbf{1}}$. A standard argument (generalized from categories to multicategories) now implies that $\mathbf{1}$ is terminal.

Conversely, suppose $\mathbb{1}$ is terminal. Then in particular, we have a unique morphism $\chi : () \to \mathbb{1}$, and acting on χ by the injection $\emptyset \to \{1, \dots, n\}$ can only yield the unique morphism $(A_1, \dots, A_n) \to \mathbb{1}$. Now we have to show that

$$(- \circ_{n+1} \chi) : \mathcal{M}(A_1, \dots, A_n, \mathbb{1}, B_1, \dots, B_m; C) \to \mathcal{M}(A_1, \dots, A_n, B_1, \dots, B_m; C)$$

is a bijection. But we have a map in the other direction given by acting with an appropriate injection, and the equivariance properties imply that this is an inverse.

Lastly, if we have a semicartesian monoidal category, then for any injection σ we have a map

$$A_1 \otimes \cdots \otimes A_n \longrightarrow A_{\sigma 1} \otimes \cdots \otimes A_{\sigma m}$$

defined by mapping each A_j not in the image of σ to the terminal object 1, then removing those copies of 1 from the tensor product since they are also the tensor unit (and finally permuting if necessary). It is straightforward to verify that these actions give a semicartesian multicategory, and that that these constructions are inverses up to isomorphism. □

Theorem 2.6.7. *If $\mathfrak{S}$ consists of all functions (i.e. we are in a cartesian multicategory), then products $A \times B$ are in bijective correspondence with tensor products $A \otimes B$. Moreover, the equivalence of Theorem 2.2.4 induces an equivalence between representable cartesian multicategories and cartesian monoidal categories (i.e. categories with finite products).*

Proof. By acting with injections, for any A, B we obtain morphisms $(A, B) \to A$ and $(A, B) \to B$. Thus, if $A \times B$ is a product, we have an induced map $\chi : (A, B) \to A \times B$. Now if we have any morphism $(C_1, \ldots, C_n, A, B, D_1, \ldots, D_m) \to E$, we can compose with the two projections of the product to get a morphism $(C_1, \ldots, C_n, A \times B, A \times B, D_1, \ldots, D_m) \to E$, and then act by a surjection to get $(C_1, \ldots, C_n, A \times B, D_1, \ldots, D_m) \to E$. The equivariance properties of a cartesian multicategory, and the universal property of the product, imply that this operation is inverse to composing with χ, so that the latter is a tensor product.

Conversely, if $\chi : (A, B) \to A \otimes B$ is a tensor product, by applying its universal property to the above morphisms $(A, B) \to A$ and $(A, B) \to B$ we obtain projections $A \otimes B \to A$ and $A \otimes B \to B$. Now given $f : (C_1, \ldots, C_n) \to A$ and $g : (C_1, \ldots, C_n) \to B$, we have $\chi \circ (f, g) : (C_1, \ldots, C_n, C_1, \ldots, C_n) \to A \otimes B$, and by acting with a suitable surjection we get $(C_1, \ldots, C_n) \to A \otimes B$. Again, the equivariance properties and the universal property of the tensor product imply that this is a unique factorization of f and g through the projections. □

Note although the first conclusion of Theorem 2.6.7 refers only to binary products, it still requires the presence of *injections* in $\mathfrak{S}$ in addition to surjections. Indeed, the monoidal category of pointed sets with its smash product has an underlying multicategory that is relevance (i.e. admits an action by all surjections), but the smash product is different from the cartesian product. It is also possible to characterize the $\mathfrak{S}$-monoidal categories when $\mathfrak{S}$ is the injections, but we leave this to the interested reader; see Exercise 2.6.6.

Remark 2.6.8. Theorems 2.6.6 and 2.6.7 identify an object having a "mapping out" universal property (a tensor product or unit object in a multicategory) with an object having a "mapping in" universal property (a cartesian product or terminal object), in the strong sense that if either exists then it is also the other.

This sort of "ambidextrous" universal property appears elsewhere in category theory as well. For instance, the splitting of an idempotent can be regarded as either a limit or a colimit; in a category enriched over abelian monoids, finite products and coproducts coincide; and more generally for any kind of enrichment there is a notion of "absolute (co)limit" [**?**]. Thus, although multicategories are not "enriched categories" in the usual sense, we could say informally that in a cartesian multicategory products are absolute limits, while in a semicartesian multicategory terminal objects are. See also Exercise 2.6.5.

Finally, we observe that closedness can be naturally characterized multicategorically. Suppose for simplicity that $\mathfrak{S}$ contains at least all bijections. Then we say an $\mathfrak{S}$-multicategory is **closed** if for each pair of objects A and B there is a specified object $A \multimap B$ and a morphism $\chi : (A \multimap B, A) \to B$ postcomposition with which defines bijections

$$(\chi \circ_1 -) : \mathcal{M}(C_1, \dots, C_n; A \multimap B) \xrightarrow{\sim} \mathcal{M}(C_1, \dots, C_n, A; B)$$

for all $C_1, \dots, C_n$. (If $\mathfrak{S}$ does not contain the bijections, we would just have to consider "left and right closedness" separately.) The following is then straightforward.

Theorem 2.6.9. *A symmetric monoidal category is closed if and only if its underlying multicategory is. Moreover, for all the above values of* $\mathfrak{S}$ *that contain the bijections, this defines an equivalence of categories.* □

Of course, cartesian closed categories are just closed cartesian monoidal categories, so they are equivalent to closed cartesian multicategories.

Exercises

Exercise 2.6.1. Fill in the details in the proof of Theorems 2.6.5 to 2.6.7.

Exercise 2.6.2. Let $\mathfrak{S}$ be a faithful cartesian club.

(a) Prove that if $\mathfrak{S}$ contains the transposition $\{1,2\} \xrightarrow{\sim} \{1,2\}$, then it contains all bijections.

(b) Prove that if $\mathfrak{S}$ contains the transposition $\{1,2\} \xrightarrow{\sim} \{1,2\}$ and also the injection $\emptyset \to \{1\}$, then it contains all injections.

(c) Prove that if $\mathfrak{S}$ contains the transposition $\{1,2\} \xrightarrow{\sim} \{1,2\}$ and also the surjection $\{1,2\} \to \{1\}$, then it contains all surjections.

Exercise 2.6.3. Define one-place versions of $\mathfrak{S}$-multicategories and show that they are equivalent to the multi-composition version defined in the text.

Exercise 2.6.4. Show that representable cartesian multicategories with coproducts are equivalent to distributive categories.

Exercise 2.6.5. Of course, for any $\mathfrak{S}$ a **functor** between $\mathfrak{S}$-multicategories is required to preserve the σ-actions. Prove that:

(a) Any functor between semicartesian multicategories must preserve unit objects / terminal objects.

(b) Any functor between cartesian multicategories must preserve tensor products / cartesian products.

Exercise 2.6.6. Define a notion of **relevance monoidal category**, by adding "natural diagonals" to a symmetric monoidal category, and show that such monoidal categories are equivalent to representable relevance multicategories. (See [**?**].)

Exercise 2.6.7. Define a notion of **faithful cocartesian club** and a corresponding notion of generalized multicategory that includes *cocartesian* monoidal categories as the maximal case.

2.7 Intuitionistic logic

We are now aiming at type theories for the generalized multicategories considered in §2.6, along with the extra structures that they may have (tensor products, cartesian products, coproducts, and closedness). In this section we start with the posetal case, which is also where our type theory at last begins to look rather like *logic*.

2.7.1 $\mathfrak{S}$-monoidal lattices

According to principle (‡) from §2.1, the additional action by σ's in an $\mathfrak{S}$-multicategory should be represented by *structural rules* in a type theory. These rules are generally formulated and named as follows.

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\ \textsc{exchange} \qquad\qquad \frac{\Gamma, \Delta \vdash C}{\Gamma, A, \Delta \vdash C}\ \textsc{weakening}$$

$$\frac{\Gamma, A, A, \Delta \vdash C}{\Gamma, A, \Delta \vdash C}\ \textsc{contraction}$$

The correspondence between kinds of multicategory and structural rules[2] should not be surprising:

$\mathfrak{S}$-multicategories	structural rules
symmetric multicategories	exchange
cartesian multicategories	exchange, weakening, contraction
semicartesian (symmetric) multicategories	exchange, weakening
relevance multicategories	exchange, contraction

[2]One can consider weakening and/or contraction without exchange, just as one might consider non-symmetric semicartesian or relevance multicategories. But this takes us rather far afield from categorical structures of general interest, so we leave it to the reader.

Note that these structural rules refer only to single transpositions, projections, and duplications, rather than arbitrary functions in $\mathfrak{S}$. This is similar to how, as we have noted, cut is usually stated in type theory using one-place composites rather than a multi-composition. As in that case, the smaller operations suffice to generate the more general ones (c.f. Exercise 2.6.2).

What is somewhat less clear is how these rules can be made *admissible* in line with principle (§). For now let us ignore this question and take these rules (when we want them) as *primitive* (recall Remark 1.2.6). This makes the treatment more parametric in $\mathfrak{S}$, and makes little difference for presenting free posets, since in that case we are only interested in the existence or nonexistence of derivations. We will address the question of admissibility in §§2.7.2, 2.8 and 2.10.

For the rest of this subsection, let $\mathfrak{S}$ be one of the four possibilities above (so in particular, it will always contain the bijections). All our type theories will then include the appropriate primitive structural rules, according to the above table. Our type operations will be the posetal versions of all the ones we saw in §2.5 — $\otimes, \mathbf{1}, \wedge, \top, \vee, \bot$ — and also an internal-hom for $\otimes$, which we denote by $A \multimap B$. (We postponed introducing the internal-hom until now only to avoid worrying about left- versus right-closedness in non-symmetric multicategories.) Thus, the categorical structure in question is *closed* $\mathfrak{S}$*-monoidal lattices*. Of course, as in §2.5 we can remove any of these operations without affecting the others, obtaining a type theory for weaker categorical structures.

The primitive rules of the **natural deduction for closed $\mathfrak{S}$-monoidal lattices** are shown in Figure 2.7. Except for the structural rules (discussed above) and $\multimap$, they are all obtained by removing the term annotations from the theory of §2.5. The only other change is that since we always include the exchange rule as primitive, in the rules $\otimes E, \mathbf{1}E, \vee E, \mathbf{0}E$ we don't need to put Ψ in the middle of the context but are free to put it on one side. As usual, we have also omitted the rules for the judgment $\vdash A\ \mathsf{type}$, which just say that all the objects of $\mathcal{G}$ are types, as are $\mathbf{1}, \top, \bot$ and $A \otimes B, A \wedge B, A \vee B, A \multimap B$ if A and B are.

The introduction rule for $\multimap$ is simply one direction of its universal property from §2.2. The elimination rule is the inverse direction, but with a cut built in to make the context of the conclusion general (modulo a splitting). That is, $\multimap E$ can be derived from the opposite of $\multimap I$ and cut:

$$\dfrac{\Psi \vdash A \qquad \dfrac{\Gamma \vdash A \multimap B}{\Gamma, A \vdash B}}{\Gamma, \Psi \vdash B}$$

Note that, as promised in §2.1, by using sequents with multiple types in the context, we can formulate the rules for $\multimap$ without reference to $\wedge/\times$.

The contraction rule gives the cut-admissibility theorem a new wrinkle. Let us first consider the cases without contraction, which are more straightforward.

Lemma 2.7.1. *If $\mathfrak{S}$ consists of the bijections or the injections, then cut is admissible in the natural deduction for closed $\mathfrak{S}$-monoidal lattices: if we have derivations of $\Psi \vdash A$ and $\Gamma, A, \Delta \vdash B$ then we also have $\Gamma, \Psi, \Delta \vdash B$.*

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\ \textsc{exchange}$$

$$\frac{\Gamma, \Delta \vdash C}{\Gamma, A, \Delta \vdash C}\ \textsc{weakening}\ \text{if injections} \subseteq \mathfrak{S}$$

$$\frac{\Gamma, A, A, \Delta \vdash C}{\Gamma, A, \Delta \vdash C}\ \textsc{contraction}\ \text{if surjections} \subseteq \mathfrak{S}$$

$$\frac{\vdash A\ \mathsf{type}}{A \vdash A} \qquad \frac{(A_1, \dots, A_n \le B) \in \mathcal{G} \qquad \Gamma_1 \vdash A_1 \qquad \dots \qquad \Gamma_n \vdash A_n}{\Gamma_1, \dots, \Gamma_n \vdash B}$$

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ \otimes I \qquad \frac{\Psi \vdash A \otimes B \qquad \Gamma, A, B \vdash C}{\Gamma, \Psi \vdash C}\ \otimes E$$

$$\frac{}{() \vdash \mathbf{1}}\ \mathbf{1}I \qquad \frac{\Psi \vdash \mathbf{1} \qquad \Gamma \vdash A}{\Gamma, \Psi \vdash C}\ \mathbf{1}E$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge I \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge E1 \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge E2$$

$$\frac{}{\Gamma \vdash \top}\ \top I \qquad \frac{\Psi \vdash \bot}{\Gamma, \Psi \vdash C}\ \bot E$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee I1 \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee I2$$

$$\frac{\Psi \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, \Psi \vdash C}\ \vee E$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ \multimap I \qquad \frac{\Psi \vdash A \qquad \Gamma \vdash A \multimap B}{\Gamma, \Psi \vdash B}\ \multimap E$$

Figure 2.7: Natural deduction for closed $\mathfrak{S}$-monoidal lattices

Proof. As always, we induct on the derivation of $\Gamma, A, \Delta \vdash B$. The cases for most of the connectives are just like those in Theorem 2.5.4, and those for $\multimap$ are nothing new. However, now we have a new possibility: the derivation might end with a primitive structural rule (exchange or weakening — our hypothesis on $\mathfrak{S}$ rules out contraction).

Firstly, if the structural rule does not affect the type A, then we can simply commute it past the cut. For instance, if we have $\Psi \vdash A$ and $\Gamma, A, \Delta_1, C, \Delta_2 \vdash B$ arising by weakening from $\Gamma, A, \Delta_1, \Delta_2 \vdash B$, we can inductively obtain $\Gamma, \Psi, \Delta_1, \Delta_2 \vdash B$ and then apply weakening again to get $\Gamma, \Psi, \Delta_1, C, \Delta_2 \vdash B$.

Secondly, essentially the same is true if it is an exchange that does affect A. For instance, if we have $\Psi \vdash A$ and $\Gamma, A, C, \Delta \vdash B$ arising by exchange from $\Gamma, C, A, \Delta \vdash B$, we can inductively obtain $\Gamma, C, \Psi, \Delta \vdash B$, and then re-apply exchange once for each type in Ψ to get $\Gamma, \Psi, C, \Delta \vdash B$. (It does matter here that we have formulated the admissible cut rule with A in the middle of the context rather than on one side, even though we have the exchange rule; otherwise the induction would fail to go through here.)

Finally, suppose it is a weakening that affects A, so we have $\Psi \vdash A$ and $\Gamma, A, \Delta \vdash B$ arising by weakening from $\Gamma, \Delta \vdash B$. In this case we can forget about the derivation of $\Psi \vdash A$ and just weaken $\Gamma, \Delta \vdash B$ once for each type in Ψ to get $\Gamma, \Psi, \Delta \vdash B$. □

If we try to extend this to theories with contraction, however, we have a problem. Suppose the derivation of $\Gamma, A, \Delta \vdash B$ ends with a contraction that affects A, so that we have $\Psi \vdash A$ and $\Gamma, A, \Delta \vdash B$ arising by contraction from $\Gamma, A, A, \Delta \vdash B$. Then we would like to inductively cut the latter with $\Psi \vdash A$ twice to obtain $\Gamma, \Psi, \Psi, \Delta \vdash B$, transforming

$$\dfrac{\Psi \vdash A \qquad \dfrac{\Gamma, A, A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}\,\textsc{contraction}}{\Gamma, \Psi, \Delta \vdash B}\,\textsc{cut}$$

into

$$\dfrac{\Psi \vdash A \qquad \dfrac{\Psi \vdash A \qquad \Gamma, A, A, \Delta \vdash B}{\Gamma, \Psi, A, \Delta, \vdash B}\,\textsc{cut}}{\Gamma, \Psi, \Psi, \Delta \vdash B}\,\textsc{cut}$$

After this we could apply exchanges to pair up the two copies of each type in Ψ, and finally a contraction on each of them to eliminate the duplicates. However, now we have the sort of problem that we did in the proof of Theorem 2.3.5: the derivation of $\Gamma, \Psi, A, \Delta, \vdash B$ that we obtain from our first application of the inductive hypothesis may not be "smaller" than our given derivation, so we cannot apply the inductive hypothesis to it again. Moreover, the solution sketched there (inducting first on types and then on derivations) does not work here, since the types are not changing.

The standard solution used in type theory is to generalize the cut rule to a rule called "mix" that enables the induction to go through. In our case, the mix rule says that if we have derivations of $\Psi \vdash A$ and $\Gamma \vdash B$, where Γ contains one or more copies of A, then we can construct a derivation of $\Psi, \Gamma^A \vdash B$, where Γ^A is Γ with one or more copies of A removed. In other words, we build a certain amount of contraction into the induction hypothesis. This works, but a more categorically principled solution is to use the multi-cut as in Theorem 2.3.8. This amounts to approximately the same thing, but feels less *ad hoc* to a category theorist (at least, it does to the author).

Lemma 2.7.2. *For any of our four $\mathfrak{S}$'s, multi-cut is admissible in the natural deduction for closed $\mathfrak{S}$-monoidal lattices: if we have derivations of $\Psi_i \vdash A_i$ for $1 \le i \le n$, and also $A_1, \ldots, A_n \vdash B$, then we can construct a derivation of $\Psi_1, \ldots, \Psi_n \vdash B$.*

Proof. The non-structural rules are easy, just as before. (Recall that in general, cut is very straightforward for natural deductions because all the rules act only on the right. With this in mind it is unsurprising that primitive structural rules are problematic, since they act on the left.)

Now, however, the structural rules are almost just as easy. If our derivation of $A_1, \ldots, A_n \vdash B$ ends with an exchange, we can simply switch two of the derivations $\Psi_i \vdash A_i$ and induct. Similarly, if it ends with a weakening, we can just forget about one of the $\Psi_i \vdash A_i$ and induct. Finally, if it ends with a contraction, we can again induct on the premise, using one of the derivations $\Psi_i \vdash A_i$ twice. □

Now we can prove the initiality theorem just as usual.

Theorem 2.7.3. *For any relational multigraph $\mathcal{G}$ and any of our four $\mathfrak{S}$'s, the free closed $\mathfrak{S}$-monoidal lattice on $\mathcal{G}$ can be presented by this natural deduction, with $(A_1, \ldots, A_n) \le B$ holding just when $A_1, \ldots, A_n \vdash B$ is derivable.*

Proof. Lemma 2.7.2 (together with the identity rule) gives us a multiposet, the rules for the type operations make it representable, closed, and a lattice, and the structural rules make it an $\mathfrak{S}$-multiposet. Thus it lives in the correct category; and its freeness follows by induction as usual. □

We can also generalize from relational multigraphs to an appropriate kind of "presentation" as in §1.7. Since we are in the posetal case, things are much simpler because everything coincides with its 1-skeleton. Specifically, a **relational $(\otimes, \wedge, \vee, \multimap)$-presentation** consists of

(a) A set $\mathcal{P}_0$ of objects; and

(b) A relation $\mathcal{P}_1$ between finite lists of types and single types, where the types are generated from $\mathcal{P}_0$ by the rules for the judgment $\vdash A\ \mathsf{type}$ in the type theory for closed ($\mathfrak{S}$-)monoidal lattices.

We can then construct a two-level tower of adjunctions generating a closed $\mathfrak{S}$-monoidal lattice from any such presentation. The only other thing to be said about this is that in the posetal case, it is common to refer to the relations in $\mathcal{P}_1$ as the **axioms**, since they are mere properties, in contrast to how in §1.7.4 we used that word for the generating equalities (at level 2 rather than level 1).

2.7.2 Heyting algebras

Let us now specialize to the cartesian case, where we have all three structural rules. Thus the categorical structure in question is *cartesian closed lattices*, which are also known as **Heyting algebras**. This theory is simpler because $\otimes$ and $\mathbf{1}$ coincide with $\wedge$ and $\top$ (see Exercise 2.7.1), so we can omit the former ones. A second reason it is simpler is because it is easy to make the structural rules admissible. The key observation is the following.

Lemma 2.7.4. *In the presence of exchange, contraction, and weakening, the following rules are inter-derivable with the rules $\bot E$, $\vee E$, $\multimap E$ from Figure 2.7.*

$$\frac{\vdash A\ \mathsf{type} \qquad A \in \Gamma}{\Gamma \vdash A}\ \mathsf{id}'$$

$$\frac{(A_1,\dots,A_n \le B) \in \mathcal{G} \qquad \Gamma \vdash A_1 \qquad \dots \qquad \Gamma \vdash A_n}{\Gamma \vdash B}\ f' \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash C}\ \bot E'$$

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee E' \qquad \frac{\Gamma \vdash A \multimap B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \multimap E'$$

Proof. Here are the referenced rules from Figure 2.7:

$$\frac{\vdash A\ \mathsf{type}}{A \vdash A}\ \mathsf{id} \qquad \frac{(A_1,\dots,A_n \le B) \in \mathcal{G} \qquad \Gamma_1 \vdash A_1 \qquad \dots \qquad \Gamma_n \vdash A_n}{\Gamma_1,\dots,\Gamma_n \vdash B}\ f$$

$$\frac{\Psi \vdash \bot}{\Gamma, \Psi \vdash C}\ \bot E \qquad \frac{\Psi \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, \Psi \vdash C}\ \vee E$$

$$\frac{\Gamma \vdash A \multimap B \qquad \Psi \vdash A}{\Gamma, \Psi \vdash B}\ \multimap E$$

Clearly id is a special case of id', while conversely id' can be derived from id followed by weakening. And $\bot E'$ is a special case of $\bot E$, while f' and $\vee E'$ and $\multimap E'$ can be derived from f and $\vee E$ and $\multimap E$ followed by exchange and contraction to turn contexts like Γ, Γ into Γ. Conversely, given the premises of any of these "unprimed" rules, we can weaken each Γ and Ψ to Γ, Ψ (or Γ_i to $\Gamma_1, \dots, \Gamma_n$ in the case of f), then apply the primed version of that rule to deduce the conclusion of the unprimed rule. □

If we replace the rules in question by their modified versions, then all the rules will have the property that the context of the conclusion is arbitrary, while the context of the premises differ from the context of the conclusion at most by addition of a new type. In other words, as we proceed *down* a derivation tree, we only ever *remove* types from the context; and dually as we proceed *up* a tree we only ever *add* to the context. This will enable us to "push the structural rules up" past all primitive rules until we get to id, thereby making them admissible.

$$\frac{\vdash A\ \mathsf{type} \qquad A \in \Gamma}{\Gamma \vdash A}\ \mathsf{id}$$

$$\frac{(A_1, \ldots, A_n \leq B) \in \mathcal{G} \qquad \Gamma \vdash A_1 \qquad \ldots \qquad \Gamma \vdash A_n}{\Gamma \vdash B}\ f$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ {\wedge}I \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ {\wedge}E1 \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ {\wedge}E2$$

$$\frac{}{\Gamma \vdash \top}\ {\top}I \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash C}\ {\bot}E$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ {\vee}I1 \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ {\vee}I2$$

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}\ {\vee}E$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ {\Rightarrow}I \qquad \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ {\Rightarrow}E$$

Figure 2.8: Natural deduction for Heyting algebras

For convenience, we collect all the rules of this modified **natural deduction for Heyting algebras** in Figure 2.8. Note that we change our notation and write $A \multimap B$ as $A \Rightarrow B$.

Lemma 2.7.5. *All the structural rules of exchange, weakening, and contraction are admissible in the natural deduction for Heyting algebras.*

Proof. It will suffice to prove admissibility of the following rule, for any function $\sigma : \{1, \ldots, m\} \to \{1, \ldots, n\}$:

$$\frac{A_{\sigma 1}, \ldots, A_{\sigma m} \vdash B}{A_1, \ldots, A_n \vdash B}$$

This is almost immediate from the fact that the premises of all rules have the same context as the conclusion, perhaps with a type added: regardless of how a derivation of $A_{\sigma 1}, \ldots, A_{\sigma m} \vdash B$ ends, we can apply the inductive hypothesis to its premises (perhaps passing to $\sigma \sqcup \mathsf{id} : \{1, \ldots, m+1\} \to \{1, \ldots, n+1\}$) and then re-apply the final rule.

The only exception is the rule id, for which we observe that if A appears in the context $A_{\sigma 1}, \ldots, A_{\sigma m}$, then $A = A_{\sigma j}$ for some $1 \leq j \leq m$, and hence $A = A_i$ for some $1 \leq i \leq n$ (namely, $i = \sigma j$). Thus, we can also apply the same rule to obtain $A_1, \ldots, A_n \vdash A_i$. □

Lemma 2.7.6. *The free Heyting algebra on a relational multigraph $\mathcal{G}$ can be described by the natural deduction for Heyting algebras.*

Proof. Left to the reader. This will also follow as a special case of Theorem 2.8.8. □

The generalization to presentations is also easy.

2.7.3 Natural deduction and logic

Let us now say a few words about what the natural deduction for Heyting algebras has to do with logic. For a reader who thinks of logical connectives in terms of their action on truth values (e.g. "if A then B" is true unless A is true and B is false), one way to make the connection to logic is to note that the poset of truth values

$$\mathbf{2} = \{\text{false} < \text{true}\}$$

is a Heyting algebra, where the operations $\wedge, \top, \vee, \bot, \Rightarrow$ correspond to "and", "true", "or", "false", and "implies". (One way to see this easily is to identify $\mathbf{2}$, up to equivalence, with the full subcategory of **Set** consisting of sets having at most one element.)

Now suppose $\mathcal{G}$ is a relational multigraph, or more generally a relational $(\wedge, \vee, \Rightarrow)$-presentation. We call the objects of $\mathcal{G}$ **propositional variables**. Suppose furthermore that we have a map of relational multigraphs (or presentations) $\nu : \mathcal{G} \to \mathbf{2}$. In other words, we assign a truth value to each propositional variable, in such a way that if $(A_1, A_2, \ldots, A_n) \le B$ in $\mathcal{G}$, and if $\nu(A_i)$ is true for all i, then also $\nu(B)$ is true. Then by Theorem 2.7.3 we have an induced map $\mathfrak{F}_{\mathbf{Heyting}}\mathcal{G} \to \mathbf{2}$ of Heyting algebras.

The objects of $\mathfrak{F}_{\mathbf{Heyting}}\mathcal{G}$ are *propositional formulas*, built out of the propositional variables by the operations $\wedge, \top, \vee, \bot, \Rightarrow$ which we now regard as denoting the logical connectives "and", "true", "or", "false", and "implies". Since $\mathfrak{F}_{\mathbf{Heyting}}\mathcal{G} \to \mathbf{2}$ is a map of Heyting algebras, it extends the truth assignment ν to all such formulas by using the "truth tables" for all the connectives, e.g. $\nu(A \wedge B)$ is true just when $\nu(A)$ and $\nu(B)$ are both true, etc. Finally, the fact that $\mathfrak{F}_{\mathbf{Heyting}}\mathcal{G} \to \mathbf{2}$ preserves inequalities means that if $A_1, \ldots, A_n \vdash B$ is derivable in the natural deduction for Heyting algebras, and if $\nu(A_i)$ is true for all i, then also $\nu(B)$ is true.

As a special case, if $\mathcal{G}$ is discrete (i.e. is nothing but a set of propositional variables), then any derivable judgment $() \vdash B$ exhibits the propositional formula B as a **tautology**: a statement that becomes true *whatever* truth values are substituted for its propositional variables. For instance, here is a derivation exhibiting $(A \wedge (B \vee C)) \Rightarrow ((A \wedge B) \vee (A \wedge C))$ as a tautology:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A\wedge(B\vee C) \vdash A\wedge(B\vee C)}}{A\wedge(B\vee C) \vdash B\vee C} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A\wedge(B\vee C), B \vdash A\wedge(B\vee C)}}{A\wedge(B\vee C), B \vdash A} \quad \dfrac{}{A\wedge(B\vee C), B \vdash B}}{A\wedge(B\vee C), B \vdash A\wedge B}}{A\wedge(B\vee C), B \vdash (A\wedge B)\vee(A\wedge C)} \qquad \text{(and dually)}}{A\wedge(B\vee C) \vdash (A\wedge B)\vee(A\wedge C)}}{() \vdash (A\wedge(B\vee C)) \Rightarrow ((A\wedge B)\vee(A\wedge C))}$$

Thus, the natural deduction for Heyting algebras can be used as a means to derive tautologies in propositional logic. More generally, if $\mathcal{G}$ has nontrivial relations (a.k.a. *axioms*), then we can derive universally valid consequences of those axioms.

However, there is more to the relationship between type theory and logic than this. There are many ways to derive tautologies, including methods such as simply plugging in all possible truth assignments for the propositional variables and checking that the formula is always true. But the natural deduction for Heyting algebras has the important property that it (at least roughly) *mirrors the process of ordinary informal mathematical reasoning.*

It is easiest to see this if we reformulate the theory a little. Let us omit the contexts "$\Gamma \vdash$" from all judgments in a derivation tree, instead writing simply the consequent A. In place of the id rule deriving $\Gamma \vdash A$, we write simply "A" without any justification, and call it a *hypothesis.* Finally, when a type A is removed from the context on our way down the tree, we cross off that hypothesis everywhere that it appears above, and say that the hypothesis has been *discharged.* At the end, the set of remaining hypothesis is the antecedent of the conclusion; if no hypotheses remain undischarged, we have derived a tautology.

For instance, the above derivation of the distributive law would be written in this style as

$$
\dfrac{\dfrac{\dfrac{\cancel{A \wedge (B \vee C)}}{B \vee C} \qquad \dfrac{\dfrac{\dfrac{\cancel{A \wedge (B \vee C)}}{A} \qquad \cancel{B}}{A \wedge B}}{(A \wedge B) \vee (A \wedge C)} \qquad \dfrac{\dfrac{\dfrac{\cancel{A \wedge (B \vee C)}}{A} \qquad \cancel{C}}{A \wedge C}}{(A \wedge B) \vee (A \wedge C)}}{(A \wedge B) \vee (A \wedge C)}\,{\vee E}}{(A \wedge (B \vee C)) \Rightarrow ((A \wedge B) \vee (A \wedge C))}\,{\Rightarrow I}
$$

Note that there is some ambiguity; it is not obvious from looking at the derivation which rule caused which hypothesis to be discharged. In the above example, the hypotheses B and C are discharged by the $\vee E$ rule, while the hypothesis $A \wedge (B \vee C)$ (everywhere it appears) is discharged by the $\Rightarrow I$ rule. Sometimes people annotate the discharges in some way to indicate this.

However, the real point of a representation like this is that the *process of writing it*, from the top down, is supposed to mirror the process of informal reasoning. First we assume $A \wedge (B \vee C)$, and deduce from it both A and $B \vee C$. Then we use $B \vee C$ by additionally assuming B and C in two separate cases (subderivations), and in each of those cases we separately deduce $(A \wedge B) \vee (A \wedge C)$ (by way of $A \wedge B$ and $A \wedge C$ respectively). Thus, completing those cases (and ending our assumptions of B and C) we have $(A \wedge B) \vee (A \wedge C)$. Finally, ending our assumption of $A \wedge (B \vee C)$, we have $(A \wedge (B \vee C)) \Rightarrow ((A \wedge B) \vee (A \wedge C))$.

From this perspective, the rules in Figure 2.8 can also be glossed in the language of "proof strategies". For instance, $\wedge I$ says that "to prove $A \wedge B$, it suffices to prove A and B separately", while $\Rightarrow E$ says that "if we know $A \Rightarrow B$, and we also know A, then we can conclude B" (the rule of *modus ponens*). We encourage the reader to similarly gloss the other rules.

While it is arguable whether this *exactly* mirrors the process of informal reasoning, it certainly has a close kinship with it — much closer than the production of tautologies by checking all possible truth assignments. In particular, it includes one essential aspect of informal reasoning: the ability to *reason under a temporary assumption* and then "discharge" that assumption in reaching some other conclusion. This sort of *hypothetical reasoning* is central to everyday mathematics, so the fact that it also appears in natural deduction logic is a strong argument in favor of the "naturalness" of the latter.

This is the real origin of the name "natural deduction". In fact, historically, this representation with discharged hypotheses came first, and only later was it rewritten to carry along the context, and then generalized to theories without contraction and weakening. Other systems of formal logic, such as "Hilbert-style calculi" (see Exercise 2.7.9), though they can derive the same class of tautologies, do not really include hypothetical reasoning as such, and hence do not model informal reasoning as well.

Remark 2.7.7 (TODO: Frobenius/Hopf for $\vee$, for reasoning with extra hypotheses, and distributivity without $\Rightarrow$.).

Now, it may seem that the logical expressivity of the natural deduction for Heyting algebras is lacking because there is no operation corresponding to *negation*. However, we can do pretty well by defining $\neg A$ to mean $A \Rightarrow \bot$, so that its rules are

$$\frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A} \qquad\qquad \frac{\Gamma \vdash \neg A \qquad \Gamma \vdash A}{\Gamma \vdash \bot}$$

In other words, to prove $\neg A$, it suffices to show that assuming A leads to a contradiction, while if we have both $\neg A$ and A we obtain a contradiction. Using these rules, here is a derivation of one of "de Morgan's laws" as a tautology:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(A \vee B), A \vdash \neg(A \vee B)} \qquad \dfrac{\dfrac{}{\neg(A \vee B), A \vdash A}}{\neg(A \vee B), A \vdash A \vee B}}{\neg(A \vee B), A \vdash \bot}}{\neg(A \vee B) \vdash \neg A} \qquad \text{(and dually)}}{\neg(A \vee B) \vdash \neg A \wedge \neg B}}{() \vdash \neg(A \vee B) \Rightarrow (\neg A \wedge \neg B)}$$

However, not *every* tautology can be derived this way. In particular, $\neg\neg A \Rightarrow A$ (the "law of double negation") and $A \vee \neg A$ (the "law of excluded middle") are not derivable, because although they hold in **2**, their analogues fail to hold in other Heyting algebras. (In fact, they hold in a Heyting algebra exactly when that Heyting algebra is a *Boolean* algebra; see Exercise 2.7.3.) Thus, although we have something that "looks like logic", it is not exactly classical logic.

One way to resolve this is to simply add another rule, such as the following for "proof by contradiction":

$$\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}$$

(The rule for $\neg A$ derived from $\Rightarrow$ is the form of "proof by contradiction" where we prove a statement is *false* by assuming it is true and deriving a contradiction; here we are considering the opposite form where we prove a statement to be *true* by assuming it to be false and deriving a contradiction.) This mirrors the process of informal reasoning in classical mathematics fairly closely, though it is a bit problematic from a type-theoretic perspective (e.g. it fails the principles enunciated in §2.1). As we will see in chapter 3, one can also formulate a well-behaved type theory that it *can* prove all classical tautologies, by restoring the left/right and $\wedge/\vee$ symmetries.

However, it is also valuable to observe that conversely, if we are willing to generalize our notion of "logic", we obtain something much more generally applicable. Indeed, this is really the whole point of categorical logic, as put forward in §0.1: we can apply "set-like" reasoning to objects of arbitrary categories as long as we are careful about what sort of reasoning we use.

So far, we have applied this principle mainly to equational reasoning about different kinds of terms. However, we now have a type theory that is powerful enough to codify significant amounts of mathematical reasoning (though not yet anything involving quantifiers such as "for all" and "there exists"; that will come in chapter 4). Thus, we can lift our notion of "generalized logic" back to informal mathematical reasoning. It takes a bit of practice to learn to write informal mathematical proofs that could (at least in principle) be codified in such a generalized logic, but it is eminently possible. (It is much *more* possible because, as discussed above, our "generalized logic" is expressed in a style that already closely mirrors ordinary mathematical reasoning; we simply have to learn which familiar styles of argument are valid in what situations.)

The payoff is that the result is much more general than it appears, since it is true "internally to any Heyting algebra". By contrast, ordinary ("classical") mathematical reasoning is only valid in *Boolean* algebras (see Exercise 2.7.3). Lest the reader think that Heyting algebras seem esoteric, we point out that the lattice of open subsets of any topological space is a Heyting algebra (Exercise 2.7.5).

Remark 2.7.8. In the context of logic, the initiality theorem (Lemma 2.7.6) corresponds to what are traditionally called *soundness* and *completeness* theorems. A *soundness* theorem says informally "if something is provable, then it is true in all models". This follows from Lemma 2.7.6 because the inequalities in a free Heyting algebra are exactly those that are provable (i.e. derivable) in the type theory; thus, if something is provable, then it is true in the free Heyting algebra, and therefore also in every other Heyting algebra. Dually, a *completeness* theorem says informally "if something is true in all models, then it is provable". This also follows from Lemma 2.7.6 because if something is true in all Heyting algebras, then it is in particular true in a free Heyting algebra; and hence, by our construction of the latter, it is provable in the type theory.

The "generalized logic" corresponding to Heyting algebras is called **intuitionistic** or **constructive logic**, because of its similarity to the mathematics advocated by certain mathematicians calling themselves "intuitionist" or "constructive" in the early 20th century. While we are stuck with these labels, it is probably best (for a classically trained category theorist first encountering the notion) not to read too much into them. The point is simply that we make our mathematics more general by generalizing our logic, and this is the logic that corresponds naturally to cartesian closed lattices, which are certainly a categorically natural notion.

The observation that the logical operations of "and", "or", "if-then", and so on in the poset **2** have the same universal properties (and hence can be represented by the same type operations) as the operations $A \times B$, $A + B$, B^A in the category **Set** has a distinguished pedigree and many names: *propositions as types*, *proofs as terms*, or the *Curry–Howard correspondence* (see [**?**] for some history). As we will see, this correspondence is also central to the use of dependent type theory (chapter 6) as a foundation for mathematics. Some "constructivist" mathematicians have argued that this correspondence should determine the *meanings* of the logical operations in terms of proofs — that is, a proof of "P and Q" should be a pair (p, q) where p is a proof of P and q is a proof of Q; a proof of "if P then Q" should be a function transforming any proof of P into a proof of Q; and so on. This is sometimes called the *Brouwer–Heyting–Kolmogorov (BHK) interpretation.* However, we will have little to say about the philosophical side of constructive logic.

In any case, having made these observations in the case of *cartesian* closed lattices, it is natural to entertain similar ideas for other values of $\mathfrak{S}$. Roughly speaking, the names of the corresponding "generalized logics" are:

$\mathfrak{S}$	generalized logic
cartesian	intuitionistic logic
symmetric	linear logic
semicartesian	affine logic
relevance	relevance logic

To be precise, we are currently talking about variants of all these logics that should be qualified as "intuitionistic"; there are also "classical" versions of linear, affine, and relevance logics in which the laws of double negation and excluded middle hold. Moreover, at least in the linear case one should also add a phrase like "multiplicative-additive"[3] to describe our current theory, because the name "linear logic" usually refers to a system with some additional modalities. Furthermore, at this point all of them should have the prefix "propositional", since we are not yet considering quantifiers of any sort ("there exists" and "for all").

The name "linear logic" comes from the same intuition as our use of "linearity" to describe Lemma 2.4.2. The name "affine logic" is similarly inspired by the fact that while a linear transformation $T(\vec{v}) = A\vec{v}$ must use its argument exactly once in each term, an affine transformation $T(\vec{v}) = A\vec{v} + \vec{b}$ also has terms that do not use its argument at all. Both of these logics are primarily studied by computer scientists; the distinction between $\otimes$ and $\wedge$ can be interpreted in terms of "resource usage" (but that is far beyond our scope here).

Finally, "relevance logic" was invented by some philosophers seeking to avoid certain facts about implication that they regarded as "paradoxical" because their "if" parts are not "relevant" to their "then" parts, such as $A \Rightarrow (B \Rightarrow A)$. The straightforward derivation of this tautology in our type theory requires weakening:

$$\dfrac{\dfrac{\dfrac{A \vdash A}{A, B \vdash A}\ \textsc{weakening}}{A \vdash (B \Rightarrow A)}}{() \vdash A \Rightarrow (B \Rightarrow A)}$$

and in fact the type theory for closed relevance monoidal lattices cannot derive $() \vdash A \multimap (B \multimap A)$ (although this is not obvious; see Exercises 2.7.7 and 2.7.8).

The most commonly used relevance logics satisfy other principles that our type theory does not, notably the distributive law $A \wedge (B \vee C) \cong (A \wedge B) \vee (A \wedge C)$

[3]In the lingo of linear logic, $\otimes$ is a "multiplicative" connective, while $\wedge$ and $\vee$ are "additive". Classical linear logic also includes another multiplicative connective called $⅋$ that is dual to $\otimes$ in the same way that $\vee$ is dual to $\wedge$; see §3.4.

(note that our derivation of this above also used weakening). Of course, any closed monoidal lattice satisfies the distributive law $A \otimes (B \vee C) \cong (A \otimes B) \vee (A \otimes C)$, but as we have observed, both weakening and contraction are necessary to force $\otimes$ to coincide with $\wedge$. (It is possible to formulate type theories that ensure the $\wedge/\vee$ distributive law as well, but this requires a fancier notion of generalized multicategory.)

Exercises

Exercise 2.7.1. Prove Theorems 2.6.6 and 2.6.7 using our posetal type theories. Specifically:

(a) If we have exchange and weakening, prove that $\mathbf{1} \cong \top$.

(b) If we have exchange, weakening, and contraction, prove that $A \otimes B \cong A \times B$.

Exercise 2.7.2. Prove that $\neg\neg(P \vee \neg P)$ is an intuitionistic tautology, i.e. construct a derivation of $() \vdash \neg\neg(P \vee \neg P)$ in the natural deduction for Heyting algebras.

Exercise 2.7.3. Prove that the following are equivalent for a Heyting algebra:

(a) The law of excluded middle $P \vee \neg P$ is true, i.e. $P \vee \neg P$ is the top element for all P.

(b) The law of double negation $\neg\neg P \Rightarrow P$ is true.

(c) The Heyting algebra is a Boolean algebra, i.e. every element P has a "complement" $\overline{P}$ such that $P \wedge \overline{P} = \bot$ and $P \vee \overline{P} = \top$.

Exercise 2.7.4. Of the four "de Morgan's laws", three are intuitionistic tautologies and one is not. Construct derivations of three of the following sequents in the natural deduction for Heyting algebras:

$$\neg(P \vee Q) \vdash \neg P \wedge \neg Q$$
$$\neg(P \wedge Q) \vdash \neg P \vee \neg Q$$
$$\neg P \wedge \neg Q \vdash \neg(P \vee Q)$$
$$\neg P \vee \neg Q \vdash \neg(P \wedge Q)$$

Exercise 2.7.5. A **frame** is a lattice with infinitary joins satisfying the infinite distributive law $A \wedge (\bigvee_i B_i) \cong \bigvee_i (A \wedge B_i)$.

(a) Prove that any (small) frame is a Heyting algebra.

(b) Prove that the lattice of open sets of any topological space is a frame.

(c) Describe a type theory for frames. This is called (propositional) **geometric logic**.

Exercise 2.7.6. Give concrete examples of Heyting algebras satisfying the following:

(a) There is an element P for which $P \vee \neg P$ is not the top element.

(b) There are elements P and Q for which the fourth de Morgan's law (see Exercise 2.7.4) does not hold.

Exercise 2.7.7. Describe a concrete example of a closed relevance monoidal lattice containing two objects A and B such that there is no morphism from $\mathbf{1}$ (the unit object) to $A \multimap (B \multimap A)$. Deduce that $() \vdash A \multimap (B \multimap A)$ is not derivable in the type theory for closed relevance monoidal lattices.

Exercise 2.7.8. One of the advantages of sequent calculus over natural deduction is that because all of its rules *introduce* operations on the left or the right, it is easier to conclude underivability theorems.

(a) Define a sequent calculus for closed $\mathfrak{S}$-monoidal lattices, and prove the cut admissibility and initiality theorems.

(b) Prove that $() \vdash A \multimap (B \multimap A)$ is not derivable in the sequent calculus for closed relevance monoidal lattices, by ruling out all possible ways that such a derivation could end.

Exercise 2.7.9. TODO: A whole section on this? There should be a general notion of "closed category" for any suitable kind of multicategory, with a resulting combinatory logic and hilbert system. https://nforum.ncatlab.org/discussion/4632/closed-category/

Another way of deriving tautologies is called a **Hilbert system**. A Hilbert system can be formulated as a sort of type theory where the judgments all have empty context, i.e. are of the form $\vdash A$ where A is a propositional formula. Instead of the "modular" left/right rules of sequent calculus or the introduction/elimination rules of natural deduction, where the rules for each connective do not refer to any other connective, a Hilbert system gives a special place to implication $\Rightarrow$. The *only* rule with premises[4] is the empty-context form of $\Rightarrow E$, *modus ponens*:

$$\frac{\vdash A \Rightarrow B \qquad \vdash A}{\vdash B}$$

The behavior of all other connectives is specified by *axioms* (rules with no premises, other than the well-formedness of the formulas appearing in them). For instance, we complete the description of $\Rightarrow$ with the following axioms:

$$\frac{\vdash A\ \mathsf{type}}{\vdash A \Rightarrow A} \qquad\qquad \frac{\vdash A\ \mathsf{type} \qquad \vdash B\ \mathsf{type}}{\vdash A \Rightarrow (B \to A)}$$

$$\frac{\vdash A\ \mathsf{type} \qquad \vdash B\ \mathsf{type} \qquad \vdash C\ \mathsf{type}}{\vdash (A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))}$$

[4] Hilbert systems for more complicated logics have one or two more rules with premises, but in general there are very few.

The axioms for the remaining connectives are (omitting the obvious premises and the $\vdash$):

$$A \Rightarrow (B \Rightarrow (A \wedge B)) \qquad (A \wedge B) \Rightarrow A \qquad (A \wedge B) \Rightarrow B$$

$$A \Rightarrow (A \vee B) \qquad B \Rightarrow (A \vee B) \qquad (A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow ((A \vee B) \Rightarrow C))$$

$$A \Rightarrow \top \qquad \bot \Rightarrow A$$

Prove that this Hilbert system derives exactly the same tautologies as the natural deduction for Heyting algebras.

(The main reasons for using a Hilbert system seem to be that it *never* changes the context and has very few rules. This sometimes makes metatheoretic arguments easier, but at the cost of greater distance from informal mathematics, since as we have remarked the latter gives a central place to hypothetical reasoning. It should also be noted that the symbol $\vdash$ is often used differently in the context of Hilbert systems; rather than $\vdash$ being part of each judgment, the notation "$\Gamma \vdash A$" means that we can derive A (that is, $\vdash A$ in our notation) in the Hilbert system augmented by all the formulas in Γ as additional axioms.)

Exercise 2.7.10. Is there a well-behaved type theory (i.e. having admissible cut and an initiality theorem) corresponding to the (posetal version of the) "cocartesian multicategories" of Exercise 2.6.7? *(As of this writing, the answer is not known to the author.)*

2.8 Simply typed λ-calculus

We now move back up the ladder from posets to categories. In this case it becomes more important to adhere to principle (§) and make our structural rules admissible. Otherwise our derivations would become polluted with applications of these rules, and our terms (which, as ever, we want to be simply syntax for derivations) would be likewise quite messy-looking. We have already seen in Lemma 2.7.5 that the structural rules can be made admissible in the cartesian case where we want all of them, so we consider that case first. In addition to being the easiest, this is probably also the most commonly used case.

We begin by introducing terms for the rules from §2.7.2, as shown in Figure 2.9. As in §2.7.2, we omit $\otimes$ and $\mathbb{1}$ since they coincide with $\wedge$ and $\top$. We also switch back to categorical notations $\times, \mathbb{1}, +, \mathbf{0}$ instead of $\wedge, \top, \vee, \bot$. We also write $A \to B$ instead of $A \multimap B$; this has the pleasing consequence that the term syntax $M : A \to B$ looks the same as the common mathematical notation for functions.

Most of the term annotations should be familiar from §2.5; indeed they are even simpler, since the (expected) presence of the structural rules allows us to omit some of the more verbose annotations. The rule $\to I$ introduces a new kind of term, a **λ-abstraction**. Since the variable x appears in the premise but not the conclusion, it must be bound in the resulting term; there is nothing else to say, so we simply prefix the letter λ to indicate the rule. Intuitively, we

$$\frac{\vdash A\ \mathsf{type} \qquad (x:A)\in\Gamma}{\Gamma\vdash x:A}\ \mathsf{id}$$

$$\frac{f\in\mathcal{G}(A_1,\ldots,A_n;B) \qquad \Gamma\vdash M_1:A_1 \qquad \ldots \qquad \Gamma\vdash M_n:A_n}{\Gamma\vdash f(M_1,\ldots,M_n):B}\ f$$

$$\frac{\Gamma\vdash M:A \qquad \Gamma\vdash N:B}{\Gamma\vdash\langle M,N\rangle:A\times B}\ {\times}I \qquad \frac{\Gamma\vdash M:A\times B}{\Gamma\vdash\pi_1^{A,B}(M):A}\ {\times}E1$$

$$\frac{\Gamma\vdash M:A\times B}{\Gamma\vdash\pi_2^{A,B}(M):B}\ {\times}E2$$

$$\frac{}{\Gamma\vdash *:\mathbb{1}}\ \mathbb{1}I \qquad \frac{\Gamma\vdash M:\mathbf{0}}{\Gamma\vdash\mathsf{match}_{\mathbf{0}}(M):C}\ \mathbf{0}E$$

$$\frac{\Gamma\vdash M:A}{\Gamma\vdash\mathsf{inl}(M):A+B}\ {+}I1 \qquad \frac{\Gamma\vdash N:B}{\Gamma\vdash\mathsf{inr}(N):A+B}\ {+}I2$$

$$\frac{\Gamma\vdash M:A+B \qquad \Gamma,u:A\vdash P:C \qquad \Gamma,v:B\vdash Q:C}{\Gamma\vdash\mathsf{match}_{A+B}(M,u.P,v.Q):C}\ {+}E$$

$$\frac{\Gamma,x:A\vdash M:B}{\Gamma\vdash\lambda x.M:A\to B}\ {\to}I \qquad \frac{\Gamma\vdash M:A\to B \qquad \Gamma\vdash N:A}{\Gamma\vdash MN:B}\ {\to}E$$

Figure 2.9: The simply typed λ-calculus with products and coproducts

think of $\lambda x.M$ as meaning "the function that takes one argument, called x, and returns the value of M (which includes x)". For instance, $\lambda x.x^2$ denotes the function that squares its argument, $\lambda x.(x+3)$ denotes the function that adds three to its argument, and so on. Because of the importance of this operation, the type theory of Figure 2.9, which we expect to correspond to cartesian closed categories with coproducts, is called the **simply typed λ-calculus (STLC) with products and coproducts**. (The unqualified "STLC" would omit the rules for $\times, \mathbb{1}, +, \mathbf{0}$.)

The term annotation for the rule $\to E$ simply "pairs up" two terms, one of which has type $A \to B$ and one of which has type A. Intuitively, we are "applying" a function $M : A \to B$ to an argument $N : A$. Technically there ought also to be a label indicating the rule being applied to pair these terms up, such as $\mathsf{app}(M, N)$. However, any system of notation has room for *one* operation denoted by simple juxtaposition (e.g. in high-school algebra it is multiplication, while in group theory it is the group operation), and the importance of the type operation $\to$ leads us to choose $\to E$ for this honor in type theory. Most mathematicians write $f(a)$ for the application of a function f to an argument a; since parentheses are used as usual for grouping, this notation is also valid here, just as $(x)(y) = xy$ in high-school algebra.

Lemma 2.8.1. *If a term $\Gamma \vdash M : A$ is derivable in the simply typed λ-calculus, then it has a unique derivation.*

Proof. This is almost immediate. Since the premises of all rules have the same context as the conclusion, perhaps with a type added, there is no ambiguity about how to split things up, and hence no need for the uglier annotations used in §2.5. □

Lemma 2.8.2. *The structural rules of exchange, contraction, and weakening are admissible in the simply typed λ-calculus. Moreover, they make the derivations into a cartesian multigraph (i.e. they are functorial as in Definition 2.6.4).*

Proof. The proof is essentially the same as Lemma 2.7.5, carrying along terms and variables; we prove the following rule, for any function $\sigma : \{1, \ldots, m\} \to \{1, \ldots, n\}$:

$$\frac{y_1 : A_{\sigma 1}, \ldots, y_m : A_{\sigma m} \vdash M : B}{x_1 : A_1, \ldots, x_n : A_n \vdash \sigma^* M : B}$$

by pushing up through all rules until we get to id. Regarded as an operation on terms, σ^* is defined by the clause

$$\sigma^* y_j = x_{\sigma j}$$

along with trivial "descending into subterms" clauses for all other terms, such as

$$\sigma^* \langle M, N \rangle = \langle \sigma^* M, \sigma^* N \rangle$$
$$\sigma^* \mathsf{match}_+(M, u.P, v.Q) = \mathsf{match}_+(\sigma^* M, u.(\sigma \sqcup \mathsf{id})^* P, v.(\sigma \sqcup \mathsf{id})^* Q)$$

and so on. Intuitively, we simply substitute the variable $x_{\sigma j}$ for y_j wherever it appears, for all $1 \le j \le m$. A similar induction proves that this operation is functorial, yielding a cartesian multigraph. □

In fact, if σ is injective — that is, it is composed of exchange and weakening only — and if we choose variables $y_j = x_{\sigma j}$ (which is only possible if σ is injective), then in fact $\sigma^* M = M$. For instance, we have

$$x : A, y : B \vdash \langle x, y\rangle : A \times B$$

and by exchange and weakening we can obtain also

$$y : B, z : C, x : A \vdash \langle x, y\rangle : A \times B.$$

with the same term $\langle x, y\rangle$. This does not contradict "terms are derivations", because we only require a term to determine a unique derivation *when paired with its context and consequent.*

Remark 2.8.3. As remarked briefly at the beginning of the section, the admissibility of the structural rules is central to having a clean theory of terms and a clean proof of the initiality theorem. If we took the structural rules as primitive, then to maintain "terms as derivations" we would have to include information about the structural rules in terms, for instance annotating the derivation

$$\cfrac{\cfrac{A, B \vdash A \qquad A, B \vdash B}{A, B \vdash A \times B}}{B, A \vdash A \times B}$$

with a term like $y : B, x : A \vdash \sigma^*\langle x, y\rangle : A \times B$, distinguishing it from the *different* derivation

$$\cfrac{B, A \vdash B \qquad B, A \vdash A}{B, A \vdash A \times B}$$

that we could write as $y : B, x : A \vdash \langle x, y\rangle : A \times B$. Clearly these two derivations ought to have the same term. However, if structural rules are primitive, then using the same term for both of them would break the "terms as derivations" principle. If we did this, then to prove the initiality theorem using terms, after inducting over derivations we would have to prove that the interpretation of a term is independent of its derivation. This sort of thing is difficult and tedious, and hence often left to the reader or left unmentioned altogether. Making the structural rules admissible avoids both horns of the dilemma.

Now we need substitution. Since all our other rules maintain the same context, it is natural to do the same here.

Lemma 2.8.4. *Substitution is admissible in the simply typed λ-calculus: given derivations of $\Gamma \vdash M : A$ and $\Gamma, x : A \vdash N : B$, we can construct a derivation of $\Gamma \vdash M[N/x] : B$.*

$$
\begin{aligned}
x[M/x] &= M \\
y[M/x] &= y \qquad (y \text{ a variable } \neq x) \\
f(N_1, \ldots, N_n)[M/x] &= f(N_1[M/x], \ldots, N_n[M/x]) \\
\langle P, Q\rangle[M/x] &= \langle P[M/x], Q[M/x]\rangle \\
\pi_1(N)[M/x] &= \pi_1(N[M/x]) \\
\pi_2(N)[M/x] &= \pi_2(N[M/x]) \\
*[M/x] &= * \\
\mathsf{match_0}(N)[M/x] &= \mathsf{match_0}(N[M/x]) \\
\mathsf{inl}(N)[M/x] &= \mathsf{inl}(N[M/x]) \\
\mathsf{inr}(N)[M/x] &= \mathsf{inr}(N[M/x]) \\
\mathsf{match}_+(N, u.P, v.Q)[M/x] &= \mathsf{match}_+(N[M/x], u.P[M/x], v.Q[M/x]) \\
(\lambda y.N)[M/x] &= \lambda y.N[M/x] \\
(PQ)[M/x] &= (P[M/x])(Q[M/x])
\end{aligned}
$$

Figure 2.10: Substitution in simply typed λ-calculus

Proof. By induction on the derivation of $\Gamma, x : A \vdash N : B$, as usual. There are two mildly new features. Firstly, since the contexts are maintained rather than split, we have to recurse into *all* premises of each rule. Secondly, when we get down to id we might find a variable other than x, in which case there is no substitution to do. Thus the clauses defining substitution are as shown in Figure 2.10. As in §§2.4 and 2.5, to write substitution using terms, we need to ensure by α-equivalence that the bound variables u, v in match_+ and y in λ do not appear free in M. □

Note that the contraction rule is actually a special case of substitution, namely the substitution of one variable for another. That is, given $\Gamma, x : A, y : A \vdash M : B$, we have $\Gamma, x : A \vdash M[x/y] : B$ which is (by induction, if you wish) equal to the contraction of M obtained from Lemma 2.8.2.

The natural sort of "associativity" for this kind of substitution is also different: it combines the "associativity and interchange" properties in one, since if a variable y is free in $N[M/x]$ then it might appear in *both* M and N.

Lemma 2.8.5. *Given derivations of* $\Gamma \vdash M : A$ *and* $\Gamma, x : A \vdash N : B$ *and* $\Gamma, x : A, y : B \vdash P : C$*, we have*

$$P[N/y][M/x] = P[M/x][N[M/x]/y].$$

On the left-hand side, $P[N/y][M/x]$ means $(P[N/y])[M/x]$. On the right-hand side, when writing $P[M/x]$ we have technically to apply weakening to M (by Lemma 2.8.2) so that it has context $\Gamma, y : B$ first.

Proof. A straightforward induction on derivations. For the "base cases" of variables, we have

$$\begin{aligned} x[N/y][M/x] &= x[M/x] \\ &= M \\ &= M[N[M/x]/y] \qquad (*) \\ &= x[M/x][N[M/x]/y] \\ y[N/y][M/x] &= N[M/x] \\ &= y[N[M/x]/y] \\ &= y[M/x][N[M/x]/y] \\ z[N/y][M/x] &= z[M/x] \\ &= z \\ &= z[M/x] \\ &= z[M/x][N[M/x]/y] \end{aligned}$$

where $z \neq x, y$. The equality (*) is because y does not appear in M, i.e. M has been obtained by weakening from a context not including y as remarked above. (Formally, we ought to prove by a further induction that substituting for a variable obtained by weakening never changes the term/derivation.) □

We also need to know that substitution commutes with the other structural rules. For weakening and exchange this is immediate from the observation that these rules do not change the term. For contraction, it follows from Lemma 2.8.5 and the observation that contraction is a special case of substitution:

$$N[x/y][M/x] = N[M/x][x[M/x]/y] = N[M/x][M/y]. \tag{2.8.6}$$

$$N[M/x][y/z] = N[y/z][M[y/z]/x] \tag{2.8.7}$$

Now we can state the β- and η-conversion rules. Those for products and coproducts are the familiar ones from Figure 2.6. The β-conversion rule for $\to$:

$$(\lambda x.M)N \equiv M[N/x]$$

says that if we apply a function defined by λ-abstraction to an argument, the result is what we get by "plugging in" the argument to the expression defining the function. That is, if $f(x) = x^2$ then $f(3) = 3^2$. The η-conversion rule says that any function is a λ-abstraction:

$$M \equiv \lambda x.Mx \qquad \text{if } M : A \to B$$

A straightforward induction shows that $\equiv$ is a congruence not only for substitution, but also for the new admissible structural rules from Lemma 2.8.2.

Now we are ready to prove the initiality theorem. Note that we generate our free structure from a mere multigraph, not (as one might guess) a cartesian multigraph. A cartesian multigraph contains operations and equations, so to use it as base data we would need to incorporate those operations into $\equiv$.

Theorem 2.8.8. *The free cartesian closed category with coproducts generated by a multigraph $\mathcal{G}$ can be presented by the simply typed λ-calculus under $\mathcal{G}$: its underlying cartesian multigraph is that constructed in Lemma 2.8.2 modulo $\equiv$, and its composition is given by substitution.*

Proof. Although Lemmas 2.8.4 and 2.8.5 are not stated in the usual form of multicategory composition operations, we can easily derive those operations from them. Given $\Gamma \vdash M : A$ and $\Delta, x : A, \Psi \vdash N : B$, we can apply weakening and exchange to obtain $\Delta, \Gamma, \Psi \vdash M : A$ and $\Delta, \Gamma, \Psi, x : A \vdash N : B$; then Lemma 2.8.4 yields $\Delta, \Gamma, \Psi \vdash N[M/x] : B$. Associativity and interchange are the special cases of Lemma 2.8.5 where x does not occur in P and where x does not occur in N, respectively, and the identity laws follow as usual. Thus to have a cartesian multicategory it remains to check Definition 2.6.4(c) and (d), using in particular (2.8.6) and (2.8.7). We leave this to the reader in Exercise 2.8.1; it can be done directly or by way of Exercise 2.6.3.

The rules for all the type operations give this cartesian multicategory products, coproducts, and closed structure; thus it underlies a cartesian closed category with coproducts. Initiality follows as usual: given a map of multigraphs $P : \mathcal{G} \to \mathcal{M}$, where $\mathcal{M}$ is a cartesian closed category with coproducts, we extend P to all types by induction, then define it on all derivations by induction, then check that $\equiv$ is preserved by induction. As always, this works because the rules for type operations (including the new one $\to$) are defined to mirror those of categorical universal properties. □

Remark 2.8.9. Throughout this chapter, we have been taking the notion of "finite list" as given externally: the context of a judgment is a finite list of types, and we assume we know what a finite list means. However, it is also possible to incorporate the definition of "finite list" into the type theory, by adding a *judgment for contexts* alongside the judgment for types. The rules for this judgment are:

$$\frac{}{\vdash () \;\mathsf{ctx}} \qquad\qquad \frac{\vdash \Gamma \;\mathsf{ctx} \qquad \vdash A \;\mathsf{type}}{\vdash (\Gamma, A) \;\mathsf{ctx}}$$

In other words, there is an empty context, and from any context we can make a new one by adding a type on the end.

We then have the problem of expressing the other rules that modify the context in terms of this judgment. This is simplest and most useful in the cartesian case, when all the structural context rules (exchange, contraction, weakening) are admissible as in this section. For in this case, all the rules that change the context (like $\to I$ and $+E$) simply add an extra type at the end of it when passing from the conclusion to some of the premises, and this operation is directly represented by the second rule for the context judgment.

We also have to deal with the "identity/variable" rule. Instead of it having a condition $(x : A) \in \Gamma$ (relying on our external knowledge of what it means to "be an element of a finite list"), we can introduce another new judgment "$\Gamma \vdash x \Downarrow A$"

meaning "x is a variable of type A in context Γ", with rules

$$\frac{\vdash \Gamma\ \mathsf{ctx} \qquad \vdash A\ \mathsf{type}}{\Gamma, A \vdash \mathsf{pop} \Downarrow A} \qquad \frac{\vdash \Gamma\ \mathsf{ctx} \qquad \vdash A\ \mathsf{type} \qquad \Gamma \vdash x \Downarrow B}{\Gamma, A \vdash \mathsf{shift}(x) \Downarrow B} \qquad \frac{\Gamma \vdash x \Downarrow A}{\Gamma \vdash \mathsf{use}(x) : A}.$$

That is, there is a variable associated to the last type in a context, and variables associated to other types in the context are defined inductively. Thus, for example, the variables in the context A, B, C are

$$\frac{}{A, B, C \vdash \mathsf{pop} \Downarrow C} \qquad \frac{\overline{A, B \vdash \mathsf{pop} \Downarrow B}}{A, B, C \vdash \mathsf{shift}(\mathsf{pop}) \Downarrow B}$$

$$\frac{\dfrac{\overline{A \vdash \mathsf{pop} \Downarrow A}}{A, B \vdash \mathsf{shift}(\mathsf{pop}) \Downarrow A}}{A, B, C \vdash \mathsf{shift}(\mathsf{shift}(\mathsf{pop})) \Downarrow A}$$

Note that modulo a change of notation, these "variables" are precisely *de Bruijn indices*: the number of "shift"s says how many types we need to "count backwards" from the right. At present this is merely a curiosity, but when we come to consider dependent type theories in chapter 6 it will be much more important.

Exercises

Exercise 2.8.1. Complete the proof in Theorem 2.8.8 that type theory yields a cartesian multicategory by checking Definition 2.6.4(c) and (d), or perhaps the corresponding one-place axioms you found in Exercise 2.6.3.

Exercise 2.8.2.

(a) For any natural number n and any type A in STLC, show that there is a term of type $(A \to A) \to (A \to A)$ that iterates a function n types. Call this term $[n]_A$.

(b) Define $[n+m]_A$ in terms of $[n]_A$ and $[m]_A$.

(c) Now define $[nm]_A$ in terms of $[n]_A$ and $[m]_A$ (you may have to use $[n]_A$ or $[m]_A$ for more than one value of A).

Exercise 2.8.3.

(a) Write down terms in the simply typed λ-calculus (with $\to$ the only type constructor) that have the following types (for arbitrary A, B, C):

$$A \to A$$

$$A \to B \to A$$

$$(A \to B \to C) \to (A \to B) \to (A \to C)$$

(Remember that the type operator $\to$ associates to the right: $X \to Y \to Z$ means $X \to (Y \to Z)$.)

(b) By **combinatory logic** we will mean the type theory obtained from simply typed λ-calculus by *removing* the λ-abstraction rule:

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}$$

and instead adding axioms called I, K, and S having the above types (for any A, B, C):

$$\frac{\vdash A \ \mathsf{type}}{\Gamma \vdash I_A : A \to A} \qquad \frac{\vdash A \ \mathsf{type} \qquad \vdash B \ \mathsf{type}}{\Gamma \vdash K_{AB} : A \to B \to A}$$

$$\frac{\vdash A \ \mathsf{type} \qquad \vdash B \ \mathsf{type} \qquad \vdash C \ \mathsf{type}}{\Gamma \vdash S_{ABC} : (A \to B \to C) \to (A \to B) \to (A \to C)}$$

Prove that the removed λ-abstraction rule is *admissible* in **CL**. That is, given a derivation in combinatory logic of $\Gamma, x : A \vdash M : B$, we can construct a derivation in combinatory logic of $\Gamma \vdash [x]M : A \to B$ for some $[x]M$ (note that this is an *operation* on combinatory logic terms, like substitution).

(c) Write down some $\equiv$-laws satisfied by the I, K, and S you defined in (a), and show that when they are used as generators for an $\equiv$ for combinatory logic, it also presents a free closed cartesian multicategory. One way to do this is by a direct induction on derivations; another way is exhibit a bijection between its terms and those of the simply typed λ-calculus.

(d) Compare to Exercise 2.7.9.

2.9 Cartesian presentations

In §1.7 we mentioned that every kind of type theory can be generalized to use an appropriate kind of "presentation" (or "theory") as input, with the main technical issue being an appropriate notion of "1-skeleton" for the categorical structure in question. We have postponed dealing with 1-skeleta in this chapter until now, because they are substantially simpler in the cartesian case; but now we are ready.

As in the unary case, our goal is to build a tower of adjunctions relating "k-skeletal $(\times, +, \to)$-presentations" with "k-skeleta for cartesian closed categories with coproducts" (and similarly when some of the type operations are omitted). For instance, a **2-skeletal** $(\times, +, \to)$**-presentation** consists of:

(a) A set $\mathcal{P}_0$ of objects;

(b) A set $\mathcal{P}_1$ of morphisms, each of which has a list of types as domain and a single type as codomain, where the "types" are generated from $\mathcal{P}_0$ using the rules for the judgment $\vdash A \ \mathsf{type}$ in §2.8. For example, such a generating morphism might have domain $(A \to B, B \times B \to A)$ and codomain $(A \to C) \times B$.

(c) A set $\mathcal{P}_2$ of equalities or axioms, each of which is a pair of terms generated from $\mathcal{P}_1$ and the rules for the term judgment, with the same context and the same consequent.

Unsurprisingly, the main subtlety is in the definition of 1-skeleta for cartesian multicategories, which will require us to introduce a notion of "presheaf on a multicategory". However, before we dive into that, there is a simple but still very important case we can deal with first.

2.9.1 Finite-product theories

Suppose we start from the type theory of §2.8 but with *no* type operations — that is, the type theory for plain cartesian multicategories. Then just as in §1.7.2, the 1-skeleta for categories coincide with the categories themselves; the new thing relative to §2.8 is that we allow arbitrary generating *equalities* in our 2-skeletal presentations. These presentations are generally called **finite-product theories** or **finitary algebraic theories**.

Note that because we have morphisms in our generating multigraph with arbitrary domains, we can still express "operations" of arbitrary finite arity even a "product type" operation. Thus, finite-product theories solve the problem that we had with our *unary* finite-product theories in §1.7 (which there we mostly called "×-presentations") of having to "pack and unpack" terms into ordered pairs in order to apply generators to them (such as the multiplication of a monoid object).

For instance, the finite-product theory for a monoid has:

- One base type A;
- Two generating morphisms $m \in \mathcal{G}_1(A, A; A)$ and $e \in \mathcal{G}_1((); A)$; and
- The following axioms:

$$x : A, y : A, z : A \vdash m(x, m(y, z)) \equiv m(m(x, y), z) : A$$

$$x : A \vdash m(x, e) \equiv x : A \qquad\qquad y : A \vdash m(e, y) \equiv y : A$$

(As is common, since e is a 0-ary generator, we write just "e" instead of explicitly giving it 0 arguments like "$e()$".) In this formulation, the equational proof of uniqueness of inverses from §0.1 finally makes sense as written. To be precise, we assume additional generators $i, j \in \mathcal{G}_1(A, A)$ and axioms

$$x : A \vdash m(x, i(x)) \equiv e : A \qquad\qquad x : A \vdash m(x, j(x)) \equiv e : A$$

$$x : A \vdash m(i(x), x) \equiv e : A \qquad\qquad x : A \vdash m(j(x), x) \equiv e : A.$$

If we write $m(x, y)$ infix as $x \cdot y$, then we can perform the computation exactly as written:

$$x : A \vdash i(x) \equiv i(x) \cdot e \equiv i(x) \cdot (x \cdot j(x)) \equiv (i(x) \cdot x) \cdot j(x) \equiv e \cdot j(x) \equiv j(x) : A.$$

For emphasis, we remind the reader how this type-theoretic proof yields a conclusion about arbitrary monoid objects in arbitrary categories with products. Given a category with products $\mathcal{M}$, we first regard it as a cartesian multicategory. Then the structure of a monoid object A corresponds to morphisms $(A, A) \to A$ and $() \to A$ in this cartesian multicategory, satisfying the appropriate axioms, and similarly for inverse operators.

Now if $\mathcal{G}$ is the above finite-product theory, it generates the free cartesian multicategory $\mathfrak{F}_{\mathbf{CartMulti}}\mathcal{G}$ containing a monoid object with two inverse operators. Its freeness implies there is a unique functor of cartesian multicategories $\mathfrak{F}_{\mathbf{CartMulti}}\mathcal{G} \to \mathcal{M}$ taking the generating syntactic monoid to the given one in $\mathcal{M}$, and also its inverse operators. Finally, the above calculation shows that i and j define equal morphisms in $\mathfrak{F}_{\mathbf{CartMulti}}\mathcal{G}$; hence their images in $\mathcal{M}$ must also be equal.

In general, finite-product theories can describe algebraic structures with operations and equational axioms, such as monoids, groups, rings, modules, and so on. Thus, all such structures can be "internalized" in any category with finite products, and anything provable from their purely equational theory must be true in any such internalization (see for instance Exercise 2.9.1).

Finite-product theories cannot describe structures whose operations or axioms involve conditions, such as categories (we can only compose two morphisms if the source of one must equal the target of the other) or fields (we can only invert an element if it is nonzero), or whose axioms involve more complicated logical operations such as "or" or "there exists". Structures of this sort can also be internalized, but only in a category with more structure; we will return to them in chapter 4.

Note that we actually obtained a more general result about monoid objects in cartesian multicategories, not just categories with products, because our free object $\mathfrak{F}_{\mathbf{CartMulti}}\mathcal{G}$ is a cartesian multicategory rather than a category with products. However, for some purposes, we might also want to have a free category with products generated by a finite-product theory. Once we have established the full theory of presentations in §2.9.2, we can obtain this by simply adding the $\times, \mathbb{1}$ type operations. But there is also another way to obtain a free category with products, by applying the following left adjoint (which, for later use, we construct in the case of a general $\mathfrak{S}$).

Theorem 2.9.1. *Let* $\mathbf{StrMonCat}_{\mathfrak{S}}$ *denote the category of strict* $\mathfrak{S}$*-monoidal categories; that is,* $\mathfrak{S}$*-multicategories that are representable and whose underlying monoidal category is strict, and functors that preserve the chosen tensor products strictly. The forgetful functor from* $\mathbf{StrMonCat}_{\mathfrak{S}}$ *to* $\mathfrak{S}$*-multicategories has a left adjoint* $\mathcal{M} \mapsto \widehat{\mathcal{M}}$.

Proof. Given an $\mathfrak{S}$-multicategory $\mathcal{M}$, we define the objects of $\widehat{\mathcal{M}}$ to be finite lists $(A_1, \ldots, A_n)$ of objects of $\mathcal{M}$, with a monoidal structure given by concatenation of lists. The hom-set $\widehat{\mathcal{M}}((A_1, \ldots, A_n), (B_1, \ldots, B_k))$ is a subquotient (i.e. a quotient of a subset) of the set

$$\{ (\sigma, (f_1, \ldots, f_k)) \mid \mathrm{cod}(f_j) = B_j \text{ and } \sigma : \{1, \ldots, m\} \to \{1, \ldots, n\} \text{ in } \mathfrak{S} \}.$$

The subset consists of those tuples such that if we concatenate the domains of the f_j's in order, we get $(A_{\sigma 1}, \ldots, A_{\sigma m})$. The quotient is then by the smallest equivalence relation that identifies

$$(\sigma, (f_1\sigma_1^*, \ldots, f_k\sigma_k^*)) \qquad \text{with} \qquad (\sigma(\sigma_1 \sqcup \cdots \sqcup \sigma_k), (f_1, \ldots, f_k))$$

(This can be described more abstractly as a "tensor product of functors".) Composition is defined on representatives by

$$(\tau, (g_1, \ldots, g_n)) \circ (\sigma, (f_1, \ldots, f_k)) = \\ (\tau(\sigma \wr \ell_1, \ldots, \ell_n), (g_{\sigma 1}, \ldots, g_{\sigma m}) \circ (f_1, \ldots, f_k))$$

Here ℓ_j is the arity of g_j, and $(g_{\sigma 1}, \ldots, g_{\sigma m}) \circ (f_1, \ldots, f_k)$ means to compose the first few $g_{\sigma i}$'s with f_1 (according to the arity of f_1), then the next few with f_2, and so on. We leave the details to the reader.

There are two cases in which $\widehat{\mathcal{M}}$ has a simpler description. Firstly, if $\mathfrak{S}$ contains only identities, so we are talking about ordinary multicategories and monoidal categories, then the coproduct over σ degenerates and there is no quotienting necessary; so the above definition is complete. Secondly, in the cartesian case when $\mathfrak{S}$ contains all functions, we can instead define

$$\widehat{\mathcal{M}}((A_1, \ldots, A_n), (B_1, \ldots, B_k)) = \prod_{j=1}^{k} \mathcal{M}(A_1, \ldots, A_n; B_j).$$

To see that this definition is isomorphic to the previous one in the cartesian case, note that since $(B_1, \ldots, B_k)$ is the cartesian product $B_1 \times \cdots \times B_k$ in $\widehat{\mathcal{M}}$, we must have

$$\widehat{\mathcal{M}}((A_1, \ldots, A_n), (B_1, \ldots, B_k)) = \prod_{j=1}^{k} \widehat{\mathcal{M}}((A_1, \ldots, A_n); B_j).$$

Thus, it suffices to observe that $\widehat{\mathcal{M}}((A_1, \ldots, A_n); B) \cong \mathcal{M}(A_1, \ldots, A_n; B)$. This is because when $k = 1$, the quotient precisely cancels out the extra σ involved in the former. □

If we apply this left adjoint to the free cartesian multicategory generated by a finite-product theory $\mathcal{P}$, we obtain a category with products whose objects are the *contexts* in the type theory of $\mathcal{P}$, and whose morphisms are tuples of terms. For this reason it is often called the **category of contexts** of $\mathcal{P}$.

When $\mathcal{P}$ has exactly one type, say A, the objects of its category of contexts are just lists $(A, A, \ldots, A)$ uniquely determined by their length, and so they can be identified with natural numbers. In this case, the category of contexts is known as the **Lawvere theory** [**?**] corresponding to $\mathcal{P}$ (recall from §1.7.4 that some categorical logicians use the word "theory" to refer to the categorical structure *presented by* a type theory).

Of course, the universal property of the category of contexts given by Theorem 2.9.1 refers only to *strict* monoidal categories, which makes sense since concatenation of contexts is strictly associative. However, it turns out that up to equivalence, it also has a more general universal property.

Theorem 2.9.2. *Let $\mathscr{P}r\mathcal{C}at$ denote the 2-category of categories with products, functors that preserve products up to isomorphism in the usual sense, and natural transformations; and let $\mathscr{C}artMulti$ denote the 2-category of cartesian multicategories. The forgetful functor $\mathscr{P}rCat \to \mathscr{C}artMulti$ has a left adjoint constructed exactly as in Theorem 2.9.1.*

Proof. This can be proven directly and concretely (see Exercise 2.9.4); but it also follows from general theory of 2-monads [**?**] and the fact that categories with products are 2-monadic over cartesian multicategories. □

Thus, up to equivalence, the category of contexts of a finite-product theory $\mathcal{P}$ is the free category-with-products generated by $\mathcal{P}$. In particular, it uniquely determines, up to equivalence, the category of morphisms of theories from $\mathcal{P}$ into any category with products (such morphisms are often called **models** of the theory): they are the same as product-preserving functors out of the category of contexts. This example was the original observation of Lawvere [**?**] that gave rise to categorical logic.

Intuitively, the category of contexts can be said to contain the "extensional essence" of a finite-product theory. As mentioned in §1.7.4, two theories that generate equivalent categories of contexts have "the same models" in all categories, and are said to be **Morita equivalent**. For instance, the notion of group can be presented with a multiplication, unit, and inverse, or with a unit and "division" operation; these are distinct theories but are Morita equivalent. (The study of finite-product theories, particularly those with only one type, is also known as *universal algebra*; although classical universal algebraists mainly study models in **Set** rather than more general categories.)

Remark 2.9.3. This seems an appropriate place to mention an alternative approach to terms in type theory that is fairly common, especially for finite-product theories. Rather than giving rules that inductively generate judgments (with contexts) and then assigning terms to represent them uniquely with variables associated to the types in the context, some authors instead suppose given from the start a different collection of "variables" associated to each type. Then the terms (without contexts) are defined inductively by applying generating morphisms to variables of appropriate types, and finally a "valid context for a term" is *defined* to be any list of variables (with their associated types) that includes all the variables appearing (freely) in the term. The end result is much the same, but our way of keeping track of the context all the way through matches the category theory better (since every morphism in a category has a specified domain) and makes for cleaner inductive arguments.

2.9.2 Presheaves on multicategories

Now we move on to cartesian theories with type operations, and hence 1-skeleta that differ from the 2-skeleta. This will require us to discuss "contravariant representable functors" for a multicategory; but what are those? More specifically, what *sort of thing* are they? Inspecting the structure possessed by the family of sets $\mathcal{M}(A_1, \ldots, A_n; B)$, for a fixed B as the A_i vary, leads us to the following definition.

Definition 2.9.4. Let $\mathcal{M}$ be an $\mathfrak{S}$-multicategory and $\mathbf{C}$ a category. A **C-valued presheaf on** $\mathcal{M}$ consists of

(a) For each list $(A_1, \ldots, A_n)$ of objects of $\mathcal{M}$, an object $\mathcal{H}(A_1, \ldots, A_n) \in \mathbf{C}$.

(b) For each list $(f_1, \ldots, f_m)$ of morphisms of $\mathcal{M}$, with $f_i : (A_{i1}, \ldots, A_{in_i}) \to B_i$, a morphism in $\mathbf{C}$:

$$(f_1, \ldots, f_n)^* : \mathcal{H}(B_1, \ldots, B_m) \to \mathcal{H}(A_{11}, \ldots, A_{mn_m})$$

which are associative and unital with respect to composition in $\mathcal{M}$:

$$(f_{11}, \ldots, f_{mn_m})^* \circ (g_1, \ldots, g_m)^* = (g_1 \circ (f_{11}, \ldots, f_{1n_1}), \ldots, g_m \circ (f_{m1}, \ldots, f_{mn_m}))^*$$
$$(\mathsf{id}_{A_1}, \ldots, \mathsf{id}_{A_n})^* = \mathsf{id}_{\mathcal{H}(A_1,\ldots,A_n)}$$

(c) For each $\sigma : \{1, \ldots, m\} \to \{1, \ldots, n\}$ in $\mathfrak{S}$, a morphism in $\mathbf{C}$:

$$\sigma^* : \mathcal{H}(A_{\sigma 1}, \ldots, A_{\sigma m}) \to \mathcal{H}(A_1, \ldots, A_n)$$

such that

$$(f_1\sigma_1^*, \ldots, f_n\sigma_n^*)^* = (\sigma_1 \sqcup \cdots \sqcup \sigma_n)^* \circ (f_1, \ldots, f_n)^*$$
$$(f_1, \ldots, f_n)^* \circ \sigma^* = (\sigma \wr (k_1, \ldots, k_n))^* \circ (f_{\sigma 1}, \ldots, f_{\sigma m})^*$$

where k_i is the arity of f_i.

Conveniently, it turns out that this definition can be reformulated as an ordinary functor on a different category:

Theorem 2.9.5. *For any $\mathfrak{S}$-multicategory $\mathcal{M}$, a $\mathbf{C}$-valued presheaf on $\mathcal{M}$ as in Definition 2.9.4 is the same as an ordinary functor $\widehat{\mathcal{M}}^{\mathrm{op}} \to \mathbf{C}$, where $\widehat{\mathcal{M}}$ is as in Theorem 2.9.1.*

Proof. A functor $\mathcal{H} : \widehat{\mathcal{M}}^{\mathrm{op}} \to \mathbf{C}$ certainly associates an object of $\mathbf{C}$ to each list $(A_1, \ldots, A_n)$ of objects of $\mathcal{M}$. The action of a morphism $(\sigma, (f_1, \ldots, f_k))$ in $\widehat{\mathcal{M}}$ corresponds to the composite $\sigma^* \circ (f_1, \ldots, f_k)^*$ of a $\mathbf{C}$-valued presheaf, and the latter actions can be recovered from it by putting in identities for σ or the f's. The axiom of a $\mathbf{C}$-valued presheaf involving $\sigma_1 \sqcup \cdots \sqcup \sigma_n$ comes from the quotient involved in the hom-sets of $\widehat{\mathcal{M}}$, while the other axioms come from the definition of identities and composition in $\widehat{\mathcal{M}}$. □

For instance, this immediately implies a Yoneda lemma:

Corollary 2.9.6. *For any $\mathfrak{S}$-multicategory $\mathcal{M}$, object $B \in \mathcal{M}$, and* **Set***-valued presheaf $\mathcal{H}$ on $\mathcal{M}$, there is a natural bijection between natural transformations $\mathcal{M}(-; B) \to \mathcal{H}$ and elements of $\mathcal{H}(B)$.* □

Specializing all of this to the cartesian case, we can now define a **1-skeleton for a cartesian closed category with coproducts**. It consists of the following (compare to the rules in Figure 2.9):

(a) A set $\mathcal{M}_0$ of objects with chosen elements $\mathbb{1}, \mathbf{0}$ and binary operations $\times, +, \to$.

(b) A cartesian multicategory $\mathcal{M}$ with $\mathcal{M}_0$ as set of objects.

(c) For each $A, B \in \mathcal{M}_0$, morphisms $A \times B \to A$ and $A \times B \to B$ and a natural transformation of **Set**-valued presheaves on $\mathcal{M}$:

$$\mathcal{M}(-;A) \times \mathcal{M}(-;B) \longrightarrow \mathcal{M}(-;A \times B)$$

(d) A natural transformation of **Set**-valued presheaves $1 \to \mathcal{M}(-;\mathbb{1})$.

(e) For each $C \in \mathcal{M}_0$, a morphism $\mathbf{0} \to C$.

(f) For each $A, B \in \mathcal{M}_0$, morphisms $A \to A + B$ and $B \to A + B$; and for any A, B, C a natural transformation:

$$\mathcal{M}(-;A + B) \times \mathcal{M}(-, A; C) \times \mathcal{M}(-, B; C) \longrightarrow \mathcal{M}(-;C)$$

(g) For any $A, B \in \mathcal{M}_0$, natural transformations:

$$\mathcal{M}(-, A; B) \longrightarrow \mathcal{M}(-;A \to B)$$

$$\mathcal{M}(-;A \to B) \times \mathcal{M}(-;A) \longrightarrow \mathcal{M}(-;B).$$

With this we can construct a tower of adjunctions as in §1.7:

$$\begin{array}{ccc}
\mathbf{Pres}_{\times,+,\to,2} & \underset{\mathfrak{U}_2}{\overset{\mathfrak{F}_2}{\rightleftarrows}}\ (\bot) & \mathbf{CCCC}_2 \\
\downarrow & & \downarrow \\
\mathbf{Pres}_{\times,+,\to,1} & \underset{\mathfrak{U}_1}{\overset{\mathfrak{F}_1}{\rightleftarrows}}\ (\bot) & \mathbf{CCCC}_1 \\
\downarrow & & \downarrow \\
\mathbf{Pres}_{\times,+,\to,0} & \underset{\mathfrak{U}_0}{\overset{\mathfrak{F}_0}{\rightleftarrows}}\ (\bot) & \mathbf{CCCC}_0
\end{array}$$

where $\mathbf{Pres}_{\times,+,\to,k}$ denotes the category of k-skeletal $(\times, +, \to)$-presentations, and $\mathbf{CCCC}_k$ denotes the category of k-skeleta for cartesian closed categories with coproducts. (The 1-skeleta were defined above, the 2-skeleta are simply the categories themselves, and the 0-skeleta are sets equipped with operations $\times, +, \to$ and elements $\mathbb{1}, \mathbf{0}$.)

We can also omit any of the type operations, by the modularity of type theory (principle $(*)$). And we can show as in §1.7 that every category of the appropriate sort is presented by its underlying presentation (or "internal logic"); so that if we define the morphisms between presentations appropriately we obtain an equivalence of bicategories. In particular, $(\times, \to)$-presentations (which are often given a name like "$\lambda_\times$-theories") can be assembled into a bicategory that is equivalent to that of cartesian closed categories. The discovery by Lambek [**?**] of this relationship between simply typed λ-calculus and cartesian closed categories was another important milestone in the development of categorical logic.

Exercises

Exercise 2.9.1. Re-do Exercises 1.7.3 and 1.7.4 using finite-product theories. Notice how much nicer they are.

Exercise 2.9.2. Let $\mathcal{G}$ be a (non-unary) $\times, \mathbb{1}$-theory, i.e. a multicategorical theory whose only type operations are $\times, \mathbb{1}$, but whose generating morphisms can involve $\times, \mathbb{1}$ in their domains and codomains, and with generating equalities. Show that there is another $\times$-theory $\mathcal{H}$ whose generating arrows contain only base types in their domains and codomains and such that $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G} \simeq \mathfrak{F}_{\mathbf{PrCat}}\mathcal{H}$.

Exercise 2.9.3. Complete the proof of Theorem 2.9.1 by showing that the claimed definition does in fact define a left adjoint. Also verify the claimed simpler description of the hom-sets in the cartesian case.

Exercise 2.9.4. Prove Theorem 2.9.2.

Exercise 2.9.5. By using induction over the "context judgment" from Remark 2.8.9, prove directly that the category of contexts of a finite-product theory has a universal property up to equivalence among categories with products.

2.10 Symmetric monoidal categories

Now we consider a type theory for symmetric monoidal categories. That is, we add the exchange rule (and the $\multimap$ rules) to §2.5, or remove weakening and contraction from §2.8.

2.10.1 Shuffles

As always for structural rules, we want exchange to be admissible. But we cannot use the same trick for this that we did in §2.8, because in the absence of weakening the identity rule must be $x : A \vdash x : A$ rather than containing a whole context on the left. Thus, we cannot expect to push exchange all the way up to the top; instead we need to build it into the other rules. For instance, in the theory of §2.7 we can derive $B, A \vdash A \otimes B$ by using primitive exchange:

$$\dfrac{\dfrac{A \vdash A \qquad B \vdash B}{A, B \vdash A \otimes B}}{B, A \vdash A \otimes B}$$

To obtain this without primitive exchange, we must incorporate some exchange into the $\otimes I$ rule. That is, it must say something like

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B \qquad \Gamma, \Delta \cong \Phi}{\Phi \vdash A \otimes B}$$

allowing us to permute the concatenated context Γ, Δ of the premises to obtain the context of the conclusion.

However, this is not quite right yet: it introduces too much redundancy. For instance, with this rule we would have the following derivation of $A, B, C \vdash A \otimes (B \otimes C)$:

$$\frac{A \vdash B \qquad \dfrac{B \vdash B \qquad C \vdash C \qquad B, C \cong C, B}{C, B \vdash B \otimes C} \qquad A, C, B \cong A, B, C}{A, B, C \vdash A \otimes (B \otimes C)}$$

which would be distinct from the obvious one:

$$\frac{A \vdash B \qquad \dfrac{B \vdash B \qquad C \vdash C \qquad B, C \cong B, C}{B, C \vdash B \otimes C} \qquad A, B, C \cong A, B, C}{A, B, C \vdash A \otimes (B \otimes C)}$$

This is not what we want; both clearly represent the same morphism in a symmetric multicategory, and moreover both are naturally represented by the same term $x : A, y : B, z : C \vdash \{x, \{y, z\}\} : A \otimes (B \otimes C)$. What we need to do is incorporate "just enough" exchange to obtain any desired ordering of the context of the conclusion, but without introducing redundancy.

The redundancy comes from the fact that the contexts of the premises must already be free to occur in any order. Thus, we don't want to re-build-in permutations of those, only permutations that alter the relative order between the contexts of different premises. Formally, what we need is a *shuffle*.

Definition 2.10.1. For $p_1, \ldots, p_n \in \mathbb{N}$, a $(p_1, \ldots, p_n)$**-shuffle** is a permutation of $\bigsqcup_{i=1}^{n} \{1, \ldots, p_i\}$ with the property that it leaves invariant the internal ordering of each summand.

For instance, if we write $\{1, 2\} \sqcup \{1, 2, 3\}$ as $\{1, 2, 1', 2', 3'\}$, then here are some $(2, 3)$-shuffles:

$$121'2'3' \qquad 11'2'23' \qquad 1'2'13'2 \qquad 1'12'3'2$$

In all cases 1 comes before 2, and also $1'$ comes before $2'$ which comes before $3'$. The name "shuffle", of course, comes from the fact that when $n = 2$ this is exactly the sort of permutation that can be obtained by cutting a deck of $p + q$ cards into a p-stack and a q-stack and riffle-shuffling them together.

Now let S_p denote the symmetric group on p elements; thus the $(p_1, \ldots, p_n)$-shuffles are elements of $S_{p_1 + \cdots + p_n}$. Note that they are not a subgroup. However, we do have a (non-normal) inclusion $S_{p_1} \times \cdots \times S_{p_n} \hookrightarrow S_{p_1 + \cdots + p_n}$ given by the **block sum of permutations**, acting on $\bigsqcup_{i=1}^{n} \{1, \ldots, p_i\}$ by permuting each summand individually. The following is straightforward to verify.

Lemma 2.10.2. *Every coset of $S_{p_1} \times \cdots \times S_{p_n}$ in $S_{p_1+\cdots+p_n}$ contains exactly one $(p_1, \ldots, p_n)$-shuffle. Thus, every permutation of $p_1 + \cdots + p_n$ can be written uniquely as the product of a block sum from $S_{p_1} \times \cdots \times S_{p_n}$ and a $(p_1, \ldots, p_n)$-shuffle.* □

If Γ_i is a context of length p_i for $i = 1, \ldots, n$, then we write $\mathrm{Shuf}(\Gamma_1, \ldots, \Gamma_n; \Psi)$ for the set of $(p_1, \ldots, p_n)$-shuffles that act on the concatenated context $\Gamma_1, \ldots, \Gamma_n$ to produce the context Ψ. When using named variables, we assume that the variable names are preserved by this action (which means that the contexts $\Gamma_1, \ldots, \Gamma_n$ and Ψ uniquely determine such a shuffle if it exists). Now we can state a better version of $\otimes I$:

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B \qquad \sigma \in \mathrm{Shuf}(\Gamma, \Delta; \Phi)}{\Phi \vdash A \otimes B}$$

This allows deriving $B, A \vdash A \otimes B$, but rules out the undesired redundant derivation above, since the permutation $A, C, B \cong A, B, C$ is not a $(1, 2)$-shuffle. We can treat all the other rules from §2.5 (and also the $\multimap$ rules) similarly, moving the active types in the context to the far right as we did in §2.8; the result is shown in Figure 2.11.

Lemma 2.10.3. *Exchange is admissible in this type theory: if we have a derivation of $\Gamma \vdash A$, and a permutation $\Gamma \cong \Delta$ rearranging Γ to Δ, then we can construct a derivation of $\Delta \vdash A$. Moreover, this operation is a group action.*

Proof. We induct on the given derivation of $\Gamma \vdash A$, and most of the cases are essentially the same. Consider $\otimes I$: given its premises and a permutation $\Phi \cong \Phi'$, we decompose the composite permutation $\Gamma, \Delta \cong \Phi \cong \Phi'$ uniquely as the block sum of two permutations $\Gamma \cong \Gamma'$ and $\Delta \cong \Delta'$ with a shuffle $\Gamma', \Delta' \cong \Phi'$. Now by the inductive hypothesis we can derive $\Gamma' \vdash A$ and $\Delta' \vdash B$, whence applying $\otimes I$ again with the shuffle $(\Gamma', \Delta') \cong \Phi'$ we get $\Phi' \vdash A \otimes B$. All the other cases can be treated similarly.

We likewise show that it is a group action by induction. In the case of $\otimes I$, if we have $\Phi \cong \Phi' \cong \Phi''$, we decompose $(\Gamma, \Delta) \cong \Phi \cong \Phi'$ as $(\Gamma, \Delta) \cong (\Gamma', \Delta') \cong \Phi'$; then we decompose $(\Gamma', \Delta') \cong \Phi' \cong \Phi''$ as $(\Gamma', \Delta') \cong (\Gamma'', \Delta'') \cong \Phi''$. But since composition of permutations is associative, this is the same as decomposing $(\Gamma, \Delta) \cong \Phi \cong \Phi''$ as $(\Gamma, \Delta) \cong (\Gamma'', \Delta'') \cong \Phi''$ directly. We conclude by applying the inductive hypothesis to the actions of $\Gamma \cong \Gamma' \cong \Gamma''$ and $\Delta \cong \Delta' \cong \Delta''$ on the premises. □

Note that the shuffles appearing in the premises are *not* notated explicitly in the terms! Nevertheless, terms still uniquely determine derivations, because we can inspect the order that the variables appear in a term. As in §2.5 we need "superlinearity" first.

Lemma 2.10.4. *If $\Gamma \vdash M : A$ is derivable, then every variable in Γ appears at least once (free) in M.*

$$\frac{\vdash A\ \mathsf{type}}{x : A \vdash x : A}\ \mathsf{id}$$

$$\frac{f \in \mathcal{G}(A_1, \ldots, A_n; B) \quad \Gamma_1 \vdash M_1 : A_1 \quad \ldots \quad \Gamma_n \vdash M_n : A_n \quad \mathrm{Shuf}(\Gamma_1, \ldots \Gamma_n; \Phi)}{\Phi \vdash f(M_1, \ldots, M_n) : B}\ fI$$

$$\frac{\Gamma \vdash M : A \quad \Delta \vdash N : B \quad \mathrm{Shuf}(\Gamma, \Delta; \Phi)}{\Phi \vdash \{M, N\} : A \otimes B}\ \otimes I$$

$$\frac{\Psi \vdash M : A \otimes B \quad \Gamma, x : A, y : B \vdash N : C \quad \mathrm{Shuf}(\Gamma, \Psi; \Phi)}{\Phi \vdash \mathsf{match}_{A \otimes B}(M, xy.N) : C}\ \otimes E$$

$$\frac{}{() \vdash \star : \mathbf{1}}\ \mathbf{1}I \qquad \frac{\Psi \vdash M : \mathbf{1} \quad \Gamma \vdash N : C \quad \mathrm{Shuf}(\Gamma, \Psi; \Phi)}{\Phi \vdash \mathsf{match}_{\mathbf{1}}(M, N) : C}\ \mathbf{1}E$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash \langle M, N \rangle : A \times B}\ \times I \qquad \frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \pi_1^{A,B}(M) : A}\ \times E1$$

$$\frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \pi_2^{A,B}(M) : B}\ \times E2$$

$$\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash *(x_1, \ldots, x_n) : \mathbb{1}}\ \mathbb{1}I \qquad \frac{\Psi \vdash M : \mathbf{0} \quad \mathrm{Shuf}(\Gamma, \Psi; \Phi)}{\Phi \vdash \mathsf{match}_{\mathbf{0}}^{\Gamma}(M) : C}\ \mathbf{0}E$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \mathsf{inl}(M) : A + B}\ +I1 \qquad \frac{\Gamma \vdash N : B}{\Gamma \vdash \mathsf{inr}(N) : A + B}\ +I2$$

$$\frac{\Psi \vdash M : A + B \quad \Gamma, u : A \vdash P : C \quad \Gamma, v : B \vdash Q : C \quad \mathrm{Shuf}(\Gamma, \Psi; \Phi)}{\Phi \vdash \mathsf{match}_{A+B}(M, u.P, v.Q) : C}\ +E$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B}\ \multimap I$$

$$\frac{\Gamma \vdash M : A \multimap B \quad \Delta \vdash N : A \quad \mathrm{Shuf}(\Gamma, \Delta; \Phi)}{\Phi \vdash MN : B}\ \multimap E$$

Figure 2.11: Type theory for symmetric monoidal categories

Proof. An easy induction just like Lemma 2.5.1. Note that we again had to include the unused variables in $\mathbb{1}I$ and $\mathbf{0}E$ for this purpose. □

Lemma 2.10.5. *If $\Gamma \vdash M : A$ is derivable, then it has a unique derivation.*

Proof. By induction on derivations. Clearly the structure of a term determines the *rule* that must have been applied to produce it, so the question is whether it determines the premises uniquely as well. The interesting cases are the rules that involve shuffles: $fI, \otimes I, \otimes E, \mathbf{1}E, \mathbf{0}E, +E$.

Consider $\otimes I$: looking at the conclusion $\Phi \vdash \{M, N\} : A \otimes B$, the rule ensures that each variable in Φ can only occur in one of M or N, and by Lemma 2.10.4 it must appear in exactly one of them. Thus, it must be that Γ consists of those variables occurring in M while Δ consists of those variables occurring in N. Moreover, since the shuffle $\mathrm{Shuf}(\Gamma, \Delta; \Psi)$ cannot alter the relative order of variables in Γ and Δ, it must be that the variables in Γ and Δ occur in the same order as they do in Ψ. Thus the premises $\Gamma \vdash M : A$ and $\Delta \vdash N : B$ are uniqly determined, and once Γ and Δ are fixed the shuffle is also uniquely determined. All the other rules involving shuffles behave similarly. □

From this point onwards the theory looks almost exactly like that of §2.5: substitution is admissible, we define β- and η-conversion rules, and construct a free closed symmetric monoidal category with products and coproducts (or less, by omitting some of our type operations). We leave the details to the reader (Exercise 2.10.1).

In particular, because we have ensured that terms uniquely represent derivations, we can prove the initiality theorem as usual by a simple induction over derivations. To the author's knowledge the use of shuffles for this purpose is an improvement over the existing literature: it produces a free symmetric multicategory using nice-looking terms while still maintaining the "terms are derivations" principle, so that we can prove the initiality theorem without incurring the (rarely-satisfied) obligation to prove that definitions by induction over derivations depend only on the term.

One sometimes also finds remarks that the context in a linear type theory can be treated as a "finite multiset" (a "set that can contain some elements more than once"). Whether this is true depends on what exactly one means by a multiset. On one hand, if a multiset just means a set with a (finite) positive "count" labeling each element, then this is true for posetal linear logic as in §2.7, but not when we want to present non-posetal symmetric monoidal categories, since it doesn't allow us to distinguish between $x : A, y : A \vdash \{x, y\} : A \otimes A$ and $x : A, y : A \vdash \{y, x\} : A \otimes A$. On the other hand, if a multiset means a set with a (finite) nonempty *set* of "occurrences" labeling each element, then the occurrences play essentially the same role as named variables. This suggests a type theory corresponding somehow to the "fat symmetric multicategories" of [**?**, Appendix A]; but we will not pursue this further.

2.10.2 Symmetric ⊗-presentations

We can now enhance the theory of §2.10.1, like in §1.7, to take a "presentation" as input. That is, we allowing generating morphisms with arbitrary types in their domains and codomains, as well as generating equalities relating arbitrary pairs of parallel terms.

As in §2.9.1, the simplest case is if we omit all type operations (so that we have a theory only of symmetric multicategories, which coincide with their 1-skeleta) but allow arbitrary generating equalities. In this case we obtain what might be called **linear finitary algebraic theories**: a set of types, a set of operations with finite arities and types, and a set of axioms about the the composites of those operations such that "each variable appears exactly once on both sides of each axiom". For instance, the theory of monoids is linear, with the axioms:

$$x : A, y : A, z : A \vdash x \cdot (y \cdot z) \equiv (x \cdot y) \cdot z : A$$
$$x : A \vdash x \cdot e \equiv x : A$$
$$x : A \vdash e \cdot x \equiv x : A$$

but the theory of groups, which adds a unary operation i and the axioms

$$x : A \vdash x \cdot i(x) \equiv e : A$$
$$x : A \vdash i(x) \cdot x \equiv e : A$$

is not. Note that formally, we do not have to give a precise meaning to "each variable appears exactly once on both sides of each axiom"; instead the terms that can appear in axioms are defined inductively by rules that happen to *ensure* that this condition holds.

If $\mathcal{G}$ is a linear finitary algebraic theory, then of course it generates a free symmetric multicategory $\mathfrak{F}_{\mathbf{SymMulti}}\mathcal{G}$ whose objects are precisely the (base) types of $\mathcal{G}$. In particular, if $\mathcal{G}$ has one type, then $\mathfrak{F}_{\mathbf{SymMulti}}\mathcal{G}$ has one object. A (symmetric) multicategory with one object is called an **operad** (enriched in **Set**— though much of the interest of operads lies in operads enriched in other categories); they were originally defined by [**?**], and the terminology has since been much generalized [**?**]. (Indeed, arbitrary multicategories are sometimes called "colored operads".)

On the other hand, there are cases where we want to allow tensor products in the codomain. For instance, the theory of bimonoids mentioned in §0.1 would have one type A, four generating morphisms

$$m : (A, A) \to A \qquad e : () \to A \qquad \triangle : A \to A \otimes A \qquad \varepsilon : A \to \mathbf{1}$$

and the axioms shown in Figure 2.12.

To make sense of this, we need a notion of 1-skeleton that can describe the naturality properties of the type operations from Figure 2.11. This requires the notion of "presheaf on a multicategory" from §2.9.2 (which we intentionally defined for arbitrary $\mathfrak{S}$ so that we could use it here too). Here, however, we will need not just the category of such presheaves but a *multicategory* of them.

$$
\begin{aligned}
x : A, y : A, z : A &\vdash x \cdot (y \cdot z) \equiv (x \cdot y) \cdot z : A \\
x : A &\vdash x \cdot e \equiv x : A \\
x : A &\vdash e \cdot x \equiv x : A \\
x : A &\vdash \mathsf{match}_{\otimes}(\triangle(x), uv.\{u, \triangle(v)\}) \\
&\qquad \equiv \mathsf{match}_{\otimes}(\triangle(x), uv.\mathsf{match}_{\otimes}(\triangle(u), wz.\{w, \{z, v\}\}) \\
&\qquad\qquad : A \otimes (A \otimes A) \\
x : A &\vdash \mathsf{match}_{\otimes}(\triangle(x), uv.\mathsf{match}_{\mathbf{1}}(\varepsilon(u), v)) \equiv x : A \\
x : A &\vdash \mathsf{match}_{\otimes}(\triangle(x), uv.\mathsf{match}_{\mathbf{1}}(\varepsilon(v), u)) \equiv x : A \\
x : A, y : A &\vdash \mathsf{match}_{\otimes}(\triangle(x), uv, \mathsf{match}_{\otimes}(\triangle(y), wz.\{m(u, w), m(v, z)\})) \\
&\qquad \equiv \triangle(m(x, y)) : A \otimes A \\
x : A, y : A &\vdash \varepsilon(m(x, y)) \equiv \mathsf{match}_{\mathbf{1}}(\varepsilon(x), \mathsf{match}_{\mathbf{1}}(\varepsilon(y), \star)) : \mathbf{1} \\
() &\vdash \triangle(e) \equiv \{e, e\} : A \otimes A \\
() &\vdash \varepsilon(e) \equiv \star : \mathbf{1}
\end{aligned}
$$

Figure 2.12: Axioms for a bimonoid

Theorem 2.10.6. *Let $\mathcal{M}$ and $\mathcal{C}$ be $\mathfrak{S}$-multicategories. Then there is an $\mathfrak{S}$-multicategory whose objects are presheaves on $\mathcal{M}$ valued in the underlying ordinary category of $\mathcal{C}$, as in Definition 2.9.4.*

*A morphism $(\mathcal{H}_1, \dots, \mathcal{H}_n) \to \mathcal{K}$ (called an **n-ary natural transformation**) consists of for each $A_{11}, \dots, A_{nm_n}$, a morphism in $\mathcal{C}$:*

$$\alpha : (\mathcal{H}(A_{11}, \dots, A_{1m_1}), \dots, \mathcal{H}(A_{n1}, \dots, A_{nm_n})) \longrightarrow \mathcal{K}(A_{11}, \dots, A_{nm_n})$$

such that

$$
\begin{aligned}
\alpha \circ (\sigma_1^*, \dots, \sigma_n^*) &= (\sigma_1 \sqcup \dots \sqcup \sigma_n)^* \circ \alpha \\
\alpha \circ ((f_{11}, \dots, f_{1\ell_1})^*, \dots, (f_{k1}, \dots, f_{k\ell_k})^*) &= (f_{11}, \dots, f_{k\ell_k})^* \circ \alpha
\end{aligned}
\qquad \square
$$

Technically, the α's should have appropriate subscripts, but that would make the notation even heavier and harder to understand.

The point of this definition is to capture the behavior of substitution into a rule that concatenates two (or more) contexts in its premises (perhaps with a shuffle) to form the context of its conclusion. With it in hand, we can define a **1-skeleton for a closed symmetric monoidal category with products and coproducts** to consist of the following (refer to Figure 2.11).

(a) A set $\mathcal{M}_0$ with chosen elements $\mathbf{1}, \mathbb{1}, \mathbf{0}$ and binary operations $\otimes, \times, +, \multimap$.

(b) A symmetric multicategory $\mathcal{M}$ with object set $\mathcal{M}_0$.

(c) For each A, B, a 2-ary natural transformation

$$(\mathcal{M}(-; A), \mathcal{M}(-; B)) \longrightarrow \mathcal{M}(-; A \otimes B)$$

(d) For each A, B, C, a 2-ary natural transformation

$$(\mathcal{M}(-, A, B; C), \mathcal{M}(-; A \otimes B)) \longrightarrow \mathcal{M}(-; C)$$

(e) A morphism $() \to \mathbf{1}$.

(f) For each C, a 2-ary natural transformation

$$(\mathcal{M}(-; C), \mathcal{M}(-; \mathbf{1})) \longrightarrow \mathcal{M}(-; C)$$

(g) For each A, B, morphisms $A \times B \to A$ and $A \times B \to B$ and a (1-ary) natural transformation

$$\mathcal{M}(-; A) \times \mathcal{M}(-; B) \longrightarrow \mathcal{M}(-; A \times B)$$

(h) A natural transformation $1 \to \mathcal{M}(-; \mathbb{1})$.

(i) For each $C \in \mathcal{M}_0$, a 2-ary natural transformation

$$(1, \mathcal{M}(-; \mathbf{0})) \longrightarrow \mathcal{M}(-; C)$$

where 1 denotes the terminal **Set**-valued presheaf on $\mathcal{M}$.

(j) For each $A, B \in \mathcal{M}_0$, morphisms $A \to A + B$ and $B \to A + B$; and for any A, B, C a 2-ary natural transformation:

$$(\mathcal{M}(-, A; C) \times \mathcal{M}(-, B; C), \mathcal{M}(-; A + B)) \longrightarrow \mathcal{M}(-; C)$$

(k) For any $A, B \in \mathcal{M}_0$, a natural transformation

$$\mathcal{M}(-, A; B) \longrightarrow \mathcal{M}(-; A \multimap B)$$

(l) For any $A, B \in \mathcal{M}_0$, a 2-ary natural transformation

$$(\mathcal{M}(-; A \multimap B), \mathcal{M}(-; A)) \longrightarrow \mathcal{M}(-; B).$$

From here, the theory of presentations proceeds as before, yielding a tower of adjunctions. We can also treat the non-symmetric case using similar ideas; see Exercise 2.10.3.

As an example, let us try to reproduce the uniqueness of antipodes calculation from §0.1. We have already mentioned the theory of a bimonoid in 2.12; an antipode augments this by a generating morphism $i : A \to A$ and the axioms

$$x : A \vdash \mathsf{match}_{\otimes}(\triangle(x), uv.m(u, i(v))) \equiv \mathsf{match}_{\mathbf{1}}(\varepsilon(x), e) : A$$
$$x : A \vdash \mathsf{match}_{\otimes}(\triangle(x), uv.m(i(u), v)) \equiv \mathsf{match}_{\mathbf{1}}(\varepsilon(x), e) : A$$

Thus, let us include these and also another antipode $j : A \to A$.

[TODO]

Exercises

Exercise 2.10.1. Finish the theory of §2.10.1: prove the admissibility of substitution, state the β- and η-conversion rules, and prove the initiality theorem.

Exercise 2.10.2. Suppose $\mathcal{M}$ is a symmetric multicategory and $\mathcal{C}$ is a cocomplete closed symmetric monoidal category, regarded as a multicategory. Show that the symmetric multicategory constructed in Theorem 2.10.6 is in fact a closed symmetric monoidal category. (In fact, it is the **Day convolution** [**?**] monoidal category constructed from $\widehat{\mathcal{M}}$ and $\mathcal{C}$.)

Exercise 2.10.3. Define 1-skeleta for non-symmetric monoidal categories, and generalize the initiality theorems of §§2.4 and 2.5 to a tower of adjunctions involving an appropriate kind of presentation.

Exercise 2.10.4. Another approach to linear type theory is to annotate some types in the context with an "unused" marker such as $(-)^0$, and allow weakening of "unused" types. Thus we could have for instance $x : A^0, y : B, z : C^0 \vdash y : B$, since x and z are marked as unused. Two of these contexts Γ and Δ can be *merged* if they contain the same types in the same order, and each type is used (the opposite of unused) in at most one of them; in that case they merge to a context $\Gamma \boxplus \Delta$ containing the same types again, with those used that are used in either Γ or Δ. For instance, $(A^0, B, C^0) \boxplus (A, B^0, C^0) = (A, B, C^0)$.

(a) Formulate a type theory containing $\otimes, \times, +, \multimap$ using contexts with usage markers and merging rather than concatenation. For instance, the rule $\otimes I$ should be
$$\frac{\Gamma \vdash M : A \qquad \Delta \vdash N : B}{\Gamma \boxplus \Delta \vdash \{M, N\} : A \otimes B.}$$
Prove that exchange, and weakening for unused types, are admissible and functorial, by pushing them up the entire derivation as we did in the cartesian case in §2.8.

(b) Define a corresponding multicategory-like structure whose domains are lists with usage markers, and establish a correspondence of some sort with symmetric monoidal categories.

(c) Prove an initiality theorem.

Exercise 2.10.5. Is it possible to formulate a theory for semicartesian or relevance monoidal categories in which the appropriate structural rules are admissible?

TODO: `https://nforum.ncatlab.org/discussion/8622/church-numerals-are-realizers/`?

Collected Exercises

For convenient reference, we collect the exercises from all sections in this chapter.

Exercise 2.2.1. Prove that the definitions of multicategory in terms of multicomposition and one-place composition are equivalent, in the strong sense that they yield isomorphic categories of multicategories.

Exercise 2.2.2. Fill in the details in the proof of Theorem 2.2.4.

Exercise 2.2.3. Show that the category whose objects are representable multicategories but whose morphisms are *arbitrary* functors of multicategories is equivalent to the category of monoidal categories and *lax* monoidal functors.

Exercise 2.2.4. Show that the category of representable multicategories and functors that "preserve tensor products", in the sense that if $\chi : (A_1, \ldots, A_n) \to \bigotimes_i A_i$ is a tensor product then $F(\chi)$ is also *a* tensor product, is equivalent to the category of monoidal categories and *strong* monoidal functors.

Exercise 2.2.5. Fill in the details in the proof of Theorem 2.2.6.

Exercise 2.3.1. Prove the well-formedness, cut-admissibility, and initiality theorems for the natural deduction for monoidal posets.

Exercise 2.3.2. Prove that the rules $\otimes L$ and $\mathbf{1}L$ in the sequent calculus for monoidal posets are invertible in the sense of Exercise 1.3.3: whenever we have a derivation of their conclusions, we also have derivations of their premises.

Exercise 2.3.3. Write down either a sequent calculus or a natural deduction for monoidal posets that are also meet-semilattices, and prove its initiality theorem.

Exercise 2.3.4. Let us augment the sequent calculus for monoidal posets by the following versions of the rules for join-semilattices:

$$\frac{\vdash A \ \mathsf{type} \qquad \vdash B \ \mathsf{type}}{\vdash A \vee B \ \mathsf{type}} \qquad \frac{}{\vdash \bot \ \mathsf{type}} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$$

$$\frac{\Gamma, A, \Delta \vdash C \qquad \Gamma, B, \Delta \vdash C}{\Gamma, A \vee B, \Delta \vdash C} \qquad \frac{}{\Gamma, \bot, \Delta \vdash C}$$

(a) Construct derivations in this calculus of the following sequents:

$$(A \otimes B) \vee (A \otimes C) \vdash A \otimes (B \vee C)$$
$$A \otimes (B \vee C) \vdash (A \otimes B) \vee (A \otimes C)$$

(b) Prove that this sequent calculus constructs the initial distributive monoidal poset (see Theorem 2.2.6).

Exercise 2.4.1. Our proof of Theorem 2.4.10 relied on the fact that monoidal categories are equivalent to representable multicategories, which we sketched but did not prove carefully. If we don't assume this fact, then our proof of Theorem 2.4.10 is actually just about free representable multicategories. Using this version of the theorem, prove *using type theory* that any representable multicategory is monoidal: that is, its tensor product is coherently associative and unital.

Exercise 2.4.2. Formulate and prove the admissibility of a "multi-substitution" rule like Theorem 2.3.2 for the type theories considered in this section.

Exercise 2.4.3. The annotation $\Gamma|\Delta$ on $\mathsf{match}^{\Gamma|\Delta}_{A\otimes B}$ is something that appears only in the non-symmetric case, so we encourage the reader not to worry overmuch about it. However, for the reader who nevertheless insists on worrying, here is some extra reassurance.

(a) We noted in Lemma 2.4.8 that this annotation on $\mathsf{match}^{\Gamma|\Delta}_{A\otimes B}(M, xy.N)$ is only necessary if M contains no variables. To see that it can actually matter in that case, find an example of two distinct derivations whose corresponding terms differ *only* in their annotations $\Gamma|\Delta$.

(b) Prove that any two terms as in (a) are related by $\equiv$.

Exercise 2.4.4. Describe precisely what has to happen to de-Bruijn-style variables when concatenating contexts, and formulate the rules for the type theories of this section using de Bruijn variables.

Exercise 2.5.1. Find an example of a distributive monoidal category with products in which the two terms in (2.5.3) represent distinct morphisms.

Exercise 2.6.1. Fill in the details in the proof of Theorems 2.6.5 to 2.6.7.

Exercise 2.6.2. Let $\mathfrak{S}$ be a faithful cartesian club.

(a) Prove that if $\mathfrak{S}$ contains the transposition $\{1,2\} \xrightarrow{\sim} \{1,2\}$, then it contains all bijections.

(b) Prove that if $\mathfrak{S}$ contains the transposition $\{1,2\} \xrightarrow{\sim} \{1,2\}$ and also the injection $\emptyset \to \{1\}$, then it contains all injections.

(c) Prove that if $\mathfrak{S}$ contains the transposition $\{1,2\} \xrightarrow{\sim} \{1,2\}$ and also the surjection $\{1,2\} \to \{1\}$, then it contains all surjections.

Exercise 2.6.3. Define one-place versions of $\mathfrak{S}$-multicategories and show that they are equivalent to the multi-composition version defined in the text.

Exercise 2.6.4. Show that representable cartesian multicategories with coproducts are equivalent to distributive categories.

Exercise 2.6.5. Of course, for any $\mathfrak{S}$ a **functor** between $\mathfrak{S}$-multicategories is required to preserve the σ-actions. Prove that:

(a) Any functor between semicartesian multicategories must preserve unit objects / terminal objects.

(b) Any functor between cartesian multicategories must preserve tensor products / cartesian products.

Exercise 2.6.6. Define a notion of **relevance monoidal category**, by adding "natural diagonals" to a symmetric monoidal category, and show that such monoidal categories are equivalent to representable relevance multicategories. (See [**?**].)

Exercise 2.6.7. Define a notion of **faithful cocartesian club** and a corresponding notion of generalized multicategory that includes *cocartesian* monoidal categories as the maximal case.

Exercise 2.7.1. Prove Theorems 2.6.6 and 2.6.7 using our posetal type theories. Specifically:

(a) If we have exchange and weakening, prove that $\mathbf{1} \cong \top$.

(b) If we have exchange, weakening, and contraction, prove that $A \otimes B \cong A \times B$.

Exercise 2.7.2. Prove that $\neg\neg(P \vee \neg P)$ is an intuitionistic tautology, i.e. construct a derivation of $() \vdash \neg\neg(P \vee \neg P)$ in the natural deduction for Heyting algebras.

Exercise 2.7.3. Prove that the following are equivalent for a Heyting algebra:

(a) The law of excluded middle $P \vee \neg P$ is true, i.e. $P \vee \neg P$ is the top element for all P.

(b) The law of double negation $\neg\neg P \Rightarrow P$ is true.

(c) The Heyting algebra is a Boolean algebra, i.e. every element P has a "complement" $\overline{P}$ such that $P \wedge \overline{P} = \bot$ and $P \vee \overline{P} = \top$.

Exercise 2.7.4. Of the four "de Morgan's laws", three are intuitionistic tautologies and one is not. Construct derivations of three of the following sequents in the natural deduction for Heyting algebras:

$$\neg(P \vee Q) \vdash \neg P \wedge \neg Q$$
$$\neg(P \wedge Q) \vdash \neg P \vee \neg Q$$
$$\neg P \wedge \neg Q \vdash \neg(P \vee Q)$$
$$\neg P \vee \neg Q \vdash \neg(P \wedge Q)$$

Exercise 2.7.5. A **frame** is a lattice with infinitary joins satisfying the infinite distributive law $A \wedge (\bigvee_i B_i) \cong \bigvee_i (A \wedge B_i)$.

(a) Prove that any (small) frame is a Heyting algebra.

(b) Prove that the lattice of open sets of any topological space is a frame.

(c) Describe a type theory for frames. This is called (propositional) **geometric logic**.

Exercise 2.7.6. Give concrete examples of Heyting algebras satisfying the following:

(a) There is an element P for which $P \vee \neg P$ is not the top element.

(b) There are elements P and Q for which the fourth de Morgan's law (see Exercise 2.7.4) does not hold.

Exercise 2.7.7. Describe a concrete example of a closed relevance monoidal lattice containing two objects A and B such that there is no morphism from $\mathbf{1}$ (the unit object) to $A \multimap (B \multimap A)$. Deduce that $() \vdash A \multimap (B \multimap A)$ is not derivable in the type theory for closed relevance monoidal lattices.

Exercise 2.7.8. One of the advantages of sequent calculus over natural deduction is that because all of its rules *introduce* operations on the left or the right, it is easier to conclude underivability theorems.

(a) Define a sequent calculus for closed $\mathfrak{S}$-monoidal lattices, and prove the cut admissibility and initiality theorems.

(b) Prove that $() \vdash A \multimap (B \multimap A)$ is not derivable in the sequent calculus for closed relevance monoidal lattices, by ruling out all possible ways that such a derivation could end.

Exercise 2.7.9. TODO: A whole section on this? There should be a general notion of "closed category" for any suitable kind of multicategory, with a resulting combinatory logic and hilbert system. `https://nforum.ncatlab.org/discussion/4632/closed-category/`

Another way of deriving tautologies is called a **Hilbert system**. A Hilbert system can be formulated as a sort of type theory where the judgments all have empty context, i.e. are of the form $\vdash A$ where A is a propositional formula. Instead of the "modular" left/right rules of sequent calculus or the introduction/elimination rules of natural deduction, where the rules for each connective do not refer to any other connective, a Hilbert system gives a special place to implication $\Rightarrow$. The *only* rule with premises[5] is the empty-context form of $\Rightarrow E$, *modus ponens*:

$$\frac{\vdash A \Rightarrow B \qquad \vdash A}{\vdash B}$$

The behavior of all other connectives is specified by *axioms* (rules with no premises, other than the well-formedness of the formulas appearing in them). For instance, we complete the description of $\Rightarrow$ with the following axioms:

$$\frac{\vdash A\ \mathsf{type}}{\vdash A \Rightarrow A} \qquad\qquad \frac{\vdash A\ \mathsf{type} \qquad \vdash B\ \mathsf{type}}{\vdash A \Rightarrow (B \to A)}$$

$$\frac{\vdash A\ \mathsf{type} \qquad \vdash B\ \mathsf{type} \qquad \vdash C\ \mathsf{type}}{\vdash (A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))}$$

The axioms for the remaining connectives are (omitting the obvious premises

[5]Hilbert systems for more complicated logics have one or two more rules with premises, but in general there are very few.

and the $\vdash$):

$$A \Rightarrow (B \Rightarrow (A \wedge B)) \qquad (A \wedge B) \Rightarrow A \qquad (A \wedge B) \Rightarrow B$$

$$A \Rightarrow (A \vee B) \qquad B \Rightarrow (A \vee B) \qquad (A \Rightarrow C) \Rightarrow ((B \Rightarrow C) \Rightarrow ((A \vee B) \Rightarrow C))$$

$$A \Rightarrow \top \qquad \bot \Rightarrow A$$

Prove that this Hilbert system derives exactly the same tautologies as the natural deduction for Heyting algebras.

(The main reasons for using a Hilbert system seem to be that it *never* changes the context and has very few rules. This sometimes makes metatheoretic arguments easier, but at the cost of greater distance from informal mathematics, since as we have remarked the latter gives a central place to hypothetical reasoning. It should also be noted that the symbol $\vdash$ is often used differently in the context of Hilbert systems; rather than $\vdash$ being part of each judgment, the notation "$\Gamma \vdash A$" means that we can derive A (that is, $\vdash A$ in our notation) in the Hilbert system augmented by all the formulas in Γ as additional axioms.)

Exercise 2.7.10. Is there a well-behaved type theory (i.e. having admissible cut and an initiality theorem) corresponding to the (posetal version of the) "cocartesian multicategories" of Exercise 2.6.7? *(As of this writing, the answer is not known to the author.)*

Exercise 2.8.1. Complete the proof in Theorem 2.8.8 that type theory yields a cartesian multicategory by checking Definition 2.6.4(c) and (d), or perhaps the corresponding one-place axioms you found in Exercise 2.6.3.

Exercise 2.8.2.

(a) For any natural number n and any type A in STLC, show that there is a term of type $(A \to A) \to (A \to A)$ that iterates a function n types. Call this term $[n]_A$.

(b) Define $[n+m]_A$ in terms of $[n]_A$ and $[m]_A$.

(c) Now define $[nm]_A$ in terms of $[n]_A$ and $[m]_A$ (you may have to use $[n]_A$ or $[m]_A$ for more than one value of A).

Exercise 2.8.3.

(a) Write down terms in the simply typed λ-calculus (with $\to$ the only type constructor) that have the following types (for arbitrary A, B, C):

$$A \to A$$

$$A \to B \to A$$

$$(A \to B \to C) \to (A \to B) \to (A \to C)$$

(Remember that the type operator $\to$ associates to the right: $X \to Y \to Z$ means $X \to (Y \to Z)$.)

(b) By **combinatory logic** we will mean the type theory obtained from simply typed λ-calculus by *removing* the λ-abstraction rule:

$$\cancel{\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}}$$

and instead adding axioms called I, K, and S having the above types (for any A, B, C):

$$\frac{\vdash A \text{ type}}{\Gamma \vdash I_A : A \to A} \qquad \frac{\vdash A \text{ type} \quad \vdash B \text{ type}}{\Gamma \vdash K_{AB} : A \to B \to A}$$

$$\frac{\vdash A \text{ type} \quad \vdash B \text{ type} \quad \vdash C \text{ type}}{\Gamma \vdash S_{ABC} : (A \to B \to C) \to (A \to B) \to (A \to C)}$$

Prove that the removed λ-abstraction rule is *admissible* in **CL**. That is, given a derivation in combinatory logic of $\Gamma, x : A \vdash M : B$, we can construct a derivation in combinatory logic of $\Gamma \vdash [x]M : A \to B$ for some $[x]M$ (note that this is an *operation* on combinatory logic terms, like substitution).

(c) Write down some $\equiv$-laws satisfied by the I, K, and S you defined in (a), and show that when they are used as generators for an $\equiv$ for combinatory logic, it also presents a free closed cartesian multicategory. One way to do this is by a direct induction on derivations; another way is exhibit a bijection between its terms and those of the simply typed λ-calculus.

(d) Compare to Exercise 2.7.9.

Exercise 2.9.1. Re-do Exercises 1.7.3 and 1.7.4 using finite-product theories. Notice how much nicer they are.

Exercise 2.9.2. Let $\mathcal{G}$ be a (non-unary) $\times, \mathbb{1}$-theory, i.e. a multicategorical theory whose only type operations are $\times, \mathbb{1}$, but whose generating morphisms can involve $\times, \mathbb{1}$ in their domains and codomains, and with generating equalities. Show that there is another $\times$-theory $\mathcal{H}$ whose generating arrows contain only base types in their domains and codomains and such that $\mathfrak{F}_{\mathbf{PrCat}}\mathcal{G} \simeq \mathfrak{F}_{\mathbf{PrCat}}\mathcal{H}$.

Exercise 2.9.3. Complete the proof of Theorem 2.9.1 by showing that the claimed definition does in fact define a left adjoint. Also verify the claimed simpler description of the hom-sets in the cartesian case.

Exercise 2.9.4. Prove Theorem 2.9.2.

Exercise 2.9.5. By using induction over the "context judgment" from Remark 2.8.9, prove directly that the category of contexts of a finite-product theory has a universal property up to equivalence among categories with products.

Exercise 2.10.1. Finish the theory of §2.10.1: prove the admissibility of substitution, state the β- and η-conversion rules, and prove the initiality theorem.

Exercise 2.10.2. Suppose $\mathcal{M}$ is a symmetric multicategory and $\mathcal{C}$ is a cocomplete closed symmetric monoidal category, regarded as a multicategory. Show that the symmetric multicategory constructed in Theorem 2.10.6 is in fact a closed symmetric monoidal category. (In fact, it is the **Day convolution** [**?**] monoidal category constructed from $\widehat{\mathcal{M}}$ and $\mathcal{C}$.)

Exercise 2.10.3. Define 1-skeleta for non-symmetric monoidal categories, and generalize the initiality theorems of §§2.4 and 2.5 to a tower of adjunctions involving an appropriate kind of presentation.

Exercise 2.10.4. Another approach to linear type theory is to annotate some types in the context with an "unused" marker such as $(-)^0$, and allow weakening of "unused" types. Thus we could have for instance $x : A^0, y : B, z : C^0 \vdash y : B$, since x and z are marked as unused. Two of these contexts Γ and Δ can be *merged* if they contain the same types in the same order, and each type is used (the opposite of unused) in at most one of them; in that case they merge to a context $\Gamma \boxplus \Delta$ containing the same types again, with those used that are used in either Γ or Δ. For instance, $(A^0, B, C^0) \boxplus (A, B^0, C^0) = (A, B, C^0)$.

(a) Formulate a type theory containing $\otimes, \times, +, \multimap$ using contexts with usage markers and merging rather than concatenation. For instance, the rule $\otimes I$ should be

$$\frac{\Gamma \vdash M : A \qquad \Delta \vdash N : B}{\Gamma \boxplus \Delta \vdash \{M, N\} : A \otimes B.}$$

Prove that exchange, and weakening for unused types, are admissible and functorial, by pushing them up the entire derivation as we did in the cartesian case in §2.8.

(b) Define a corresponding multicategory-like structure whose domains are lists with usage markers, and establish a correspondence of some sort with symmetric monoidal categories.

(c) Prove an initiality theorem.

Exercise 2.10.5. Is it possible to formulate a theory for semicartesian or relevance monoidal categories in which the appropriate structural rules are admissible?

Chapter 3

Classical type theories

3.1 Props and symmetric monoidal categories

We would like a type theory for symmetric monoidal categories that can also deal "primitively" with tensors in the codomain (without needing to wrap and unwrap terms belonging to a tensor product type). For this purpose we need a categorical structure like a multicategory, but which also allows sequences of objects in the codomain of morphisms. Such a structure is called a **prop**, and the simplest way to define it is the following:

Definition 3.1.1. A **prop** consists of

(a) A set of **objects**, and

(b) A symmetric strict monoidal category (that is, a symmetric monoidal category whose associators and unitors are identities) whose underlying monoid of objects is freely generated by the set of objects of the prop.

We write the objects of the monoidal category in (b) as finite lists $(A, B, \ldots, Z)$ of generating objects. The monoidal structure is given by concatenation of lists; the unit object is the empty list $()$. We will use the notation $\bullet$ for the tensor product in this monoidal category, to remind ourselves that it plays a different role than $\otimes$: in type theory, $\otimes$ is an operation on types, whereas $\bullet$ simply corresponds to concatenation of contexts.

(The original Adams–MacLane definition of prop had only one object; thus our "props" are sometimes called "colored props". However, we dispense with the adjective.)

The underlying data of a prop, from which we intend to freely generate one, is a *polygraph*.

Definition 3.1.2. A **polygraph** $\mathcal{G}$ is a set of objects together with a set of arrows, each assigned a domain and codomain that are both finite lists of objects.

In our type theories for categories and multicategories in §§1.2.2 and 2.4 (before introducing any operations such as products or tensors), we did not have to impose any equivalence relation on the derivations. However, in the case of props, the interchange rule for $\bullet$ makes things more complicated. For instance, if we have $f \in \mathcal{G}(A; B)$ and $g \in \mathcal{G}(C; D)$, here are two plausible-looking derivations of a sequent representing $f \bullet g$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{() \vdash ()}}{A \vdash A}}{A \vdash B}\, f}{\dfrac{A, C \vdash B, C}{A, C \vdash B, D}\, g} \qquad\qquad \dfrac{\dfrac{\dfrac{\dfrac{}{() \vdash ()}}{C \vdash C}}{C \vdash D}\, g}{\dfrac{A, C \vdash A, D}{A, C \vdash B, D}\, f}$$

If we write down a term calculus whose terms correspond exactly to derivations, as we usually do, then the desired equality between these two derivations would look something like

$$x : A, y : C \vdash (f, \mathsf{id})((\mathsf{id}, g)(x, y)) \equiv (\mathsf{id}, g)((f, \mathsf{id})(x, y)) : (B, D) \tag{3.1.3}$$

Note that unlike the β- and η-conversions, the equality (3.1.3) is not directional: it makes no sense to regard one or the other side as "simpler" or "more canonical" than the other. We would like to avoid having to assume such equalities in $\equiv$, and furthermore the terms appearing in (3.1.3) are rather ugly. One approach to deal with this would be to break the bijection between terms and deductions, in a way that enables us to represent both of the above two derivations by the same term. However, a better approach is to design a different theory in which there is only *one* derivation of $f \bullet g$, allowing us to maintain the principle that terms correspond uniquely to derivations.

The non-directionality of (3.1.3) also makes it unclear how to design a type theory in which one would be permitted but not the other. Instead we will forbid *both* of them, replacing the generator rule by a "multi-generator" rule allowing only a one-step derivation

$$\dfrac{x : A, y : C \vdash (x, y) : (A, C)}{x : A, y : C \vdash (f(x), g(y)) : (B, D)}\, f, g$$

The intuition in this term notation is of course that $f(x) : B$ and $g(y) : D$. We could write it as "$f(x) : B, g(y) : D$", but we choose to tuple the terms up as in $(f(x), g(y)) : (B, D)$ for a couple of reasons. The first reason is that when doing equational reasoning (such as for the antipode calculation), the equalities must relate entire tuples rather than single terms. The second reason is that in general, we also need to include some "terms without a type" (e.g. coming from morphisms with empty codomain $()$, which is a judgmental representation of the unit object), and this looks a little nicer when all the terms are grouped together: we write for instance $(f(x), g(y) \mid h(z))$ to mean that $h(z) : ()$. We refer to terms of this sort as **scalars**.

There are also, of course, function symbols with *multiple* outputs. To deal with this case we write $f_{(1)}(x)$, $f_{(2)}(x)$, and so on for the terms corresponding to all the types in the codomain. (This notation is motivated by the classical "Sweedler notation" for comonoids and comodules; later on we will compare them formally.) For example, we write the composite of $f : (A, B) \to (C, D)$ with $g : (E, D) \to (F, G)$ as

$$x : A, y : B, z : E \vdash (f_{(1)}(x, y), g_{(1)}(z, f_{(2)}(x, y)), g_{(2)}(z, f_{(2)}(x, y))) : (C, F, G)$$

Note that the variables in a context are not literally treated "linearly", since they can occur multiple times in the multiple "components" of a map f. Instead the "usages" of a variable are controlled by the codomain arity of the morphisms applied to them. In general, we write $\vec{f}(\vec{M})$ for the list of "all the values of f" applied to the list of arguments $\vec{M}$. Thus if $f : (A, B, C) \to (D, E)$ then $\vec{f}(\vec{M})$ would be $(f_{(1)}(M_1, M_2, M_3), f_{(2)}(M_1, M_2, M_3))$. If f has only one output, we ought technically to write the corresponding term as $f_{(1)}(\vec{M})$, but we will generally omit this and write simply $f(\vec{M})$.

This does require one further technical device (that will be almost invisible in practice). Suppose we have $f : () \to (B, C)$, written in our type theory as $() \vdash (f_{(1)}, f_{(2)}) : (B, C)$, and we compose/tensor it with itself to get a morphism $() \to (B, B, C, C)$. We would naïvely write this as $() \vdash (f_{(1)}, f_{(1)}, f_{(2)}, f_{(2)})$, but this is ambiguous since we can't tell which $f_{(1)}$ matches which $f_{(2)}$. We disambiguate the possibilities by writing $() \vdash (f_{(1)}, f'_{(1)}, f_{(2)}, f'_{(2)})$ or $() \vdash (f_{(1)}, f'_{(1)}, f'_{(2)}, f_{(2)})$. Although this issue seems to only arise for morphisms with empty domain and greater than unary codomain, for consistency we formulate the syntax with a label (like $'$) on *every* term former, and simply omit them informally when there is no risk of ambiguity (which includes the vast majority of cases). We assume given an infinite alphabet of symbols $\mathfrak{A}$ for this purpose (such as $', '', ''', \dots$, or $1, 2, 3, \dots$), and we annotate our judgments with a finite subset $\mathfrak{B} \subseteq \mathfrak{A}$ indicating which labels might have been used already in the terms (so that we can avoid re-using them).

Thus, a first approximation to our generator rule is

$$\frac{\begin{array}{c}\Gamma \vdash^{\mathfrak{B}} (\vec{M}, \dots, \vec{N}, \vec{P} \mid \vec{Z}) : (\vec{A}, \dots, \vec{B}, \vec{C}) \\ f \in \mathcal{G}(\vec{A}, \vec{D}) \qquad \cdots \qquad g \in \mathcal{G}(\vec{B}, \vec{E}) \\ \mathfrak{a}, \dots, \mathfrak{b} \notin \mathfrak{B} \text{ and pairwise distinct}\end{array}}{\Gamma \vdash^{\mathfrak{B} \cup \{\mathfrak{a}, \dots, \mathfrak{b}\}} (\vec{f}^{\mathfrak{a}}(\vec{M}), \dots, \vec{g}^{\mathfrak{b}}(\vec{N}), \vec{P} \mid \vec{Z}) : (\vec{D}, \dots, \vec{E}, \vec{C})}$$

We also have to allow scalar generators, which get collected into the $\vec{Z}$'s. Thus we have

$$\frac{\begin{array}{c}\Gamma \vdash^{\mathfrak{B}} (\vec{M}, \dots, \vec{N}, \vec{P}, \dots, \vec{Q}, \vec{R} \mid \vec{Z}) : (\vec{A}, \dots, \vec{B}, \vec{C}, \dots, \vec{D}, \vec{E}) \\ f \in \mathcal{G}(\vec{A}, \vec{F}^{\geq 1}) \qquad \cdots \qquad g \in \mathcal{G}(\vec{B}, \vec{G}^{\geq 1}) \\ h \in \mathcal{G}(\vec{C}, ()) \qquad \cdots \qquad k \in \mathcal{G}(\vec{D}, ()) \\ \mathfrak{a}, \dots, \mathfrak{b}, \mathfrak{c}, \dots, \mathfrak{d} \notin \mathfrak{B} \text{ and pairwise distinct}\end{array}}{\Gamma \vdash^{\mathfrak{B} \cup \{\mathfrak{a}, \dots, \mathfrak{b}, \mathfrak{c}, \dots, \mathfrak{d}\}} \left(\vec{f}^{\mathfrak{a}}(\vec{M}), \dots, \vec{g}^{\mathfrak{b}}(\vec{N}), \vec{R} \,\middle|\, \vec{h}^{\mathfrak{c}}(\vec{P}), \dots, \vec{k}^{\mathfrak{d}}(\vec{Q}), \vec{Z}\right) : (\vec{F}, \dots, \vec{G}, \vec{E})}$$

(Here $\vec{F}^{\geq 1}$ means that $\vec{F}$ contains at least one type.) Eventually we will also have to incorporate shuffles into the rule as in §2.10, but we postpone that for now. Let us consider instead how to prevent duplication of derivations. In addition to our desired term

$$x : A, y : C \vdash (f(x), g(y)) : (B, D) \tag{3.1.4}$$

we must also be able to write

$$x : A, y : C \vdash (f(x), y) : (B, C) \tag{3.1.5}$$

and

$$x : A, y : C \vdash (x, g(y)) : (A, D) \tag{3.1.6}$$

so how do we prevent ourselves from being able to apply the generator rule again to the latter two, obtaining two more derivations of the same morphism as (3.1.4)? The idea is to force ourselves to "apply all functions as soon as possible": we cannot apply g to y in (3.1.5) because we *could have* already applied it to produce (3.1.4). On the other hand, we could apply $h : (B, C) \to E$ in (3.1.5) to get

$$x : A, y : C \vdash (h(f(x), y)) : E$$

because h uses f as one of its inputs and so could not have been applied at the same time as f.

To be precise, we augment our judgments (*not* their terms) by labeling some of the types in the consequent as **active**, denoted by $A^\star$. We write $\vec{A}^{\star\geq 1}$ to mean that at least one of the types in $\vec{A}$ is active, $\vec{A}^\star$ to mean that they are all active, and $\vec{A}^{\star=0}$ to mean that none of them are active; if we write just $\vec{A}$ then we are not specifying whether or not any of the types are active.

The identity rule will make all types active, while the generator rule makes only the outputs of the generators active. We then restrict the generator rule to require that at least one of the *inputs* of each generator being applied must be active in the premise; this means that none of them could have been applied any sooner, since at least one of their arguments was just introduced by the previous rule. Thus, our desired derivation

$$\dfrac{\overline{A, C \vdash A^\star, C^\star}}{A, C \vdash B^\star, D^\star}\; f, g$$

is allowed, while the undesired one

$$\dfrac{\dfrac{\overline{A, C \vdash A^\star, C^\star}}{A, C \vdash B^\star, C}\; f}{A, C \vdash B, D}\; g???$$

is not allowed, since in the attempted application of g the input type C is not active. Thus our generator rule now becomes

$$\frac{\begin{array}{c}\Gamma \vdash^{\mathfrak{B}} (\vec{M},\dots,\vec{N},\vec{P},\dots,\vec{Q},\vec{R} \mid \vec{Z}) : (\vec{A}^{\star\geq 1},\dots,\vec{B}^{\star\geq 1},\vec{C}^{\star\geq 1},\dots,\vec{D}^{\star\geq 1},\vec{E}) \\ f \in \mathcal{G}(\vec{A},\vec{F}^{\geq 1}) \qquad \cdots \qquad g \in \mathcal{G}(\vec{B},\vec{G}^{\geq 1}) \\ h \in \mathcal{G}(\vec{C},()) \qquad \cdots \qquad k \in \mathcal{G}(\vec{D},()) \\ \mathfrak{a},\dots,\mathfrak{b},\mathfrak{c},\dots,\mathfrak{d} \notin \mathfrak{B} \text{ and pairwise distinct}\end{array}}{\Gamma \vdash^{\mathfrak{B}\cup\{\mathfrak{a},\dots,\mathfrak{b},\mathfrak{c},\dots,\mathfrak{d}\}} \left(\vec{f}^{\mathfrak{a}}(\vec{M}),\dots,\vec{g}^{\mathfrak{b}}(\vec{N}),\vec{R} \,\middle|\, \vec{h}^{\mathfrak{c}}(\vec{P}),\dots,\vec{k}^{\mathfrak{d}}(\vec{Q}),\vec{Z}\right) : (\vec{F}^{\star},\dots,\vec{G}^{\star},\vec{E}^{\star=0})}$$

Of course, this rule can now never apply to generators with nullary domain. Since these can always be applied at the very beginning, we incorporate them into the identity rule. Thus the identity rule is now

$$\frac{\begin{array}{c} f \in \mathcal{G}((),\vec{B}^{\geq 1}) \qquad \cdots \qquad g \in \mathcal{G}((),\vec{C}^{\geq 1}) \\ h \in \mathcal{G}((),()) \qquad \cdots \qquad k \in \mathcal{G}((),()) \\ \mathfrak{a},\dots,\mathfrak{b},\mathfrak{c},\dots,\mathfrak{d} \in \mathfrak{B} \text{ and pairwise distinct}\end{array}}{\vec{x} : \vec{A} \vdash^{\mathfrak{B}} \left(\vec{x},\vec{f}^{\mathfrak{a}},\dots,\vec{g}^{\mathfrak{b}} \,\middle|\, h^{\mathfrak{c}},\dots,k^{\mathfrak{d}}\right) : (\vec{A}^{\star},\vec{B}^{\star},\dots,\vec{C}^{\star})}$$

Finally, since we want to make the exchange rule admissible, we have to build permutations into the rules as well. As in §2.10, each rule should add exactly the part of a permutation that can't be "pushed into the premises". Because we've formulated the generator rule so that the premise and conclusion have the same context, any desired permutation in the domain can be pushed all the way up to the identity rule. Thus, for the generator rule it remains to deal with permutation in the codomain.

The freedom we have in the premises of the generator rule is to (inductively) permute the types *within* each list $\vec{A},\vec{B},\vec{C},\vec{D},\vec{E}$, and also to block-permute the lists $\vec{A},\dots,\vec{B}$ and separately the lists $\vec{C},\dots,\vec{D}$ (with a corresponding permutation of the generators $f,\dots,g$ and $h,\dots,k$). (If we permuted the main premise any more than this, it would no longer have the requisite shape to apply the rule to.) Permutations of $\vec{C},\dots,\vec{D}$ don't do us any good in terms of permuting the codomain of the conclusion, but we can push permutations of $\vec{E}$ directly into the premise, and also a block-permutation of $\vec{F},\dots,\vec{G}$ into a block-permutation of $\vec{A},\dots,\vec{B}$.

What remains that we have to build into the rule can be described precisely by a permutation of $\vec{F},\dots,\vec{G},\vec{E}$ that (1) preserves the relative order of the types in $\vec{E}$, and (2) preserves the relative order of the *first* types $F_1,\dots,G_1$ in the lists $\vec{F},\dots,\vec{G}$. That is, any permutation of $\vec{F},\dots,\vec{G},\vec{E}$ can be factored uniquely as one with these two properties followed by a block sum of a block-permutation of $\vec{F},\dots,\vec{G}$ with a permutation of $\vec{E}$. (The choice of the first types is arbitrary; we could just as well use the last types, etc.)

There is no real need to allow ourselves to permute the scalar terms, since semantically their order doesn't matter anyway. But it is convenient to allow ourselves to write the scalar terms in any order, so we incorporate permutations there too. The freedom in the premises allows us to permute the term in $\vec{Z}$

$$\frac{\begin{array}{c}\Gamma \vdash^{\mathfrak{B}} (\vec{M},\dots,\vec{N},\vec{P},\dots,\vec{Q},\vec{R} \mid \vec{Z}) : (\vec{A}^{\star\ge 1},\dots,\vec{B}^{\star\ge 1},\vec{C}^{\star\ge 1},\dots,\vec{D}^{\star\ge 1},\vec{E}) \\ f \in \mathcal{G}(\vec{A},\vec{F}^{\ge 1}) \quad \cdots \quad g \in \mathcal{G}(\vec{B},\vec{G}^{\ge 1}) \\ h \in \mathcal{G}(\vec{C},()) \quad \cdots \quad k \in \mathcal{G}(\vec{D},()) \\ \mathfrak{a},\dots,\mathfrak{b},\mathfrak{c},\dots,\mathfrak{d} \notin \mathfrak{B} \text{ and pairwise distinct} \\ \sigma : (\vec{F}^{\star},\dots,\vec{G}^{\star},\vec{E}^{\star=0}) \xrightarrow{\sim} \Delta \text{ preserving activeness} \\ \sigma \text{ preserves the relative order of types in } \vec{E} \\ \sigma \text{ preserves the relative order of } F_1,\dots,G_1 \qquad \tau \in \mathrm{Shuf}(h,\dots,k;\vec{Z})\end{array}}{\Gamma \vdash^{\mathfrak{B}\cup\{\mathfrak{a},\dots,\mathfrak{b},\mathfrak{c},\dots,\mathfrak{d}\}} \left(\sigma\left(\vec{f}^{\mathfrak{a}}(\vec{M}),\dots,\vec{g}^{\mathfrak{b}}(\vec{N}),\vec{R}\right) \,\middle|\, \tau\left(\vec{h}^{\mathfrak{c}}(\vec{P}),\dots,\vec{k}^{\mathfrak{d}}(\vec{Q}),\vec{Z}\right)\right) : \Delta}$$

$$\frac{\begin{array}{c} f \in \mathcal{G}((),\vec{B}^{\ge 1}) \quad \cdots \quad g \in \mathcal{G}((),\vec{C}^{\ge 1}) \\ h \in \mathcal{G}((),()) \quad \cdots \quad k \in \mathcal{G}((),()) \\ \mathfrak{a},\dots,\mathfrak{b},\mathfrak{c},\dots,\mathfrak{d} \in \mathfrak{B} \text{ and pairwise distinct} \\ \sigma : (\vec{A}^{\star},\vec{B}^{\star},\dots,\vec{C}^{\star}) \xrightarrow{\sim} \Delta \text{ preserving activeness} \\ \sigma \text{ preserves the relative order of } B_1,\dots,C_1\end{array}}{\vec{x} : \vec{A} \vdash^{\mathfrak{B}} \left(\sigma\left(\vec{x},\vec{f}^{\mathfrak{a}},\dots,\vec{g}^{\mathfrak{b}}\right) \,\middle|\, h^{\mathfrak{c}},\dots,k^{\mathfrak{d}}\right) : \Delta}$$

Figure 3.1: Type theory for props

arbitrarily, and also to permute the terms $h,\dots,k$ among themselves; thus what remains is precisely a shuffle. The final generator rule is therefore the first rule shown in Figure 3.1.

In the identity rule, the only useful freedom in the premises is to block-permute the $\vec{B},\dots,\vec{C}$. Thus what remains is a permutation that preserves the relative order of the first types $B_1,\dots,C_1$. Any permutation in the scalar terms can be pushed into the premises, so we have the final rule shown second in Figure 3.1. Note that this also allows us to incorporate an arbitrary permutation in the domain. This completes our definition of the **type theory for props under** $\mathcal{G}$.

Since our terms are less directly connected to derivations than usual, there is more content to the following lemma.

Theorem 3.1.7. *If there is some assignment of activeness to the types in* Δ *such that* $\Gamma \vdash^{\mathfrak{B}} (\vec{M} \mid \vec{Z}) : \Delta$ *is derivable in the type theory for props, then that assignment is unique, as is the derivation.*

Proof. We first define an auxiliary notion: the *depth* of a variable is 0, and the *depth* of an occurrence of a function symbol in a term is the least natural number strictly greater than the depths of the head symbols of all its arguments. This is a purely syntactic definition, which is well-defined by the well-foundedness of syntax. Note that a nullary function symbol always has depth 0, while a function symbol applied to a positive number of variables alone has depth 1.

Now note that in any derivable term in the type theory for props, any two function symbols marked with the same label $\mathfrak{a} \in \mathfrak{B}$ must be applied to exactly the same arguments, and therefore have the same depth. Therefore, given that $\Gamma \vdash^{\mathfrak{B}} (\vec{M} \mid \vec{Z}) : \Delta$ is derivable, we may regard the depth as a function defined on $\mathfrak{B}$ rather than on occurrences of function symbols.

We claim that in any derivable term judgment, the terms associated to active types are precisely those non-scalar ones whose head symbol has maximum depth. The proof is by induction on derivations. In the identity rule, all terms have depth 0 and all types are active. Now consider the generator rule, and suppose inductively that the claim is true for the main premise, with maximal depth n, say. Then since each of the new labels introduced by the rule is applied to at least one term from an active type, which therefore has the maximal depth n, it must have depth $n + 1$. It follows that the new maximum depth is $n + 1$, and that these new symbols are precisely those of maximal depth; but they are also precisely those associated to active types. This proves the claim.

It follows immediately that the terms uniquely determine the activeness of the types, since depth is a syntactic invariant of the terms. Moreover, we can tell from the terms which rule must have been applied last (if the maximum depth is 0, it must come from the identity rule; otherwise it must come from the generator rule) and which labels that rule must have introduced (those of maximum depth). If it is the generator rule, then the ordering of the corresponding function symbols as $f, \ldots, g$ and $h, \ldots, k$ must be the order in which $f_{(1)}, \ldots, g_{(1)}$ and $h, \ldots, k$ appear in the term list, since the permutations σ and τ preserve those orders. Then σ^{-1} is uniquely determined by the fact that it must place all the outputs of f first, and so on until all the outputs of g, then all the terms of non-maximum depth in the same order that they were given in the conclusion. Similarly, τ^{-1} is uniquely determined by the fact that it has to place $h, \ldots, k$ first and the scalar terms of non-maximum depth last, preserving internal order in each group. Finally, this determines the main premise uniquely as well. The argument for the identity rule is similar, with no τ and with σ^{-1} placing all the variables first in the order of the context. Inductively, therefore, the entire derivation is uniquely determined. □

Note that this proof is a little more complicated than most type-checking algorithms. In particular, it requires crawling through the structure of the terms twice: once to calculate depths, and then again to construct the derivation by peeling off terms of maximum depth step by step.

Because the activeness of types is uniquely determined by the terms, and hence also by the derivations, in the future we will omit the activeness labels as long as there is a specified term or derivation. We now proceed to show that our type theory has the structure of a prop, beginning with the admissibility of exchange on the right.

Lemma 3.1.8. *If we have a derivation of $\Gamma \vdash \Delta$ and a permutation ρ of Δ, then we can construct a derivation of $\Gamma \vdash \rho\Delta$. Moreover, this action is functorial.*

Proof. This essentially follows from how we built the rules. If the derivation ends with the identity rule, then we can compose ρ with the specified permutation σ

from that rule, and reorder the generators $f, \ldots, g$ in the rule according to the order that $\rho\sigma$ puts them in. If the derivation ends with the generator rule, then we similarly compose ρ with σ, reorder the generators $f, \ldots, g$, and inductively push the remaining part of the permutation (that acting on the non-active terms) into the main premise. Functoriality follows as in Lemma 2.10.3. □

For cut admissibility, it seems helpful to first prove the admissibility of a single-generator rule. Note that we formulate it with the domain of the generator at the *end* of the given codomain context.

Lemma 3.1.9. *Given a derivation of $\Gamma \vdash \Delta, \vec{A}$ and a generator $f \in \mathcal{G}(\vec{A}, \vec{B})$, we can construct a derivation of $\Gamma \vdash \Delta, \vec{B}$. Moreover, if none of the types in $\vec{A}$ are active in the given derivation, then all of the types in Δ that are active in the given derivation are still active in the result.*

Proof. If any of the types in $\vec{A}$ are active, we can simply apply the generator rule with f as the only generator. Otherwise, none of them were introduced by the final rule in the derivation of $\Gamma \vdash \Delta, \vec{A}$. If that rule was the identity rule, then $\vec{A}$ must be empty (since all types in the conclusion of the identity rule are active), so we can just add f to that application of the identity rule.

If that rule was the generator rule, then $\vec{A}$ must also appear at the end of its main premise. If none of the types in $\vec{A}$ are active therein, then we can inductively apply f to that premise; by the second clause of the inductive hypothesis, this does not alter the activeness of the other types in the premise, so we can re-apply the generator rule. Finally, if at least one of the types in $\vec{A}$ *is* active in the main premise, then we can add f to the generator rule, applying it alongside all the other generators, since it satisfies the condition that at least one of its arguments be active. (Technically, this may require us to first permute the consequent of the main premise so that $\vec{A}$ appears before all the other non-inputs to the generator rule. This is not a problem for the induction since in this case we are not actually using the inductive hypothesis at all.) In all cases, the second claim of the lemma is obvious. □

Now by combining Lemmas 3.1.8 and 3.1.9, we can postcompose with a generator $f \in \mathcal{G}(\vec{A}, \vec{B})$ whose domain types $\vec{A}$ appear anywhere in the consequent of a judgment $\Gamma \vdash \Delta$, in any order.

Theorem 3.1.10. *Cut is admissible: given derivations of $\Gamma \vdash \Delta$ and $\Delta \vdash \Phi$, we can construct a derivation of $\Gamma \vdash \Phi$.*

Proof. We induct on the derivation of $\Delta \vdash \Phi$. If it comes from the identity rule, then we just have to compose $\Gamma \vdash \Delta$ with some number of nullary-domain generators and permute its codomain; we do this one by one using Lemma 3.1.9 and then Lemma 3.1.8. Similarly, if it comes from the generator rule, we inductively cut with its main premise, then apply all of the new generators one by one using Lemma 3.1.9. □

As an example, suppose we want to cut the following terms:

$$x : A, y : B \vdash (f_{(1)}(y), k(g, f_{(3)}(y)), f_{(2)}(y) \mid h(x)) : (C, D, E) \tag{3.1.11}$$

$$u : C, v : D, w : E \vdash (m(u, \ell_{(2)}(w)), s, \ell_{(1)}(w) \mid n(v)) : (F, G, H) \tag{3.1.12}$$

Here the generators are

$$f : B \to (C, E, P) \qquad g : () \to Q \qquad h : A \to () \qquad k : (Q, P) \to D$$

$$\ell : E \to (H, R) \qquad m : (C, R) \to F \qquad n : D \to () \qquad s : () \to G$$

The depths are

$$f = 1 \qquad g = 0 \qquad h = 1 \qquad k = 2 \qquad \ell = 1 \qquad m = 2 \qquad n = 1 \qquad s = 0$$

Thus, the final rule of the second derivation must apply m only, so our inductive job is to cut

$$x : A, y : B \vdash (f_{(1)}(y), k(g, f_{(3)}(y)), f_{(2)}(y) \mid h(x)) : (C, D, E) \tag{3.1.13}$$

$$u : C, v : D, w : E \vdash (u, \ell_{(2)}(w), s, \ell_{(1)}(w) \mid n(v)) : (C, R, G, H) \tag{3.1.14}$$

Now the final rule of the second derivation must apply ℓ and n together, so our inductive job is to cut

$$x : A, y : B \vdash (f_{(1)}(y), k(g, f_{(3)}(y)), f_{(2)}(y) \mid h(x)) : (C, D, E) \tag{3.1.15}$$

$$u : C, v : D, w : E \vdash (w, v, u, s \mid) : (E, D, C, G) \tag{3.1.16}$$

The latter is obtained from the identity rule, so our task is now to apply Lemma 3.1.9 to the former and the single generator $s : () \to G$. Peeling down the derivation of the former, we obtain

$$x : A, y : B \vdash (g, f_{(3)}(y), f_{(1)}(y), f_{(2)}(y) \mid h(x)) : (Q, P, C, E)$$

and then

$$x : A, y : B \vdash (y, x, g \mid) : (B, A, Q)$$

which is also obtained from the identity rule. The identity rule can therefore also give us

$$x : A, y : B \vdash (y, x, g, s \mid) : (B, A, Q, G).$$

Re-applying f, h and then k, we obtain

$$x : A, y : B \vdash (g, f_{(3)}(y), f_{(1)}(y), f_{(2)}(y), s \mid h(x)) : (Q, P, C, E, G)$$

and then

$$x : A, y : B \vdash (f_{(1)}(y), k(g, f_{(3)}(y)), f_{(2)}(y), s \mid h(x)) : (C, D, E, G).$$

Permuting this, we obtain

$$x : A, y : B \vdash (f_{(2)}(y), f_{(1)}(y), k(g, f_{(3)}(y)), s \mid h(x)) : (E, C, D, G).$$

as the result of cutting (3.1.15) and (3.1.16).

Backing out the induction one more step, we must apply ℓ and n to this using Lemma 3.1.9. We cannot apply ℓ directly since its domain E is not active (its term $f_{(2)}(y)$ has depth 1 while the maximum depth is 2). Thus, we back up to the main premise

$$x : A, y : B \vdash (g, f_{(3)}(y), f_{(1)}(y), f_{(2)}(y), s \mid h(x)) : (Q, P, C, E, G)$$

in which E is active. Thus, we can apply ℓ in the same generator rule as k, obtaining

$$x : A, y : B \vdash (\ell_{(1)}(f_{(2)}(y)), \ell_{(2)}(f_{(2)}(y)), f_{(1)}(y), k(g, f_{(3)}(y)), s \mid h(x)) : (H, R, C, D, G). \tag{3.1.17}$$

Now the domain D of the generator n *is* active, so we can directly apply it with another generator rule, obtaining (after permutation)

$$x : A, y : B \vdash (f_{(1)}(y), \ell_{(2)}(f_{(2)}(y)), s, \ell_{(1)}(f_{(2)}(y)) \mid n(k(g, f_{(3)}(y))), h(x)) : (C, R, G, H). \tag{3.1.18}$$

as the result of cutting (3.1.13) and (3.1.14).

Finally, we must compose this with m using Lemma 3.1.9. Neither of the domain types C and R is active in (3.1.18) (in fact, *no* types are active in (3.1.18), since the last rule applied was a generator rule with only a scalar generator), so we have to inductively peel back to (3.1.17) in which R is active (though not C). Thus, we can then apply m in the same generator rule as n, obtaining

$$x : A, y : B \vdash (m(f_{(1)}(y), \ell_{(2)}(f_{(2)}(y))), s, \ell_{(1)}(f_{(2)}(y)) \mid n(k(g, f_{(3)}(y))), h(x)) : (F, G, H) \tag{3.1.19}$$

as our end result.

Note that the terms in (3.1.19) from those in (3.1.12) by substituting $f_{(1)}(y)$ for u, $k(g, f_{(3)}(y))$ for v, and $f_{(2)}(y)$ for w, and appending the scalar term $h(x)$ of (3.1.11) to the scalar terms of (3.1.12). (Our less than completely explicit proof of Lemma 3.1.9 does not really determine the order of the scalar terms in the end result; we henceforth adopt this convention that those associated to the terms being substituted into come first, followed by those associated to the terms being substituted.) Because the distance between terms and derivations is greater than usual, there is also more content to this observation than usual. We can make it formal by introducing a notion of substitution for untyped terms, and then proving that the terms produced by Theorem 3.1.10 are actually instances of this notion. We call these "untyped term" **pre-terms**; they are defined by the judgments:

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x \;\mathsf{preterm}} \qquad \frac{\Gamma \vdash M \;\mathsf{preterm} \quad \dots \quad \Gamma \vdash N \;\mathsf{preterm} \qquad f \in \mathcal{G}(A, \dots, B; \vec{C}) \quad \mathfrak{a} \in \mathfrak{A} \quad k \in \mathbb{N}}{\Gamma \vdash f^{\mathfrak{a}}_{(k)}(M, \dots, N) \;\mathsf{preterm}}$$

We define substitution into preterms in a fairly obvious way; for convenience we define a notion of "simultaneous substitution" of a list of preterms $\vec{M}$ for a list of variables $\vec{x}$

$$\begin{aligned} x_k[\vec{M}/\vec{x}] &= M_k \\ y[\vec{M}/\vec{x}] &= y && (y \notin \vec{x}) \\ f^{\mathfrak{a}}_{(k)}(N,\ldots,P)[\vec{M}/\vec{x}] &= f^{\mathfrak{a}}_{(k)}(N[\vec{M}/\vec{x}],\ldots,P[\vec{M}/\vec{x}]) \end{aligned}$$

Then we can prove:

Lemma 3.1.20. *If* $\Gamma \vdash (\vec{M} \mid \vec{Z}) : \vec{A}$ *and* $\vec{x} : \vec{A} \vdash (\vec{N} \mid \vec{W}) : \vec{B}$*, then the composite constructed by Theorem 3.1.10 is* $\Gamma \vdash (\vec{N}[\vec{M}/\vec{x}] \mid \vec{W}[\vec{M}/\vec{x}], \vec{Z}) : \vec{B}$*.*

Proof. Inducting on the derivation of the second judgment, we trace through the proofs and find that the resulting terms are eventually obtained by applying a generator or identity rule, producing terms that match the inductive definition of substitution into pre-terms. □

This actually turns out to be quite useful. For instance, the following is quite easy to prove:

Lemma 3.1.21. *Substitution into pre-terms is associative:* $P[\vec{N}/\vec{y}][\vec{M}/\vec{x}] = P[\vec{N}[\vec{M}/\vec{x}]/\vec{y}]$*.* □

Therefore, we can immediately conclude:

Theorem 3.1.22. *Cut is associative.*

Proof. Since derivations are determined uniquely by their terms by Theorem 3.1.7, this follows from Lemmas 3.1.20 and 3.1.21. □

This would be rather messier to prove by a direct induction on the construction of Theorem 3.1.10. (The construction of Theorem 3.1.10 is still necessary, however, to show that the substituted pre-terms in Lemma 3.1.20 are well-typed.) Unitality follows similarly, so we have a category whose objects are contexts and whose morphisms are terms/derivations of $\Gamma \vdash \Delta$.

Of course, we have a strictly associative and unital operation of concatenation on contexts, so we are approaching the construction of a prop with types as its objects. But in order for the category of contexts to be monoidal, we need to add an equality rule imposing invariance under permutation of the scalar terms:

$$\frac{\Gamma \vdash (\vec{M} \mid Z_1,\ldots,Z_n) : \Delta \qquad \rho \in S_n}{\Gamma \vdash (\vec{M} \mid Z_1,\ldots,Z_n) \equiv (\vec{M} \mid Z_{\rho 1},\ldots,Z_{\rho n}) : \Delta}$$

It may seem silly to have incorporated permutations in the scalar terms earlier, but to now quotient out by that freedom. However, this equality rule would be necessary even if we hadn't incorporated any permutations. The paradigmatic case is when we have two nullary scalar generators $f : () \to ()$ and $g : () \to ()$, leading unavoidably to two distinct valid terms

$$() \vdash (\mid f, g) : () \qquad\qquad () \vdash (\mid g, f) : ()$$

that must be equal in a monoidal category.

Theorem 3.1.23. *The contexts and derivable term judgments in the type theory for props under $\mathcal{G}$, modulo the above equality rule, form a symmetric strict monoidal category.*

Proof. The monoidal structure on contexts is concatenation, with the empty context as unit. To tensor morphisms, it is easiest to first tensor with identities: given $\Gamma \vdash (\vec{M} \mid \vec{Z}) : \Delta$, we construct $\Gamma, \vec{x} : \vec{A} \vdash (\vec{M}, \vec{x} \mid \vec{Z}) : \Delta, \vec{A}$ by inducting until we get down to the identity rule and then just adding variables to the context. Now we obtain the tensor product of $\Gamma \vdash \Delta$ and $\Phi \vdash \Psi$ by first tensoring with identities to get $\Gamma, \Phi \vdash \Delta, \Phi$ and $\Delta, \Phi \vdash \Delta, \Psi$ and then composing to get $\Gamma, \Phi \vdash \Delta, \Psi$. Since composition is by substitution, which is unital, it follows (using again the fact that derivations are uniquely determined by their terms) that this is equal, up to permutation of the scalar terms, to what we would get by doing it the other way (using $\Gamma, \Phi \vdash \Gamma, \Psi$ and $\Gamma, \Psi \vdash \Delta, \Psi$). In particular, this implies functoriality of the tensor product; associativity and unitality follow similarly. Finally, the symmetry isomorphism is $\vec{x} : \vec{A}, \vec{y} : \vec{B} \vdash (\vec{y}, \vec{x} \mid) : \vec{B}, \vec{A}$; it is easy to verify the axioms. □

Thus we have a prop, which we denote $\mathfrak{F}_{\mathbf{Prop}}\mathcal{G}$.

Theorem 3.1.24. *$\mathfrak{F}_{\mathbf{Prop}}\mathcal{G}$ is the free prop generated by $\mathcal{G}$.*

Proof. Let $\mathcal{M}$ be a prop and $\omega : \mathcal{G} \to \mathcal{M}$ a morphism of polygraphs. As always, we extend it to $\mathfrak{F}_{\mathbf{Prop}}\mathcal{G}$ by induction on derivations. By the coherence theorem for symmetric monoidal categories, there is a unique choice at each step if we are to have a (symmetric strict monoidal) functor, and likewise the equality rule corresponds to an actual equality that must hold in $\mathcal{M}$. Afterwards we prove that this actually is a symmetric strict monoidal functor, using the definition of composition and the tensor product in $\mathfrak{F}_{\mathbf{Prop}}\mathcal{G}$. □

We can also construct "presented" props, by adding arbitrary generators of $\equiv$. The uniqueness of antipodes in a bimonoid presented in §0.1 is an example of this: $\mathcal{G}$ has one object M and four morphisms

$$\begin{aligned} m &: (M, M) \to M && \text{(written infix as } m(x, y) = x \cdot y\text{)} \\ e &: () \to M && \\ \triangle &: M \to (M, M) && \text{(written a little abusively as } \triangle_{(i)}(x) = x_{(i)}\text{)} \\ \varepsilon &: M \to () && \text{(written } \varepsilon(x) = \not{x}\text{)} \end{aligned}$$

while the generators of $\equiv$ are the bimonoid axioms.

The slightly abusive notation $\triangle_{(i)}(x) = x_{(i)}$ is part of the traditional "Sweedler notation" for comodules. Since there is no other meaning of $x_{(i)}$ when x is a variable (or a term that already has a subscript, including a one-output function $f(\vec{M})$, which we recall technically means $f_{(1)}(\vec{M})$), it is unambiguous as long as no type has more than one relevant comultiplication. It may be regarded as a sort of "dual" to the usual shorthand notation "xy" (rather than $x \cdot y$ or $m(x, y)$) for the multiplication of elements of a monoid.

Traditional Sweedler notation also goes one step further. With the convention $\triangle_{(i)}(x) = x_{(i)}$, applying the comultiplication twice would yield three terms $x_{(1)(1)}$ and $x_{(1)(2)}$ and $x_{(2)}$ — or $x_{(1)}$ and $x_{(2)(1)}$ and $x_{(2)(2)}$, depending on how we apply the comultiplication. However, by the coassociativity of $\triangle$, these two triples of terms are actually equal; thus Sweedler writes them as $x_{(1)}$ and $x_{(2)}$ and $x_{(3)}$. In general, the principle is that if subscripts are applied to a variable or a term that is already subscripted (where "a term" is technically identified by the label $\mathfrak{a}$ of its head function symbol), with the maximum subscript appearing on it being n, then the subscript ${}_{(k)}$ is to be interpreted as $\overbrace{{}_{(2)(2)\cdots(2)}}^{k-1}{}_{(1)}$ if $k < n$, and $\overbrace{{}_{(2)(2)\cdots(2)}}^{n-1}$ if $k = n$.

Here is another example using presentations for props. If A is an object of a prop, a **dual** of A is an object A^* with morphisms $\eta : () \to (A, A^*)$ and $\varepsilon : (A^*, A) \to ()$ such that $(\mathsf{id}_A \bullet \varepsilon) \circ (\eta \bullet \mathsf{id}_A) = \mathsf{id}_A$ and $(\varepsilon \bullet \mathsf{id}_{A^*}) \circ (\mathsf{id}_{A^*} \bullet \eta) = \mathsf{id}_{A^*}$. (In a symmetric monoidal category this reduces to the usual notion of dual.) In the type theory for props, these axioms say

$$x : A \vdash (\eta_{(1)} \mid \varepsilon(\eta_{(2)}, x)) \equiv x : A \qquad y : A^* \vdash (\eta_{(2)} \mid \varepsilon(y, \eta_{(1)})) \equiv y : A^*.$$

Recall that $\equiv$ is a congruence for substitution on both sides; thus the first dual axiom means that *any term* $M : A$ (appearing even as a sub-term of some other term) can be replaced by $\eta^{\mathfrak{a}}_{(1)}$ if we simultaneously add $\varepsilon^{\mathfrak{b}}(\eta^{\mathfrak{a}}_{(2)}, M)$ to the scalars (here $\mathfrak{a}$ and $\mathfrak{b}$ are fresh labels). And conversely, $\varepsilon^{\mathfrak{b}}(\eta^{\mathfrak{a}}_{(2)}, M)$ appears in the scalars, for any term $M : A$, then it can be removed by replacing $\eta^{\mathfrak{a}}_{(1)}$ (wherever it appears) with M. The other dual axiom is similar.

Now if A has a dual A^*, and $f : A \to A$, the **trace** of f is the composite

$$() \xrightarrow{\eta} (A, A^*) \xrightarrow{(f, \mathsf{id})} (A, A^*) \xrightarrow{\cong} (A^*, A) \xrightarrow{\varepsilon} ()$$

In type theory the trace is $(\mid \varepsilon(\eta_{(2)}, f(\eta_{(1)})))$. Now a classical fact about the trace is that it is *cyclic*: if A and A' both have duals A^* and $(A')^*$, and $f : A \to A'$ and $g : A' \to A$, then $\mathrm{tr}(gf) = \mathrm{tr}(fg)$. To prove this using traditional commutative-diagram reasoning is quite involved. It does have a pretty and intuitive proof using string diagrams. However, its proof in the type theory for props is one line:

$$\mathrm{tr}(gf) = (\mid \varepsilon(\eta_{(2)}, g(f(\eta_{(1)})))) = (\mid \varepsilon(\eta_{(2)}, g(\eta'_{(1)})), \varepsilon'(\eta'_{(2)}, f(\eta_{(1)}))) = (\mid \varepsilon(\eta'_{(2)}, f(g(\eta'_{(1)})))) = \mathrm{tr}(fg).$$

Here η, ε exhibit the dual of A, while η', ε' exhibit the dual of A'. The first and last equality are by definition; the second applies the first duality equation for

A' with $x = f(\eta_{(1)})$; and the third applies the first duality equation for A with $x = g(\eta'_{(1)})$.

3.2 Cyclic multicategories and cosubunary polycategories

Motivate with two examples: multivariable adjunctions (symmetric), and classical logic (cartesian).

But we want the cyclic action to have a universal property to get a good type theory. We can do this by allowing conullary arrows too; then a map $\varepsilon : (A, A^\bullet) \to ()$ is the counit of a duality if composing with it induces a bijection between arrows $(\Gamma, A) \to ()$ and $\Gamma \to A^\bullet$, and the duality is symmetric if $\varepsilon\sigma$ is also such a counit. Conullary arrows are mutual left adjunctions and proofs of contradictions; the universal property is precisely proof by contradiction.

Maybe start with the cosubunary case at the beginning of the chapter, as the simplest one? Note that if the conullary arrows are representable, and we have function-types, then all duals exist, but are not generally symmetric.

The elim for $A^\bullet$ is ε, which "applies" a term of $A^\bullet$ to a term of A to produce a conullary term. Its intro "abstracts" a conullary term over a variable of A to a term of $A^\bullet$, and the universal property is β/η for these. ("Parigot-style μ-abstraction"?) To ensure symmetry, hence $(A^\bullet)^\bullet \cong A$, we also have to be able to abstract a conullary term over a variable of $A^\bullet$ to get a term of A. This is closely related to Koh-Ong; it's more or less just making their substitutions implicit rather than explicit.

For instance, if $f : (A, B) \to C$ is the left adjoint of a multivariable adjunction, with term $x : A, y : B \vdash f(x, y) : C$, then its two right adjoints are $x : A, z : C^\bullet \vdash \mu y.z(f(x, y)) : B^\bullet$ and $y : B, z : C^\bullet \vdash \mu x.z(f(x, y)) : A^\bullet$. (We have to dualize their domains and codomains to regard them as *left* adjoints.)

Adding connectives, we get de Morgan duality $\wedge/\vee$ for classical logic and $\otimes/\parr$ in the noncartesian case, leading to linearly distributive and $*$-autonomous categories, and linear logic.

But this isn't ideal from a type-theoretic POV, since abstracting over $A^\bullet$ produces a term in an *arbitrary* type, in contrast to how intro rules are supposed to live in their corresponding type former. This corresponds to the weirdness of classical logic by which "proof by contradiction" is a special rule that applies to any goal.

A cleaner approach from this POV, though yielding a messier term syntax, is to generalize from the cosubunary case to arbitrary polycategories. In a symmetric polycategory, a bijection between arrows $(\Gamma, A) \to \Delta$ and $\Gamma \to (\Delta, A^\bullet)$ is automatically symmetric, since we can Yoneda to get a unit and counit with triangle identities. Get a term syntax for polycategories by mapping them into a prop and using a "proof net" condition to characterize the image. (Actually, use this as a lead-in to prop type theory.) Mention the mix rules.

Mention negation normal form at least briefly, since it's in the literature.

3.3 Classical logic

3.4 Polycategories and linear logic

Chapter 4

First-order logic

4.1 Predicate logic

In §2.7 we saw that the posetal reduction of a simple type theory can be regarded as a deductive system for logic (intuitionistic, linear, relevant, classical, etc. depending on the type theory). However, these logics are only *propositional*, lacking variables and the ability to quantify over them with statements such as "for all x" or "there exists an x such that". Similarly, in §2.9.1 we saw that simple type theory is adequate to express finite-product theories such as groups and rings, but not more complicated theories such as categories or fields. The solution to both of these problems is the same: we combine *two* type theories, one representing the objects (like a finite-product theory) and one representing the logic in which we speak about those objects.

The types in the second type theory, which we will henceforth call **propositions** instead of types to avoid confusion, will be *dependent* on the types in the first type theory (which we sometimes call the *base type theory*). This means that terms belonging to types can appear in propositions. More formally, it means that unlike the judgment $\vdash A\ \mathsf{type}$ for types (in the base type theory), the judgment for propositions *has a context of types*, so we write it $\Gamma \vdash \varphi\ \mathsf{prop}$. We will have rules such as

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : A}{\Gamma \vdash (M =_A N)\ \mathsf{prop}}$$

allowing the logic (the type theory of propositions) to "talk about" equality of terms (morphisms between types). Finally, since propositions depend on a context of types, their morphism judgment (which we also call **entailment**) must also depend on such a context. Thus it has *two* contexts, one of types and one of propositions, which we separate with a vertical bar: $\Gamma \mid \Theta \vdash \varphi$.

In this section, we will describe and study type theories of this sort, with one simple type theory dependent on another simple type theory. Unlike the type theories considered in chapter 2, which were directly motivated by a corresponding categorical structure, in the present case it seems more natural to describe the type theory first and then define an appropriate categorical structure in order to match it. (This is not to say that there are not lots of naturally occurring examples of this categorical structure; there are! It's just that without the type theory in mind, we might not be led to define and study that exact class of categorical structures.) Thus, we postpone consideration of their categorical semantics to §§4.2 and 4.3.

We will also make several simplifying assumptions in this section. Firstly, the base type theory will always be a bare theory of cartesian multicategories under some multigraph, with no type operations and no axioms. The lack of axioms is not much of a problem, since once we have equality propositions we can use those instead. The lack of type operations is a temporary simplification, but identifies our current subject as *first-order* logic; in chapter 5 on "higher-order logic" we will reintroduce type operations. The *cartesianness* of the base type theory is also a simplifying assumption, but one that we will not (in this book) ever generalize away from. People have attempted to define first-order logics over non-cartesian base theories, but in general the results are more complicated and less intuitive, and there are fewer interesting categorical examples.

Secondly, in this section the logic will be posetal, so that we care only about the existence of derivations rather than their values, and hence we will not introduce terms belonging to propositions. We will generalize away from this assumption in §4.6.

4.1.1 Structural rules and simple rules

With all that out of the way, we move on to actually describing the rules. As mentioned above, the base type theory is that for cartesian multicategories under a multigraph $\mathcal{G}$:

$$\frac{\vdash A \ \mathsf{type} \qquad (x : A) \in \Gamma}{\Gamma \vdash x : A}\ \mathsf{id}$$

$$\frac{f \in \mathcal{G}(A_1, \ldots, A_n; B) \qquad \Gamma \vdash M_1 : A_1 \qquad \cdots \qquad \Gamma \vdash M_n : A_n}{\Gamma \vdash f(M_1, \ldots, M_n) : B}\ f$$

As usual, cut/substitution is admissible for this theory. For the propositions, we have two kinds of judgment:

$$\Gamma \vdash \varphi \ \mathsf{prop} \qquad\qquad \Gamma \mid \Theta \vdash \varphi$$

where Θ is a context (i.e. a list) of propositions. Here the proposition φ should be regarded as a sort of "term" for the proposition judgment, that can be shown to uniquely determine a derivation of $\Gamma \vdash \varphi \ \mathsf{prop}$.

Before discussing the rules for these judgments, however, we have to decide what to do about the structural rules such as cut. As with propositional logic, we can formulate first-order logic in either a natural deduction or a sequent

$$\frac{\Gamma \mid \Theta, \varphi, \psi, \Delta \vdash \chi}{\Gamma \mid \Theta, \psi, \varphi, \Delta \vdash \chi}\ \textsc{exchange} \qquad \frac{\Gamma \mid \Theta, \Delta \vdash \chi}{\Gamma \mid \Theta, \varphi, \Delta \vdash \chi}\ \textsc{weakening (maybe)}$$

$$\frac{\Gamma \mid \Theta, \varphi, \varphi, \Delta \vdash \chi}{\Gamma \mid \Theta, \varphi, \Delta \vdash \chi}\ \textsc{contraction (maybe)} \qquad \frac{}{\Gamma \mid \varphi \vdash \varphi}$$

$$\frac{\Gamma \mid \Theta \vdash \varphi \qquad \Gamma \mid \Psi, \varphi \vdash \psi}{\Gamma \mid \Psi, \Theta \vdash \psi} \qquad \frac{\Gamma \vdash M : A \qquad \Gamma, x : A \vdash \varphi\ \mathsf{prop}}{\Gamma \vdash \varphi[M/x]\ \mathsf{prop}}$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma, x : A \mid \Theta \vdash \varphi}{\Gamma \mid \Theta[M/x] \vdash \varphi[M/x]}$$

Figure 4.1: Structural rules for first-order logic

calculus style to make cut admissible. However, I feel that both choices require formulating at least one of the rules for quantifiers and equality in a less-than-maximally-intuitive way. Of course, intuitions differ from person to person. But it is also a more objective fact that the most *categorically natural* versions of the rules, as we will see in §4.2, also do not exactly match either the sequent calculus or the natural deduction versions. There are also new structural rules, namely substitution of terms into propositions and entailments, that we would eventually like to be admissible as well.

For these reasons we take the following approach. In this section we state all the structural rules, including both those that will stay primitive and those that will eventually be admissible. Then in §§4.1.2–4.1.4 we discuss the rules for the quantifiers and equality, mentioning all the ways that each rule can be formulated and showing that they are equivalent in the presence of all the structural rules. Finally, in §4.1.6 we show that the natural deduction rules do make an appropriate selection of the structural rules admissible.

The complete list of structural rules is shown in Figure 4.1. As in §2.7, we always have exchange for propositions, but we allow ourselves the freedom to take or omit weakening and contraction, corresponding to a choice of a faithful cartesian club $\mathfrak{S}$ (as in §2.6). Depending on which we choose, we speak of **intuitionistic first-order logic** (all the structural rules), **intuitionistic first-order linear logic** (exchange only), etc.

Then there are the identity and cut rule for propositions; the latter is just the cut rule from §2.7.1 with an extra type context Γ. There are also two new structural rules arising from the dependency of propositions on types: substitution of terms into propositions and into entailments.

Of all these structural rules, there is one that it is most important (for the purpose of categorical semantics) to make admissible: substitution of terms into propositions. This is for the same reason that we want substitution into terms to be admissible. Namely, we certainly want to be *able* to make such substitutions, but if we asserted them as primitive then (to maintain the unique correspondence between names for propositions φ and the derivations of $\Gamma \vdash \varphi \ \mathsf{prop}$) we would have to introduce "$\varphi[M/x]$" as basic syntax, rather than an operation on syntax.

For instance, we want to be able to substitute M for x and N for y into $x = y$, and we want to be able to actually *do* that substitution on the syntax to get $M = N$, rather than having to write $(x = y)[M/x, N/y]$ everywhere. Another possibility would be to break the "propositions are derivations" correspondence and allow one proposition to have multiple derivations, but that has the same problems as breaking the "terms are derivations" correspondence in simple type theory; we do care about *which* proposition we are talking about.

Fortunately, it is just as easy to ensure that substitution into propositions is admissible as it is to ensure that cut is admissible in a natural deduction. We just make sure to "build enough substitutions" into the rules for the proposition judgment, so that their conclusions always have a fully general context. Thus, we will always do this in our rules for the proposition judgment.

The other structural rules are all for entailment, and since at the moment we are interested in semantics where entailment corresponds to inequality in a poset, we only care about *whether or not* an entailment is derivable rather than what all its derivations are. Thus, it makes little difference (for the purpose of categorical semantics) whether these rules are primitive or admissible. (However, there are still other technical advantages to admissibility.)

Now we move on to the logical rules for the proposition and entailment judgments. To start with, there will be the usual rules for propositional logic from §2.7. We import these rules into our present theory by assigning all of them an arbitrary context of types in the base theory that remains unchanged between the premises and the conclusion. For instance, the rules for $\vee$ are

$$\frac{\Gamma \mid \Theta \vdash A}{\Gamma \mid \Theta \vdash A \vee B}\ \vee I1 \qquad\qquad \frac{\Gamma \mid \Theta \vdash B}{\Gamma \mid \Theta \vdash A \vee B}\ \vee I2$$

$$\frac{\Gamma \mid \Psi \vdash A \vee B \qquad \Gamma \mid \Theta, A \vdash C \qquad \Gamma \mid \Theta, B \vdash C}{\Gamma \mid \Theta, \Psi \vdash C}\ \vee E$$

and likewise we have rules for $\bot, \wedge, \top, \otimes, \mathbf{1}$, and $\multimap$. Of course, in the cartesian case we can dispense with $\otimes$ and $\mathbf{1}$ (since they coincide with $\wedge$ and $\top$), and write $\multimap$ instead as $\Rightarrow$ or $\to$. The modularity of type theory means we can also mix and match, choosing the rules corresponding to some of these connectives but not others; in §4.3 we will see that some groups of connectives are particularly natural from a categorical perspective.

The interesting new things happen with the *new* operations on propositions that *do* change the type context. We will consider three such operations, which are particularly natural both categorically and logically. The first two are the *quantifiers* "for all" (the "universal quantifier") and "there exists" (the "existential quantifier"). The rules introducing these two propositions both look the same:

$$\frac{\Gamma, x : A \vdash \varphi \ \mathsf{prop}}{\Gamma \vdash (\forall x{:}A.\, \varphi) \ \mathsf{prop}} \qquad \frac{\Gamma, x : A \vdash \varphi \ \mathsf{prop}}{\Gamma \vdash (\exists x{:}A.\, \varphi) \ \mathsf{prop}}$$

(Note that in both cases the variable x is *bound* in the resulting proposition, just as it is in $\lambda x.M$. If there is no danger of confusion, we may abbreviate these to $\forall x.\varphi$ and $\exists x.\varphi$, but in general the type annotation is necessary to make type-checking possible.) But the rules governing entailments involving them, of course, are different.

Recall that in natural deduction each type operation has either *introduction* and *elimination* rules, while in sequent calculus these are reformulated as *right* and *left* rules. In the past we have motivated these rules by appeal to universal properties in a categorical structure, with one group of rules giving the basic data and the other giving their universal factorization property. The rules for $\exists$ and $\forall$ do correspond to universal properties, but since we have postponed the semantics of first-order logic to §4.2 we will attempt to instead motivate their rules from an intuitive understanding of logic.

4.1.2 The universal quantifier

Informally, how do we prove that $\forall x{:}A.\, \varphi$? Arguably the most basic way to do it is to assume given an arbitrary $x : A$ and prove that φ is true (here φ is a statement involving x, hence involving our arbitrary assumed $x : A$). This suggests the following introduction (or right) rule:

$$\frac{\Gamma, x : A \mid \Theta \vdash \varphi}{\Gamma \mid \Theta \vdash \forall x{:}A.\, \varphi} \ \forall I$$

Note that since Θ appears in the conclusion, where x is no longer in the type context, Θ cannot depend on x, even though syntactically the premise would allow that.

Similarly, what good does it do to know that $\forall x{:}A.\, \varphi$? The most basic thing it tells us is that if we have any particular element M of A, then φ is true about M, i.e. with M replacing x. The simplest way to formulate this is

$$\frac{\Gamma \vdash M : A}{\Gamma \mid (\forall x{:}A.\, \varphi) \vdash \varphi[M/x]} \ \forall S$$

But there are many other ways to say the same thing, including a sequent-calculus-style left rule, a natural-deduction-style elimination rule, and the opposite of the introduction rule:

$$\frac{\Gamma \vdash M : A \qquad \Gamma \mid \Theta \vdash \forall x{:}A.\, \varphi}{\Gamma \mid \Theta \vdash \varphi[M/x]} \ \forall E \qquad \frac{\Gamma \vdash M : A \qquad \Gamma \mid \Theta, \varphi[M/x] \vdash \psi}{\Gamma \mid \Theta, (\forall x{:}A.\, \varphi) \vdash \psi} \ \forall L$$

$$\frac{\Gamma \mid \Theta \vdash \forall x{:}A.\, \varphi}{\Gamma, x : A \mid \Theta \vdash \varphi} \ \forall I^{-1}$$

All of these rules are inter-derivable in the presence of cut and substitution. For instance, we can derive $\forall E$ from $\forall I^{-1}$ using substitution:

$$\dfrac{\Gamma \vdash M : A \qquad \dfrac{\Gamma \mid \Theta \vdash \forall x{:}A.\,\varphi}{\Gamma, x : A \mid \Theta \vdash \varphi}\,\forall I^{-1}}{\Gamma \mid \Theta \vdash \varphi[M/x]}\,\textsc{sub}$$

We can derive $\forall S$ as a special case of $\forall E$ using the identity rule:

$$\dfrac{\Gamma \vdash M : A \qquad \overline{\Gamma \mid (\forall x{:}A.\,\varphi) \vdash \forall x{:}A.\,\varphi}}{\Gamma \mid (\forall x{:}A.\,\varphi) \vdash \varphi[M/x]}\,\forall E$$

We can derive $\forall L$ from $\forall S$ using cut:

$$\dfrac{\dfrac{\Gamma \vdash M : A}{\Gamma \mid (\forall x{:}A.\,\varphi) \vdash \varphi[M/x]}\,\forall S \qquad \Gamma \mid \Theta, \varphi[M/x] \vdash \psi}{\Gamma \mid \Theta, (\forall x{:}A.\,\varphi) \vdash \psi}\,\textsc{cut}$$

and finally $\forall I^{-1}$ from $\forall L$ using cut and weakening:

$$\dfrac{\dfrac{\Gamma \mid \Theta \vdash \forall x{:}A.\,\varphi}{\Gamma, x : A \mid \Theta \vdash \forall x{:}A.\,\varphi}\,\textsc{weak} \qquad \dfrac{\overline{\Gamma, x : A \vdash x : A} \qquad \overline{\Gamma, x : A \mid \varphi \vdash \varphi}}{\Gamma, x : A \mid (\forall x{:}A.\,\varphi) \vdash \varphi}\,\forall L}{\Gamma, x : A \mid \Theta \vdash \varphi}\,\textsc{cut}$$

In practice, therefore, we are free to use whichever rule we find most intuitive or convenient. To make substitition and cut admissible in §4.1.6 we will use $\forall E$, while for categorical semantics in §4.2 we will use $\forall I^{-1}$.

Remark 4.1.1. Note that many of these rules involve substitution into propositions. Thus, formally speaking we have to state all the rules for the proposition judgment $\Gamma \vdash \varphi\ \mathsf{prop}$ first, *then* prove that substitution into propositions is admissible (thereby defining the notation $\varphi[M/x]$), and only after that can we state all the rules for the entailment judgment $\Gamma \mid \Theta \vdash \varphi$. A similar situation obtained for the equality judgment $\equiv$ for simple and unary type theories, which often involved substitution into terms (e.g. $(\lambda x.M)N \equiv M[N/x]$), so that we had to prove the admissibility of the latter before stating the rules for $\equiv$ (and likewise, when proving the initiality theorems, we had to show that our functor-in-progress took substitution to composition before defining it on equalities). However, in practice we actually state all the rules at once, with the implicit understanding that afterwards we will define substitution so that the rules involving it make sense.

We do have to be careful, when taking such a shortcut, to notice whether we are introducing any "cyclic dependencies". For instance, if there are any rules for the term or proposition judgments whose premises involve the entailment judgment, it is no longer possible to complete the definition of the former, *then* define substitution for them, and *then* give the definition of the latter: we would have to give the definition all at once, including (somehow) defining substitution at the same time. It is possible to do this, but it is much more difficult and leads us into the realm of dependent type theory; see chapter 6.

In this chapter and chapter 5 none of our rules will introduce such cyclic dependencies. We mention the possibility only as a warning to the reader, because it is easy (especially when adding rules to a type theory one by one) to fail to notice a cyclic dependency when it appears.

4.1.3 The existential quantifier

The most basic way to prove $\exists x{:}A.\,\varphi$ is to exhibit a particular element M of A and prove that it has the property φ (that is, φ with M replacing x is true). This is of course a "constructive" proof. In classical mathematics one can also give "nonconstructive" existence proofs, but these arise by use of the law of excluded middle or its equivalent law of double negation. The *basic* way to prove existence, which uses no other logical laws than the meaning of "existence", is to supply a witness. This leads to the following introduction (or right) rule for $\exists$:

$$\frac{\Gamma \vdash M : A \qquad \Gamma \mid \Theta \vdash \varphi[M/x]}{\Gamma \mid \Theta \vdash \exists x{:}A.\,\varphi}\ \exists I$$

On the other hand, what good does it do to know that $\exists x{:}A.\,\varphi$? It means we are free to assume that we have some element of A satisfying φ (but about which we assume nothing else). This is most simply expressed by a left rule:

$$\frac{\Gamma, x : A \mid \Theta, \varphi \vdash \psi}{\Gamma \mid \Theta, (\exists x{:}A.\,\varphi) \vdash \psi}\ \exists L$$

This is perhaps the least intuitive of the quantifier rules: it says that if we can prove some other statement ψ under the assumption of some arbitrary $x : A$ that satisfies φ, then we can also conclude ψ under the assumption of $\exists x{:}A.\,\varphi$. (Note the similarity in structure between $\exists L$ and $\otimes L$; this suggests the eventual universal property we will find corresponding to $\exists$.)

By building in a cut, we can re-express $\exists L$ as an elimination rule instead:

$$\frac{\Gamma \mid \Psi \vdash \exists x{:}A.\,\varphi \qquad \Gamma, x : A \mid \Theta, \varphi \vdash \psi}{\Gamma \mid \Theta, \Psi \vdash \psi}\ \exists E$$

Of course $\exists E$ follows from $\exists L$ plus cut, while we can obtain $\exists L$ from $\exists E$ by taking Ψ to be $(\exists x{:}A.\,\varphi)$.

Technically, we should actually add some additional premises to $\exists E$ and $\exists L$ to ensure that ψ and Θ are defined in context Γ rather than $\Gamma, x : A$, since otherwise the premises would permit the latter. Otherwise we would not want to let ourselves write $\Gamma \mid \Theta \vdash \psi$ (with x not appearing in Γ, as implied by our conventions and the fact that in a premise we wrote $\Gamma, x : A$). Thus we ought to write them as

$$\frac{\Gamma \vdash \psi\ \mathsf{prop} \qquad \Gamma \vdash \Theta\ \mathsf{ctx} \qquad \Gamma, x : A \mid \Theta, \varphi \vdash \psi}{\Gamma \mid \Theta, (\exists x{:}A.\, \varphi) \vdash \psi}\ \exists L$$

$$\frac{\Gamma \vdash \psi\ \mathsf{prop} \qquad \Gamma \vdash \Theta\ \mathsf{ctx} \qquad \Gamma \mid \Psi \vdash \exists x{:}A.\, \varphi \qquad \Gamma, x : A \mid \Theta, \varphi \vdash \psi}{\Gamma \mid \Theta, \Psi \vdash \psi}\ \exists E$$

where $\Gamma \vdash \Theta\ \mathsf{ctx}$ is an abbreviation for

$$\Gamma \vdash B_1\ \mathsf{prop} \quad \cdots \quad \Gamma \vdash B_n\ \mathsf{prop}$$

if $\Theta = (B_1, \ldots, B_n)$. However, we often neglect to write such conditions explicitly.

Finally, now that we have $\exists E/\exists L$, we note that $\exists I$ can also be reformulated in a couple of other ways:

$$\frac{\Gamma \mid \Theta, (\exists x{:}A.\, \varphi) \vdash \psi}{\Gamma, x : A \mid \Theta, \varphi \vdash \psi}\ \exists L^{-1} \qquad\qquad \frac{}{\Gamma, x : A \mid \varphi \vdash \exists x{:}A.\, \varphi}\ \exists S$$

These are both inter-derivable with $\exists I$ in the presence of cut and substitution. For instance, we can deduce $\exists S$ as a special case of $\exists I$:

$$\frac{\dfrac{}{\Gamma, x : A \vdash x : A} \qquad \dfrac{}{\Gamma, x : A \mid \varphi \vdash \varphi}}{\Gamma, x : A \mid \varphi \vdash \exists x{:}A.\, \varphi}\ \exists I$$

We can get $\exists L^{-1}$ from $\exists S$ and cut:

$$\frac{\dfrac{}{\Gamma, x : A \mid \varphi \vdash \exists x{:}A.\, \varphi}\ \exists S \qquad \dfrac{\Gamma \mid \Theta, (\exists x{:}A.\, \varphi) \vdash \psi}{\Gamma, x : A \mid \Theta, (\exists x{:}A.\, \varphi) \vdash \psi}\ \textsc{weak}}{\Gamma, x : A \mid \Theta, \varphi \vdash \psi}\ \textsc{cut}$$

And we can get $\exists I$ by substituting and cutting in $\exists L^{-1}$:

$$\frac{\Gamma \mid \Theta \vdash \varphi[M/x] \qquad \dfrac{\Gamma \vdash M : A \qquad \dfrac{\Gamma \mid (\exists x{:}A.\, \varphi) \vdash \exists x{:}A.\, \varphi}{\Gamma, x : A \mid \varphi \vdash \exists x{:}A.\, \varphi}\ \exists L^{-1}}{\Gamma \mid \varphi[M/x] \vdash \exists x{:}A.\, \varphi}\ \textsc{sub}}{\Gamma \mid \Theta \vdash \exists x{:}A.\, \varphi}\ \textsc{cut}$$

Thus, we are free to use whichever of these rules is most convenient. To make substitition and cut admissible in §4.1.6 we will use $\exists E$ and $\exists I$, while for categorical semantics in §4.2 we will use $\exists L$ and $\exists L^{-1}$.

4.1.4 Equality

The third and last new operation on propositions is perhaps the subtlest of all: the *equality proposition*. Its formation rule is unsurprising: it says that for any two terms of the same type, we can consider the proposition that they are equal.

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : A}{\Gamma \vdash (M =_A N)\ \mathsf{prop}}$$

(The subscript annotation A in $M =_A N$ is needed for type-checking; but as usual, we will often omit it.) But how are we to describe its behavior? The most classical approach to equality is to assert that it is reflexive, symmetric, transitive, and "substitutive" (i.e. if $\varphi[M/x]$ and $M = N$, then also $\varphi[N/x]$). This is very much like how we described the equality *judgment* $M \equiv N$ in chapters 1 and 2. It works here too, but it doesn't fit the general introduction/elimination pattern of natural deduction, and therefore its categorical semantics are not as obvious.

It is one of the great insights of Lawvere [?] (presaged by Leibniz, and approximately contemporaneous with a similar observation by Martin-Löf) that the rules of reflexivity, symmetry, transitivity, and substitutivity are equivalent to the following pair of rules:

$$\frac{}{\Gamma, x : A \mid () \vdash (x =_A x)} =R \qquad\qquad \frac{\Gamma, x : A \mid \Theta[x/y] \vdash \varphi[x/y]}{\Gamma, x : A, y : A \mid \Theta, (x =_A y) \vdash \varphi} =L$$

The first, right/introduction, rule is simply reflexivity. When combined with a substitution (to make substitution into entailment admissible) it becomes

$$\frac{\Gamma \vdash M : A}{\Gamma \mid () \vdash (M =_A M)} \tag{4.1.2}$$

and if we have weakening, we can more generally derive $\Gamma \mid \Theta \vdash (M =_A M)$ for any proposition context Θ.

The left rule is the tricky one to understand. Intuitively, it says that if we have a statement about x and y, and that statement becomes true when we substitute x for y, then that statement is true under the hypothesis that $x = y$. More generally, we can replace the truth of a statement with the truth of an entailment $\Theta \vdash \varphi$, where we also substitute x for y in Θ in the premise. In other words, *if we have a hypothesis that $x = y$, then we may as well write x instead of y everywhere that it appears.*

To help motivate this rule further, let us derive symmetry and transitivity from it. Here is symmetry:

$$\frac{x : A, y : A \vdash (y =_A x)\ \mathsf{prop} \qquad \dfrac{}{x : A \mid () \vdash (x =_A x)}}{x : A, y : A \mid (x =_A y) \vdash (y =_A x)}$$

We use the left rule once, with φ being $y =_A x$, so that $\varphi[x/y]$ is $x =_A x$, which we can prove by reflexivity.

And here is transitivity:

$$\frac{x : A, y : A, z : A \vdash (x =_A z)\ \mathsf{prop} \qquad \dfrac{}{x : A, y : A \mid (x =_A y) \vdash (x =_A y)}}{x : A, y : A, z : A \mid (x =_A y), (y =_A z) \vdash (x =_A z)}$$

We again use the left rule once on the hypothesis $y =_A z$, with φ being $x =_A z$, so that $\varphi[y/z]$ is $x =_A y$, which we can prove by the identity rule from the other hypothesis. Note that both symmetry and transitivity are *derivable rules* in the sense of Remark 1.2.6.

As with so many things, the only way to really understand this rule is to practice it. We recommend the reader try their hand at Exercise 4.1.3.

There are a few more technical things to be said about $=L$. Firstly, like $\exists L$, it should technically have additional premises making clear what Θ and φ are:

$$\frac{\Gamma, x : A, y : A \vdash \varphi \ \mathsf{prop} \qquad \Gamma, x : A, y : A \vdash \Theta \ \mathsf{ctx} \qquad \Gamma, x : A \mid \Theta[x/y] \vdash \varphi[x/y]}{\Gamma, x : A, y : A \mid \Theta, (x =_A y) \vdash \varphi}$$

Secondly, to make substitution into entailments admissible, it needs substitutions for M and N built in:

$$\frac{\begin{array}{c}\Gamma, x : A, y : A \vdash \varphi \ \mathsf{prop} \qquad \Gamma, x : A, y : A \vdash \Theta \ \mathsf{ctx} \\ \Gamma \vdash M : A \qquad \Gamma \vdash N : A \qquad \Gamma, x : A \mid \Theta[x/y] \vdash \varphi[x/y]\end{array}}{\Gamma \mid \Theta[M/x, N/y], (M =_A N) \vdash \varphi[M/x, N/y]} \tag{4.1.3}$$

Thirdly, to make cut for propositions admissible, it needs another cut built in as well; see Figure 4.2.

4.1.5 First-order theories

The last thing we need is some "generator" rules that would allow us to speak of a "first-order theory". In addition to our multigraph $\mathcal{G}$ giving the base types and terms, we would like to also have a set $\mathcal{P}$ of "base propositions" (usually called **atomic propositions**). Each of these should have an assigned *type context*, i.e. a list of objects of $\mathcal{G}$; we write $\mathcal{P}(A_1, \dots, A_n)$ for the set of atomic propositions with context $A_1, \dots, A_n$. Then we will have a generator rule for propositions, with substitutions built in just like the generator rule for terms:

$$\frac{P \in \mathcal{P}(A_1, \dots, A_n) \qquad \Gamma \vdash M_1 : A_1 \qquad \cdots \qquad \Gamma \vdash M_n : A_n}{\Gamma \vdash P(M_1, \dots, M_n) \ \mathsf{prop}}$$

Remark 4.1.4. Note that while we write φ for a *generic* proposition that might contain a variable x, and $\varphi[M/x]$ for the result of substituting M for that variable x, if P is an *atomic* proposition we write $P(x)$ and $P(M)$ for its instantiations at a variable x or a more general term M. As always, substitution $\varphi[M/x]$ is an *operation* on propositions; while the application $P(M)$ is, like the application of a function symbol $f(M)$, a *primitive* part of syntax. The relationship between them is that $(P(x))[M/x]$ is, by definition, $P(M)$ (see Theorem 4.1.6).

Remark 4.1.5. There is a substantial tradition of terminology according to which the phrase *atomic proposition* includes not just these "generating" propositions, but also *equality* propositions ($M =_A N$). This is entirely understandable historically, since when equality is presented using laws such as reflexivity, symmetry, transitivity, and substitution it appears "axiomatic" rather than governed by principled rules like those of the connectives and quantifiers. However, from a modern (i.e. post-Lawvere [**?**]) perspective, we can see that the rules $=R$ and

$=L$ have the same shape as those of the other connectives and quantifiers, and in §4.2 we will see that they similarly express a categorical universal property. Thus, it makes much more sense to call the equality rules *logical*, like those of the connectives and quantifiers, and restrict the adjective *atomic* to the generating propositions.

Finally, we should have some generating entailments, i.e. *axioms*. Each of these should have an assigned type context $A_1, \ldots, A_n$, an assigned proposition context Θ, and an assigned consequent φ. Here φ and the elements of Θ should be propositions in context $x_1 : A_1, \ldots, x_n : A_n$ — not just atomic propositions, but arbitrary ones derivable from the atomic ones and the rules for making new propositions. If we write $\mathcal{A}(A_1, \ldots, A_n; \Theta; \varphi)$ for the assertion that there is such an axiom, then simplest form of the generator rule introducing axioms will be

$$\frac{\mathcal{A}(A_1, \ldots, A_n; \Theta; \varphi)}{x_1 : A_1, \ldots, x_n : A_n \mid \Theta \vdash \varphi}$$

To make substitution into entailments admissible, we should build one in:

$$\frac{\mathcal{A}(A_1, \ldots, A_n; \Theta; \varphi) \qquad \Gamma \vdash M_1 : A_1 \qquad \cdots \qquad \Gamma \vdash M_n : A_n}{\Gamma \mid \Theta[\vec{M}/\vec{x}] \vdash \varphi[\vec{M}/\vec{x}]}$$

And to make cut admissible we should also build in a cut; see Figure 4.2.

With this rule added to the other rules for entailment, we complete the definition of **intuitionistic first-order $\mathfrak{S}$-logic**. If we include both weakening and contraction, we speak simply of **intuitionistic first-order logic**, while other values of $\mathfrak{S}$ have appropriate names like **intuitionistic first-order linear logic** (no weakening or contraction) and so on. A **first-order theory** in any such logic consists of all the generating data:

(a) A set of *objects* (also called *types* or *sorts*);

(b) A set of *morphisms* (also called *function symbols*), each with a list of objects as its domain and a single object as its codomain;

(c) A set of *atomic propositions* (also called *predicates* or *relation symbols*), each with a list of objects as its domain or arity; and

(d) A set of *axioms*, each consisting of a type context, a proposition context, and a consequent.

The qualifier "intuitionistic" is because, like in §2.7, we cannot prove the law of excluded middle $\varphi \vee \neg\varphi$ (where $\neg\varphi$ means $\varphi \multimap \bot$), or its equivalent the law of double negation $\neg\neg\varphi \multimap \varphi$. In §2.7 we motivated this by noting that leaving it out just means our "logic" has models in all Heyting algebras rather than just Boolean algebras. We will be able to say something similar, and hopefully even more convincing, about first-order logic in §4.3.

A few important subsystems of intuitionistic first-order logic that will reappear later are:

- *Coherent* logic: includes $\wedge, \top, \vee, \bot, \exists, =$ but not $\Rightarrow$ or $\forall$ (hence also not $\neg$).
- *Regular* logic: includes $\wedge, \top, \exists, =$ but not $\vee, \bot, \Rightarrow, \neg, \forall$.
- *Horn* logic: includes $\wedge, \top, =$ but not $\vee, \bot, \Rightarrow, \neg, \forall, \exists$.
- Another important logic is *geometric* logic, which is like coherent logic but also includes the "infinitary disjunction" from Exercise 2.7.5.
- In §4.5 we will study a somewhat more complicated logic to define called *finite-limit* or *lex logic*.

4.1.6 Natural deduction

At last we are ready to consider admissibility of substitution and cut. To be precise, we work with the **natural deduction for first-order intuitionistic $\mathfrak{S}$-logic** consisting of:

(a) The rules for forming terms and propositions;

(b) The exchange and possibly (depending on $\mathfrak{S}$) weakening and contraction rules;

(c) The identity rule $\Gamma \mid \varphi \vdash \varphi$ for all propositions φ;

(d) The natural deduction rules for intuitionistic $\mathfrak{S}$-logic from Figure 2.7, with an arbitrary type context Γ; and

(e) The natural deduction rules for quantifiers, equality, and axioms summarized in Figure 4.2.

Of course, the modularity of type theory means we can mix and match these rules, removing any number of type operations and their corresponding rules without altering the main theorems.

Note also that the rules $=E$ and AXIOM from Figure 4.2 incorporate an additional cut on the left. For this we have used a shortcut notation $\Gamma \mid \Phi \vdash \Theta$ where Θ is a proposition context, meaning that $\Gamma \mid \Phi \vdash \theta$ for each $\theta \in \Theta$.

We start with the admissibility of substitution into propositions.

Theorem 4.1.6. *Substitution into propositions is admissible: given derivations of $\Gamma, x : A \vdash \varphi\ \mathsf{prop}$ and $\Gamma \vdash M : A$, we can construct a derivation of $\Gamma \vdash \varphi[M/x]\ \mathsf{prop}$.*

Proof. As with substitution into terms, this is entirely straightforward because we have written all the rules for such judgments with an arbitrary type context. Some of the defining clauses are

$$(\varphi \wedge \psi)[M/x] = \varphi[M/x] \wedge \psi[M/x]$$
$$(\forall y{:}B.\, \varphi)[M/x] = \forall y{:}B.\, \varphi[M/x]$$
$$(N =_B P)[M/x] = (N[M/x] =_B P[M/x])$$

$$\frac{\Gamma, x : A \mid \Theta \vdash \varphi}{\Gamma \mid \Theta \vdash \forall x{:}A.\, \varphi}\ \forall I \qquad \frac{\Gamma \vdash M : A \qquad \Gamma \mid \Theta \vdash \forall x{:}A.\, \varphi}{\Gamma \mid \Theta \vdash \varphi[M/x]}\ \forall E$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \mid \Theta \vdash \varphi[M/x]}{\Gamma \mid \Theta \vdash \exists x{:}A.\, \varphi}\ \exists I$$

$$\frac{\Gamma \mid \Psi \vdash \exists x{:}A.\, \varphi \qquad \Gamma, x : A \mid \Theta, \varphi \vdash \psi}{\Gamma \mid \Theta, \Psi \vdash \psi}\ \exists E \qquad \frac{\Gamma \vdash M : A}{\Gamma \mid () \vdash (M =_A M)}\ {=}I$$

$$\frac{\Gamma, x : A \mid \Theta[x/y] \vdash \varphi[x/y] \qquad \begin{array}{c}\Gamma \vdash M : A \qquad \Gamma \vdash N : A \\ \Gamma \mid \Psi \vdash (M =_A N)\end{array} \qquad \Gamma \mid \Phi \vdash \Theta[M/x, N/y]}{\Gamma \mid \Phi, \Psi \vdash \varphi[M/x, N/y]}\ {=}E$$

$$\frac{\begin{array}{c}\mathcal{A}(A_1, \ldots, A_n; \Theta; \varphi) \\ \Gamma \vdash M_1 : A_1 \qquad \cdots \qquad \Gamma \vdash M_n : A_n \qquad \Gamma \mid \Phi \vdash \Theta[\vec{M}/\vec{x}]\end{array}}{\Gamma \mid \Phi \vdash \varphi[\vec{M}/\vec{x}]}\ \text{AXIOM}$$

Figure 4.2: Natural deduction rules for quantifiers, equality, and axioms

In the case of $\forall$ (and also $\exists$), we have to ensure (by α-equivalence if necessary) that x and y are distinct variables, and that y does not occur in M. This is the same issue that arose in §§2.4, 2.5 and 2.8 when substituting into terms with bound variables such as match_+ and λ-abstractions. As always, this is only an issue when representing derivations by terms; the underlying operation on derivations has no notion of "bound variable".

Note also that substitution into an equality *proposition* is defined using substitution into the *terms* appearing in it. But since terms never involve propositions, there is no cyclic dependency: we can first prove the admissibility of substitution into terms, and then use it to prove the admissibility of substitution into propositions. □

Just as substitution into terms is associative, substitution into propositions satisfies as "functoriality" property that can be proven in the same way:

$$\varphi[N/y][M/x] = \varphi[M/x][N[M/x]/y] \tag{4.1.7}$$

Theorem 4.1.8. *Substitution into entailments is admissible: if we have derivations of* $\Gamma, x : A \vdash \Theta \vdash \varphi$ *and* $\Gamma \vdash M : A$*, we can construct a derivation of* $\Gamma \vdash \Theta[M/x] \vdash \varphi[M/x]$.

Proof. Just like Theorem 4.1.6, we substitute recursively into the derivation of $\Gamma, x : A \vdash \Theta \vdash \varphi$. This works because all the entailment rules have a fully general type context in the conclusion, so the substitution can always be done inductively on their premises. □

Theorem 4.1.9. *Cut for propositions is admissible in the natural deduction for first-order intuitionistic $\mathfrak{S}$-logic: given derivations of $\Gamma \mid \Theta \vdash \varphi$ and $\Gamma \mid \Psi, \varphi \vdash \psi$, we can construct a derivation of $\Gamma \mid \Psi, \Theta \vdash \psi$.*

Proof. As usual, we induct on the derivation of $\Gamma \mid \Psi, \varphi \vdash \psi$. This works because all the rules for the natural deduction have a fully general proposition context in the conclusion as well. □

As in §2.7.2, in the cartesian case we can make exchange, weakening, and contraction admissible as well, by reformulating the rules $\exists E$ and $=E$ to keep the same proposition context in the premises and the conclusion, and the identity rule to incorporate a weakening. The reformulated rules are

$$\frac{\Gamma \mid \Theta \vdash \exists x{:}A.\,\varphi \qquad \Gamma, x : A \mid \Theta, \varphi \vdash \psi}{\Gamma \mid \Theta \vdash \psi}\;\exists E'$$

$$\frac{\Gamma, x : A \mid \Phi, \Theta[x/y] \vdash \varphi[x/y] \qquad \begin{array}{c}\Gamma \vdash M : A \qquad \Gamma \vdash N : A \\ \Gamma \mid \Phi \vdash (M =_A N)\end{array} \qquad \Gamma \mid \Phi \vdash \Theta[M/x, N/y]}{\Gamma \mid \Phi \vdash \varphi[M/x, N/y]}\;{=}E'$$

We leave the proof to the reader (Exercise 4.1.5).

Recall also from §2.7.3 that with such a reformulation, the natural deduction of intuitionistic propositional logic can be formulated without explicit contexts, instead "discharging" temporary assumptions by crossing them out. The same is true for intuitionistic first-order logic with $\forall$ and $\exists$, if we also allow "assumptions of variables" (i.e. additions to the type context, in addition to the proposition context, as we move up the derivation tree) that can be discharged by the quantifier rules. Usually we do not bother to include the derivations of term judgments in this style; we just write down the terms wherever they are needed.

For instance, here is a derivation in this style of the intuitionistic tautology $\varphi \wedge (\exists x{:}A.\,\psi) \Rightarrow (\exists x{:}A.\,(\varphi \wedge \psi))$.

$$\dfrac{\dfrac{\dfrac{\cancel{\varphi \wedge (\exists x{:}A.\,\psi)}}{\exists x{:}A.\,\psi} \qquad \dfrac{\cancel{x : A} \qquad \dfrac{\dfrac{\cancel{\varphi \wedge (\exists x{:}A.\,\psi)}}{\varphi} \qquad \cancel{\psi}}{\varphi \wedge \psi}\;\wedge I}{\exists x{:}A.\,(\varphi \wedge \psi)}\;\exists I}{\exists x{:}A.\,(\varphi \wedge \psi)}\;\exists E'}{\varphi \wedge (\exists x{:}A.\,\psi) \Rightarrow (\exists x{:}A.\,(\varphi \wedge \psi))}\;{\Rightarrow} I$$

The hypotheses $x : A$ and ψ are discharged by the $\exists E'$, while the two occurences of the hypothesis $\varphi \wedge (\exists x{:}A.\,\psi)$ are discharged by the $\Rightarrow I$. Without too much stretch, this can be regarded as a direct formalization of the following informal English proof:

> Suppose $\varphi \wedge (\exists x{:}A.\, \psi)$, that is φ and $\exists x{:}A.\, \psi$. By the latter, we may assume given an x such that ψ. Now we have both φ and ψ, so $\varphi \wedge \psi$. Thus, $\exists x{:}A.\, (\varphi \wedge \psi)$, and so we have $\varphi \wedge (\exists x{:}A.\, \psi) \Rightarrow (\exists x{:}A.\, (\varphi \wedge \psi))$.

Remark 4.1.10. In particular, this (nontrivial!) interaction between $\wedge$ and $\exists$ is, like the distributive law of $\wedge/\otimes$ over $\vee$ from Exercise 2.3.4 and §2.7, implied automatically by the structure of our contexts and how they interact with the rules for $\wedge$ and $\exists$. There is sometimes a temptation to "simplify" logic by presenting it as a unary type theory, arguing that a context $\Theta = (\varphi_1, \ldots, \varphi_n)$ can always be replaced by the conjunction $\varphi_1 \wedge \cdots \wedge \varphi_n$, and perhaps even replacing entailments $\varphi \vdash \psi$ by implications $\varphi \Rightarrow \psi$. This is technically possible, but it forces one to assert laws like these "by hand", breaking principle $(*)$ and making for a less congenial theory. More importantly, as remarked in §2.7.3, allowing arbitrarily many propositions in the context yields a formal theory that matches informal reasoning much better: as in the example above, informally we frequently apply inference rules in the presence of other unaffected hypotheses.

Furthermore, for categorical semantics it is important to maintain the distinction between entialment and implication, since entailment corresponds to a morphism in a category, whereas implication corresponds to an internal-hom in a category. In particular, the former always exists, but the latter may not. Phrasing the rules for logical operations such as $\exists$ and $\vee$ in a way that matches ordinary reasoning, *and* doesn't refer to any other operations such as $\Rightarrow$, ensures that ordinary informal (constructive) reasoning can be formalized and remain valid in any category as long as it uses only operations that exist in that category. This is important because we will see in §4.3 that certain fairly natural conditions on categories allow them to model some, but not always all, of the logical operations.

The rule $=E'$ is a bit tricker to write in this style because of the arbitrary context Θ that has to be substituted into. One approach is to use Exercise 4.1.1, which shows that as long as we also have implication we can get around this. A more direct approach is to allow the proof of $\varphi[x, y]$ to discharge an arbitrary number of hypotheses of the form $\theta[x/y]$, as long as we also supply corresponding proofs of $\theta[M/x, N/y]$ to the rule $=E'$. For instance, with one θ formula the rule would look like this:

$$\dfrac{\dfrac{\vdots}{M =_A N} \qquad \dfrac{\vdots}{\theta[M/x, N/y]} \qquad \dfrac{\cancel{x : A} \quad \cancel{\theta[x/y]} \\ \vdots}{\varphi[x/y]}}{\varphi[M/x, N/y]}$$

Exercises

Exercise 4.1.1. Assuming we have $\multimap$, show that the rule $=R$ is derivable (recall Remark 1.2.6) from the following simpler rule with no proposition context Θ:

$$\frac{\begin{array}{c}\Gamma \vdash M : A \qquad \Gamma \vdash N : A \\ \Gamma, x : A, y : A \vdash \varphi \ \mathsf{prop} \qquad \Gamma \vdash \Theta \ \mathsf{ctx} \qquad \Gamma, x : A \mid \Theta \vdash \varphi[x/y]\end{array}}{\Gamma \mid \Theta, (M =_A N) \vdash \varphi[M/x, N/y]}$$

Exercise 4.1.2. Three of the following four sequents are derivable in intuitionistic first-order logic (for any type A, context Γ, and proposition $\Gamma, x : A \vdash \varphi \ \mathsf{prop}$); derive them.

$$\Gamma \mid \exists x{:}A.\, \neg\varphi \vdash \neg\forall x{:}A.\, \varphi$$
$$\Gamma \mid \forall x{:}A.\, \neg\varphi \vdash \neg\exists x{:}A.\, \varphi$$
$$\Gamma \mid \neg\forall x{:}A.\, \varphi \vdash \exists x{:}A.\, \neg\varphi$$
$$\Gamma \mid \neg\exists x{:}A.\, \varphi \vdash \forall x{:}A.\, \neg\varphi$$

Exercise 4.1.3. In a first-order theory with three types A, B, C, two generating arrows $f : A \to B$ and $g : B \to A$, one atomic proposition P with domain (A, B), and no axioms, derive the following judgments:

(a) $x_1 : A, x_2 : A, y : B \mid \varphi(x_1, y), (x_1 =_A x_2) \vdash \varphi(x_2, y)$

(b) $x_1 : A, x_2 : A \mid (x_1 =_A x_2) \vdash f(x_1) =_B f(x_2)$

(c) $() \mid (\forall x{:}A.\, g(f(x)) =_A x) \vdash \forall x_1{:}A.\, \forall x_2{:}A.\, ((f(x_1) =_B f(x_2)) \to (x_1 =_A x_2))$

Exercise 4.1.4. Write down a first-order theory for each of the following structures. If you can, formulate them so that they fit inside the specified fragment.

(a) Partially ordered sets (Horn)

(b) Totally ordered sets (coherent)

(c) Fields (coherent)

(d) Categories (regular)

Exercise 4.1.5. Prove that in intuitionistic first-order logic with $\exists E$ and $=E$ replaced by $\exists E'$ and $=E'$ as mentioned at the end of the section, the structural rules of exchange, weakening, and contraction for proposition contexts are admissible.

4.2 First-order hyperdoctrines

Now we move on to the categorical semantics of first-order logic. Continued adherence to principle (‡) suggests that the *structural rules*, including for instance the substitution of terms into propositions and entailments, should correspond to basic operations in an appropriate categorical structure. This would lead us to the following structure.

Let $\mathfrak{S}$ be a faithful cartesian club, and recall from §2.6 the notion of $\mathfrak{S}$-multicategory and $\mathfrak{S}$-multiposet. In contrast to chapters 1 and 2, in this chapter we will assume for simplicity that our multiposets *do* satisfy antisymmetry: if $x \le y$ and $y \le x$ then $x = y$. Allowing distinct isomorphic objects, while morally correct, would lead us down a 2-categorical road that we prefer to postpone until §4.6.

Definition 4.2.1. Let $\mathcal{S}$ be a cartesian multicategory and $\mathbf{C}$ a category. A **$\mathbf{C}$-valued presheaf on $\mathcal{S}$** consists of

(a) For each list $(A_1, \dots, A_n)$ of objects of $\mathcal{S}$, an object $\mathcal{P}(A_1, \dots, A_n) \in \mathbf{C}$.

(b) For each list $(f_1, \dots, f_m)$ of morphisms of $\mathcal{S}$, with $f_i : (A_{i1}, \dots, A_{in_i}) \to B_i$, a morphism in $\mathbf{C}$:

$$(f_1, \dots, f_n)^* : \mathcal{P}(B_1, \dots, B_m) \to \mathcal{P}(A_{11}, \dots, A_{mn_m})$$

(c) These morphisms are associative and unital with respect to composition in $\mathcal{S}$:

$$(f_{11}, \dots, f_{mn_m})^* \circ (g_1, \dots, g_m)^* = (g_1 \circ (f_{11}, \dots, f_{1n_1}), \dots, g_m \circ (f_{m1}, \dots, f_{mn_m}))^*$$
$$(\mathsf{id}_{A_1}, \dots, \mathsf{id}_{A_n})^* = \mathsf{id}_{\mathcal{P}(A_1, \dots, A_n)}$$

(d) For each $\sigma : \{1, \dots, m\} \to \{1, \dots, n\}$, a morphism in $\mathbf{C}$:

$$\mathcal{P}(A_{\sigma 1}, \dots, A_{\sigma m}; B) \to \mathcal{P}(A_1, \dots, A_n; B)$$

satisfying analogues of the axioms in Definition 2.6.4.

One way to understand the definition is that it is precisely the structure possessed by the contravariant representables: for any object B in a cartesian multicategory $\mathcal{S}$, there is a **Set**-valued presheaf $\mathcal{S}(-; B)$.

Now the categorical structure corresponding to first-order logic should consist of a cartesian multicategory $\mathbf{S}$ and a presheaf $\mathcal{P}$ on $\mathbf{S}$ valued in the category of $\mathfrak{S}$-multiposets. The objects and morphisms of $\mathcal{S}$ represent the types and terms, respectively; while the objects of $\mathcal{P}(A_1, \dots, A_n)$ represent the propositions in context $(A_1, \dots, A_n)$ and its morphisms/inequalities represent the entailments in that same context. Composition in $\mathcal{S}$ represents substitution into terms, composition in each $\mathcal{P}(A_1, \dots, A_n)$ represents the cut rule for propositions, and the functorial action of $\mathcal{P}$ represents substitution of terms into propositions and entailments.

However, in addition to being nonstandard, this structure is rather unnecessarily complicated. It can be simplified greatly by the following observation, whose proof we leave to the reader (Exercise 4.2.1).

Lemma 4.2.2. **C**-*valued presheaves on a cartesian multicategory* $\mathcal{S}$ *are equivalent to ordinary* **C**-*valued presheaves on the category with finite products freely generated by* $\mathcal{S}$ *as in Theorem 2.9.2.* □

Moreover, in practice we rarely care about semantics in cartesian multicategories that do not arise from categories with products. Thus, we retreat slightly from the principled position of chapter 2, and simplify our lives by taking the base $\mathcal{S}$ to be a category with products rather than a cartesian multicategory throughout. This leads to the following definition.

Definition 4.2.3. An **$\mathcal{S}$-indexed $\mathfrak{S}$-multiposet** is a functor $\mathcal{P}$ from $\mathcal{S}^{\mathrm{op}}$ to the category of $\mathfrak{S}$-multiposets.

Since we do not include product types in our base theory, this means that the free structure generated from a first-order logic will involve the *category of contexts* introduced in §2.9.1, and possess a universal property only up to equivalence. For this reason we will often use letters like Γ, Δ for objects of $\mathcal{S}$. Thus, in this definition we have categorical counterparts of the type contexts (objects of $\mathcal{S}$), terms (morphisms of $\mathcal{S}$), substitution into terms (composition in $\mathcal{S}$), propositions (objects of $\mathcal{P}(\Gamma)$), entailments (morphisms of $\mathcal{P}(\Gamma)$), cut for propositions (composition in $\mathcal{P}(\Gamma)$), and substitution of terms into propositions and entailments (the functorial action of $\mathcal{P}$). In general for a morphism $f : \Gamma \to \Delta$ in $\mathcal{S}$, we write $f^* : \mathcal{P}(\Delta) \to \mathcal{P}(\Gamma)$ for this latter action and call it a **reindexing** or **substitution** functor.

The propositional operations imported from §2.7 are also easy to describe categorically.

Definition 4.2.4. Let $\mathcal{P}$ be an $\mathcal{S}$-indexed $\mathfrak{S}$-multiposet. We say that $\mathcal{P}$ has **products**, **coproducts**, is **representable**, or is **closed**, if each $\mathfrak{S}$-multiposet $\mathcal{P}(\Gamma)$ has the corresponding structure, and that structure is preserved by the reindexing functors f^*.

We did not define formally in §2.2 what it means for a functor to preserve all these properties of a multicategory, but we trust the reader can do it. The requirement that f^* preserve these properties is necessary because substitution in type theory does, by definition, preserve the type operations: $(\varphi \wedge \psi)[M/x] = (\varphi[M/x] \wedge \psi[M/x])$ and so on. Thus, in the free structure built from type theory the reindexing functors do preserve all the relevant structure, so we can't hope for it to be initial except in a world where that structure is always preserved.

Of course, one may naturally wonder, where do indexed multiposets with these properties come from? We will consider this question in more depth in §4.3, but here are three fundamental examples to help the intuition.

Example 4.2.5. Let $\mathcal{S} = \mathbf{Set}$ be the category of sets, and define $\mathcal{P}(\Gamma)$ to be the poset of subsets of the set Γ, with its cartesian multiposet structure. The latter is in fact a Heyting algebra, and moreover a Boolean algebra: $\wedge$ is intersection, $\vee$ is union, $\neg$ is complement.

Example 4.2.6. Let $\mathcal{S}$ be any category with finite limits, and define $\mathcal{P}(\Gamma)$ to be the poset of subobjects of Γ, i.e. isomorphism classes of monomorphisms with codomain Γ. The reindexing functors are given by pullback. When $\mathcal{S} = \mathbf{Set}$, this reproduces Example 4.2.5 up to isomorphism. In general, we need more structure on $\mathcal{S}$ to ensure that this $\mathcal{P}$ has the structure of Definition 4.2.4; we will study this question in §4.3.

Example 4.2.7. Let H be any complete Heyting algebra, let $\mathcal{S} = \mathbf{Set}$, and define $\mathcal{P}(\Gamma) = H^\Gamma$, the poset of Γ-indexed families $\{h_i\}_{i\in\Gamma}$ of objects of H. The Heyting algebra operations on H applied pointwise (e.g. $\{h_i\}_{i\in\Gamma} \wedge \{k_i\}_{i\in\Gamma} = \{h_i \wedge k_i\}_{i\in\Gamma}$) make $\mathcal{P}(\Gamma)$ a Heyting algebra as well. Note that when $H = \mathbf{2}$, this again reproduces Example 4.2.5 up to isomorphism.

It remains to consider categorical analogues of the quantifiers and equality. Lawvere's fundamental insight [**?**, **?**] was that these correspond categorically to *adjoint functors*.

Consider, for instance, the universal quantifier. We saw in §4.1.2 that its rules could be given as the following pair:

$$\frac{\Gamma, x : A \mid \Theta \vdash \varphi}{\Gamma \mid \Theta \vdash \forall x{:}A.\, \varphi}\ \forall I \qquad\qquad \frac{\Gamma \mid \Theta \vdash \forall x{:}A.\, \varphi}{\Gamma, x : A \mid \Theta \vdash \varphi}\ \forall I^{-1}$$

which are clearly inverses to each other. Categorically, they say that to have a morphism from Θ to $\forall x{:}A.\, \varphi$ in $\mathcal{P}(\Gamma)$ is equivalent to having a morphism from Θ to φ in $\mathcal{P}(\Gamma, A)$. Here the second Θ technically denotes the weakening of Θ to the context $\Gamma, x : A$, which categorically will be the functorial action of $\mathcal{P}$ applied to the projection $(\Gamma, A) \to \Gamma$. Note that the latter is one of the projections of a cartesian product in the category of contexts. This leads to the following definition.

Definition 4.2.8. Let $F : \mathcal{M} \to \mathcal{N}$ be a functor of $\mathfrak{S}$-multicategories. We say it **has a right adjoint** if for each object $B \in \mathcal{N}$ there is an object $GB \in \mathcal{M}$ and a morphism $\varepsilon_B : FGB \to B$ in $\mathcal{N}$ such that for any $A_1, \ldots, A_n \in \mathcal{M}$, the composite

$$\mathcal{M}(A_1, \ldots, A_n; GB) \xrightarrow{F} \mathcal{N}(FA_1, \ldots, FA_n; FGB) \xrightarrow{\varepsilon_B \circ -} \mathcal{N}(FA_1, \ldots, FA_n; B)$$

is a bijection.

The case $n = 1$ of this definition implies immediately that the underlying ordinary functor of F has a right adjoint in the usual sense. Conversely, in the case when $\mathcal{M}$ and $\mathcal{N}$ are representable, it is sufficient to have such an underlying adjoint together with the fact that F preserves tensor products; see Exercise 4.2.3. Moreover, if G exists, it can be made into a functor $\mathcal{N} \to \mathcal{M}$, that is right adjoint to $\mathcal{M}$ in an appropriate 2-category of $\mathfrak{S}$-multicategories; see Exercise 4.2.2.

We need one more thing for a categorical analogue of $\forall$: we need to know that this structure is "preserved by the reindexing functors" in an appropriate sense. The appropriate sense is the following.

Definition 4.2.9. Let $\mathcal{S}$ be a category, let $\mathcal{P} : \mathcal{S}^{\mathrm{op}} \to \mathbf{Cat}$ be a functor, and suppose we have a commutative square in $\mathcal{S}$:

$$\begin{array}{ccc} A & \xrightarrow{h} & C \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[k]{} & D. \end{array}$$

Suppose furthermore that the functors $f^* : \mathcal{P}(B) \to \mathcal{P}(A)$ and $g^* : \mathcal{P}(D) \to \mathcal{P}(C)$ have right adjoints f_* and g_*. We say that $\mathcal{P}$ satisfies the **right Beck–Chevalley condition** with respect to this square (or sometimes that the square satisfies the Beck-Chevalley condition with respect to $\mathcal{P}$) if the composite natural transformation

$$k^* g_* \xrightarrow{\eta k^* g_*} f_* f^* k^* g_* = f_* h^* g^* g_* \xrightarrow{f_* h^* \varepsilon} f_* h^*$$

is an isomorphism. Dually, if f^* and g^* have left adjoints $f_!$ and $g_!$, we say $\mathcal{P}$ satisfies the **left Beck–Chevalley condition** with respect to the above square if the composite

$$f_! h^* \xrightarrow{f_! h^* \eta} f_! h^* g^* g_! = f_! f^* k^* g_! \xrightarrow{\varepsilon k^* g_!} k^* g_!$$

is an isomorphism.

When $\mathcal{P}$ is an $\mathcal{S}$-indexed $\mathfrak{S}$-multiposet, we apply this definition to its underlying functor into posets (regarded as categories). Since our posets are antisymmetric, every isomorphism is an equality, and so in this case we have $k^* g_* = f_* h^*$ (or $f_! h^* = k^* g_!$). Now we can state:

Definition 4.2.10. An $\mathcal{S}$-indexed $\mathfrak{S}$-multiposet **has universal quantifiers** if

(a) For any objects $\Gamma, A \in \mathcal{S}$, the reindexing functor $\mathcal{P}(\Gamma) \to \mathcal{P}(\Gamma \times A)$ has a right adjoint in the sense of Definition 4.2.8; and

(b) For any morphism $f : \Gamma \to \Delta$ and object A in $\mathcal{S}$, $\mathcal{P}$ satisfies the right Beck–Chevalley condition with respect to the square

$$\begin{array}{ccc} \Gamma \times A & \xrightarrow{f \times \mathsf{id}_A} & \Delta \times A \\ \downarrow & & \downarrow \\ \Gamma & \xrightarrow[f]{} & \Delta. \end{array}$$

Note that the Beck–Chevalley condition is true in the syntax because the universal quantifier is preserved by substitution, by definition of substitution: $(\forall x{:}A.\, \varphi)[M/y] = \forall x{:}A.\, \varphi[M/y]$ as long as $y \neq x$. For the indexed poset of subsets from Example 4.2.5, the right adjoint to $(\pi_A)^* : \mathcal{P}(\Gamma) \to \mathcal{P}(\Gamma \times A)$ is similarly defined by

$$(\pi_A)_*(\varphi) = \{\, x \in \Gamma \mid \forall y \in A.(x,y) \in \varphi \,\}\,.$$

Such right adjoints for Example 4.2.6 will be studied in §4.3.4; while for Example 4.2.7, they can defined by

$$(\pi_A)_*\left(\{h_{(i,a)}\}_{(i,a)\in\Gamma\times A}\right) = \left\{\bigwedge_a h_{(i,a)}\right\}_{i\in\Gamma}$$

The existential quantifier is similar, but a bit more subtle. We saw in §4.1.3 that its rules could be expressed as the pair

$$\frac{\Gamma, x : A \mid \Theta, \varphi \vdash \psi}{\Gamma \mid \Theta, (\exists x{:}A.\,\varphi) \vdash \psi}\;\exists L \qquad\qquad \frac{\Gamma \mid \Theta, (\exists x{:}A.\,\varphi) \vdash \psi}{\Gamma, x : A \mid \Theta, \varphi \vdash \psi}\;\exists L^{-1}$$

which likewise seem to express some kind of adjunction; but there is an extra context Θ hanging around. Translating directly across the correspondence to multicategories, this leads to the following definition.

Definition 4.2.11. Let $G : \mathcal{M} \to \mathcal{N}$ be a functor of $\mathfrak{S}$-multicategories. We say it **has a Hopf left adjoint** if for each object $B \in \mathcal{N}$ there is an object $FB \in \mathcal{M}$ and a morphism $\eta : B \to GFB$ in $\mathcal{N}$ such that for any objects $A_1, \dots, A_n, C_1, \dots, C_m, D \in \mathcal{M}$, the composite

$$\mathcal{M}(\vec{A}, FB, \vec{C}; D) \xrightarrow{G} \mathcal{N}(G\vec{A}, GFB, G\vec{C}; GD) \xrightarrow{-\circ_{(n+1)}\eta} \mathcal{N}(G\vec{A}, B, G\vec{C}; GD)$$

is a bijection.

As before, the case $n = m = 1$ implies that the underlying ordinary functor has a left adjoint in the usual sense. Conversely, when $\mathcal{M}$ and $\mathcal{N}$ are representable and G preserves tensor products, an underlying left adjoint is a Hopf left adjoint just when some canonical maps are isomorphisms; see Exercise 4.2.6. Unlike the case of right adjoints, however, in general a Hopf left adjoint cannot be made into a functor of multicategories.

Definition 4.2.12. An $\mathcal{S}$-indexed $\mathfrak{S}$-multiposet **has existential quantifiers** if

(a) For any objects Γ and A of $\mathcal{S}$, the reindexing functor $\mathcal{P}(\Gamma) \to \mathcal{P}(\Gamma \times A)$ has a Hopf left adjoint in the sense of Definition 4.2.11; and

(b) For any morphism $f : \Gamma \to \Delta$ and object A in $\mathcal{S}$, $\mathcal{P}$ satisfies the left Beck–Chevalley condition with respect to the square

$$\begin{array}{ccc} \Gamma \times A & \xrightarrow{f \times \mathsf{id}_A} & \Delta \times A \\ \downarrow & & \downarrow \\ \Gamma & \xrightarrow[f]{} & \Delta. \end{array}$$

Unsurprisingly, for the indexed poset of subsets from Example 4.2.5, the left adjoint to $(\pi_A)^* : \mathcal{P}(\Gamma) \to \mathcal{P}(\Gamma \times A)$ is similarly defined by

$$(\pi_A)_!(\varphi) = \{\, x \in \Gamma \mid \exists y \in A.(x, y) \in \varphi \,\}.$$

Left adjoints for Example 4.2.6 will be studied in §4.3.2, while those in Example 4.2.7 can be defined like the right adjoints using joins instead of meets:

$$(\pi_A)_! \left(\{h_{(i,a)}\}_{(i,a)\in\Gamma\times A}\right) = \left\{ \bigvee_a h_{(i,a)} \right\}_{i\in\Gamma}$$

Finally, we consider the rules for equality:

$$\frac{}{\Gamma, x : A \mid () \vdash (x =_A x)} \qquad\qquad \frac{\Gamma, x : A \mid \Theta[x/y] \vdash \varphi[x/y]}{\Gamma, x : A, y : A \mid \Theta, (x =_A y) \vdash \varphi}$$

Although we didn't mention it in §4.1.4, the first of these rules is equivalent to the opposite of the second, similarly to what happened for the quantifiers. One direction is immediate:

$$\frac{\dfrac{}{\Gamma, x : A, y : A \mid (x =_A y) \vdash (x =_A y)}}{\Gamma, x : A \mid () \vdash (x =_A x)}$$

while the other uses a cut and a substitution:

$$\frac{\dfrac{}{\Gamma, x : A \mid () \vdash x =_A x} \qquad \dfrac{\Gamma, x : A, y : A \mid \Theta, (x =_A y) \vdash \varphi}{\Gamma, x : A \mid \Theta[x/y], (x =_A x) \vdash \varphi[x/y]}\ \textsc{subst}\ x/y}{\Gamma, x : A \mid \Theta[x/y] \vdash \varphi[x/y]}\ \textsc{cut}$$

Thus, what we have looks very much like a (Hopf) left adjoint to substitution along the diagonal $(\Gamma, A) \to (\Gamma, A, A)$ in the category of contexts; but there is *no* proposition in context (Γ, A) that it is applied to. This suggests the following definitions.

Definition 4.2.13. Let $G : \mathcal{M} \to \mathcal{N}$ be a functor of $\mathfrak{S}$-multicategories. We say it **has a Hopf left adjoint at** $()$ if there is an object $F \in \mathcal{M}$ and a morphism $\eta : () \to GF$ in $\mathcal{N}$ such that for any objects $A_1, \dots, A_n, C_1, \dots, C_m, D \in \mathcal{M}$, the composite

$$\mathcal{M}(\vec{A}, F, \vec{C}; D) \xrightarrow{G} \mathcal{N}(G\vec{A}, GF, G\vec{C}; GD) \xrightarrow{-\circ_{(n+1)}\eta} \mathcal{N}(G\vec{A}, G\vec{C}; GD)$$

is a bijection.

Definition 4.2.14. Suppose given an $\mathcal{S}$-indexed $\mathfrak{S}$-multiposet $\mathcal{P}$ and a commutative square in $\mathcal{S}$:

$$\begin{array}{ccc} A & \xrightarrow{h} & C \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[k]{} & D, \end{array}$$

and suppose that the reindexing functors f^* and g^* have Hopf left adjoints at $()$, given by objects $f_! \in \mathcal{P}(B)$ and $g_! \in \mathcal{P}(D)$. Then there is a unique morphism $f_! \to k^* g_!$ in $\mathcal{P}(B)$ such that the composite $() \xrightarrow{\eta} f^* f_! \to f^* k^* g_!$ in $\mathcal{P}(A)$ is equal to the composite $() \xrightarrow{h^*\eta} h^* g^* g_!$ (note $f^* k^* = h^* g^*$). We say that $\mathcal{P}$ satisfies the **left Beck–Chevalley condition at** $()$ with respect to this square if this morphism $f_! \to k^* g_!$ is an isomorphism.

Definition 4.2.15. An $\mathcal{S}$-indexed $\mathfrak{S}$-multiposet with unit objects **has equality** if

(a) For any objects Γ and A of $\mathcal{S}$, the reindexing functor $\mathcal{P}(\Gamma \times A \times A) \to \mathcal{P}(\Gamma \times A)$ has a Hopf left adjoint at $()$; and

(b) For any morphism $f : \Gamma \to \Delta$ and object A in $\mathcal{S}$, $\mathcal{P}$ satisfies the left Beck–Chevalley condition at $()$ with respect to the square

$$\begin{array}{ccc} \Gamma \times A & \xrightarrow{f \times \mathsf{id}_A} & \Delta \times A \\ \downarrow & & \downarrow \\ \Gamma \times A \times A & \xrightarrow[f \times \mathsf{id}_A \times \mathsf{id}_A]{} & \Delta \times A \times A. \end{array}$$

As before, the Beck–Chevalley condition is true in the syntax because “equality is preserved by substitution”. The relevant substitution here is not the one built into the equality rule, though, but the substitution for different variables, which doesn’t change the equality proposition at all: $(x =_A y)[M/z] = (x =_A y)$ as long as $z \neq x$ and $z \neq y$. For the indexed poset of subsets from Example 4.2.5, the left adjoint to $(\Delta_A)^* : \mathcal{P}(\Gamma \times A \times A) \to \mathcal{P}(\Gamma \times A)$ at $()$ is defined by

$$(\Delta_A)_! = \{\, (i, x, y) \in \Gamma \times A \times A \mid x = y \,\}\,.$$

Left adjoints in Example 4.2.6 actually always exist (see Theorem 4.3.1), while those in Example 4.2.7 can be defined by

$$((\Delta_A)_!)_{(i,x,y)\in\Gamma\times A\times A} = \begin{cases} \top & \text{if } x = y \\ \bot & \text{if } x \neq y. \end{cases}$$

Note that we are sticking doggedly to the principle that just as the rules for a given type operation should be independent of any other type operations, the corresponding universal property should be statable without reference to any other objects with universal properties.[1] If we *do* have additional structure, particularly tensor products and units in the multiposets $\mathcal{P}(\Gamma)$, then our various kinds of adjoints can be formulated in terms of those and ordinary adjunctions — see Exercises 4.2.2, 4.2.3, 4.2.6 and 4.2.8 — and our examples in §4.3 will mainly arise in this way. However, to make a closer connection to the type theory we prefer to formulate them independently first.

[1] At least, other universal properties in the multiposets $\mathcal{P}(\Gamma)$. We do still refer to cartesian products in $\mathcal{S}$, but we could also remove those by working with “presheaves on multicategories” as sketched at the beginning of this section.

Definition 4.2.16. A **first-order $\mathfrak{S}$-hyperdoctrine** consists of a category $\mathcal{S}$ with finite products together with an $\mathcal{S}$-indexed $\mathfrak{S}$-multiposet that is closed and representable and has products (finite meets), coproducts (finite joins), universal and existential quantifiers, and equality.

By default, a **first-order hyperdoctrine** refers to the cartesian case where $\mathfrak{S}$ contains all functions; in this case representability is equivalent to having finite meets. More generally, an $\mathcal{S}$-indexed cartesian multiposet is called a:

(a) **coherent hyperdoctrine** if it has finite meets, finite joins, existential quantifiers, and equality;

(b) **geometric hyperdoctrine** if it has finite meets, infinite joins, existential quantifiers, and equality;

(c) **regular hyperdoctrine** if it has finite meets, existential quantifiers, and equality; and a

(d) **Horn hyperdoctrine** if it has finite meets and equality.

Note that since all the structure of a first-order $\mathfrak{S}$-hyperdoctrine is determined by universal properties, it is unique up to isomorphism, and hence unique on the nose in an (antisymmetric) poset. Thus, there is no need to suppose separately that we have *chosen* such operations.

Theorem 4.2.17. *The free first-order $\mathfrak{S}$-hyperdoctrine generated by a first-order $\mathfrak{S}$-theory can be presented, up to equivalence, by the type theory of the latter:*

- *$\mathcal{S}$ is the category of type contexts; and*
- *the poset $\mathcal{P}(\Gamma)$ is obtained from the poset of proposition judgments $\Gamma \vdash \varphi \ \mathsf{prop}$ and derivable entailments $\Gamma \mid \Theta \vdash \varphi$ by identifying isomorphic objects (since in this section our posets are antisymmetric).*

(And similarly for the other fragments with fewer type operations.)

Proof. We have already observed that this structure defines an indexed $\mathfrak{S}$-multicategory and that the simple type operations $\wedge, \top, \vee, \bot, \otimes, \mathbf{1}, \multimap$ yield the appropriate multicategorical strurcture. Moreover, we defined the categorical notions of universal and existential quantifiers and equality precisely so that they would hold in the syntax; thus the description above does yield a first-order $\mathfrak{S}$-hyperdoctrine.

Now, the underlying multigraph of a first-order $\mathfrak{S}$-theory is of course a finite-product theory without axioms, and we showed in §2.9.1 that the category of contexts of its type theory is, up to equivalence, the free category with products it generates. Thus, it maps uniquely (up to isomorphism) into the base category of any other first-order $\mathfrak{S}$-hyperdoctrine; it remains to show that this map extends uniquely to a map of hyperdoctrines, i.e. a natural transformation between the $\mathcal{P}$-functors preserving all the structure.

As usual, we do this by induction on derivations. The proposition judgment $\Gamma \vdash \varphi \ \mathsf{prop}$ is easy: each rule corresponds to one of the objects with a universal property that we have assumed to exist in any first-order $\mathfrak{S}$-hyperdoctrine. Next, since the rules for entailment involve substitution of terms into propositions, before defining our functor on entailments we have to first prove that it maps such substitutions to the reindexing functors in the target; this is another straightforward induction on derivations of $\Gamma \vdash \varphi\ \mathsf{prop}$. Now the rules for entailment involving simple type operations are also easy, just as in §2.7. Finally, in §§4.1.2–4.1.4 and this section we showed that the natural deduction rules for quantifiers and equality are inter-derivable (in the presence of substitution and cut) with the rules that exactly express the appropriate kind of adjunctions.

This completes the definition on entailments. Since everything is posetal there is not much left to do: we show that our map preserves all the hyperdoctrine structure, essentially by definition, and then that it is unique (modulo the up-to-isomorphism uniqueness of the functor on base categories), because its definition was forced at every step. □

Remark 4.2.18. As noted in Remark 2.7.8 for propositional logic, Theorem 4.2.17 implies the traditional *soundness* and *completeness* theorems for first-order logic with respect to hyperdoctrines. The soundness theorem says that if we can prove $\Gamma \mid () \vdash \varphi$, then when we interpret our logic into any hyperdoctrine, φ must go to the top element, i.e. it must "be true". In particular, this applies to models in the hyperdoctrine of sets and subsets from Example 4.2.5, which are the classical notion of "model". (In §4.3 we will construct hyperdoctrines from more general categories than **Set**.) Conversely, the completeness theorem says that if something is true in all hyperdoctrines, then it must in particular be true in the free one constructed from the type theory, and therefore must be provable in the type theory.

Remark 4.2.19. Note that all categorical structure corresponding to quantifiers and equality takes the form of *certain* adjoints to reindexing functors. As we will see in §4.3, most examples arising in practice naturally have adjoints to *all* the reindexing functors (if they have any). However, this is not actually an additional condition; given only the adjoints assumed in our definition of first-order hyperdoctrine, we can *construct* adjoints to arbitrary reindexing functors and *prove* that they satisfy some Beck–Chevalley conditions. At the moment, we leave this proof to the reader; see Exercise 4.2.9. (In fact, it is more usual to include all such adjoints, and their Beck–Chevalley conditions, in the definition of "hyperdoctrine".)

Exercises

Exercise 4.2.1. Prove Lemma 4.2.2.

Exercise 4.2.2. Suppose a functor $F : \mathcal{M} \to \mathcal{N}$ of $\mathfrak{S}$-multicategories has a right adjoint in the sense of Definition 4.2.8.

(a) Prove that if $\mathcal{M}$ and $\mathcal{N}$ are both representable, then F preserves tensor products in the sense of Exercise 2.2.4.

(b) Extend G to a functor $G : \mathcal{N} \to \mathcal{M}$.

(c) Define a 2-category of $\mathfrak{S}$-multicategories and show that G is right adjoint to F in this 2-category (i.e. there are 2-cells $\eta : 1 \to GF$ and $\varepsilon : FG \to 1$ in this 2-category satisfying the triangle identities).

Exercise 4.2.3. Let $\mathcal{M}$ and $\mathcal{N}$ be representable $\mathfrak{S}$-multicategories. Prove that a functor $F : \mathcal{M} \to \mathcal{N}$ has a right adjoint in the sense of Definition 4.2.8 if and only if (1) it preserves tensor products in the sense of Exercise 2.2.4 and (2) its underlying ordinary functor has a right adjoint.

Exercise 4.2.4. Show that an $\mathfrak{S}$-multicategory $\mathcal{M}$ has binary products, in the sense defined before Theorem 2.2.5, if and only if the diagonal $\mathcal{M} \to \mathcal{M} \times \mathcal{M}$ has a right adjoint in the sense of Definition 4.2.8.

Exercise 4.2.5. Let $\mathcal{P} : \mathcal{S}^{\mathrm{op}} \to \mathbf{Cat}$ and suppose we have a commutative square in $\mathcal{S}$:

$$\begin{array}{ccc} A & \xrightarrow{h} & C \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[k]{} & D. \end{array}$$

such that f^* and g^* have left adjoints and also h^* and k^* have right adjoints. Prove that $\mathcal{P}$ satisfies the left Beck-Chevalley condition with respect to this square if and only if it satisfies the right Beck–Chevalley condition with respect to the transposed square

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle k} \\ C & \xrightarrow[g]{} & D. \end{array}$$

Exercise 4.2.6. Let $\mathcal{M}$ and $\mathcal{N}$ be representable $\mathfrak{S}$-multicategories, and $G : \mathcal{M} \to \mathcal{N}$ a functor preserving tensor products.

(a) Show that G has a Hopf left adjoint if and only if its underlying ordinary functor has a left adjoint F such that the canonical map

$$F(A \otimes GB) \to F(GFA \otimes GB) \xrightarrow{\sim} FG(FA \otimes B) \to FA \otimes B$$

is an isomorphism for any $A \in \mathcal{N}$ and $B \in \mathcal{M}$.

(b) If $\mathcal{M}$ and $\mathcal{N}$ are additionally closed, and G is also closed in the sense that the canonical maps $G(A \multimap B) \to GA \multimap GB$ are isomorphisms, prove that g has a Hopf left adjoint if and only if its underlying ordinary functor has a left adjoint.

Exercise 4.2.7. Show that an $\mathfrak{S}$-multicategory $\mathcal{M}$ has binary coproducts, in the sense defined before Theorem 2.2.6, if and only if the diagonal $\mathcal{M} \to \mathcal{M} \times \mathcal{M}$ has a Hopf left adjoint.

Exercise 4.2.8. Let $\mathcal{M}$ and $\mathcal{N}$ be $\mathfrak{S}$-multicategories, let $G : \mathcal{M} \to \mathcal{N}$ be a functor having a Hopf left adjoint, and assume that $\mathcal{N}$ has a unit object. Prove that G also has a Hopf left adjoint at $()$.

Exercise 4.2.9. Suppose $\mathcal{P} : \mathcal{S}^{\mathrm{op}} \to \mathbf{Heyt}$ is a first-order $\mathfrak{S}$-hyperdoctrine as defined in the text.

(a) Prove (using type theory or commutative diagrams, your choice) that in fact the reindexing functor $f^* : \mathcal{P}(\Delta) \to \mathcal{P}(\Gamma)$ has a Hopf left adjoint for *all* morphisms $f : \Gamma \to \Delta$ in $\mathcal{S}$. *(Hint: in the hyperdoctrine of subsets over* **Set***, these left adjoints can be defined by* $f_!(\varphi) = \{\, y \in \Delta \mid \exists x \in \Gamma.(x \in \varphi \wedge f(x) = y) \,\}$.*)*

(b) Similarly, prove that f^* has a right adjoint for all f.

(c) Prove that these left adjoints satisfy both Beck–Chevalley conditions for commutative squares of the following form:

$$\begin{array}{ccc} A & \xrightarrow{(1,f)} & A \times B \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f\times 1} \\ B & \xrightarrow[\Delta]{} & B \times B \end{array} \qquad\qquad \begin{array}{ccc} A & \xrightarrow{\Delta} & A \times A \\ {\scriptstyle \Delta}\downarrow & & \downarrow{\scriptstyle 1\times \Delta} \\ A \times A & \xrightarrow[\Delta\times 1]{} & A \times A \times A \end{array}$$

4.3 Hyperdoctrines of subobjects

Finally, we turn to the question of where hyperdoctrines come from. From now on we will focus entirely on the cartesian monoidal case (with all the structural rules), which is the most-studied and most-applicable.

4.3.1 Horn hyperdoctrines from finite limits

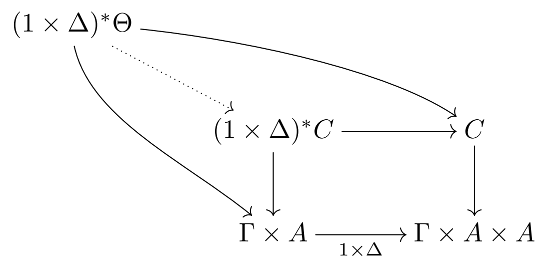
Example 4.2.5 suggests that from a category $\mathcal{S}$, we should try to construct a hyperdoctrine such that for $\Gamma \in \mathcal{S}$, $\mathcal{P}(\Gamma)$ is a poset of "subobjects" of Γ. Moreover, there is a standard way to define a subobject of Γ, namely as an isomorphism class of monomorphisms with target Γ.

We write this poset as $\mathrm{Sub}_{\mathcal{S}}(\Gamma)$, or just $\mathrm{Sub}(\Gamma)$. To make $\mathrm{Sub}_{\mathcal{S}}$ into an $\mathcal{S}$-indexed poset in a natural way, we need $\mathcal{S}$ to have pullbacks of monomorphisms along arbitrary morphisms. However, a category with finite products and pullbacks of monomorphisms automatically has all finite *limits*, since the equalizer of $f, g : A \to B$ can be constructed as the pullback of the monomorphism $\Delta : B \to B \times B$ along $(f, g) : A \to B \times B$.

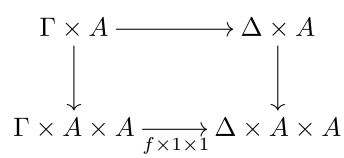
Thus, *from now on we assume that $\mathcal{S}$ has finite limits*, so that $\mathrm{Sub}_{\mathcal{S}}$ is an $\mathcal{S}$-indexed poset (which we already mentioned in Example 4.2.6). Moreover this $\mathcal{S}$-indexed poset has products (meets) and a terminal (greatest) object; the former are given by pullback of monomorphisms (which we henceforth call *intersections*) and the latter by the monomorphism $\mathsf{id}_\Gamma : \Gamma \to \Gamma$. We can also show:

Theorem 4.3.1. *If $\mathcal{S}$ has finite limits, then* $\mathrm{Sub}_{\mathcal{S}}$ *has equality. Therefore, it is a Horn hyperdoctrine.*

Proof. For any objects Γ and A, the diagonal $1 \times \Delta : \Gamma \times A \to \Gamma \times A \times A$ is itself a monomorphism, so we can regard it as a subobject of $\Gamma \times A \times A$. For this to give the desired Hopf left adjoint at $()$, we must show that for any monomorphisms $\Theta \rightarrowtail \Gamma \times A \times A$ (being the intersection of some number of subobjects) and $C \rightarrowtail \Gamma \times A \times A$, we have $\Theta \cap (\Gamma \times A) \leq C$ as subobjects of $\Gamma \times A \times A$ if and only if we have $(1 \times \Delta)^*\Theta \leq (1 \times \Delta)^*C$ as subobjects of $\Gamma \times A$. However, $\Theta \cap (\Gamma \times A)$ and $(1 \times \Delta)^*\Theta$ are the same object, and so this bijection is just using the universal property of the pullback $(1 \times \Delta)^*C$:



Finally, we have a pullback square:



which implies the Beck–Chevalley condition. ☐

Therefore, any Horn theory can be interpreted into any category with finite limits. (In fact, more than this can be done in categories with finite limits, but it is slightly tricky to characterize exactly what; we will come back to this in §4.5.)

4.3.2 Regular categories

Now we move on to regular logic. For $\mathrm{Sub}_{\mathcal{S}}$ to have existential quantifiers, we need some way to make a subobject $C \rightarrowtail \Gamma \times A$ into a subobject of Γ. In $\mathrm{Sub}_{\mathbf{Set}}$, the desired subset of Γ is the *image* of the composite function $C \rightarrowtail \Gamma \times A \to \Gamma$, so it seems natural to consider categories that have a well-behaved notion of "image factorization".

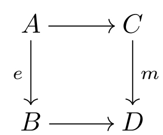
Definition 4.3.2. An **extremal epimorphism** is a morphism $e : A \to B$ in a category such that if $e = mg$ with m a monomorphism, then m is an isomorphism. A **regular category** is a category with finite limits such that every morphism f factors as me where m is a monomorphism and e an extremal epimorphism, and moreover extremal epimorphisms are stable under pullback.

We start with a lemma about extremal epimorphisms:

Lemma 4.3.3. *Let $\mathcal{S}$ be a category with finite limits.*

(a) Every extremal epimorphism is an epimorphism.

(b) If we have a commutative square in $\mathcal{S}$



*in which e is an extremal epimorphism and m a monomorphism, there exists a unique morphism $B \to C$ making both triangles commute. (This means, by definition, that e is also a **strong epimorphism**.)*

(c) If a morphism f in $\mathcal{S}$ factors as me with e an extremal epi and m a monomorphism, then such a factorization is unique up to isomorphism.

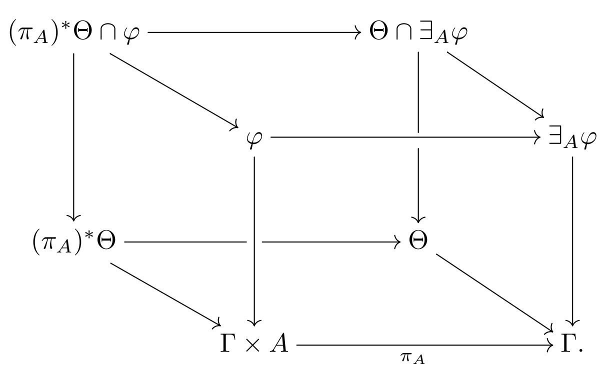
Proof. For (a), if e is extremal epi and $fe = ge$, then the equalizer of f and g is a monomorphism through which e factors; so it is an isomorphism and thus $f = g$.

For (b), the projection $B \times_D C \to B$ is a monomorphism through which e factors, so it is an isomorphism. The composite $B \to B \times_D C \to C$ is then the desired morphism; its uniqueness follows from the fact that m is mono.

For (c), two such factorizations give a squares as in (b) whose transpose is also such a square, so it has diagonal fillers in both directions, giving inverse isomorphisms. □

Theorem 4.3.4. *A category $\mathcal{S}$ with finite limits is regular if and only if* $\mathrm{Sub}_{\mathcal{S}}$ *has existential quantifiers.*

Proof. First suppose $\mathcal{S}$ is regular, and that we have a subobject $\varphi \rightarrowtail \Gamma \times A$. Factor the composite $\varphi \rightarrowtail \Gamma \times A \to \Gamma$ as an extremal epi $\varphi \to \exists_A \varphi$ followed by a mono $\exists_A \varphi \rightarrowtail \Gamma$. Then we must show that given any other monos $\Theta \rightarrowtail \Gamma$ and $\psi \rightarrowtail \Gamma$, we have $\Theta \cap \exists_A \varphi \leq \psi$ if and only if $(\pi_A)^* \Theta \cap \varphi \leq (\pi_A)^* \psi$. Now by the functoriality of pullback, we have a diagram:



Thus in one direction, if $\Theta \cap \exists_A \varphi \le \psi$, we have a composite map $(\pi_A)^* \Theta \cap \varphi \to \psi$ over Γ, which induces a map $(\pi_A)^* \Theta \cap \varphi \le (\pi_A)^* \psi$ by the universal property of pullback $(\pi_A)^*$. And in the other direction, if $(\pi_A)^* \Theta \cap \varphi \le (\pi_A)^* \psi$, we have a square

$$\begin{array}{ccccc} (\pi_A)^* \Theta \cap \varphi & \longrightarrow & (\pi_A)^* \psi & \longrightarrow & \psi \\ \downarrow & & & & \downarrow \\ \Theta \cap \exists_A \varphi & & \longrightarrow & & \Gamma \end{array}$$

as in Lemma 4.3.3(b). (The fact that the left-hand arrow is an extremal epi uses the assumption on a regular category that pullback preserves extremal epis.) Thus there is a diagonal filler giving $\Theta \cap \exists_A \varphi \le \psi$.

For the Beck–Chevalley condition, if we have $f : \Gamma \to \Delta$ and a mono $\varphi \rightarrowtail \Delta \times A$, by pasting pullback squares we see that the outer rectangle below is a pullback:

$$\begin{array}{ccc} (f \times \mathsf{id}_A)^* \varphi & \longrightarrow & \varphi \\ \downarrow & & \downarrow \\ \Gamma \times A & \xrightarrow{f \times \mathsf{id}_A} & \Delta \times A \\ \downarrow & & \downarrow \\ \Gamma & \xrightarrow[f]{} & \Delta \end{array}$$

Now if we pull back the factorization $\varphi \to \exists_A \varphi \to \Delta$ along f we get another pair of pullback squares

$$\begin{array}{ccc} (f \times \mathsf{id}_A)^* \varphi & \longrightarrow & \varphi \\ \downarrow & & \downarrow \\ f^*(\exists_A \varphi) & \xrightarrow{f \times \mathsf{id}_A} & \exists_A \varphi \\ \downarrow & & \downarrow \\ \Gamma & \xrightarrow[f]{} & \Delta. \end{array}$$

Since monos and extremal epis are both stable under pullback, the left-hand maps form a factorization of the map $(f \times \mathsf{id}_A)^* \varphi \to \Gamma$; and since such factorizations are unique by Lemma 4.3.3(c), we must have $f^*(\exists_A \varphi) \cong \exists_A ((f \times \mathsf{id}_A)^* \varphi)$, which is what the Beck–Chevalley condition requires.

Now suppose $\mathcal{S}$ has finite limits and $\mathrm{Sub}_{\mathcal{S}}$ has existential quantifiers. Given a morphism $f : A \to B$, its "graph" $(f, 1) : A \to B \times A$ is a monomorphism. Applying the existential quantifier for the projection $\pi_A : B \times A \to B$, we obtain a monomorphism $\exists (f, 1) : C \rightarrowtail B$ with the property that $\exists (f, 1) \le D$ as subobjects of B if and only if $(f, 1) \le (\pi_A)^* D$ as subobjects of $B \times A$. By the universal property of pullback, the latter is equivalent to there being a map $A \to D$ such that the composite $A \to D \to B$ is equal to the composite

$A \to B \times A \to B$; but the latter is f, so this just means that f factors through D.

In particular, since $\exists(f,1) \le \exists(f,1)$, it follows that f factors through it, by some map $e : A \to \exists(f,1)$, say. Moreover, if we have a monomorphism $D \rightarrowtail \exists(f,1)$ that e factors through, then f factors through the composite mono $D \to B$, and thus $\exists(f,1) \le D$ as subobjects of B; hence $D \cong \exists(f,1)$. Thus, e is extremal epic, and so $A \xrightarrow{e} \exists(f,1) \to B$ is a factorization of f as required in the definition of regular category.

The uniqueness of factorizations means that if f itself is extremal epic, then $\exists(f,1) \to B$ is an isomorphism. And of course, conversely, if $\exists(f,1) \to B$ is an isomorphism, then f, like e, is extremal epic. Now the Beck–Chevalley condition for existential quantifiers implies that the construction of $\exists(f,1)$ is preserved by pullback. Thus so is the property of $\exists(f,1) \to B$ being an isomorphism, and thus so is the property of f being extremal epic. Therefore, $\mathcal{S}$ is a regular category. □

Regular categories are quite common. Of course, **Set** is regular. So is any presheaf category; and, as we will see later, so is any "elementary topos". Moreover, the category of models of any finite-product theory (like monoids, groups, rings, etc.) is also regular; see Exercise 4.3.3. Thus, regular logic can be used to reason about any such category.

In fact, regular logic is quite useful in proving basic facts about regular categories. To get started, we make the following observations.

Lemma 4.3.5. *Consider the regular theory with two types A, B, one morphism $f : A \to B$, and one axiom $y : B \mid () \vdash \exists x{:}A.\, f(x) = y$. A model of this theory in a regular category $\mathcal{S}$ is precisely an extremal epimorphism in $\mathcal{S}$.*

Proof. By the proof of Theorem 4.3.4, we can construct the interpretation of $y : B \vdash (\exists x{:}A.\, f(x) = y)\ \mathsf{prop}$ as follows:

(a) Start with the diagonal $B \to B \times B$, for $y_1 : B, y_2 : B \vdash (y_1 = y_2)\ \mathsf{prop}$.

(b) Pull it back along $(f \times \mathsf{id}) : A \times B \to B \times B$, representing the substitution $x : A, y : B \vdash (f(y) = y)\ \mathsf{prop}$. This yields the graph $(f,1) : A \to A \times B$.

(c) Take the image of the composite $A \to A \times B \to B$. This composite is just f, so its image is also the image of f.

Therefore, to interpret $y : B \mid () \vdash \exists x{:}A.\, f(x) = y$ is to say that the image of f is all of B, i.e. that f is extremal epic. □

The next lemma requires only Horn logic, but there was not much point to stating it before now.

Lemma 4.3.6. *Consider the Horn theory with two types A, B, one morphism $f : A \to B$, and one axiom $x_1 : A, x_2 : A \mid (f(x_1) = f(x_2)) \vdash x_1 = x_2$. A model of this theory in a category with finite limits is precisely a monomorphism.*

Proof. The interpretation of $x_1 : A, x_2 : A \vdash (f(x_1) = f(x_2))\ \mathsf{prop}$ is the pullback of the diagonal $B \to B \times B$ along $f \times f$. This is otherwise known as the *kernel pair* of f, namely the pullback of f along itself. Thus, the axiom of our theory says pricely that this kernel pair is contained in the diagonal of A (as a subobject of $A \times A$). Now if we have $h, k : X \to A$ such that $fh = fk$, then (h, k) factors through the kernel pair; hence it also factors through the diagonal, which means $h = k$; so f is monic. □

Our third lemma starts to reveal some of the real value of the logical approach.

Lemma 4.3.7. *Suppose we have a regular theory containing two types A, B and a proposition (not necessarily an atomic one) $x : A, y : B \vdash \varphi\ \mathsf{prop}$ such that the following sequents are provable:*

$$x : A \mid () \vdash \exists y{:}B.\, \varphi \qquad x : A, y_1 : B, y_2 : B \mid \varphi[y_1/y], \varphi[y_2/y] \vdash y_1 = y_2$$

Then for any interpretation of this theory in a regular category, the interpretation of φ is a monomorphism $\varphi \rightarrowtail A \times B$ such that the composite $\varphi \to A \times B \to A$ is an isomorphism; hence the composite $A \cong \varphi \to A \times B \to B$ defines a morphism from A to B.

Proof. Let us consider what the two assumptions say. By construction of $\exists$, the first says that the image of $\varphi \to A \times B \to A$ is all of A, which is to say that this composite is extremal epi.

The second says that if we pull $\varphi \to A \times B$ back along the two projections $\pi_1, \pi_2 : A \times B \times B \rightrightarrows A \times B$, then the intersection $(\pi_1)^*\varphi \cap (\pi_2)^*\varphi$ lies inside the diagonal $\Delta : A \times B \to A \times B \times B$. We claim this means that the composite $\varphi \to A \times B \to A$ is mono. For if we have $f, g : X \to \varphi$ that are equalized in A, we have an induced map $X \to A \times B \times B$ that factors through $(\pi_1)^*\varphi \cap (\pi_2)^*\varphi$. Hence it also factors through Δ, which is to say that the two composites $X \rightrightarrows \varphi \to A \times B$ are equal; but since $\varphi \rightarrowtail A \times B$ is mono, this implies $f = g$.

Thus, $\varphi \to A \times B \to A$ is both extremal epi and mono. But since it factors through itself, this implies it is an isomorphism. □

We leave the proof of the final lemma to the reader (Exercise 4.3.8).

Lemma 4.3.8.

(a) *If in Lemma 4.3.7 the proposition φ is $f(x) = y$ for some morphism $f : A \to B$ in the theory, then the morphism $A \to B$ defined by Lemma 4.3.7 is just the interpretation of f.*

(b) *If in a regular theory we have three types A, B, C and propositions*

$$x : A, y : B \vdash \varphi\ \mathsf{prop} \qquad y : B, z : C \vdash \psi\ \mathsf{prop} \qquad x : A, z : C \vdash \chi\ \mathsf{prop}$$

all satisfying the hypotheses of Lemma 4.3.7, and moreover we can prove

$$x : A, y : B, z : C \mid \varphi, \psi \vdash \chi$$

then under interpretation in any regular category, the induced morphisms $A \to B$ and $B \to C$ compose to the induced morphism $A \to C$. □

$$\dfrac{(\text{axiom}) \qquad \dfrac{g(x) : C \qquad \dfrac{\dfrac{\dfrac{}{y : B, x : A \mid (f(x) = y) \vdash (f(x) = y)} \qquad \dfrac{}{y : B, x : A \mid (f(x) = y) \vdash (g(x) = g(x))}\,{=}I}{y : B, x : A \mid (f(x) = y) \vdash (f(x) = y) \wedge (g(x) = g(x))}\,\wedge I}{y : B, x : A \mid (f(x) = y) \vdash \exists x{:}A.\, ((f(x) = y) \wedge (g(x) = g(x)))}\,\exists I}{y : B, x : A \mid (f(x) = y) \vdash \exists z{:}C.\, \exists x{:}A.\, ((f(x) = y) \wedge (g(x) = z))}\,\exists I}{y : B \mid () \vdash \exists z{:}C.\, \exists x{:}A.\, ((f(x) = y) \wedge (g(x) = z))}\,\exists E$$

Figure 4.3: Derivation tree of (a) in proof of Theorem 4.3.9

Putting all these lemmas together, we can prove a nontrivial theorem about regular categories.

Theorem 4.3.9. *In a regular category, every extremal epi is in fact a regular epi (the coequalizer of some parallel pair).*

Proof. We will show that every extremal epi is the coequalizer of its kernel pair. Note that since an extremal epi is epi by Lemma 4.3.3(a), factorizations through it are unique if they exist. Now, given $f : A \to B$ and $g : A \to C$, we can say that g coequalizes the kernel pair of f if and only if the kernel pair of f is contained in the kernel pair of g as a subobject of $A \times A$.

Thus, consider the regular theory with three types A, B, C, two morphisms $f : A \to B$ and $g : A \to C$, and the axioms

$$y : B \mid () \vdash \exists x{:}A.\, f(x) = y \qquad x_1 : A, x_2 : A \mid (f(x_1) = f(x_2)) \vdash (g(x_1) = g(x_2))$$

The first says exactly that f is extremal epi, while the second says that the kernel pair of f is contained in the kernel pair of g. In this theory, define φ to be the proposition

$$y : B, z : C \vdash \exists x{:}A.\, ((f(x) = y) \wedge (g(x) = z)) \ \mathsf{prop}$$

We will prove the following sequents in this theory:

(a) $y : B \mid () \vdash \exists z{:}C.\, \varphi$

(b) $y : B, z_1 : C, z_2 : C \mid \varphi[z_1/z], \varphi[z_2/z] \vdash z_1 = z_2$

(c) $x : A, y : B, z : C \mid \varphi, (f(x) = y) \vdash (g(x) = z)$

Then by Lemmas 4.3.7 and 4.3.8, the interpretation of φ will define a morphism $B \to C$ that factors g through f.

(a) Informally, suppose $y : B$. By one of our axioms, there exists an $x : A$ such that $f(x) = y$. Let $z = g(x)$; then of course $f(x) = y$ and $g(x) = z$.

    A corresponding derivation tree is shown (with some parts abbreviated) in Figure 4.3. The derivation trees of the next two would be even harder to fit on a page, but there is nothing tricky about translating the informal proofs into derivations. Thus we leave it to the reader, with some hints about which rules are being used.

(b) Suppose we have $y : B$ and $z_1, z_2 : C$, and assume $\varphi[z_1/z]$ and $\varphi[z_2/z]$, that is to say $\exists x{:}A.\,(f(x) = y \wedge g(x) = z_1)$ and $\exists x{:}A.\,(f(x) = y \wedge g(x) = z_2)$. Let $x_1, x_2 : A$ be such (using $\exists E$), so that $f(x_1) = y$ and $g(x_1) = z_1$, while $f(x_2) = y$ and $g(x_2) = z_2$. Then $f(x_1) = f(x_2)$ (using transitivity of equality), so by our other axiom, $g(x_1) = g(x_2)$; hence (using transitivity of equality again) $z_1 = z_2$.

(c) Suppose we have $x : A$ and $y : B$ and $z : C$, and that $f(x) = y$ and φ, i.e. $\exists x{:}A.\,(f(x) = y \wedge g(x) = z)$. Let $x' : A$ be such an element (using $\exists E$), so that $f(x') = y$ and $g(x') = z$. Then $f(x) = f(x')$ (by transitivity), so by our second axiom, $g(x) = g(x')$, and therefore (by transitivity) $g(x) = z$. □

As always, this logical proof can be "compiled out" to a proof using commutative diagrams; see for instance [**?**, A1.3.4]. However, I find the logical proof much easier to understand.

4.3.3 Coherent categories

For coherent logic, there are few surprises.

Definition 4.3.10. A **coherent category** is a regular category in which the posets $\mathrm{Sub}(\Gamma)$ have finite unions that are preserved by pullback.

Theorem 4.3.11. *A regular category $\mathcal{S}$ is coherent if and only if $\mathrm{Sub}_{\mathcal{S}}$ is a coherent hyperdoctrine.*

Proof. If $\mathrm{Sub}_{\mathcal{S}}$ is a coherent hyperdoctrine, then clearly its joins are unions in the subobject posets of $\mathcal{S}$, and the Beck–Chevalley condition implies these are stable under pullback. The converse is just as easy except for the presence of an additional context Θ in the rule for $\vee$ (and similarly $\bot$, but we leave that case to the reader): we must show that if $\Theta \cap \varphi \le \chi$ and $\Theta \cap \psi \le \chi$ in $\mathrm{Sub}(\Gamma)$, then $\Theta \cap (\varphi \cup \psi) \le \chi$. But $\Theta \cap \varphi \le \chi$ in $\mathrm{Sub}(\Gamma)$ is equivalent to $m^*\varphi \le m^*\chi$ in $\mathrm{Sub}(\Theta)$, where $m : \Theta \rightarrowtail \Gamma$ is the given monomorphism, and similarly for the other conditions; so this also follows from pullback-stability of unions. □

In particular, in a coherent category, every object has a smallest subobject $0_A \rightarrowtail A$. It is not obvious, but true, that for any object A, the domain 0_A of this smallest subobject is an initial object (and hence isomorphic to 0_B for any other B). For this purpose we need an additional lemma.

First note that although Lemmas 4.3.5 to 4.3.8 in §4.3.2 were stated for regular theories, they are in fact valid for theories in any fragment of logic *containing* regular logic, and for any category $\mathcal{S}$ such that $\mathrm{Sub}_{\mathcal{S}}$ interprets that fragment of logic. In particular, they are valid for coherent theories and coherent categories. Second, we need the following further enhancement of Lemma 4.3.7, whose proof we leave to the reader (Exercise 4.3.11).

Lemma 4.3.12. *Suppose we have a theory in a logic containing regular logic containing two types A, B and propositions*

$$x : A \vdash \alpha\ \mathsf{prop} \qquad y : B \vdash \beta\ \mathsf{prop} \qquad x : A, y : B \vdash \varphi\ \mathsf{prop}$$

such that the following sequents are provable:

$$x : A, y : B \mid \varphi \vdash \alpha \qquad x : A, y : B \mid \varphi \vdash \beta \qquad x : A \mid \alpha \vdash \exists y{:}B.\, (\beta \wedge \varphi)$$

$$x : A, y_1 : B, y_2 : B \mid \alpha, \beta[y_1/y], \beta[y_2/y], \varphi[y_1/y], \varphi[y_2/y] \vdash y_1 = y_2$$

Then for any interpretation of this theory in a category $\mathcal{S}$ such that $\mathrm{Sub}_{\mathcal{S}}$ models the appropriate logic, the interpretation of φ yields (as in Lemma 4.3.7) a morphism from the interpretation of α to the interpretation of β. □

Theorem 4.3.13. *If $0_A \rightarrowtail A$ is the smallest subobject of A in a coherent category, then 0_A is an initial object.*

Proof. Let B be any other object, and consider the coherent theory with two types A and B and nothing else. In this theory, let $\alpha = \bot$, $\beta = \top$, and $\varphi = \bot$. Then all the sequents Lemma 4.3.12 have a $\bot$ in their proposition context, hence follow immediately from $\bot E$. Since 0_A is the interpretation of α and B is the interpretation of β, we get a morphism $0_A \to B$.

To show that it is unique, consider the theory with two objects Z and B and two morphisms $f, g : Z \to B$, and the axiom $z : Z \mid () \vdash \bot$. This is modeled by any two parallel morphisms in a coherent category whose domain has exactly one subobject (up to isomorphism), which is the case whenever its domain is the smallest subobject of some other object (like 0_A). In this theory, we can prove $z : Z \mid () \vdash f(z) = g(z)$ by $\bot E$, which easily implies $f = g$. □

We leave some further basic facts about coherent categories to the reader as Exercises 4.3.12 and 4.3.13.

4.3.4 Heyting categories

Finally, we add the rest of the structure of first-order logic: universal quantification and implication.

Definition 4.3.14. A **Heyting category** is a coherent category $\mathcal{S}$ such that for every $f : \Gamma \to \Delta$ in $\mathcal{S}$, the pullback functor $f^* : \mathrm{Sub}(\Delta) \to \mathrm{Sub}(\Gamma)$ has a right adjoint.

Theorem 4.3.15. *A coherent category $\mathcal{S}$ is a Heyting category if and only if $\mathrm{Sub}_{\mathcal{S}}$ is a first-order hyperdoctrine.*

Proof. Since $\mathcal{S}$ is assumed to be coherent and in particular regular, by Exercise 4.3.7 the *left* adjoints of f^* satisfy the Beck–Chevalley condition for pullbacks of projections. Thus, by Exercise 4.2.5, if these functors also have right adjoints, they automatically satisfy the Beck–Chevalley condition for all pullback squares as well, and in particular for pullbacks of projections. Thus, if $\mathcal{S}$ is Heyting, then $\mathrm{Sub}_{\mathcal{S}}$ has universal quantifiers. For implication, we note that the Heyting exponential $A \to B$ in $\mathrm{Sub}(\Gamma)$ can equivalently be constructed by first pulling back from $\mathrm{Sub}(\Gamma)$ to $\mathrm{Sub}(A)$, then applying the right adjoint to this pullback; this operation is stable under pullback by the same Beck–Chevalley condition.

Conversely, suppose $\mathrm{Sub}_{\mathcal{S}}$ is a first-order hyperdoctrine, and in particular has universal quantifiers. We consider two first-order theories, both with two types A, B, a morphism $f : A \to B$, and atomic propositions $x : A \vdash P(x)$ prop and $y : B \vdash Q(y)$ prop.

(a) Our first theory adds to this the axiom $x : A \mid Q(f(x)) \vdash P(x)$. We will show from this that $y : B \mid Q(y) \vdash (\forall x{:}A.\,(f(x) = y) \Rightarrow P(x))$. By the rules for $\forall$ and $\Rightarrow$, it suffices to derive $x : A, y : B \mid Q(y), (f(x) = y) \vdash P(x)$. But then applying the rule for equality, it suffices to derive $x : A \mid Q(f(x)) \vdash P(x)$, which was an axiom.

(b) Our second theory instead takes $y : B \mid Q(y) \dashv\vdash (\forall x{:}A.\,(f(x) = y) \Rightarrow P(x))$ as an axiom. Applying the rules for $\forall$ and $\Rightarrow$ in the other direction, we get $x : A, y : B \mid Q(y), (f(x) = y) \vdash P(x)$. Substituting $f(x)$ for y in this, we get $x : A \mid Q(f(x)), (f(x) = f(x)) \vdash P(x)$; but since $f(x) = f(x)$ is true by reflexivity we can cut to get $x : A \mid Q(f(x)) \vdash P(x)$.

Thus, if we define $\forall f(P)$ by $y : B \vdash (\forall x{:}A.\,(f(x) = y) \Rightarrow P(x))$ prop, we see that it has the correct universal property to be a right adjoint of pullback f^* (the latter given by substitution of $f(x)$ for y). □

Heyting categories are thus in some sense the most natural categorical home for first-order logic. One origin of Heyting categories is explored in Exercises 4.3.16 and 4.3.17: any locally cartesian closed category with finite colimits is a Heyting category, including all elementary toposes and quasitoposes (which, as we will see in chapter 5, also model *higher*-order logic). This book is not about the category theory of (quasi)toposes, but we encourage the reader to learn more about them; some good sources are [?, ?, ?, ?]. In particular [?] includes a comprehensive discussion of regular, coherent, and Heyting categories from a purely category-theoretic viewpoint.

Exercises

Exercise 4.3.1. Suppose given a finite-product theory in the sense of §2.9.1. Then we can make it into a Horn theory by replacing its $\equiv$ axioms with $=$ axioms (with empty proposition context). If $\mathcal{S}$ is any category with finite limits, prove that models of the original finite-product theory in $\mathcal{S}$ correspond bijectively to models of the Horn theory in $\mathrm{Sub}_{\mathcal{S}}$. In this sense, first-order logic subsumes finite-product logic; see Exercise 4.3.6 for a further enhancement.

Exercise 4.3.2. Prove that every regular epimorphism (in any category) is an extremal epimorphism.

Exercise 4.3.3. Prove that if $\mathcal{S}$ is a regular category and $\mathcal{T}$ is any finite-product theory (see §2.9.1), then the category of $\mathcal{T}$-models in $\mathcal{S}$ is also regular.

Exercise 4.3.4. Give an example of a regular category that is not a coherent category.

Exercise 4.3.5. A (**unique** or **orthogonal**) **factorization system** on a category $\mathcal{S}$ is a pair $(\mathcal{E}, \mathcal{M})$ of classes of morphisms in $\mathcal{S}$ such that

(a) $\mathcal{E}$ and $\mathcal{M}$ are both closed under composition with isomorphisms;

(b) every morphism f in $\mathcal{S}$ factors as $f = me$ where $m \in \mathcal{M}$ and $e \in \mathcal{E}$; and

(c) if $mh = ke$ with $m \in \mathcal{M}$ and $e \in \mathcal{E}$, there exists a unique ℓ such that $m\ell = k$ and $\ell e = h$ (as in Lemma 4.3.3(b)).

A factorization system is **stable** if $\mathcal{E}$ is stable under pullback ($\mathcal{M}$ is automatically so), and **proper** if every morphism in $\mathcal{E}$ is an epimorphism and every morphism in $\mathcal{M}$ is a monomorphism.

(a) Prove that if $(\mathcal{E}, \mathcal{M})$ is a proper, stable, factorization system on a category $\mathcal{S}$ with finite limits, there is a regular hyperdoctrine $\mathrm{Sub}_{\mathcal{M}}$ where $\mathrm{Sub}_{\mathcal{M}}(\Gamma)$ is the sub-poset of $\mathrm{Sub}_{\mathcal{S}}(\Gamma)$ consisting only of monomorphisms in $\mathcal{M}$.

(b) If $\mathcal{S}$ additionally has finite coproducts that are stable under pullback, prove that $\mathrm{Sub}_{\mathcal{M}}(\Gamma)$ is a coherent hyperdoctrine.

(c) Show that both of the previous parts apply when $\mathcal{S}$ is the category of topological spaces and $\mathcal{M}$ consists of the subspace inclusions.

(d) Is there an analogue of Lemma 4.3.7 for $\mathrm{Sub}_{\mathcal{M}}$?

Exercise 4.3.6. For any category $\mathcal{S}$ with finite products, define an $\mathcal{S}$-indexed poset $\mathrm{Sieve}_{\mathcal{S}}$ by letting $\mathrm{Sieve}_{\mathcal{S}}(\Gamma)$ be the poset of sieves on Γ. (A **sieve** on an object Γ is a sub-functor of the representable functor $\mathcal{S}(-, \Gamma)$. More concretely, it is a set of morphisms with codomain Γ, closed under precomposition with arbitrary morphisms of $\mathcal{S}$.)

(a) Prove that $\mathrm{Sieve}_{\mathcal{S}}$ is always a first-order hyperdoctrine.

(b) If we make a finite-product theory into a Horn theory as in Exercise 4.3.1, prove that models of the finite-product theory in $\mathcal{S}$ correspond bijectively to models of the Horn theory in $\mathrm{Sieve}_{\mathcal{S}}$. Thus, first-order logic subsumes finite-product logic even for categories having only finite products.

Exercise 4.3.7. Prove that in a regular category, the pullback functor $f^* : \mathrm{Sub}(\Delta) \to \mathrm{Sub}(\Gamma)$ has a left adjoint for *every* morphism $f : \Gamma \to \Delta$, and that these left adjoints satisfy the Beck–Chevalley condition with respect to every pullback square in $\mathcal{S}$.

Exercise 4.3.8. Prove Lemma 4.3.8.

Exercise 4.3.9. Write out derivation trees for statements (b) and (c) in the proof of Theorem 4.3.9. Feel free to use transitivity of equality as a (derivable) rule, rather than writing it out explicitly in terms of $=R$.

Exercise 4.3.10. Rewrite the proof of the "if" direction of Theorem 4.3.4 using regular logic rather than category theory.

Exercise 4.3.11. Prove Lemma 4.3.12.

Exercise 4.3.12. Prove using coherent logic that in a coherent category, any morphism whose codomain is initial is an isomorphism.

Exercise 4.3.13. Prove using coherent logic that if we have monomorphisms $A \rightarrowtail C$ and $B \rightarrowtail C$ in a coherent category, then the square

$$\begin{array}{ccc} A \cap B & \longrightarrow & B \\ \downarrow & & \downarrow \\ A & \longrightarrow & A \cup B \end{array}$$

is a pushout as well as a pullback. Conclude that if two objects of a coherent category can be embedded as disjoint subobjects of some third object, then they have a coproduct. (A coherent category in which this is true for any two objects is called *positive* or *extensive*.)

Exercise 4.3.14. Let $\mathcal{D}$ be a distributive lattice that is not a complete lattice, and let $\mathcal{S}$ be its free coproduct completion; the elements of $\mathcal{S}$ are set-indexed families $\{a_i\}_{i\in I}$ of elements of $\mathcal{D}$, and the morphisms $\{a_i\}_{i\in I} \to \{b_j\}_{j\in J}$ are functions $f : I \to J$ such that $a_i \le b_{f(i)}$ for all i. Prove that $\mathrm{Sub}_{\mathcal{S}}$ has the structure to model the type operations $\wedge, \top, \vee, \bot$, but not $\exists$.

Exercise 4.3.15. Give an example of a coherent category that is not a Heyting category.

Exercise 4.3.16. Suppose $\mathcal{S}$ is a category with finite limits. Prove:

(a) If $\mathcal{S}$ has coequalizers that are stable under pullback, then it is a regular category.

(b) If $\mathcal{S}$ has all finite colimits that are stable under pullback, then it is a coherent category.

(c) If $\mathcal{S}$ has all finite colimits and is locally cartesian closed, then it is a Heyting category.

Exercise 4.3.17.

(a) Show that the category of presheaves on any small category has finite limits and colimits and is locally cartesian closed, hence is a Heyting category.

(b) Show that if $\mathcal{S}$ is locally cartesian closed with finite colimits, and $\mathcal{T}$ is a reflective subcategory of $\mathcal{S}$ whose reflector preserves finite limits, then $\mathcal{T}$ is also locally cartesian closed with finite colimits.

The categories obtained by applying (b) to (a) are called **Grothendieck topoi**.

4.4 Comprehension

In Theorem 4.2.17 we showed that the syntax of first-order logic presents free hyperdoctrines, and in §4.3 that certain kinds of categories give rise to hyperdoctrines of subobjects. This is sufficient to enable us to interpret first-order logic into these kinds of categories.

However, for some purposes one would like to be able to present a free *category* using logic (e.g. the free coherent category generated by a coherent theory), rather than just a free hyperdoctrine. This is analogous to how the syntax of a finite-product theory directly presents a free cartesian multicategory, but for some purposes we would actually like it to generate a free category with products. (Hyperdoctrines can be thought of as a sort of "multicategory-like" version of a category with well-behaved subobjects.)

In the case of finite-product theories, we had two solutions to this problem: we could include a product type operation in the type theory, or we could consider the category of contexts of the type theory rather than the multicategory of types. Let us consider first an analogy of the latter construction. The most directly analogous thing would be to start with a cartesian multicategory $\mathbf{S}$ and a presheaf $\mathcal{P}$ on $\mathbf{S}$ valued in the category of cartesian multiposets, as considered briefly at the beginning of §4.2. We would then build a category whose objects compare to the most general sort of "context" that appears first-order logic: pairs $(\Gamma \mid \Theta)$, where Γ is a list of objects of $\mathcal{S}$ and Θ is a list of objects of $\mathcal{P}(\Gamma)$.

However, we will simplify this in a couple of ways. Firstly, we continue to consider instead ordinary presheaves on categories $\mathcal{S}$ with finite products. Secondly, as we saw in §4.3.1, if a category has an indexed poset of subobjects then the latter is automatically a Horn hyperdoctrine, and in particular has finite meets. Thus, we can similarly replace Θ by a single object of $\mathcal{P}(A)$.

In the end, therefore, we will start with a category $\mathcal{S}$ with products and a $\mathcal{S}$-indexed poset $\mathcal{P}$ that is at least a Horn hyperdoctrine, and construct a category whose objects are pairs $(A \mid \alpha)$ where A is an object of $\mathcal{S}$ and α is an object of $\mathcal{P}(A)$. But what should the morphisms be? A first guess might be that a morphism from $(A \mid \alpha)$ to $(B \mid \beta)$ would be a morphism $f : A \to B$ such that $\alpha \le f^*\beta$. However, this would not give enough morphisms.

In particular, by Lemma 4.3.7 we know that if $\mathcal{P}$ is a regular hyperdoctrine, then any proposition φ satisfying certain conditions gives rise to a morphism in any regular category that $\mathcal{P}$ maps into. But our guess in the previous paragraph would not produce such a morphism in general, so it cannot produce a regular category that $\mathcal{P}$ maps into.

The solution is to take Lemma 4.3.7, or more precisely Lemma 4.3.12, as a *definition* of the morphisms. That is, we *define* a morphism from $(A \mid \alpha)$ to $(B \mid \beta)$ to be an object $\varphi \in \mathcal{P}(A \times B)$ such that the sequents in Lemma 4.3.12 are valid in $\mathcal{P}$. More precisely, we define a morphism to be a model in $\mathcal{P}$ of the regular theory that has two types A, B, three atomic propositions α, β, φ in contexts A and B and (A, B) respectively, and four axioms

$$x : A, y : B \mid \varphi(x,y) \vdash \alpha(x) \qquad\qquad x : A, y : B \mid \varphi(x,y) \vdash \beta(y)$$

$$x : A \mid \alpha(x) \vdash \exists y{:}B.\, \varphi(x,y)$$

$$x : A, y_1 : B, y_2 : B \mid \varphi(x,y_1), \varphi(x,y_2) \vdash (y_1 = y_2)$$

We will call this the **theory of maps**. This definition can, of course, be reinterpreted in terms of the categorical structure of $\mathcal{P}$, but we will have no need to do that.

Note that this definition only makes sense if $\mathcal{P}$ is at least a *regular* hyperdoctrine, with existential quantifiers. We will come back to this point in §4.5; for the moment, we simply assume that $\mathcal{P}$ is at least regular.

Theorem 4.4.1. *If $\mathcal{P}$ is a regular hyperdoctrine, then the above definition yields a category* $\mathrm{Map}(\mathcal{P})$*.*

Proof. TODO: Formulate in terms of co-categories.

We define the composite of $\varphi : (A \mid \alpha) \to (B \mid \beta)$ and $\psi : (B \mid \beta) \to (C \mid \gamma)$ to be

$$x : A, z : C \vdash (\exists y{:}B.\, (\varphi(x, y) \wedge \psi(y, z))) \;\mathsf{prop}$$

We show that this defines another morphism by the following informal arguments, translated into formal regular logic. For instance, here are the proofs that the composite satisfies the axioms to be a morphism from $(A \mid \alpha)$ to $(C \mid \gamma)$.

- If $\exists y{:}B.\, (\varphi(x, y) \wedge \psi(y, z))$, then we have a $y : B$ such that $\varphi(x, y)$, and so (since φ satisfies the axioms) $\alpha(x)$.
- Similarly, if $\exists y{:}B.\, (\varphi(x, y) \wedge \psi(y, z))$, then we have a $y : B$ such that $\psi(y, z)$, and so (since ψ satisfies the axioms) $\gamma(z)$.
- If $\alpha(x)$, then since φ satisfies the axioms, there exists (and so we may assume we have) a $y : B$ such that $\varphi(x, y)$. Moreover, since $\varphi(x, y)$, we have $\beta(y)$. Now since ψ also satisfies the axioms, there exists (and so we may assume we have) a $z : C$ such that $\psi(y, z)$. Using y, we conclude $\exists y{:}B.\, (\varphi(x, y) \wedge \psi(y, z))$, and therefore $\exists z{:}C.\, \exists y{:}B.\, (\varphi(x, y) \wedge \psi(y, z))$.
- Suppose we have $x : A$ and $z_1 : C$ and $z_2 : C$ such that $\exists y{:}B.\, (\varphi(x, y) \wedge \psi(y, z_1))$ and $\exists y{:}B.\, (\varphi(x, y) \wedge \psi(y, z_2))$. By using these latter two hypotheses, we may assume we have $y_1 : B$ and $y_2 : B$ such that $\varphi(x, y_1)$ and $\psi(y_1, z_1)$, and also $\varphi(x, y_2)$ and $\psi(y_2, z_2)$. Now since φ satisfies the axioms, we have $y_1 = y_2$. Using this equality, we can change our assumption of $\psi(y_2, z_2)$ to $\psi(y_1, z_2)$. Therefore, since also $\psi(y_1, z_2)$ and ψ satisfies the axioms, we have $z_1 = z_2$.

More formally, what we are doing is considering a theory with three objects A, B, C, five atomic propositions $\alpha, \beta, \gamma, \varphi, \psi$, and the axioms making φ and ψ both morphisms of $\mathrm{Map}(\mathcal{P})$. This could be described as the pushout, in the category of theories, of two copies of the theory of maps over the *theory of objects* (which has one object A and one atomic proposition $x : A \vdash \alpha(x)$ prop). We have just shown that the free hyperdoctrine generated by this theory contains *another* model of the theory of maps. Therefore, by the universal property of free hyperdoctrines, any composable morphisms of $\mathrm{Map}(\mathcal{P})$ give rise to a third one, which we define to be their composite.

Similarly, we define the identity morphism of $(A \mid \alpha)$ is $x : A, y : A \vdash (\alpha(x) \wedge (x = y))$ prop. Proving that this is a morphism gives a morphism from the theory of maps to the free hyperdoctrine generated by the theory of objects.

We can similarly prove that composition is associative and unital; we leave the details to the reader (Exercise 4.4.1). □

To show that $\mathrm{Map}(\mathcal{P})$ has products, we need the following logical characterization of products.

Lemma 4.4.2. *Consider the Horn theory with two objects A, B, and P, three morphisms $\pi_1 : P \to A$, $\pi_2 : P \to B$, and $p : (A, B) \to P$, and axioms*

$$x : A, y : B \mid () \vdash \pi_1(p(x, y)) = x \qquad x : A, y : B \mid () \vdash \pi_2(p(x, y)) = y$$

$$z : P \mid () \vdash p(\pi_1(z), \pi_2(z)) = z$$

For any category $\mathcal{S}$ with finite limits, a model of this theory in $\mathrm{Sub}_{\mathcal{S}}$ is precisely a binary product diagram.

Proof. Under interpretation in $\mathcal{S}$, p becomes a morphism $A \times B \to P$, while π_1 and π_2 pair up to yield a morphism $P \to A \times B$. The axioms ensure that the composites of these morphisms in both directions are the identity. (Note that this theory is obtained from a finite-product theory as in Exercise 4.3.1; also compare to Theorem 2.6.7.) □

Similarly, we have:

Lemma 4.4.3. *Consider the Horn theory with one object T, one morphism $t : () \to T$, and one axiom*

$$x : T \mid () \vdash x = t$$

Then for any category $\mathcal{S}$ with finite limits, a model of this theory in $\mathrm{Sub}_{\mathcal{S}}$ is precisely a terminal object of $\mathcal{S}$. □

Theorem 4.4.4. *If $\mathcal{P}$ is a regular hyperdoctrine, then $\mathrm{Map}(\mathcal{P})$ (as defined in Theorem 4.4.1) has finite limits.*

TODO. The product of $(A \mid \alpha)$ and $(B \mid \beta)$ is $(A \times B \mid \alpha \times \beta)$, where "$\alpha \times \beta$" denotes the proposition

$$z : A \times B \vdash (\alpha(\pi_1(z)) \wedge \beta(\pi_2(z))) \;\mathsf{prop}$$

The terminal object is $(\mathbb{1} \mid \top)$. □

Exercises

Exercise 4.4.1. Fill in the details of Theorem 4.4.1.

4.5 Finite-limit theories

4.6 Indexed monoidal categories

Collected Exercises

For convenient reference, we collect the exercises from all sections in this chapter.

Exercise 4.1.1. Assuming we have $\multimap$, show that the rule $=R$ is derivable (recall Remark 1.2.6) from the following simpler rule with no proposition context Θ:

$$\frac{\Gamma, x : A, y : A \vdash \varphi \ \mathsf{prop} \qquad \begin{array}{c}\Gamma \vdash M : A \qquad \Gamma \vdash N : A \\ \Gamma \vdash \Theta \ \mathsf{ctx}\end{array} \qquad \Gamma, x : A \mid \Theta \vdash \varphi[x/y]}{\Gamma \mid \Theta, (M =_A N) \vdash \varphi[M/x, N/y]}$$

Exercise 4.1.2. Three of the following four sequents are derivable in intuitionistic first-order logic (for any type A, context Γ, and proposition $\Gamma, x : A \vdash \varphi \ \mathsf{prop}$); derive them.

$$\Gamma \mid \exists x{:}A.\, \neg\varphi \vdash \neg\forall x{:}A.\, \varphi$$
$$\Gamma \mid \forall x{:}A.\, \neg\varphi \vdash \neg\exists x{:}A.\, \varphi$$
$$\Gamma \mid \neg\forall x{:}A.\, \varphi \vdash \exists x{:}A.\, \neg\varphi$$
$$\Gamma \mid \neg\exists x{:}A.\, \varphi \vdash \forall x{:}A.\, \neg\varphi$$

Exercise 4.1.3. In a first-order theory with three types A, B, C, two generating arrows $f : A \to B$ and $g : B \to A$, one atomic proposition P with domain (A, B), and no axioms, derive the following judgments:

(a) $x_1 : A, x_2 : A, y : B \mid \varphi(x_1, y), (x_1 =_A x_2) \vdash \varphi(x_2, y)$

(b) $x_1 : A, x_2 : A \mid (x_1 =_A x_2) \vdash f(x_1) =_B f(x_2)$

(c) $() \mid (\forall x{:}A.\, g(f(x)) =_A x) \vdash \forall x_1{:}A.\, \forall x_2{:}A.\, ((f(x_1) =_B f(x_2)) \to (x_1 =_A x_2))$

Exercise 4.1.4. Write down a first-order theory for each of the following structures. If you can, formulate them so that they fit inside the specified fragment.

(a) Partially ordered sets (Horn)

(b) Totally ordered sets (coherent)

(c) Fields (coherent)

(d) Categories (regular)

Exercise 4.1.5. Prove that in intuitionistic first-order logic with $\exists E$ and $=E$ replaced by $\exists E'$ and $=E'$ as mentioned at the end of the section, the structural rules of exchange, weakening, and contraction for proposition contexts are admissible.

Exercise 4.2.1. Prove Lemma 4.2.2.

Exercise 4.2.2. Suppose a functor $F : \mathcal{M} \to \mathcal{N}$ of $\mathfrak{S}$-multicategories has a right adjoint in the sense of Definition 4.2.8.

(a) Prove that if $\mathcal{M}$ and $\mathcal{N}$ are both representable, then F preserves tensor products in the sense of Exercise 2.2.4.

(b) Extend G to a functor $G : \mathcal{N} \to \mathcal{M}$.

(c) Define a 2-category of $\mathfrak{S}$-multicategories and show that G is right adjoint to F in this 2-category (i.e. there are 2-cells $\eta : 1 \to GF$ and $\varepsilon : FG \to 1$ in this 2-category satisfying the triangle identities).

Exercise 4.2.3. Let $\mathcal{M}$ and $\mathcal{N}$ be representable $\mathfrak{S}$-multicategories. Prove that a functor $F : \mathcal{M} \to \mathcal{N}$ has a right adjoint in the sense of Definition 4.2.8 if and only if (1) it preserves tensor products in the sense of Exercise 2.2.4 and (2) its underlying ordinary functor has a right adjoint.

Exercise 4.2.4. Show that an $\mathfrak{S}$-multicategory $\mathcal{M}$ has binary products, in the sense defined before Theorem 2.2.5, if and only if the diagonal $\mathcal{M} \to \mathcal{M} \times \mathcal{M}$ has a right adjoint in the sense of Definition 4.2.8.

Exercise 4.2.5. Let $\mathcal{P} : \mathcal{S}^{\mathrm{op}} \to \mathbf{Cat}$ and suppose we have a commutative square in $\mathcal{S}$:

$$\begin{array}{ccc} A & \xrightarrow{h} & C \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[k]{} & D. \end{array}$$

such that f^* and g^* have left adjoints and also h^* and k^* have right adjoints. Prove that $\mathcal{P}$ satisfies the left Beck-Chevalley condition with respect to this square if and only if it satisfies the right Beck–Chevalley condition with respect to the transposed square

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle k} \\ C & \xrightarrow[g]{} & D. \end{array}$$

Exercise 4.2.6. Let $\mathcal{M}$ and $\mathcal{N}$ be representable $\mathfrak{S}$-multicategories, and $G : \mathcal{M} \to \mathcal{N}$ a functor preserving tensor products.

(a) Show that G has a Hopf left adjoint if and only if its underlying ordinary functor has a left adjoint F such that the canonical map

$$F(A \otimes GB) \to F(GFA \otimes GB) \xrightarrow{\sim} FG(FA \otimes B) \to FA \otimes B$$

is an isomorphism for any $A \in \mathcal{N}$ and $B \in \mathcal{M}$.

(b) If $\mathcal{M}$ and $\mathcal{N}$ are additionally closed, and G is also closed in the sense that the canonical maps $G(A \multimap B) \to GA \multimap GB$ are isomorphisms, prove that g has a Hopf left adjoint if and only if its underlying ordinary functor has a left adjoint.

Exercise 4.2.7. Show that an $\mathfrak{S}$-multicategory $\mathcal{M}$ has binary coproducts, in the sense defined before Theorem 2.2.6, if and only if the diagonal $\mathcal{M} \to \mathcal{M} \times \mathcal{M}$ has a Hopf left adjoint.

Exercise 4.2.8. Let $\mathcal{M}$ and $\mathcal{N}$ be $\mathfrak{S}$-multicategories, let $G : \mathcal{M} \to \mathcal{N}$ be a functor having a Hopf left adjoint, and assume that $\mathcal{N}$ has a unit object. Prove that G also has a Hopf left adjoint at $()$.

Exercise 4.2.9. Suppose $\mathcal{P} : \mathcal{S}^{\mathrm{op}} \to \mathbf{Heyt}$ is a first-order $\mathfrak{S}$-hyperdoctrine as defined in the text.

(a) Prove (using type theory or commutative diagrams, your choice) that in fact the reindexing functor $f^* : \mathcal{P}(\Delta) \to \mathcal{P}(\Gamma)$ has a Hopf left adjoint for *all* morphisms $f : \Gamma \to \Delta$ in $\mathcal{S}$. *(Hint: in the hyperdoctrine of subsets over* **Set***, these left adjoints can be defined by* $f_!(\varphi) = \{\, y \in \Delta \mid \exists x \in \Gamma.(x \in \varphi \wedge f(x) = y) \,\}$*.)*

(b) Similarly, prove that f^* has a right adjoint for all f.

(c) Prove that these left adjoints satisfy both Beck–Chevalley conditions for commutative squares of the following form:

$$\begin{array}{ccc} A & \xrightarrow{(1,f)} & A \times B \\ {\scriptstyle f}\big\downarrow & & \big\downarrow{\scriptstyle f\times 1} \\ B & \xrightarrow[\Delta]{} & B \times B \end{array} \qquad\qquad \begin{array}{ccc} A & \xrightarrow{\Delta} & A \times A \\ {\scriptstyle \Delta}\big\downarrow & & \big\downarrow{\scriptstyle 1\times\Delta} \\ A \times A & \xrightarrow[\Delta\times 1]{} & A \times A \times A \end{array}$$

Exercise 4.3.1. Suppose given a finite-product theory in the sense of §2.9.1. Then we can make it into a Horn theory by replacing its $\equiv$ axioms with $=$ axioms (with empty proposition context). If $\mathcal{S}$ is any category with finite limits, prove that models of the original finite-product theory in $\mathcal{S}$ correspond bijectively to models of the Horn theory in $\mathrm{Sub}_{\mathcal{S}}$. In this sense, first-order logic subsumes finite-product logic; see Exercise 4.3.6 for a further enhancement.

Exercise 4.3.2. Prove that every regular epimorphism (in any category) is an extremal epimorphism.

Exercise 4.3.3. Prove that if $\mathcal{S}$ is a regular category and $\mathcal{T}$ is any finite-product theory (see §2.9.1), then the category of $\mathcal{T}$-models in $\mathcal{S}$ is also regular.

Exercise 4.3.4. Give an example of a regular category that is not a coherent category.

Exercise 4.3.5. A (**unique** or **orthogonal**) **factorization system** on a category $\mathcal{S}$ is a pair $(\mathcal{E}, \mathcal{M})$ of classes of morphisms in $\mathcal{S}$ such that

(a) $\mathcal{E}$ and $\mathcal{M}$ are both closed under composition with isomorphisms;

(b) every morphism f in $\mathcal{S}$ factors as $f = me$ where $m \in \mathcal{M}$ and $e \in \mathcal{E}$; and

(c) if $mh = ke$ with $m \in \mathcal{M}$ and $e \in \mathcal{E}$, there exists a unique ℓ such that $m\ell = k$ and $\ell e = h$ (as in Lemma 4.3.3(b)).

A factorization system is **stable** if $\mathcal{E}$ is stable under pullback ($\mathcal{M}$ is automatically so), and **proper** if every morphism in $\mathcal{E}$ is an epimorphism and every morphism in $\mathcal{M}$ is a monomorphism.

(a) Prove that if $(\mathcal{E}, \mathcal{M})$ is a proper, stable, factorization system on a category $\mathcal{S}$ with finite limits, there is a regular hyperdoctrine $\mathrm{Sub}_{\mathcal{M}}$ where $\mathrm{Sub}_{\mathcal{M}}(\Gamma)$ is the sub-poset of $\mathrm{Sub}_{\mathcal{S}}(\Gamma)$ consisting only of monomorphisms in $\mathcal{M}$.

(b) If $\mathcal{S}$ additionally has finite coproducts that are stable under pullback, prove that $\mathrm{Sub}_{\mathcal{M}}(\Gamma)$ is a coherent hyperdoctrine.

(c) Show that both of the previous parts apply when $\mathcal{S}$ is the category of topological spaces and $\mathcal{M}$ consists of the subspace inclusions.

(d) Is there an analogue of Lemma 4.3.7 for $\mathrm{Sub}_{\mathcal{M}}$?

Exercise 4.3.6. For any category $\mathcal{S}$ with finite products, define an $\mathcal{S}$-indexed poset $\mathrm{Sieve}_{\mathcal{S}}$ by letting $\mathrm{Sieve}_{\mathcal{S}}(\Gamma)$ be the poset of sieves on Γ. (A **sieve** on an object Γ is a sub-functor of the representable functor $\mathcal{S}(-, \Gamma)$. More concretely, it is a set of morphisms with codomain Γ, closed under precomposition with arbitrary morphisms of $\mathcal{S}$.)

(a) Prove that $\mathrm{Sieve}_{\mathcal{S}}$ is always a first-order hyperdoctrine.

(b) If we make a finite-product theory into a Horn theory as in Exercise 4.3.1, prove that models of the finite-product theory in $\mathcal{S}$ correspond bijectively to models of the Horn theory in $\mathrm{Sieve}_{\mathcal{S}}$. Thus, first-order logic subsumes finite-product logic even for categories having only finite products.

Exercise 4.3.7. Prove that in a regular category, the pullback functor $f^* : \mathrm{Sub}(\Delta) \to \mathrm{Sub}(\Gamma)$ has a left adjoint for *every* morphism $f : \Gamma \to \Delta$, and that these left adjoints satisfy the Beck–Chevalley condition with respect to every pullback square in $\mathcal{S}$.

Exercise 4.3.8. Prove Lemma 4.3.8.

Exercise 4.3.9. Write out derivation trees for statements (b) and (c) in the proof of Theorem 4.3.9. Feel free to use transitivity of equality as a (derivable) rule, rather than writing it out explicitly in terms of $=R$.

Exercise 4.3.10. Rewrite the proof of the "if" direction of Theorem 4.3.4 using regular logic rather than category theory.

Exercise 4.3.11. Prove Lemma 4.3.12.

Exercise 4.3.12. Prove using coherent logic that in a coherent category, any morphism whose codomain is initial is an isomorphism.

Exercise 4.3.13. Prove using coherent logic that if we have monomorphisms $A \rightarrowtail C$ and $B \rightarrowtail C$ in a coherent category, then the square

$$\begin{array}{ccc} A \cap B & \longrightarrow & B \\ \downarrow & & \downarrow \\ A & \longrightarrow & A \cup B \end{array}$$

is a pushout as well as a pullback. Conclude that if two objects of a coherent category can be embedded as disjoint subobjects of some third object, then they have a coproduct. (A coherent category in which this is true for any two objects is called *positive* or *extensive*.)

Exercise 4.3.14. Let $\mathcal{D}$ be a distributive lattice that is not a complete lattice, and let $\mathcal{S}$ be its free coproduct completion; the elements of $\mathcal{S}$ are set-indexed families $\{a_i\}_{i\in I}$ of elements of $\mathcal{D}$, and the morphisms $\{a_i\}_{i\in I} \to \{b_j\}_{j\in J}$ are functions $f : I \to J$ such that $a_i \le b_{f(i)}$ for all i. Prove that $\mathrm{Sub}_{\mathcal{S}}$ has the structure to model the type operations $\wedge, \top, \vee, \bot$, but not $\exists$.

Exercise 4.3.15. Give an example of a coherent category that is not a Heyting category.

Exercise 4.3.16. Suppose $\mathcal{S}$ is a category with finite limits. Prove:

(a) If $\mathcal{S}$ has coequalizers that are stable under pullback, then it is a regular category.

(b) If $\mathcal{S}$ has all finite colimits that are stable under pullback, then it is a coherent category.

(c) If $\mathcal{S}$ has all finite colimits and is locally cartesian closed, then it is a Heyting category.

Exercise 4.3.17.

(a) Show that the category of presheaves on any small category has finite limits and colimits and is locally cartesian closed, hence is a Heyting category.

(b) Show that if $\mathcal{S}$ is locally cartesian closed with finite colimits, and $\mathcal{T}$ is a reflective subcategory of $\mathcal{S}$ whose reflector preserves finite limits, then $\mathcal{T}$ is also locally cartesian closed with finite colimits.

The categories obtained by applying (b) to (a) are called **Grothendieck topoi**.

Exercise 4.4.1. Fill in the details of Theorem 4.4.1.

Chapter 5

Higher-order logic

Chapter 6

Dependent type theory

Appendix A

Deductive systems

The purpose of this appendix is to explain the formal apparatus underlying type theory from a mathematical perspective, giving precise meanings to words like "judgment", "rule", "derivation", and "binder". This is rarely explained in detail, yet in my experience the unfamiliar terminology is a large part of what makes type theory difficult for mathematicians to understand.

Formally speaking, this appendix should come before chapter 1. However, its technicalities seem unlikely to be appreciated without some concrete exposure to the ideas that it is trying to explain, so I have placed it as an appendix instead. I encourage the reader to skip back and forth between it and the main text as needed.

I should say that probably not all type theorists would agree with the meanings assigned herein to words like "judgment". Constructive type theory also has a philosophical/foundational aspect that I will not attempt to explain or engage with. The purpose of this appendix, like that of the entire book, is to explain type theory *only* in its role as a language for reasoning about categorical structures, without meaning thereby to disparage its other roles or regard them as unimportant.

A.1 Trees and free algebras

As remarked in §0.3, our perspective on type theory is that it presents *free categorical structures* in a particularly convenient way. Since categorical structures are particular kinds of *algebraic* structures, it seems reasonable to think first about what free algebraic structures look like in general. In this section we begin by considering "algebras without axioms".

A **signature** Σ is a set Σ_1 of **operations** with a function $\mathrm{ar} : \Sigma_1 \to \mathbb{N}$ assigning to each operation a natural number[1] called its **arity**. A Σ-**algebra** is

[1] Everything in this chapter works just as well if arities are arbitrary cardinal numbers (except that in §A.3 we would require the axiom of choice). However, for simplicity we restrict to the case of finite arities, since that is where our ultimate interest lies. On the other hand,

a set A together with, for each $m \in \Sigma_1$, a function $[m] : A^{\mathrm{ar}(m)} \to A$. There is an obvious notion of Σ-algebra morphism, forming a category.

Algebras over a signature are a very "primordial" sort of algebraic structure, with arbitrary operations but no axioms allowed. For instance, if $\Sigma_1 = \{e, m\}$ with $\mathrm{ar}(e) = 0$ and $\mathrm{ar}(m) = 2$, then Σ-algebras are *pointed magmas*: sets equipped with a basepoint and a binary operation. We will see how to add axioms in §A.3.

Free Σ-algebras are conveniently described in terms of trees. A **tree** is a set whose elements are called **nodes**, together with a binary relation between them called **edge existence** (a "relational graph") that is connected and loop-free. A tree is **rooted** if it is equipped with a chosen node called the **root**. In a rooted tree every node x has a unique (non-retracing) path to the root; if x is not the root, this path goes through a unique edge connected to x that we call **outgoing**, and the node at the other end of that edge is the **parent** of x. The non-outgoing edges connected to x are called **incoming**, and the nodes they connect it to are called its **children**. A node is a **descendant** of x if its path to the root passes through x, which is to say it is a child of a child of a. . . of x. A node x together with all its descendants forms another rooted tree with x as the root. A **branch** is a non-retracing path starting at the root; a rooted tree is **well-founded** if there are no infinite branches.

If Σ is a signature, then a **Σ-labeled tree** is a rooted tree equipped with a **labeling** function from nodes to Σ_1, along with for every node x labeled by $m \in \mathcal{O}$, a bijection from the incoming edges of x to $\{1, \ldots, \mathrm{ar}(m)\}$ (hence, in particular, that there are exactly $\mathrm{ar}(m)$ such edges). There is an obvious notion of *isomorphism* between labeled trees. We write $W\Sigma$ for the set of all isomorphism classes of *well-founded* Σ-labeled trees. (Note that $W\Sigma$ is empty unless there is at least one nullary operation.) Then $W\Sigma$ has a Σ-algebra structure defined as follows: given $m \in \Sigma_1$ and a list of trees $t_1, \ldots, t_{\mathrm{ar}(m)}$, define a tree $[m](t_1, \ldots, t_{\mathrm{ar}(m)})$ with nodes $\{\star\} \sqcup \bigsqcup_i t_i$, where $\star$ is the root, with label m, and its children are the roots of the trees t_i.

The central fact is that $W\Sigma$ is the initial Σ-algebra. We will give a classical set-theoretic proof of this for the comfort of a certain kind of reader, but readers of a different kind, or who already believe this fact, are welcome to skip the proof. (From a constructive type-theoretic point of view, $W\Sigma$ and its initiality are sometimes a fundamental axiom not reducible to sets.)

Theorem A.1.1. *Suppose $P \subseteq W\Sigma$ has the property that for any m and trees $t_1, \ldots, t_{\mathrm{ar}(m)}$ such that each $t_i \in P$, then also $[m](t_1, \ldots, t_{\mathrm{ar}(m)}) \in P$. Then $P = W\Sigma$.*

Proof. Suppose not, so there is a well-founded Σ-labeled tree not in P. Let m be the label of its root and $t_1, \ldots, t_{\mathrm{ar}(m)}$ its children; then our given tree is (isomorphic to) $[m](t_1, \ldots, t_{\mathrm{ar}(m)})$. By the contrapositive of our assumption, therefore, there must be some i such that $t_i \notin P$. Iterating, we obtain an infinite branch, contradicting well-foundedness. □

there may certainly be infinitely many operations.

Theorem A.1.2. *For any Σ-algebra A, there is a unique Σ-algebra morphism $W\Sigma \to A$.*

Proof. TODO: standard argument. □

Now that we have *initial* Σ-algebras, note that *free* Σ-algebras can be constructed by a simple modification. Given Σ and any set X, define a new signature $\Sigma[X]$ by $\Sigma[X]_1 = \Sigma_1 \sqcup X$, where $\mathrm{ar}(x) = 0$ for all $x \in X$. Then a $\Sigma[X]$-algebra is just a Σ-algebra together with a map from X into its underlying set, so the initial such algebra is exactly the free Σ-algebra on X. Thus, the forgetful functor from Σ-algebras to sets has a left adjoint.

A different way to express Theorem A.1.2 is that *given an arbitrary set A, to define a function $W\Sigma \to A$ it suffices to define a Σ-algebra structure on A*. This may seem like a trivial reformulation, but it better reflects the way we use it to describe type theories.

In yet other words, we may define a function $f : W\Sigma \to A$ by specifying $f([m](t_1, \ldots, t_{\mathrm{ar}(m)}))$ for each m, assuming recursively that $f(t_1), \ldots, f(t_{\mathrm{ar}(m)})$ have already been defined. Formally this is the same as specifying a Σ-algebra structure on A — the definition of "$f([m](t_1, \ldots, t_{\mathrm{ar}(m)}))$" given the "values of $f(t_1), \ldots, f(t_{\mathrm{ar}(m)})$" is precisely the action $[m]$ on A — but it often matches our thought processes best.

Exercises

Exercise A.1.1. Prove that a well-founded Σ-labeled tree has no nonidentity automorphisms. Thus, the passage to isomorphism classes in the definition of $W\Sigma$ is "categorically harmless".

Exercise A.1.2. Exhibit a signature Σ such that $W\Sigma \cong \mathbb{N}$ and Theorem A.1.1 reduces to ordinary mathematical induction.

A.2 Indexed trees

The signatures and algebras in §A.1 have only one underlying set, or *sort*, but sometimes algebraic structures have more than one sort. As a simple example, we could consider a set together with a group acting on that set to be a single algebraic structure; then the group and the set are two sorts.

Categories could be regarded as having two sets, namely objects and arrows; but it is generally better to treat them differently. Specifically, for a fixed set $\mathcal{O}$, we regard categories with object set $\mathcal{O}$ as an algebraic structure whose set of sorts is $\mathcal{O} \times \mathcal{O}$. Thus each hom-set is a separate sort, and each triple A, B, C gives a different binary composition operation

$$\circ_{A,B,C} : (\hom(B, C), \hom(A, B)) \to \hom(A, C)$$

This may seem a little odd, but as we will see it makes sense.

To deal with multi-sorted algebraic structures in general, we augment our signatures with a set Σ_0 of **sorts** together with, for each operation $m \in \Sigma_1$, an **output sort** $c_m \in \Sigma_0$ and also a list of **input sorts** $d_{m,1}, \ldots, d_{m,\mathrm{ar}(m)}$. For brevity we write such an operation as $m : (d_{m,1}, \ldots, d_{m,\mathrm{ar}(m)}) \to c_m$. From now on we call these *multi-sorted signatures* simply **signatures**; the simpler signatures of §A.1 we re-christen **one-sorted signatures**. (In fact, a multi-sorted signature is essentially the same as a "multigraph", Definition 2.2.1.)

For a multi-sorted signature Σ, a Σ-**algebra** is a Σ_0-indexed family of sets $\{A_i\}_{i\in\Sigma_0}$ together with for each $m \in \Sigma_1$ a function $A_{d_{m,1}} \times \cdots \times A_{d_{m,\mathrm{ar}(m)}} \to A_{c_m}$. For instance, if $\Sigma_0 = \{g, s\}$ and $\Sigma_1 = \{m, t\}$ with $m : (g, g) \to g$ and $t : (g, s) \to s$, then an indexed algebra consists of two sets A_g and A_s, a binary operation on A_g, and an action of A_g on A_s.

Similarly, we define a Σ-**labeled tree** as before, with the additional requirement that if x is the k^{th} child of y, and x is labeled by $m \in \Sigma_1$ while y is labeled by $p \in \Sigma_1$, then $c_m = d_{p,k}$. For each $i \in \Sigma_0$, let $W\Sigma_i$ be the set of isomorphism classes of Σ-labeled trees for which the output sort of the root is i. Then $\{W\Sigma_i\}_{i\in\Sigma_0}$ has a similar tautological Σ-algebra structure, and is the initial one.

Exercises

Exercise A.2.1. Prove that $\{W\Sigma_i\}_{i\in\Sigma_0}$ is the initial Σ-algebra.

A.3 Free algebras with axioms

Of course, most algebraic structures of interest contain axioms as well as operations; for instance, multiplication in a group or monoid must be associative and unital. The free monoid on a set X is naturally regarded as a quotient of the free pointed magma on X that forces associativity and unitality to hold. It turns out that we can construct free algebras of this sort quite generally by defining an equivalence relation *as another indexed free algebra*.

Making this completely precise in general is a bit technical, so we will begin with a concrete example. Suppose we want to generate the free semigroup on a set X. Let $\mathfrak{F}_{\mathbf{Mag}}X$ denote the free magma on X, constructed as in §A.1. (A magma is a set with a single binary operation; a semigroup is a magma whose operation is associative.)

Now define a signature $\Sigma^{\equiv}$ with $\Sigma_0^{\equiv} = \mathfrak{F}_{\mathbf{Mag}}X \times \mathfrak{F}_{\mathbf{Mag}}X$ and the following operations.

- For each $x \in \mathfrak{F}_{\mathbf{Mag}}X$, a nullary operation $() \to (x, x)$.
- For each $x, y \in \mathfrak{F}_{\mathbf{Mag}}X$, a unary operation $((x, y)) \to (y, x)$.
- For each $x, y, z \in \mathfrak{F}_{\mathbf{Mag}}X$, a binary operation $((x, y), (y, z)) \to (x, z)$.
- For each $x, y, z, w \in \mathfrak{F}_{\mathbf{Mag}}X$, a binary operation

$$((x, y), (z, w)) \to (x \cdot z, y \cdot w),$$

where $\cdot$ denotes the binary magma operation on $\mathfrak{F}_{\mathbf{Mag}}X$.

- For each $x, y, z \in \mathfrak{F}_{\mathbf{Mag}}X$, a nullary operation

$$() \to (x \cdot (y \cdot z), (x \cdot y) \cdot z).$$

An algebra for this signature is an $(\mathfrak{F}_{\mathbf{Mag}}X \times \mathfrak{F}_{\mathbf{Mag}}X)$-indexed family of sets $R(x, y)$ equipped with elements and operations

$$\begin{aligned} e_x &\in R(x,x) \\ R(x,y) &\to R(y,x) \\ R(x,y) \times R(y,z) &\to R(x,z) \\ R(x,y) \times R(z,w) &\to R(x \cdot z, y \cdot w) \\ a_{x,y,z} &\in R(x \cdot (y \cdot z), (x \cdot y) \cdot z) \end{aligned}$$

Now for any such R, "$R(x, y)$ is nonempty" is a binary *relation* on $\mathfrak{F}_{\mathbf{Mag}}X$, which we abusively denote also by $R(x, y)$. The above elements and operations imply that this is an equivalence relation that is a congruence for the magma operation and moreover relates $x \cdot (y \cdot z)$ to $(x \cdot y) \cdot z$ for all x, y, z. And conversely, if we have any such binary relation $\sim$, we can construct an indexed algebra R by setting $R(x, y) = \mathbb{1}$ if $x \sim y$ and $R(x, y) = \emptyset$ otherwise.

Let $\equiv$ denote the binary relation obtained as above from nonemptiness of the *initial* algebra for this indexed signature.

Theorem A.3.1. *The quotient of $\mathfrak{F}_{\mathbf{Mag}}X$ by $\equiv$ is the free semigroup generated by X.*

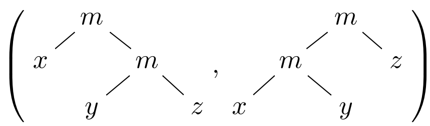
Proof. First we show that it is a semigroup. Given $u, v \in \mathfrak{F}_{\mathbf{Mag}}X/{\equiv}$, choose representatives $x, y \in \mathfrak{F}_{\mathbf{Mag}}X$ for them, and let $u \cdot v$ be the equivalence class of $x \cdot y$. Since $\equiv$ is a congruence for the magma operation, the result is independent of the choice of representatives; thus $\mathfrak{F}_{\mathbf{Mag}}X/{\equiv}$ is a magma. Now given $u, v, w \in \mathfrak{F}_{\mathbf{Mag}}X/{\equiv}$, choose representatives x, y, z; then since $x \cdot (y \cdot z) \equiv (x \cdot y) \cdot z$, we have $u \cdot (v \cdot w) = (u \cdot v) \cdot w$. Thus $\mathfrak{F}_{\mathbf{Mag}}X/{\equiv}$ is a semigroup

Now let M be any other semigroup and $\psi : X \to M$ a map. Since M is in particular a magma, we have a unique induced magma morphism $\phi : \mathfrak{F}_{\mathbf{Mag}}X \to M$. Define a binary relation R on $\mathfrak{F}_{\mathbf{Mag}}X$ by saying that $R(x, y)$ means $\phi(x) = \phi(y)$. Since ϕ is a magma morphism and M is a semigroup, R can be regarded as an algebra for the above indexed signature. Thus it admits a map from the initial such algebra. Hence, if $x \equiv y$, then $R(x, y)$, i.e. $\phi(x) = \phi(y)$; so ϕ factors through $\mathfrak{F}_{\mathbf{Mag}}X/{\equiv}$. It is straightforward to check that this factorization is a semigroup morphism and is the unique such extending ψ. $\square$

In the general case, we proceed as follows. Suppose Σ is a (multi-sorted) signature and we have additionally a set Λ of **axioms**, each of which is a pair (a, b) of elements of the free algebra $W\Sigma[V]_i$ for some $i \in \Sigma_0$ and some finite set V. Then for any Σ-algebra A, any axiom $a, b \in W\Sigma[V]_i$, and any function $g : V \to A$ (picking out some finite set of elements of A), we have an induced

Σ-algebra map $\overline{g} : W\Sigma[V] \to A$. We define a (Σ, Λ)-**algebra** to be a Σ-algebra A such that $\overline{g}(a) = \overline{g}(b)$ for any $(a, b) \in \Lambda$ and $g : V \to A$.

For instance, associativity in a magma is represented by the axiom



$$\in W\Sigma[\{x, y, z\}]$$

The (Σ, Λ)-algebras in this case are exactly semigroups.

Now, given a set X, we define a signature $\Sigma^{\equiv}$ with

$$\Sigma_0^{\equiv} = \{ (i, x, y) \mid i \in \Sigma_0; x, y \in W\Sigma[X]_i \}$$

and the following operations:

- For each $x \in W\Sigma[X]_i$, a nullary operation $() \to (i, x, x)$.
- For each $x, y \in W\Sigma[X]_i$, a unary operation $((i, x, y)) \to (i, y, x)$.
- For each $x, y, z \in W\Sigma[X]_i$, a binary operation $((i, x, y), (i, y, z)) \to (i, x, z)$.
- For each operation $m : (d_{m,1}, \ldots, d_{m,\mathrm{ar}(m)}) \to c_m$ in Σ, and each collection of pairs of elements $x_k, y_k \in W\Sigma[X]_{d_{m,k}}$ for $1 \leq k \leq \mathrm{ar}(m)$, an operation

$$((d_{m,1}, x_1, y_1), \ldots, (d_{m,\mathrm{ar}(m)}, x_{\mathrm{ar}(m)}, y_{\mathrm{ar}(m)})) \\ \longrightarrow (c_m, [m](x_1, \ldots, x_{\mathrm{ar}(m)}), [m](y_1, \ldots, y_{\mathrm{ar}(m)})).$$

- For each axiom $a, b \in W\Sigma[V]_i$ in Λ and each function $g : V \to W\Sigma[X]$ with unique extension $\overline{g} : W\Sigma[V] \to W\Sigma[X]$, a nullary operation

$$() \to (i, \overline{g}(a), \overline{g}(b)).$$

Let $\equiv_i$ be the binary relation on $W\Sigma[X]_i$ defined by $a \equiv_i b$ if the sort (i, a, b) is nonempty in the initial $\Sigma^{\equiv}$-algebra.

Theorem A.3.2. *Each $\equiv_i$ is an equivalence relation and a congruence for the Σ-algebra structure, and the quotients $W\Sigma[X]_i/{\equiv_i}$ form the free (Σ, Λ)-algebra.* $\square$

As in §A.1, we will usually think of this theorem slightly differently: to define a family of maps $f_i : W\Sigma[X]_i/{\equiv_i} \to A_i$, it suffices to define each $f_{c_m}([m](t_1, \ldots, t_{\mathrm{ar}(m)}))$ assuming recursively that $f_{d_{m,1}}(t_1), \ldots, f_{d_{m,\mathrm{ar}(m)}}(t_{\mathrm{ar}(m)})$ have been defined, and also to check that for any axiom $(a, b) \in W\Sigma[V]_i$ and $g : V \to W\Sigma[X]_i$ we have $f_i(\overline{g}(a)) = f_i(\overline{g}(b))$.

Exercises

Exercise A.3.1. Prove Theorem A.3.2.

Exercise A.3.2. Why is the axiom of choice required to generalize Theorem A.3.2 to the case of infinitary operations?

A.4 Rules and deductive systems

The basic machinery of type theory is an iteration and reformulation of the preceding sections in different language, simultaneously introducing convenient notations.

We consider a sequence of signatures $\Sigma^{(1)}, \Sigma^{(2)}, \dots, \Sigma^{(n)}$ for which the sorts of $\Sigma^{(k)}$ are defined in terms of the initial algebras $W\Sigma^{(j)}$ for the previous signatures $j < k$. For instance, we might have $\Sigma_0^{(2)} = W\Sigma^{(1)} \times W\Sigma^{(1)}$. A particularly important special case is when $\Sigma^{(k)}$ is $(\Sigma^{(j)})^{\equiv}$ for some $j < k$ and some set of axioms, as in §A.3.

Each sort in one of the signatures $\Sigma^{(k)}$ is called a **judgment**. We write $\mathcal{J}$ for a generic judgment, but we use more specific and congenial notation in particular cases, such as:

- When categories with object set $\mathcal{O}$ are regarded as an $(\mathcal{O} \times \mathcal{O})$-sorted theory as mentioned in §A.2, the sort (A, B) is usually written $A \vdash B$. This signature (with an $\equiv$ on top of it) corresponds to the cut-ful type theory for categories from §1.2.1. The cut-free type theory for categories has different operations but the same sorts, and uses the same notation.
- If $\Sigma^{(1)}$ is a one-sorted signature regarded as describing the *objects* of some categorical structure, then we denote its sort by "type". We generally then have $\Sigma_0^{(2)} = W\Sigma^{(1)} \times W\Sigma^{(1)}$ (for a unary type theory), with sorts again written as $A \vdash B$, where now A and B are elements of the initial $\Sigma^{(1)}$-algebra rather than elements of a fixed set $\mathcal{O}$.
- The multicategorical and polycategorical theories of chapters 2 and 3 use a similar notation $\Gamma \vdash \Delta$ for sorts (Γ, Δ) where one or both of Γ and Δ is a list rather than a single item.
- If $\Sigma^{(k)} = (\Sigma^{(j)})^{\equiv}$, then its sort $(\mathcal{J}, x, y)$ is usually written $x \equiv y : \mathcal{J}$.

In general, each operation $m : (\mathcal{J}_1, \dots, \mathcal{J}_n) \to \mathcal{J}'$ in one of the signatures $\Sigma^{(k)}$ is called a **rule**, and written

$$\frac{\mathcal{J}_1 \qquad \cdots \qquad \mathcal{J}_n}{\mathcal{J}'}\, m.$$

The input judgments $\mathcal{J}_1, \dots, \mathcal{J}_n$ of a rule are called its **premises**, and the output judgment $\mathcal{J}'$ is called its **conclusion**.

Finally, each element of $W\Sigma^{(k)}$ is called a **derivation** (sometimes a derivation *of* its root judgment) and written by placing rules on top of each other to form its tree structure. For instance, if $\mathcal{J}$ denotes the single sort of the signature for semigroups, then the associativity axiom of a monoid is

$$\dfrac{\dfrac{}{\mathcal{J}}\,x \qquad \dfrac{\dfrac{}{\mathcal{J}}\,y \quad \dfrac{}{\mathcal{J}}\,z}{\mathcal{J}}\,m}{\mathcal{J}}\,m \quad\equiv\quad \dfrac{\dfrac{\dfrac{}{\mathcal{J}}\,x \quad \dfrac{}{\mathcal{J}}\,y}{\mathcal{J}}\,m \qquad \dfrac{}{\mathcal{J}}\,z}{\mathcal{J}}\,m$$

Note the rules with empty premises, corresponding to nullary operations. Similarly, for the cut-ful type theory for categories, associativity is the collection of axioms (one for each $A, B, C \in \mathcal{O}$)

$$\dfrac{\dfrac{}{A \vdash B}\,x \qquad \dfrac{\dfrac{}{B \vdash C}\,y \qquad \dfrac{}{C \vdash D}\,z}{B \vdash D}\circ_{B,C,D}}{A \vdash D}\circ_{A,B,D}$$

$$\equiv \qquad \dfrac{\dfrac{\dfrac{}{A \vdash B}\,x \qquad \dfrac{}{B \vdash C}\,y}{A \vdash C}\circ_{A,B,C} \qquad \dfrac{}{C \vdash D}\,z}{A \vdash D}\circ_{A,C,D}$$

The whole sequence of signatures $\Sigma^{(1)}, \Sigma^{(2)}, \ldots, \Sigma^{(n)}$ is called a **deductive system**. Thus, for instance, the signature $\Sigma[X]$ for semigroups under a fixed set X, together with the axiom-signature for monoids under X on top of it, form a single deductive system. Some deductive systems (probably not all) deserve to be called *type theories*; but we will not attempt to give any definition of this class except by the examples we consider (throughout the entire book).

Remark A.4.1. To be sure, not all type theories fit exactly into the picture presented here. In particular, *dependent* type theories break the clean "stratification" of a deductive system $\Sigma^{(1)}, \Sigma^{(2)}, \ldots, \Sigma^{(n)}$, since in the judgment $\vdash A\ \mathsf{type}$ the type A can now contain terms from the "higher level" judgment $\Gamma \vdash M : B$. Thus the whole system must be defined by one big mutual induction (in type-theoretic lingo it is an "inductive-inductive definition"). The general idea is similar, however.

A.5 Terms

Since the judgments in each signature $\Sigma^{(k)}$ in a deductive system are defined in terms of the *elements* of $W\Sigma^{(j)}$ for $j < k$, and the latter are rooted trees, the notation would rapidly get unwieldy if each $\mathcal{J}$ in a rule contained within it some number of derivation trees. Thus, we generally represent derivations by **terms**, which are a more concise syntax containing enough information to reconstruct the derivation. For instance, the expressions $x \cdot (y \cdot z)$ and $(x \cdot y) \cdot z$ for the two sides of associativity are terms, in which we have represented the rule m by an infix operation "$\cdot$".

If M is a term representing a derivation of the judgment $\mathcal{J}$, we generally write $M : \mathcal{J}$. (A notable exception is that if $\mathcal{J}$ is the sort of a one-sorted $\Sigma^{(1)}$ presenting the objects of a category, as mentioned above, we usually write "$A\ \mathsf{type}$" or "$\vdash A\ \mathsf{type}$" rather than "$A : \mathsf{type}$".) We describe a syntax for terms by annotating the rules of a deductive system with terms, so that for instance the multiplication of a semigroup would be

$$\frac{M : \mathcal{J} \qquad N : \mathcal{J}}{M \cdot N : \mathcal{J}}\, m$$

Here M and N are "metavariables" standing for terms, indicating that whatever terms we have representing two derivations of $\mathcal{J}$, we represent their combination by m by juxtaposing them with an infix dot. (We always assume that parentheses are added as necessary to ensure correct grouping.)

For purposes of this discussion, "terms with variables from the context" such as $x : A \vdash M : B$ can be regarded as merely a variant notation of something like $x.M : (A \vdash B)$. Thus we still have a single thing (namely $x.M$) that represents the entire derivation, even though we generally apply the word "term" only to part of this thing (namely M). Similarly, an equality judgment like $x : A \vdash M \equiv N : B$ is shorthand for $(x.M) \equiv (x.N) : (A \vdash B)$. There is one actual difference here in that we generally consider terms of this form modulo "α-equivalence", i.e. the consistent renaming of variables. For now, let us assume that we know what that means; in §A.6 we will explain it precisely.

There is no unique way to assign terms to a deductive system; all that is necessary is to describe some kind of syntax from which a derivation tree can be algorithmically extracted. When a human mathematician reads an expression such as $x \cdot (y \cdot z)$, they generally mentally organize it as a tree without really thinking about it: here the first $\cdot$, being the "outer" operation, is the root, with children x and $y \cdot z$, and the latter decomposes further into another $\cdot$ node with children y and z. This "internal syntax tree" has exactly the same shape as the intended derivation tree. An alternative reading where the second $\cdot$ is the root with children "$x \cdot (y$" and "$z)$" is ruled out by our intuitive understanding of the meaning of parentheses. When a computer reads such an expression it likewise constructs an internal tree representation, but the programmer has to explicitly instruct it how to do so; this is called **parsing**.

If we are given a putative term claiming to represent a derivation of some judgment, then after parsing there is a further step of verifying that the "parse tree" indeed corresponds to a valid derivation tree. This is called **type-checking**. Technically it could be done at the same time as parsing, but both human and electronic mathematicians generally separate them. Thus the parse tree is a sort of "raw abstract syntax" that knows how operations are grouped but not whether the operations actually mean anything yet.

We will not say anything more about parsing; we trust the human reader to do it unconciously and the programmer to have good algorithms for it. Thus, in our formal description of terms, the mathematical objects we call "terms" will be representations of parse trees. And as trees, they will be elements of some other free algebra — but a simpler one than the one whose derivations we are using them to represent. For instance, for the cut-ful type theory of categories under $\mathcal{G}$, whose judgments are of the form $A \vdash B$ for $A, B \in \mathcal{G}_0$ (and in particular there are $\mathcal{G}_0 \times \mathcal{G}_0$ of them), the terms will be elements of a *one-sorted* free algebra with a nullary operation id_A and a binary operation $\circ_A$ for each $A \in \mathcal{G}_0$. Thus this free algebra contains many "ill-typed" terms such as $g \circ_B \mathrm{id}_A$ where $g \in \mathcal{G}(C, D)$; the goal of type-checking is to discard these undesirables. (For technical reasons, rather than a single set of terms as here, in the general case we will allow each judgment of our intended theory to be assigned a different set of "potential terms"; see below.)

Now in practice, the input to type-checking is usually a parsed term *together with* a putative type for that term, and so the term notations only need to contain enough information to reconstruct the derivation tree when supplemented with the latter. For instance, we have noted that the cut-ful type theory for categories technically involves a different composition operation $\circ_{A,B,C}$ for each triple of objects, so that terms would technically have to be written as $h \circ_{A,C,D} (g \circ_{A,B,C} f)$. However, if we are given a term whose outer operation is a composition and that claims to represent a derivation of a judgment $A \vdash D$, then the composition must be $\circ_{A,?,D}$. Thus in general it suffices to indicate the object being composed over, as in $h \circ_C (g \circ_B f)$.

Remark A.5.1. In many cases we can omit further information because it can be inferred from context; for instance, if we know that $h : C \to D$ then a term of the form "$h \circ (-)$" can only mean "$h \circ_C (-)$". Human mathematicians omit information informally and unsystematically, and we have done the same throughout the book. The implementors of electronic mathematicians have elaborate and precise algorithms for "inferring from context" enabling the omission of information, but most of these are far beyond our scope.

With type-checking (and also "proof search") in mind, type theorists tend to read the rules of a deductive system "bottom-up". That is, instead of thinking of a rule

$$\frac{\mathcal{J}_1 \qquad \mathcal{J}_2}{\mathcal{J}}$$

as meaning "if we have $\mathcal{J}_1$ and $\mathcal{J}_2$ we can deduce $\mathcal{J}$", they instead think "if we want to deduce $\mathcal{J}$, it suffices to have $\mathcal{J}_1$ and $\mathcal{J}_2$". This is the direction that a type-checking algorithm applies the rule: given a parsed term M and a putative judgment $\mathcal{J}$, the rule tells us how to break down the job of checking that $M : \mathcal{J}$ into simpler type-checking tasks that can be done recursively.[2] It is also the direction that the rule is often applied when *searching* for a derivation of $\mathcal{J}$, by the same sort of recursive procedure.

With all of this in mind, we make the following formal definition.

Definition A.5.2. Let Σ be a signature; a **term system** for Σ is a Σ-algebra $\mathbb{T}$, whose elements are called (potentially ill-typed) **terms**, such that

(a) For any judgment $c \in \Sigma_0$ and term $t \in \mathbb{T}_c$, there is at most one rule $m : (d_1, \dots, d_n) \to c$ and terms $s_j \in \mathbb{T}_{d_j}$ such that $t = [m](s_1, \dots, s_n)$.

[2]However, some more advanced theories are type-checked in a "bidirectional" way, with some judgments being read upwards in this way and others being read downwards as "type synthesis", where only the term is given and the type is inferred by the algorithm.

(b) If we define a relation $s \prec t$ on $\bigsqcup_i \mathbb{T}_i$ to hold just when $t = [m](s_1, \ldots, s_n)$ and $s = s_j$ for some j, then $\prec$ is well-founded: there are no infinite chains $t_1 \succ t_2 \succ t_3 \succ \cdots$.

Since a term system $\mathbb{T}$ is a Σ-algebra, there is a unique Σ-algebra morphism $W\Sigma \to \mathbb{T}$. This is the function that assigns to each derivation a unique representing term. Axiom (a) above ensures that a derivation is determined by its term:

Lemma A.5.3. *If $\mathbb{T}$ is a term system, then the unique Σ-algebra morphism $W\Sigma \to \mathbb{T}$ is injective.*

Proof. Let $x, y \in W\Sigma_i$ have the same image in $\mathbb{T}_i$. By axiom (a), and the fact that $W\Sigma \to \mathbb{T}$ is a Σ-algebra morphism, we must have $x = [m](x_1, \ldots, x_n)$ and $y = [m](y_1, \ldots, y_n)$ for the same operation m and each pair x_j, y_j having the same image in $\mathbb{T}$. By structural induction, therefore, each $x_j = y_j$, and thus $x = y$. □

However, the converse of Lemma A.5.3 does not hold. Indeed, its conclusion does not even imply axiom (a) (which is all that was used in its proof): the former is only about "globally" well-typed terms, while the latter is a "local" condition that says something even about ill-typed terms.

The reason for the extra strength of (a), and for condition (b), is to make typechecking a "deterministic terminating recursive algorithm", as follows. Given a term t, we check whether it is of the form $[m](s_1, \ldots, s_n)$. If so, then by (a) m and the s_j's are uniquely determined, and we can recursively consider each s_j. If the answer is ever no, then the term t is ill-typed. Otherwise, axiom (b) ensures that the algorithm must terminate (by bottoming out at nullary rules), and we have now constructed a derivation (the tree of rules m) represented by the term t.

In the main text, we generally stated our "terms are derivations" lemmas in the simple form of "if a term judgment is derivable, then it has a unique derivation." As stated this is only the conclusion of Lemma A.5.3, but in all cases our proofs had the simple inductive form that actually establishes all of Definition A.5.2.

Of course, for this to actually be an algorithm in the computer science sense, the test for whether $t = [m](s_1, \ldots, s_n)$ would have to be "computable". Making that precise is far beyond our current scope, but it may be worth mentioning that it generally holds because $\mathbb{T}$ is constructed using an initial algebra for some other signature, and initial algebras are very computable (they are "abstract datatypes"). Such a construction of $\mathbb{T}$ generally also ensures axiom (b) immediately.

Notationally, we regard the common "annotation of rules" as specifying a signature along with a term syntax for it. For instance, when we annotate the composition rule in the cut-ful type theory of categories

$$\frac{A \vdash B \qquad B \vdash C}{A \vdash C}$$

by terms to get

$$\frac{\phi : (A \vdash B) \qquad \psi : (B \vdash C)}{\psi \circ_B \phi : (A \vdash C)}$$

we mean that if m is this rule, then the corresponding operation $[m]$ on $\mathbb{T}$ is given by the operation $(- \circ_B -)$. Technically, this requires us to specify in advance the set (or sets) $\mathbb{T}$ of terms, so that the annotated rules are describing which *previously existing operations* on $\mathbb{T}$ we are using to represent each rule. However, since in most cases $\mathbb{T}$ is a free algebra for a different signature with an operation corresponding directly to each rule in Σ (though not in a one-to-one manner), we can generally omit this preliminary step and assume that $\mathbb{T}$ is freely generated as necessary by the operations named in the annotations.

A.6 Variable binding and α-equivalence

Finally, we come to the vexing question of α-equivalence. We could wave our hands at it by claiming to use de Bruijn variables everywhere, but this would be a bit dishonest. As is evident, we actually do use named variables all over the place, so it behooves us to say something about what they mean. In this section we will describe a general way to construct "terms with binders" such as match and λ and define a notion of α-equivalence. There are many ways to do this; our approach follows [**?**, **?**, **?**] (see also [**?**]).

Let $\mathbb{A}$ be a fixed infinite set (usually countable), whose elements we call **variables**. Let Σ be a signature, one-sorted for simplicity, together with injective functions $v, b : \mathbb{A} \to \Sigma_1$ such that $\mathrm{ar}(v(x)) = 0$ and $\mathrm{ar}(b(x)) = 1$ for all $x \in \mathbb{A}$. What we have in mind is that the initial Σ-algebra will supply the set of terms in a term syntax for some other signature, with the operations of Σ corresponding to the term notations for the rules in that other signature.

The inclusion v simply says that variables can occur in terms, while the operation $b(x)$ is intended to "bind" the variable x in its argument; usually $b(x)(M)$ is written $x.M$. When term notations bind variables, their corresponding operations will put a specially named Σ-operation together with one or more uses of b. For instance, when describing the terms in the unary type theory for categories with coproducts, there will be operations match_{A+B} of arity 3, which we combine with two uses of b to represent the terms annotating $+E$:

$$\mathsf{match}_{A+B}(M, u.P, v.Q) = \mathsf{match}_{A+B}(M, b(u)(P), b(v)(Q)).$$

As usual, let $W\Sigma$ be the initial Σ-algebra; and let $\mathrm{Aut}(\mathbb{A})$ be the group of automorphisms (permutations) of the set $\mathbb{A}$. We write the action of $\sigma \in \mathrm{Aut}(\mathbb{A})$ on $x \in \mathbb{A}$ by x^σ. Now we define, by recursion, an action of $\mathrm{Aut}(\mathbb{A})$ on $W\Sigma$ as follows:

$$\sigma \cdot [v(x)] = [v(x^\sigma)]$$
$$\sigma \cdot [b(x)](M) = [b(x^\sigma)](\sigma \cdot M)$$

with $\sigma \cdot M$ defined recursively in the latter. In all other cases, $\sigma \cdot (-)$ simply recurses into all subtrees. It is easy to show that this is a group action.

Because all operations in Σ have finite arity[3] and all trees in $W\Sigma$ are well-founded, only finitely many variables can occur in any such tree (either through v or b). So, since $\mathbb{A}$ is infinite, for any $M \in W\Sigma$ there is some variable $z \in \mathbb{A}$ that does not occur in M. We call such a z **fresh** (relative to M) and write $z \notin M$.

We now define α-equivalence $\equiv$ on $W\Sigma$, by defining a new signature $\Sigma^{\equiv}$ similar to how we did it in §A.3. We include operations making $\equiv$ a congruence for all operations of Σ except b. In the case of v, this means we have "reflexivity at variables" $v(x) \equiv v(x)$. We also include one further operation that in rule form looks like this:

$$\frac{z \notin M \qquad z \notin N \qquad z \neq x \qquad z \neq y \qquad (zx) \cdot M \equiv (zy) \cdot N}{b(x)(M) \equiv b(y)(N)} \tag{A.6.1}$$

Here (zx) and (zy) denote the transposition permutations that swap z with x and z with y, respectively. Since z does not occur in M and N, the permutation actions $(zx) \cdot M$ and $(zy) \cdot N$ amount to replacing all occurences of x in M and y in N (even bound ones) by z. The rule then says that if these two results are α-equivalent, then so are the terms $x.M$ and $y.N$ with new bound variables.

For instance, $x.x$ and $y.y$ are α-equivalent because $(zx) \cdot x = z$ and $(zy) \cdot y = z$. We also have $x.(x.x) \equiv x.(y.y)$ because $(zx) \cdot (x.x) = (z.z)$ and $(zy) \cdot (y.y) = (z.z)$ as well, so the inner bound x really does "shadow" the outer one, making the latter invisible even though it has the same name. But neither of these is equivalent to $x.(y.x)$, since $(zx) \cdot (y.x) = (y.z)$.

Note also that if $M \in W\Sigma$, then $x.M$ is α-equivalent to $y.((yx) \cdot M)$ for any variable y not occurring in M, since if neither y nor z occur in M then

$$(zy) \cdot (yx) \cdot M = (yx) \cdot (zx) \cdot M = (zx) \cdot M.$$

Thus, we can always replace a bound variable by any another fresh variable.

Unlike in §A.3, we do not explicitly include operations making $\equiv$ an equivalence relation. However, we can nevertheless prove that it is; this is itself a sort of cut-admissibility.

Lemma A.6.2. *α-equivalence $\equiv$, as defined above, has the following properties:*

(a) *Equivariance: if $M \equiv N$, then $\sigma \cdot M \equiv \sigma \cdot N$ for any $\sigma \in \mathrm{Aut}(\mathbb{A})$.*

(b) *Congruence for binding: if $M \equiv N$, then $x.M \equiv x.N$.*

(c) *Rule (A.6.1) is invertible: if $x.M \equiv y.N$, then $(zx) \cdot M \equiv (zy) \cdot N$ for some fresh z.*

(d) *Reflexivity: $M \equiv M$ for any $M \in W\Sigma$.*

[3] If Σ were allowed to contain infinitary operations, then to make this work, the cardinality of $\mathbb{A}$ would have to be of cofinality greater than any of their arities.

(e) Symmetry: if $M \equiv N$ then $N \equiv M$.

(f) Transitivity: if $M \equiv N$ and $N \equiv P$, then $M \equiv P$.

(g) Bound variables can be altered freely: if $z \notin M$ then $x.M \equiv z.((zx) \cdot M)$.

Proof. Perhaps surprisingly, the tricky and important one is (a). Of course, the proof is by induction on the derivation of $M \equiv N$, and all the congruence rules are immediate, so it remains to deal with (A.6.1). That is, suppose $x.M \equiv y.N$ is obtained from $(zx)\cdot M \equiv (zy)\cdot N$, and let $\sigma \in \mathrm{Aut}(\mathbb{A})$. Now $\sigma\cdot(x.M) = x^\sigma.(\sigma\cdot M)$ and similarly for N, so to conclude $\sigma \cdot (x.M) \equiv \sigma \cdot (y.N)$ it will suffice to show $(wx^\sigma) \cdot \sigma \cdot M \equiv (wy^\sigma) \cdot \sigma \cdot N$ for some fresh w. The obvious choice for w is z^σ. Then if we let $\tau = (z^\sigma y^\sigma)\sigma(zx) \in \mathrm{Aut}(\mathbb{S})$, we have

$$\begin{aligned}(z^\sigma x^\sigma) \cdot \sigma \cdot M &= \tau \cdot (zx) \cdot M \\ &\equiv \tau \cdot (zy) \cdot N \\ &= (z^\sigma y^\sigma) \cdot \sigma \cdot N\end{aligned}$$

using the inductive hypothesis of equivariance for $(zx) \cdot M \equiv (zy) \cdot N$.

Now (b) is immediate, since $M \equiv N$ implies $(zx)\cdot M \equiv (zx)\cdot N$, whence (A.6.1) gives $x.M \equiv x.N$. And (c) is clear since (A.6.1) is the only rule that can produce an α-equivalence between terms of the form $x.M$ and $y.N$ (since we did not include (b) or (d)–(f) as primitive). Combining the primitive congruence rules with (b) yields straightforward inductive proofs of (d) and (e).

For (f) we induct on both $M \equiv N$ and $N \equiv P$. By inspection of the form of N, they must both be derived from the same rule. If it is a primitive congruence rule, we just apply the inductive hypothesis to all the inputs and then congruence again. The interesting case is (A.6.1), where we have $(ux) \cdot M \equiv (uy) \cdot N$ and also $(vy) \cdot N \equiv (vz) \cdot P$ for potentially different variables u and v, with u fresh for M, N, x, y and v fresh for N, P, y, z. Since $\mathbb{A}$ is infinite there exists a variable w that is fresh for all of M, N, P, x, y, z, and so we have

$$(wx) \cdot M = (wu) \cdot (ux) \cdot M \equiv (wu) \cdot (uy) \cdot N = (wy) \cdot N$$

using (a). Similarly, $(wy) \cdot N \equiv (wz) \cdot P$, so we ought to be able to conclude by the inductive hypothesis that $(wx) \cdot M \equiv (wz) \cdot P$ and so $x.M \equiv z.P$ by (A.6.1). However, this is not the usual structural inductive hypothesis, since the derivations of and $(wx) \cdot M \equiv (wy) \cdot N$ and $(wy) \cdot N \equiv (wz) \cdot P$ are produced by (a) and are not subtrees of our given derivations of $x.M \equiv y.N$ and $y.N \cdot z.P$. Instead we have to do something like assign a natural number "height" to all derivations, observe that (a) preserves the height of derivations, and then induct on height.

Finally, for (g) we choose a fresh w and observe that $(wx)\cdot M = (wz)\cdot(zx)\cdot M$. Thus, by reflexivity (d) we have $(wx) \cdot M \equiv (wz) \cdot (zx) \cdot M$ and hence by (A.6.1) $x.M \equiv z.((zx) \cdot M)$. □

The quotient $W\Sigma/\equiv$ of this equivalence relation is, of course, our set of "terms modulo α-equivalence of bound variables". Since $\equiv$ is a congruence for all the operations of Σ, these all descend to the quotient, including (by Lemma A.6.2(b)) variable binding; we also denote this operation by $x.M$ where now $M \in W\Sigma/\equiv$.

Our goal is to use this quotient as the term syntax for another signature. In practice we will write terms as elements of $W\Sigma$ itself, but we regard them formally as representing their equivalence class. We also usually want to restrict to some subsets of terms that have the right number of variables bound to represent the context.

For instance, in unary type theories (chapter 1) we have said that a term judgment such as $x : A \vdash M : B$ can be read as $x : M : (A \vdash B)$. We really do want this x to be a *bound* variable in the formal sense of this section, since derivations to determine unique terms we have to quotient by renaming the variables in the context as well. That is, we represent "free" variables as variables that are bound "on the outside". Thus, we should take our set $\mathbb{T}$ of terms to be the subset of $W\Sigma/\equiv$ consisting of terms having a variable binding outermost. Similarly, in a simple type theory (chapter 2) the terms for $\Gamma \vdash B$ should have n variable bindings outermost, where n is the length of Γ (this is why in §A.5 we allowed different judgments to have different sets of potential terms).

We then need to define operations on these sets $\mathbb{T}$ that represent the rules of our desired signature. These will generally be constructed from basic operations in Σ combined with one or more variable bindings.

Let us consider match_+ from §1.5 as a paradigmatic example. Since the rule $+E$ has three premises, what we have to give is an operation $\mathbb{T} \times \mathbb{T} \times \mathbb{T} \to \mathbb{T}$, where $\mathbb{T}$ is the set of α-equivalence classes of terms of the form $x.M$. We have presumably included "match_+" as a 3-ary operation in our term signature Σ, but this does not take account yet of the binding structure. The inputs to our desired operation will be (given the above restriction defining $\mathbb{T}$) of the form $x.M$, $u.P$, and $v.Q$. The basic 3-ary operation in Σ could give $\mathsf{match}_+(x.M, u.P, v.Q)$, but of course we want "$x.\mathsf{match}_+(M, u.P, v.Q)$" instead.

To define this, we first choose representatives for the equivalence classes of $x.M$, $u.P$, and $v.Q$. By Lemma A.6.2(g) we can do this so that x does not appear in $u.P$ or $v.Q$ (which have only finitely many variables each). Now we can write $x.\mathsf{match}_+(M, u.P, v.Q)$; but for this to define an operation $\mathbb{T} \times \mathbb{T} \times \mathbb{T} \to \mathbb{T}$ we have to check that it is independent of the chosen representatives. For $u.P$ and $v.Q$ this is easy since $\equiv$ is a congruence for all operations of Σ, including match_+. And if $x.M \equiv y.N$, then by Lemma A.6.2(c) we have $(zx) \cdot M \equiv (zy) \cdot N$ for some z, which we may also take to not appear in $u.P$ or $v.Q$. Thus, using the congruence rules and transitivity, we have

$$\begin{aligned} x.\mathsf{match}_+(M, u.P, v.Q) &\equiv z.\mathsf{match}_+((zx) \cdot M, u.P, v.Q) \\ &\equiv z.\mathsf{match}_+((zy) \cdot N, u.P, v.Q) \\ &\equiv y.\mathsf{match}_+(N, u.P, v.Q). \end{aligned}$$

The same principle applies to all other term systems using variable binding. Sometimes we also need to poke down into all the terms to ensure that certain variables in their context are disjoint or equal. For instance, the term operation representing $\times I$ takes as given $x.M$ and $y.N$, but its output has only one shared variable. Thus we have to first note $x.M \equiv z.((zx)\cdot M)$ and $y.N \equiv z.((zy)\cdot N)$ for some z that is fresh for both, and then write $z.\langle (zx) \cdot M, (zy) \cdot N\rangle$ for the pairing term. Based on these examples, we trust that the reader could formulate precise definitions of all the terms used in this book as operations on α-equivalence classes.

Of course, in any particular case it is still (technically) necessary to prove that what we get *is* a term system in the sense of Definition A.5.2. Since $\mathbb{T}$ is a subset of an initial algebra, and our operations are built using at least one operation of that algebra, A.5.2(b) is straightforward. Finally, the proof of A.5.2(a) essentially means checking that we chose the operations of the term signature to contain enough information to reconstruct a derivation step-by-step. This is a formal version of what in the main text we called *type-checking is possible* or *terms are derivations*. Note that since all our "terms" are actually α-equivalence classes, we never have to prove anything about α-equivalence.