An Introduction to Categories in Computing

C. Barry Jay

University of Technology, Sydney
School of Computing Sciences
P.O. Box 123 Broadway
Australia. 2007
e-mail:cbj@socs.uts.edu.au

Contents

Preface

These notes arose from introductory graduate courses in category theory for computer scientists, taught at the University of Edinburgh and the University of Technology, Sydney from 1989 til 1993. The intention is to give a brief survey of some central categorical concepts, including adjoints and monads, with examples drawn from a computing perspective. Others have already taken this route, so that there is little that is novel to be found here. It does, however, address the theory of monads (or triples) which is of increasing importance in both syntax and semantics, and also my theory of loops, their fixpoints and invariants.

This work is typeset using Latex with diagrams constructed using Paul Taylor's diagrams.tex, and inferences constructed using Makoto Tatsuta's proof.sty.

Chapter 1

Categories

1.1 Definition and Examples

Every category has an underlying *graph* which, in these lectures, means a directed multigraph

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[dr, "k"'] & B \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] \arrow[d, "l"] \arrow[dr, "m"'] & C \arrow[d, "n"] \\
& D \arrow[r, "p"'] & E
\end{tikzcd}$$

In the earliest examples, the objects (vertices) were always sets with some algebraic or topological structure, and the morphisms (edges) were functions that *preserved* the structure. Now even without knowing anything about what the structure actually is, it is possible to say something about this graph.

- If $f : A \to B$ and $g : B \to C$ preserve the structure then so does the composite function $g \circ f : A \to C$.
- The identity function $\mathtt{id}_A : A \to A$ preserves the structure.

Indeed, composition (with its identities) is the fundamental operation of category theory. Let's consider some examples:

Category	Objects	Morphisms
Sets	sets	functions
Mon	monoids	homomorphisms
Grp	groups	homomorphisms
Gph	graphs	homomorphisms
Pos	posets	order-preserving functions
Pre	pre-orders	order-preserving functions
Top	topological spaces	continuous maps
$\mathbf{Pos}_\omega$	ω-c.p.o.'s	continuous maps

Of course, there are many other examples, given by different kinds of algebra or topology, but we won't dwell on them here.

Definition 1.1.1 *A* category $\mathcal{C}$ *is given by a collection* $ob(\mathcal{C})$ *of* objects *and for each pair* A, B *of objects a set* $\mathcal{C}(A, B)$ *of* morphisms*, with some additional properties. If* $f \in \mathcal{C}(A, B)$ *then the* domain *or* source *of* f *is* $\mathtt{dom}(f) = A$ *and the* codomain *or* target *of* f *is* $cod(f) = B$. *This may be written* $f : A \to B$ *or* $A \xrightarrow{f} B$. *For each triplet* A, B, C *of objects there is a* composition function

$$\begin{array}{rcl} \mathcal{C}(B,C)\times\mathcal{C}(A,B) & \to & \mathcal{C}(A,C) \\ (g,f) & \mapsto & g \circ f \end{array}$$

which is associative and unitary, i.e. if f, g *and* h *are as above then*

$$\begin{array}{rcl} h \circ (g \circ f) & = & (h \circ g) \circ f \\ \mathtt{id}_B \circ f & = & f = f \circ \mathtt{id}_A \end{array}$$

In these notes the composite $g \circ f$ above may also be written $f;g$ or even as gf, but never as $f \circ g$ or $g;f$ or fg. The functional notation $g \circ f$ is preferred when thinking of functions, or the morphisms are denoted by single letters. The sequential notation is preferred for programs (see below) or when the morphisms are named by character strings, e.g. `push;pop`. Eliding the symbol for composition should be avoided unless there is no danger of confusion with the operation of application in the λ-calculus.

Let's consider the example of monoids. A *monoid* is a set M equipped with a binary operation $m : M^2 \to M$ called *multiplication* and a constant $e \in M$ called the *unit* such that m is associative

$$\begin{array}{rcl} m(a, m(b,c)) & = & m(m(a,b),c) \\ a \cdot (b \cdot c) & = & (a \cdot b) \cdot c \end{array}$$

and unitary

$$\begin{array}{rcl} m(e,a) & = & a = m(a,e) \\ e \cdot a & = & a = e \cdot a \end{array}$$

where $a \cdot b$ is infix notation for $m(a,b)$.

A *monoid homomorphism* $f : (M, m, e) \to (M', m', e')$ is a function $f : M \to M'$ which preserves the multiplication and unit, i.e.

$$\begin{array}{rcl} f(a \cdot b) & = & f(a) \cdot f(b) \\ f(e) & = & e' \end{array}$$

Now if $g : (M', m', e') \to (M'', m'', e'')$ is another homomorphism then the composite function $g \circ f$ also yields a homomorphism since

$$\begin{array}{rcl} (g \circ f)(a \cdot b) & = & g(f(a \cdot b)) \\ & = & g(f(a) \cdot f(b)) \\ & = & g(f(a)) \cdot g(f(b)) \\ & = & (g \circ f)(a) \cdot (g \circ f)(b) \end{array}$$

and similarly for the unit. Thus composition in **Mon** is given by the composition of the underlying functions, and so is associative with identities given by the identity functions.

Aside from the concrete categories above, the are many others whose morphisms are not functions.

Category	Objects	Morphisms	Composition
a monoid	$*$	elements	multiplication
a poset	elements	order	transitivity
an automaton	states	transitions	transitivity
a language	types	terms	substitution
a functional programming lang.	types	programs	sequential comp'n

These examples deserve a little extra comment. If M is an object in a category $\mathcal{C}$ then the morphisms $M \to M$ (called *endomorphisms* of M) form a monoid under composition: if $f, g : M \to M$ then $m(f, g) = f;g$ is an associative and unitary operation (the unit is the identity morphism) which is always defined since the matching of domains and codomains is guaranteed. Indeed, if M is the *only* object of $\mathcal{C}$ then the monoid $(M, m, \mathtt{id})$ determines the category, and we may think of a monoid as a category with only one object. Alternatively, we can think of a category as a "monoid with many objects".

In the same vein, consider a category whose homsets all have at most one element. Then there is no need to name the morphisms at all, and we may write $A \to B$ as $A \leq B$. Now the existence of composites and identities is the transitivity and reflexivity of $\leq$. The categorical axioms hold automatically since any two morphisms $A \to B$ are automatically equal. Thus our category is a *pre-order* whose elements are the objects ordered by $\leq$. Note that it need not be a partial order since we may well have morphisms $A \to B$ and $B \to A$ without $A = B$. Hence a pre-order "is"just a category whose hom-sets have at most one element. Conversely, a general category may be viewed as a pre-order with given reasons for the ordering, i.e. $f : A \to B$ asserts that $A \leq B$ by f.

An automaton is given by (Q, Σ, O, a) where Q is a collection of states, Σ is a collection of inputs, O is a collection of outputs and $a : \Sigma \times Q \to Q \times O$ maps pairs of inputs and states to pairs of states and outputs. We can build a category from this whose objects are the states. Morphisms $q_1 \to q_2$ are given by some elements (v, w) of $\Sigma^* \times O^*$, that is, pairs of words in Σ and O (ϵ is the empty word). The requirement is that the inputs v map q_1 to q_2 and produce output w. The inductive definition is given by

- $(\epsilon, \epsilon) : q \to q$ is the identity.
- If $(v, w) : q_1 \to q_3$ and $a(i, q_3) = (q_2, o)$ then $(vi, wo) : q_1 \to q_2$.

Composition is defined by

$$(v_2, w_2) \circ (v_1, w_1) = (v_1 v_2, w_1 w_2)$$

It is associative and unitary since concatenation of words is.

A typed language L yields a category whose objects are the types and whose morphisms $X \to Y$ are given by a pair (x, s) where $x : X$ is a variable and $s : Y$ is a term which has at most one free variable, namely x. More precisely, the morphisms are equivalence classes of these pairs under α-conversion, i.e. under change of free variable. Composition is given by substitution

$$\begin{tikzcd}
 & Y \arrow[dr, "{(y,t)}"] & \\
X \arrow[ur, "{(x,s)}"] \arrow[rr, "{(x,t[s/y])}"'] & & Z
\end{tikzcd}$$

The identity on X is (the equivalence class of) (x, x).

Functional programming languages can often be thought of as categories. For example, we might the underlying graph might contain

$$\begin{tikzcd}
N \arrow[r, "\mathtt{succ}"] & N \arrow[r, "\mathtt{IsZero}"] & B \\
 & 1 \arrow[u, "\mathtt{zero}"] \arrow[ur, "\mathtt{true}"'] &
\end{tikzcd}$$

with the obvious interpretations as naturals booleans, etc, and equations such as $\mathtt{IsZero} \circ \mathtt{zero} = \mathtt{true}$.

This last equation is a fine *denotation* of its left-hand side, but tells us nothing about how to *compute* it. This difficulty can be solved by extending the theory of categories to allow the homsets to have additional structure. In this case, we would like to incorporate the reduction order, so that we can write

$$\mathtt{IsZero} \circ \mathtt{zero} \Rightarrow \mathtt{true}$$

within the category. That is, each hom-order behaves like an automaton. We will not develop these ideas in this course, though it is a subject of ongoing research.

1.2 Life without Elements

The earliest categories all consisted of sets with structure and structure-preserving morphisms, so that one could reason about the elements of the objects, if one desired. However, the second set of examples above show that we are not always in this position, and indeed we will see many more examples as we proceed. So how are we to reason without elements?

Rather than simply leap into the categorical approach, I shall attempt to show you how to pass from an element-based approach to an arrow-based one in the category **Sets**.

There is a bijection between elements $x \in A$ of A and functions $\lceil x \rceil : 1 \rightarrow A$ where 1 is a set with one element $*$ and $\lceil x \rceil(*) = x$. The function $\lceil x \rceil$ is the *name* of x. Thus elements can be replaced by arrows. Now let's see how this can be used.

Consider $A \subseteq B$. This corresponds to a function $i : A \rightarrow B$ such that $i(x) = x$. Now i is not just any function, but in particular is an *injection*. That is, a function for $x, y \in A$

$$i(x) = i(y) \text{ implies } x = y$$

We can re-express this, using our names for elements, as

$$i \circ \lceil x \rceil = i \circ \lceil y \rceil \text{ implies } \lceil x \rceil = \lceil y \rceil$$

and since every function $1 \rightarrow A$ is the name of some element we can transform this again. Thus $m : A \rightarrow B$ is an injection if, for all functions $g, h : 1 \rightarrow A$

$$m \circ g = m \circ h \text{ implies } g = h$$

Now the definition is entirely expressed in terms of arrows, but there is a spurious reference to the special set (object) 1. Eliminating this reference leads us to the following

Definition 1.2.1 *A morphism $m : A \rightarrow B$ is a* monomorphism *if, for all objects X and all parallel pairs of morphisms $g, h : X \rightarrow A$ we have*

$$\begin{tikzcd} X \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] & A \arrow[r, "m"] & B \end{tikzcd}$$

$$m \circ g = m \circ h \textit{ implies } g = h$$

Exercise 1.2.2 Show that the monomorphisms in **Sets** are exactly the injections.

Exercise 1.2.3 Consider $f : A \rightarrow B$ and $g : B \rightarrow C$ in a category $\mathcal{C}$. Show that if f and g are both monomorphisms then $g \circ f$ is, too. Conversely, show that if $g \circ f$ is a monomorphism then f is, but not necessarily g.

Morphisms from 1 to A are called *global elements* of A while morphisms $X \rightarrow A$ are *local elements* of A.

Now let us consider *surjections*. A function $f : A \rightarrow B$ is a surjection if for every $y \in B$ is an $x \in A$ such that $f(x) = y$. A warning is appropriate here. The process of replacing elements by functions in a mechanical way in this definiton is unsatisfactory, because it is too complicated (try it), and there is a simpler way, obtained by dualising the definition of monomorphism.

Definition 1.2.4 *A morphism $f : A \rightarrow B$ is an* epimorphism *if, for all objects X and all parallel pairs of morphisms $g, h : B \rightarrow X$ we have*

$$g \circ f = h \circ f \text{ implies } g = h$$

Exercise 1.2.5 Show that a function $f : A \rightarrow B$ in **Sets** is an epimorphism iff it is surjective.

This definition was obtained from that for monomorphisms by simply changing the order of composition in the equations. In general we have

Definition 1.2.6 *Let $\mathcal{C}$ be a category. Then its* dual *or* opposite *$\mathcal{C}^{op}$ is the category whose objects are the same as those of $\mathcal{C}$ but with*

$$\mathcal{C}^{op}(A, B) = \mathcal{C}(B, A)$$

Thus $f : A \rightarrow B$ in $\mathcal{C}^{op}$ iff $f : B \rightarrow A$ in $\mathcal{C}$. Although the morphisms of $\mathcal{C}^{op}$ are the actual morphisms of $\mathcal{C}$ we sometimes write $f^{op} : A \rightarrow B$ for emphasis. Composition in $\mathcal{C}^{op}$ is given by that of $\mathcal{C}$. That is with f as above and $g : B \rightarrow C$ in $\mathcal{C}^{op}$ then

$$g^{op} \circ f^{op} = (f \circ g)^{op}$$

Identities are the same in both categories. Associativity and unicity of composition in $\mathcal{C}$ implies that for $\mathcal{C}^{op}$. Note too that $(\mathcal{C}^{op})^{op} = \mathcal{C}$.

This ability to dualise doubles the number of categories available, and every definition and theorem has a dual result. For example, $f : A \rightarrow B$ is a monomorphism in $\mathcal{C}$ iff it is an epimorphism in $\mathcal{C}^{op}$.

Exercise 1.2.7 Dualise Exercise 1.2.3 above.

We shall rely on duality often. It is a powerful idea since, for example, the duality between injections and surjections is not obvious set-theoretically, but only arises from the categorical presentation. (Try to extract the definition of epimorphisms directly using elements!)

A third set-theoretic notion is that of a bijection: a function $f : A \rightarrow B$ is a *bijection* if it is both an injection and a surjection. Translating this directly into category-theoretic terms is not quite enough. A morphism which is both a monomorphism and an epimorphism is called *balanced*. For example, the monoid inclusion $i : \mathcal{N} \rightarrow \mathbf{Z}$ of the natural numbers $\{0, 1, 2, \ldots\}$ into the integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$ is a balanced homomorphism. Clearly it is a monomorphism, while if $g, h : Z \rightarrow M$ are monoid homomorphisms that agree on $\mathcal{N}$ then they must be equal since monoid homomorphisms preserve inverses whenever they exist.

So what went wrong? A bijection $f : A \rightarrow B$ in **Sets** always has an inverse $g : B \rightarrow A$ such that

$$g \circ f = \mathtt{id}_A \tag{1.1}$$
$$f \circ g = \mathtt{id}_B \tag{1.2}$$

It is defined as follows: if $y \in B$ then $g(y)$ is the unique $x \in A$ such that $f(x) = y$. In a general category there is no way to construct such a g from being balanced; we need to be given g explicitly. Thus we have:

Definition 1.2.8 *A morphism $f : A \to B$ in a category $\mathcal{C}$ is an* isomorphism *if there is a morphism $g : B \to A$ called an* inverse *to f such that Equations (1.1) and (1.2) hold.*

Exercise 1.2.9 Show that if f above is an isomorphism then it is both a monomorphism and an epimorphism, and hence that if g and h are both inverses for f then $g = h$.

This almost completes our "translation" of set-theoretic primitives. However, what happened to subsets? Not every injection $A \to B$ is given by a subset inclusion. Indeed there may be many injections $A \to B$, so how are they to be distinguished? The answer is that categorical notions can't separate out the inclusions. A little reflection shows that this is good! When we say that x is an element of A and of B then the meaning (or typing) of x is overloaded. By forcing us to mention the injection i the ambiguity is removed.

To put it another way, the naming of elements, or their representation, is an implementation issue. if you are vitally concerned with these details then categories aren't going to help much. But for most programming problems, and for reasoning about programs, we are able to ignore these implementation details, and then the categorical approach helps us to do this.

A third way of putting it is that categorical concepts are often defined up to isomorphism, that is, the defined object can be replaced by any other which is isomorphic to it and still satisfy the categorical conditions. For example, if $f : A \to B$ is a monomorphism in **Sets** then we can take the image B_0 of f which is a subset of B (with inclusion j), and f factorises as

$$\begin{tikzcd} & B_0 \arrow[dr, "j"] & \\ A \arrow[ur, "f_0"] \arrow[rr, "f"'] & & B \end{tikzcd}$$

Now f_0 is an isomorphism since f is a monomorphism (exercise) so that, up to isomorphism, every monomorphism is an inclusion!

Let us see this up-to-isomorphism business in another setting.

The cartesian product $A \times B$ of sets A and B consists of ordered pairs $\langle x, y\rangle$ where $x \in A$ and $y \in B$. It has projections $\pi : A \times B \to A$ and $\pi' : A \times B \to B$ with the obvious actions. Performing the same steps as for monomorphisms we are led to the following definition;

Definition 1.2.10 *Let A and B be objects of a category $\mathcal{C}$. A* cartesian product *(or simply* product*) of A and B is given by an object P and a pair of morphisms $\pi : P \to A$ and $\pi' : P \to B$ called* first *and* second projection *respectively such that, for any object X and pair of morphisms $f : X \to A$ and $g : X \to B$ there is a unique morphism $h : X \to P$ such that the following diagram commutes*

$$\begin{tikzcd} & X \arrow[dl, "f"'] \arrow[d, dotted, "h"] \arrow[dr, "g"] & \\ A & P \arrow[l, "\pi"] \arrow[r, "\pi'"'] & B \end{tikzcd}$$

We say that a diagram *commutes* if each pair of paths through the diagram that have the same beginning and end points define the same morphism, with the caveat that we do not require parallel arrows to be equal until they have been composed with something. Thus in the diagram above we have:

$$\begin{aligned} \pi \circ h &= f \\ \pi' \circ h &= g \end{aligned}$$

To check the commutativity of a large diagram, it suffices to check the commutativity of each cell.

Returning to **Sets** we see that both $(A \times B, \pi, \pi')$ and $(B \times A, \pi', \pi)$ are both products of A and B. Choosing between them is an implementation issue.

Exercise 1.2.11 Show that if (P, π, π') and (Q, q, q') are both products of A and B then there is an isomorphism $f : P \rightarrow Q$ which respects the product structure, i.e. $q \circ f = \pi$ and $q' \circ f = \pi'$.

If there is a canonical,or chosen product (perhaps given by the syntax of some language) of A and B then it is usually called $A \times B$.

Chapter 2

Limits and Colimits

2.1 Products

Here are some examples of products:

In **Sets** the categorical product is the usual collection of ordered pairs.

The product of monoids (M, m, e) and (M', m', e') has underlying set $M \times M'$ with the multiplication defined *pointwise* i.e.

$$(x_1, y_1)(x_2, y_2) = (x_1 x_2, y_1 y_2)$$

Products of groups, posets etc. are defined similarly.

Let $(P, \leq)$ be a poset, thought of as a category, and let $p, q \in P$. A product of p and q, if one exists, is their *greatest lower bound* $p \wedge q$. Thus we see that a category need not have any products, or may have some but not others.

Can a monoid, as category, have products? This is not so common, but can be done. Let $M = \mathbf{Sets}(\mathbf{N}, \mathbf{N})$ be the subcategory of **Sets** having one object $\mathbf{N}$ the natural numbers. Further, let $p : N^2 \rightarrow N$ be any isomorphism with inverse q. Now define the pairing of $f, g : \mathbf{N} \rightarrow \mathbf{N}$ in terms of their pairing in **Sets** by $p \circ \langle f, g \rangle$. The projections are $\pi \circ q$ and $\pi' \circ q$.

Exercise 2.1.1 Are projections always epimorphisms? Hint: consider **Sets**.

There are various morphisms definable whenever sufficient products exist:

$$\begin{aligned} a_{A,B,C} &= \langle \pi \circ \pi, \langle \pi' \circ \pi, \pi' \rangle \rangle : (A \times B) \times C \rightarrow A \times (B \times C) \\ c_{A,B} &= \langle \pi', \pi \rangle : A \times B \rightarrow B \times A \\ f \times g &= \langle f \circ \pi, g \circ \pi' \rangle : A \times B \rightarrow A' \times B' \end{aligned}$$

for $f : A \rightarrow A'$ and $g : B \rightarrow B'$.

Exercise 2.1.2 Show that $a_{A,B,C}$ and $c_{A,B}$ are isomorphisms.

The isomorphisms $a_{A.B.C}$ show that once we can define products of arbitrary pairs of objects then we can define products of three, four or more objects, by taking repeated products of pairs. No matter which order these pairings are made the result is determined up to isomorphism. The only outstanding case of finite products is the product of zero objects:

An object 1 in a category $\mathcal{C}$ is a *terminal object* if, for every object A in $\mathcal{C}$ there is a unique morphism $!_A : A \rightarrow 1$. We can think of a morphism $x : 1 \rightarrow B$ as an *(global)element* of A. Often the constant morphism $x \circ ! : A \rightarrow B$ is denoted as simply $x : A \rightarrow B$.

As with binary products, if T and T' are both terminal objects in $\mathcal{C}$ then $T \cong T'$. Here are some additional isomorphisms:

$$\begin{aligned} r_A &= \pi_{A,1} : A{\times}1{\to}A \\ l_A &= \pi'_{1,A} : 1{\times}A{\to}A \end{aligned}$$

2.2 Coproducts

The concept dual to "product" is "coproduct" or "sum". Let A and B be objects in a category $\mathcal{C}$. They have a *coproduct* if there is an object $A + B$ equipped with *injections*

$$A \xrightarrow{\iota_{A,B}} A + B \xleftarrow{\iota'_{A,B}}$$

such that for every object C and pair of morphisms $f : A{\to}C$ and $g : B{\to}C$ there is a unique morphism $[f, g] : A + B{\to}C$ making the following diagram commute.

$$\begin{tikzcd} A \arrow[r, "\iota"] \arrow[dr, "f"'] & A + B \arrow[d, dotted, "{[f,g]}"] & B \arrow[l, "\iota'"'] \arrow[dl, "g"] \\ & C & \end{tikzcd}$$

Sums are useful for representing constructors into a datatype. For example, we can set $\texttt{Bool} = 2 = 1 + 1$ with $\texttt{true} = \iota : 1{\to}2$ and $\texttt{false} = \iota' : 1{\to}2$ so that we have a coproduct diagram

$$1 \xrightarrow{\texttt{true}} \texttt{Bool} \xleftarrow{\texttt{false}} 1$$

which we can use to construct boolean conditions. For example, the following diagram

$$\begin{tikzcd} A \arrow[r, "\iota"] \arrow[d, "!"'] & A + B \arrow[d, "isA"] & B \arrow[l, "\iota'"'] \arrow[d, "!"] \\ 1 \arrow[r, "\texttt{true}"'] & \texttt{Bool} & 1 \arrow[l, "\texttt{false}"] \end{tikzcd}$$

constructs the test $isA = [\texttt{true}\circ!, \texttt{false}\circ!] : A + B{\to}\texttt{Bool}$ for being in A.

More specifically, if we have an object of natural numbers in our category, then we have the following coproduct diagram:

$$1 \xrightarrow{\texttt{zero}} N \xleftarrow{\texttt{succ}} N$$

which is to say that every number is either zero or a successor. Thus we can define $\texttt{isZero} : N{\to}\texttt{Bool}$ and $\texttt{isnotZero} : N{\to}\texttt{Bool}$ by taking A above to be 1 or N.

Two points are worth noting here. First, this is not a definition of the natural numbers - we'll come to that later. Second, recall that the definition of coproduct only specifies it up to isomorphism; the assertion here is that this diagram is one of the many possible coproducts of 1 and N (including perhaps a canonical one labelled $1 + N$).

Lists provide another example of such a construction: an object $\texttt{list}A$ of lists on an object A must satisfy the domain equation

$$\texttt{list}A \cong 1 + (A{\times}\texttt{list}A)$$

Writing this as a coproduct diagram yields

$$1 \xrightarrow{\texttt{nil}} \texttt{list}A \xleftarrow{\texttt{cons}} A{\times}\texttt{list}A$$

where $\texttt{nil}$ and $\texttt{cons}$ have the usual meaning. Now $\texttt{isNil} : \texttt{list}A{\rightarrow}\texttt{Bool}$ is defined according to the same pattern.

Again, we can dualise the notion of terminal object as follows. An object 0 in a category $\mathcal{C}$ is *initial* if, for each object A there is a unique morphism $0{\rightarrow}A$. In **Sets** it is the empty set. Note, however, that in **Mon** the one element monoid is both initial and terminal, in which case it is called a *zero object*.

Exercise 2.2.1 Show that coproducts in $\mathcal{C}$ are products in $\mathcal{C}^{op}$.

2.3 Conditionals

Aside from these very basic operations there is little that can be done simply with products and sums without knowing how they interact. Consider the following morphism:

$$(A{\times}B) + (A{\times}C) \xrightarrow{[\texttt{id}_A{\times}\iota,\, \texttt{id}_A{\times}\iota']} A{\times}(B + C)$$

The *distributive law* (for products over sums) asserts that this morphism is an isomorphism, whose inverse is denoted $d_{A,B,C} : A{\times}(B + C){\rightarrow}(A{\times}B) + (A{\times}C)$. That is,

$$A{\times}(B + C) \cong (A{\times}B) + (A{\times}C)$$

but not by accident, but in a canonical way.

Now we can prove various arithmetic facts, such as

$$\begin{array}{rcl} 2{\times}2 & = & 2{\times}(1+1) \\ & \cong & (2{\times}1) + (2{\times}1) \\ & \cong & 2 + 2 = 4 \end{array}$$

Exercise 2.3.1 Define the boolean connectives, "and" "or" "not" and "implies" by a case analysis.

Further, we can also define the meaning of conditional statements. Let $b : A{\rightarrow}\texttt{Bool}$ be a predicate, i.e. assign truth values to "elements" of A, and let $f, g : A{\rightarrow}C$ be morphisms. The conditional, $\texttt{if}\ b\ \texttt{then}\ f\ \texttt{else}\ g$ can be interpreted by $\texttt{cond}(b, f, g)$ given by

$$A \xrightarrow{\langle b, \texttt{id}\rangle} \texttt{Bool}{\times}A \quad\cong\quad A + A \xrightarrow{[f, g]} C$$

where $\cong$ denotes the isomorphism

$$\texttt{Bool}{\times}A = (1 + 1){\times}A \xrightarrow{d \circ c} (A{\times}1) + (A{\times}1) \xrightarrow{r \times r} A + A$$

2.4 Pullbacks

Let $b : A{\rightarrow}\texttt{Bool}$ be a boolean condition on A. We would like to introduce a subobject A_b of A that represents that part of A on which b is $\texttt{true}$. That b is $\texttt{true}$ on A_b is to say that the following diagram commutes

$$
\begin{tikzcd}
A_b \arrow[r, "i"] \arrow[d] & A \arrow[d, "b"] \\
1 \arrow[r, "\mathtt{true}"'] & \mathtt{Bool}
\end{tikzcd}
$$

Further, we want that any "element" $x : X \rightarrow A$ for which b is **true** lies in A_b. That is, if $b \circ x = \mathtt{true}$ then x there is a unique map $x_b : X \rightarrow A_b$ such that $i \circ x_b = x$

$$
\begin{tikzcd}
X \arrow[dr, "x_b"] \arrow[drr, "x"] \arrow[ddr] & & \\
& A_b \arrow[r, "i"'] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & A \arrow[d, "b"] \\
& 1 \arrow[r, "\mathtt{true}"'] & \mathtt{Bool}
\end{tikzcd}
$$

In other words, A_b is the pullback of b along `true`.

Definition 2.4.1 *A* pullback *of morphisms $f : A \rightarrow C$ and $g : B \rightarrow C$ in a category $\mathcal{C}$ is given by an object P and a pair of morphisms $p : P \rightarrow A$ and $p' : P \rightarrow B$ (called the* projections *such that $f \circ p = g \circ p' : P \rightarrow C$ and for any pair of morphisms $h : X \rightarrow A$ and $k : X \rightarrow B$ such that $f \circ h = g \circ k$ there is a unique morphism $x : X \rightarrow P$ such that $p \circ x = h$ and $p' \circ x = k$.*

$$
\begin{tikzcd}
X \arrow[dr, "x"] \arrow[drr, "k"] \arrow[ddr, "h"'] & & \\
& P \arrow[r, "p'"] \arrow[d, "p"] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d, "g"] \\
& A \arrow[r, "f"'] & C
\end{tikzcd}
$$

If there is a type-constructor for pullbacks then type-checking cannot be performed statically, since this requires the verification of equations involving arbitrary computations. Hence such type-constructors are usually avoided (but see Martin-Löf type theory). General pullbacks are good for semantic analysisis.

Many particular pullbacks arise as a consequence of the existence of datatypes.

For example

$$\begin{tikzcd}
1 \arrow[r, "\mathtt{nil}"] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathtt{list}A \arrow[d, "\mathtt{isNil}"] & A\times\mathtt{list}A \arrow[l, "\mathtt{cons}"'] \arrow[d] \arrow[dl, phantom, "\llcorner", very near start] \\
1 \arrow[r, "\mathtt{true}"'] & \mathtt{Bool} & 1 \arrow[l, "\mathtt{false}"]
\end{tikzcd}$$

and

$$\begin{tikzcd}
\mathtt{list}(A\times B) \arrow[r, "\mathtt{list}\pi'"] \arrow[d, "\mathtt{list}\pi"'] & \mathtt{list}B \arrow[d, "\#"] \\
\mathtt{list}A \arrow[r, "\#"'] & N
\end{tikzcd}$$

where $\mathbf{N}$ is the object of natural numbers (defined categorically in a later lecture!) and # is the length map on lists.

Exercise 2.4.2 Define matrices of A's by a pullback with projections into $\mathtt{listlist}A$ and $\mathbf{N}\times\mathbf{N}$.

Pullbacks can be used to define a category object within a category. In the first lecture it was given as a collection of objects, and for each pair of objects a set of morphisms. The use of the word "collection" was deliberately vague: if there is a set of objects then the category is *small*. For example, the category **Sets** is not small, since there is no set of all sets. The definition of small categories can be generalised.

Let $\mathcal{C}$ be a category which has all pullbacks. A *graph* in $\mathcal{C}$ is given by an *object of objects* O and an *object of arrows* A together with a pair of morphisms $d_0, d_1; A \to O$ called *domain* and *codomain*. Consider the following pullback

$$\begin{tikzcd}
C \arrow[r, "p'"] \arrow[d, "p"'] \arrow[dr, phantom, "\lrcorner", very near start] & A \arrow[d, "d_0"] \\
A \arrow[r, "d_1"'] & O
\end{tikzcd}$$

The original graph is a *category* in $\mathcal{C}$ if there is a morphism $\circ : C \to A$ called *composition* which is associative and unitary in the appropriate sense.

Exercise 2.4.3 Complete the definition of a category above.

Lemma 2.4.4 *If both squares in the following diagram*

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "x"'] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[r, "g"] \arrow[d, "y"] \arrow[dr, phantom, "\lrcorner", very near start] & C \arrow[d, "z"] \\
D \arrow[r, "h"'] & E \arrow[r, "k"'] & F
\end{tikzcd}$$

are pullbacks then the outer rectangle is also a pullback. Conversely, if the outer rectangle and right-hand square are pullbacks then so is the left-hand square.

Proof Let $\alpha : X \to C$ and $\beta : X \to E$ be such that $z \circ \alpha = k \circ h \circ \beta$. Then, since the right-hand square is a pullback, there is a unique morphism $\gamma : X \to B$ such that $g \circ \gamma = \alpha$ and $y \circ \gamma = h \circ \beta$. The latter equation then implies that there is a unique morphism $\delta : X \to A$ such that $f \circ \delta = \gamma$ and $x \circ \delta = \beta$. Thus $(g \circ f) \circ \delta = \alpha$ and $x \circ \delta = \beta$ which shows that δ has the desired property. Conversely, assume that there is another morphism $\delta' : X \to A$ satisfying these equations. Then similar calculations show that $f \circ \delta'$ yield the same results on composition with g or y and so are equal. As $x \circ \delta = x \circ \delta'$ too it follows that $\delta = \delta'$. □

Exercise 2.4.5 Prove the converse above.

Exercise 2.4.6 Dualise the notion of pullback and define pushouts.

2.5 Case Analysis

Coproduct is commonly interpreted as *disjoint* union, i.e. "that nothing in $A + B$ is in both A and B". This translates as requiring that the inclusions to a coproduct be monomorphisms and that the following diagram be a pullback

$$\begin{tikzcd}
0 \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & A \arrow[d, "\iota"] \\
B \arrow[r, "\iota'"'] & A + B
\end{tikzcd}$$

Thus any morphism $X \to A + B$ which factors through both ι and ι' factors through 0.

In **Sets** this is clearly true. In the category of monoids, **Mon**, this is a little more tricky. Recall that the sum of monoids A and B is given by the words in the disjoint union of their elements, subject to some equivalence relation. Now the unit element e of $A + B$ is the image of the units in both A and B and in fact the intersection of ι and ι' is the singleton $\{e\}$ and not the empty set! Note, however, that the empty set is not the underlying set of any monoid, and that the singleton monoid is in fact initial! Thus **Mon** has disjoint coproducts.

Another property that we may wish for our coproducts is "that everything in $A + B$ is either in A or in B". This translates into asking that sums be *stable under pullback*. That is, if $x : X \to A + B$ generates the following pullback diagrams

$$\begin{tikzcd}
X_A \arrow[r, "i_A"] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & X \arrow[d, "x"] & X_B \arrow[l, "i_B"'] \arrow[d] \arrow[dl, phantom, "\llcorner", very near start] \\
A \arrow[r, "\iota"'] & A + B & B \arrow[l, "\iota'"]
\end{tikzcd}$$

then the upper row is a coproduct diagram.

Exercise 2.5.1 Show that **Sets** has stable coproducts, but **Mon** does not.

Stable coproducts allow us to interpret general case analysis as follows. Let $b : X \to A + B$ and $f, g : X \to Y$ be morphisms. Then a program which applies f or g according to whether b lands in A or B can be interpeted by

$$[f \circ i_A, g \circ i_B] : X \to Y$$

2.6 Fixpoints and Invariants

Let $f : A \rightarrow A$ be an endomorphism of a category $\mathcal{C}$. Another name for endomorphism is a *loop*. A morphism $x : X \rightarrow A$ is *fixed* by f if $f \circ x = x$. A *fixed object* for f is given by a morphism $\mathtt{fix}(f) : \mathtt{Fix}(f) \rightarrow A$ which is fixed by f with the property that any $x : X \rightarrow A$ fixed by f factors through $\mathtt{fix}(f)$ in a unique way.

$$\begin{tikzcd} \mathtt{Fix}(f) \arrow[r, "\mathtt{fix}(f)"] & A \\ X \arrow[u, dashed] \arrow[ur, "x"'] & \end{tikzcd}$$

If $\mathcal{C} = \mathbf{Sets}$ then $\mathtt{Fix}(f)$ can be chosen to be the subset of A of elements which are fixed by f.

Dually, a morphism $g : A \rightarrow Q$ is an *invariant* for f if $g \circ f = g$. An *invariant object* for f is an invariant $\mathtt{inv}(f) : A \rightarrow \mathtt{Inv}(f)$ such that if g is any invariant for f then there is a unique morphism $h : \mathtt{Inv}(f) \rightarrow Q$ such that $h \circ \mathtt{inv}(f) = g$.

$$\begin{tikzcd} A \arrow[r, "\mathtt{inv}(f)"] \arrow[dr, "g"'] & \mathtt{Inv}(f) \arrow[d, dashed, "h"] \\ & Q \end{tikzcd}$$

Finally, the loop f on an object C is *convergent* if it has a universal invariant $\mathtt{inv}(f) : C \rightarrow C_0$ and a morphism $m : C_0 \rightarrow C$ which is fixed by f and satisfies

$$\mathtt{inv}(f) \circ m = \mathtt{id}_{C_0}$$

Typically, $m = \mathtt{fix}(f)$.

For example, let $(L, \mathtt{nil}, \mathtt{cons})$ be a (finite) list object for A in a category $\mathcal{C}$ which satisfies the distributive law, e.g. **Sets**. Then the loop $\mathtt{tail} : L \rightarrow L$ can be defined in the usual way (or see below). It has a fixed object $\mathtt{nil} : 1 \rightarrow L$ and an invariant object $! : L \rightarrow 1$ (since all lists become identified with the $\mathtt{nil}$ list) and so $\mathtt{tail}$ converges:

$$1 \xrightarrow{\mathtt{nil}} L \xrightarrow{\;!\;} 1$$

A more interesting example arises when we let $\alpha : C \times A \rightarrow C$ be a right A-action on an object C. Now we can define the loop $\mathtt{shunt}(\alpha) : C \times L \rightarrow C \times L$ by

$$C \times L \cong C \times (1 + (A \times L)) \xrightarrow{d} (C \times 1) + (C \times A \times L) \xrightarrow{[\mathtt{id} \times \mathtt{nil}, \alpha \times \mathtt{id}_L]} C \times L$$

This corresponds to the ML definition

$$\begin{array}{ll} \mathtt{fun} & \mathtt{shunt}(x, \mathtt{nil}) = (x, \mathtt{nil}) \\ \mid & \mathtt{shunt}(x, h :: t) = (\alpha(x, h), t) \end{array}$$

When $C = 1$ then α is unique and $\mathtt{shunt}(\alpha) = \mathtt{tail}$. In general $\mathtt{shunt}(\alpha)$ converges to

$$\mathtt{inv}(\mathtt{shunt}(\alpha)) = \mathtt{foldl}(\alpha)$$

In **Sets** (writing $\alpha(x, a)$ as $x * a$) we have, for example,

$$\mathtt{foldl}(\alpha)(x, [a_0, a_1, a_2]) = (((x * a_0) * a_1) * a_2)$$

The fixpoints are given by $\langle \mathtt{id}, \mathtt{nil} \rangle : C \rightarrow C \times L$.

This description of lists is a theorem with respect to the usual (initial algebra) semantics but can also be used as a tail-recursive *definition* of lists.

2.7 While-loops

Convergence also distinguishes the well-behaved while-loops from the rest. Consider a program

$$\mathtt{while}\ b\ \mathtt{do}\ f$$

We would like to characterise those while-loops which always terminate cleanly, i.e. at a point where b is false.

Think of f as a loop on some object C and $b : C \rightarrow \mathtt{Bool}$ is the test. These can be combined to yield a loop g on C corresponding to $\mathtt{cond}(()b, f, \mathtt{id})$. The picture for the fixpoints and invariants of g (at least in **Sets**) is

$$C_{\neg b} \longrightarrow \mathtt{Fix}(g) \longrightarrow C \longrightarrow C_{\neg b} + C_{\infty}$$

where

$$\begin{aligned} C_{\neg b} &= \{x \mid b(x) = \mathtt{false}\} \\ C_{\infty} &= \{ \text{ infinite loops of } g \ \} \end{aligned}$$

(the latter consists of equivalence classes of x such that $b(f^n(x)) = \mathtt{true}$ for all n).

In general, the fixpoints of g include pairs $\langle \mathtt{true}, x \rangle$ where $f(x) = x$ and the invariants of g include infinite loops.

Both these cases must be excluded for clean termination. Generalising from sets, yields the assertion (or definition):

The while-loop $\mathtt{while}\ b\ \mathtt{do}\ f$ *terminates cleanly iff* g *converges to* $C_{\neg b}$.

2.8 General Limits and Colimits

It is time to unify the concepts of product, terminal object, pullback and fixed object under the general concept of limits. Dually, coproducts, initial objects, pushouts and invariant objects are colimits. We begin with the general definition of a diagram, which requires another look at graphs.

Let I be a graph (in **Sets**) and $\mathcal{C}$ be a category. A *diagram* in $\mathcal{C}$ of type I is a graph morphism $D : I \rightarrow |\mathcal{C}|$. For example, if $I = 2$ has two objects (say 1 and 2) and no edges then a diagram is given by a pair of objects D_1 and D_2 in $\mathcal{C}$. If I has three vertices $1, 2, 3$ and two edges $u : 1 \rightarrow 3$ and $v : 2 \rightarrow 3$ then D is given by a pair of morphisms $D_u : D_1 \rightarrow D_3$ and $D_v : D_2 \rightarrow D_3$

$$\begin{tikzcd} 1 \arrow[dr, "u"] & & & D_1 \arrow[dr, "D_u"] & \\ & 3 & \mapsto & & D_3 \\ 2 \arrow[ur, "v"'] & & & D_2 \arrow[ur, "D_v"'] & \end{tikzcd} \tag{2.1}$$

Finally, if I has two objects $1, 2$ and two arrows $u, v : 1 \to 2$ then D yields a parallel pair of arrows $D_u, D_v : D_1 \to D_2$.

A *cone* for a diagram $D : I \to |\mathcal{C}|$ consists of an object X and a family of morphisms $x_i : X \to D_i$ indexed by the vertices of I such that for each edge $u : i \to j$ of I we have

$$\begin{tikzcd}
 & D_i \arrow[dd, "D_u"] \\
X \arrow[ur, "x_i"] \arrow[dr, "x_j"'] & \\
 & D_j
\end{tikzcd}$$

A *limit* for a diagram $D : I \to |\mathcal{C}|$ is a universal cone for it, i.e. if $(X, \{x_i : X \to D_i\})$ is any other cone for D then there is a unique morphism $x : X \to L$ such that $p_i \circ x = x_i$ for every vertex i.

$$\begin{tikzcd}
 & & D_i \arrow[dd, "D_u"] \\
X \arrow[urr, "x_i"] \arrow[r, "x"] \arrow[drr, "x_j"'] & L \arrow[ur, "p_i"] \arrow[dr, "p_j"'] & \\
 & & D_j
\end{tikzcd}$$

For example, if $I = \{1, 2\}$ then a limit for D is a product $D_1 \times D_2$. More generally if I is a *set* i.e. has no edges then a limit of D is a product $\prod_i D_i$ of the D_i whose projections may be called π_i. In particular, if I is empty (has no vertices) then there is a unique diagram of type I in $\mathcal{C}$ and every object has a unique cone over it. Thus a limit for D is a terminal object. If I is as in Diagram 2.1 then a limit is given by an object P and three morphism $p_i : P \to D_i$ such that

$$D_u \circ p_1 = p_3 = D_v \circ p_2$$

Equivalently, a cone is given by p_1 and p_2 which must satisfy

$$D_u \circ p_1 = D_v \circ p_2$$

with p_3 derived as the common value of the sides of this equation. Thus P is the pullback of D_u and D_v.

Finally, consider the diagram

$$\begin{tikzcd}
A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & B
\end{tikzcd}$$

Its limit, if it exists, is called the *equaliser* of f and g and is determined by the limit object E and the projection $e : E \to A$ from the limit into A. The projection into B is then the common value of $f \circ e$ and $g \circ e$. Any other morphism $x : X \to A$ such that $f \circ x = g \circ x$ must factor through e by some $y : X \to A$.

Dually, a cocone for a diagram $D : I \to \mathcal{C}$ is given by an object Q and a family of morphisms $q_i : D_i \to Q$

$$\begin{tikzcd} D_i \arrow[dd, "D_u"] \arrow[dr, "q_i"] & \\ & Q \\ D_j \arrow[ur, "q_j"'] & \end{tikzcd}$$

A *colimit* for D is then a universal cocone for it. Coproducts, etc arise in this way. The colimit of a parallel pair is its *coequaliser*.

Theorem 2.8.1 *A category $\mathcal{C}$ has all (finite) limits if it has equalisers and all (finite) products.*

Proof Let $D : I \to \mathcal{C}$ be a diagram. A cone for D is specified by an object X and a family of morphisms $x_i : X \to D_i$ or simply a morphism $y = \langle x_i \rangle : X \to \prod D_i$. Further, for each edge $u : i \to j$ in I we must satisfy

$$D_u \circ \pi_i \circ y = \pi_j \circ y$$

i.e. that y factorises through the equaliser of $D_u \circ \pi_i$ and π_j. Now this family of equalisers can be coded up as one big equaliser of a parallel pair of maps into a product indexed by the edges of I

$$\prod_i D_i \underset{g}{\overset{f}{\rightrightarrows}} \prod_{u:i \to j} D_j$$

where $\pi_u \circ f = D_u \circ \pi_i \circ y$ and $\pi_u \circ g = \pi_j \circ y$. Now cones over D correspond to maps into $\prod_i D_i$ which equalise f and g and the limit is given by their equaliser. $\square$

Exercise 2.8.2 A category has all finite limits iff it has all pullbacks and a terminal object.

Chapter 3

Categories as Objects

3.1 Functors

A *functor* is a structure-preserving morphism of categories. That is, a functor $F : \mathcal{C} \to \mathcal{D}$ between categories $\mathcal{C}$ and $\mathcal{D}$ is given by a graph morphism that preserves composition and identities. In more detail, F maps objects of $\mathcal{C}$ to objects of $\mathcal{D}$, and arrows of $\mathcal{C}$ to arrows of $\mathcal{D}$ so that domains and codomains are preserved,

$$\mathcal{C} \xrightarrow{\quad F \quad} \mathcal{D}$$

$$\mathcal{C}(A,B) \xrightarrow{F_{A,B}} \mathcal{D}(FA,FB)$$

$$\begin{array}{ccc} A & & FA \\ {\scriptstyle f}\Big\downarrow & \mapsto & \Big\downarrow{\scriptstyle Ff} \\ B & & FB \end{array}$$

and for $f : A \to B$ and $g : B \to C$ in $\mathcal{C}$ we have

$$\begin{aligned} F(g \circ f) &= (Fg) \circ (Ff) \\ F(\mathtt{id}_A) &= \mathtt{id}_{FA} \end{aligned}$$

Consider the following examples.

Let $\mathcal{C}$ be a category of sets with structure and structure-preserving morphisms, e.g. **Mon** the category of monoids and monoid homomorphisms. Then there is a forgetful functor, $U : \mathbf{Mon} \to \mathbf{Sets}$ which takes each monoid (M, m, e) to its underlying set M and each homomorphism $f : (M, m, e) \to (M', m', e')$ to its underlying function $f : M \to M'$. It isn't necessary to check that composition and identities are preserved, since these are *defined* in terms of the composition and identities in **Sets**.

Now consider M and M' as categories with one object. A functor

$$f : (M, m.e) \to (M', m', e')$$

maps the unique object of M to the unique object of M' and the morphisms (elements) of M to those of M'. Preservation of composition and identities then amounts to preserving the composition and its identity, so that f is exactly a monoid homomorphism. Conversely, every monoid homomorphism induces a unique functor.

Let $(P, \leq)$ and $(Q, \leq)$ be posets, regarded as categories. Then a map from objects to objects is a function $f : P \rightarrow Q$. Recall that there is a unique morphism $x \rightarrow y$ iff $x \leq y$. Thus f determines a functor iff $x \leq y$ implies $f(x) \leq f(y)$. The preservation of composition and identities is automatic. In summary, a functor is exactly an order-preserving function.

Here is an example of a free functor $F : \mathbf{Sets} \rightarrow \mathbf{Mon}$. It takes a set X to the free monoid FX on X whose elements are the words $x_1 x_2 \ldots x_n$ (including the empty word) with multiplication given by concatenation. If $f : X \rightarrow Y$ is a function then $Ff : FX \rightarrow FY$ is given by

$$Ff(x_1 x_2 \ldots x_n) = f(x_1) f(x_2) \ldots f(x_n)$$

The details are left to the reader.

The (small) graphs and graph morphisms form a category $\mathbf{Gph}$. If $F : \mathcal{A} \rightarrow \mathcal{B}$ and $G : \mathcal{B} \rightarrow \mathcal{C}$ are graph morphisms then $G \circ F : \mathcal{A} \rightarrow \mathcal{C}$ is the graph morphism which maps an vertex A of $\mathcal{A}$ to GFA and an edge $f : A \rightarrow B$ to $GFf : GA \rightarrow GFB$. Similarly the (small) categories and functors form a category $\mathbf{Cat}$. If $F : \mathcal{A} \rightarrow \mathcal{B}$ and $G : \mathcal{B} \rightarrow \mathcal{C}$ are functors then their composite as graph morphisms preserves composition and identities too.

By definition there is a forgetful functor $U : \mathbf{Cat} \rightarrow \mathbf{Gph}$. There is also a free functor $F : \mathbf{Gph} \rightarrow \mathbf{Cat}$ which maps a graph $\mathcal{G}$ to $F\mathcal{G}$ whose objects are the vertices of $\mathcal{G}$. A morphism $A \rightarrow B$ is a *path* from A to B in $\mathcal{G}$ i.e. a sequence of edges $f_i : A_i \rightarrow A_{i+1}$ in $\mathcal{G}$ such that $A = A_0$ and $B = A_n$. Composition is given by concatenation and identities are the empty paths. If $f : \mathcal{G} \rightarrow \mathcal{H}$ is a graph morphism then Ff has the same action on objects as f and maps the path of f_i's to the path of Ff_i's. Note that the free monoid construction is a special case.

A *subcategory* $\mathcal{C}_0$ of a category $\mathcal{C}$ is given by a collection of some of the objects of $\mathcal{C}$ and for each pair of objects A, B of $\mathcal{C}_0$ a subset $\mathcal{C}_0(A, B) \subseteq \mathcal{C}(A, B)$ which together are closed under composition and identities. Of course, the old problem of distinguishing monomorphisms from subsets arises again. For example, are $\mathbf{Mon}$ and $\mathbf{Pre}$ subcategories of $\mathbf{Cat}$? This depends on whether we define a monoid to be a category with one object or a set equipped with some operations and equations. The same sort of issues arise for pre-orders.

Thus we must describe the properties of the functors $F : \mathbf{Mon} \rightarrow \mathbf{Cat}$ and $G : \mathbf{Pre} \rightarrow \mathbf{Cat}$. The mapping of objects is not very informative since the collection of objects varies with the style of presentation of monoids. For example, if they are to be categories with one object then we must name the object, which leads to many different presentations compared to simply naming the homset (the set-theoretic monoid). However, we established above that for monoids A and B the mapping

$$F_{A,B} : \mathbf{Mon}(A, B) \rightarrow \mathbf{Cat}(FA, FB)$$

is a bijection. This information is independent of the way in which we represent monoids.

Consider the action $F_{A,B} : \mathcal{C}(A, B) \rightarrow \mathcal{D}(FA, FB)$ of a functor $F : \mathcal{C} \rightarrow \mathcal{D}$ on an arbitrary homset. If it is always an injection then F is *faithful*. If it is always a surjection then F is *full*. If it is always a bijection then F is *fully faithful*. The inclusions of $\mathbf{Mon}$ and $\mathbf{Pre}$ into $\mathbf{Cat}$ are both fully faithful.

Here are a few more examples of functors:

The powerset $\mathcal{P}A$ of a set A is the set of all subsets of A. Like many constructions it is functorial, but this construction also illustrates a warning: the mapping of objects doesn't always determine the mapping of morphisms. This is obvious enough for categories with one object: here is an example on **Sets**. There are two obvious ways to make P a functor. The *inverse image power functor*

$$\mathcal{P}_0 : \mathbf{Sets}^{op} \rightarrow \mathbf{Sets}$$

maps $f : A \rightarrow B$ in **Sets** to $\mathcal{P}_0 f = f^{-1} : \mathcal{P}B \rightarrow \mathcal{P}A$ where

$$f^{-1}(B_0) = \{x \in A | f(x) \in B_0\}$$

is the inverse image of B_0 under f. The *direct image power functor*

$$\mathcal{P}_1 : \mathbf{Sets} \rightarrow \mathbf{Sets}$$

satisfies

$$\mathcal{P}_1 f(A_0) = f(A_0) = \{f(x) | x \in A_0\}$$

for f as above. You may wish to confirm the functoriality of these constructions.

A functor $\mathcal{C}^{op} \rightarrow \mathcal{D}$ is often called a *contravariant functor* from $\mathcal{C}$ to $\mathcal{D}$. In such a setting a functor $F : \mathcal{C} \rightarrow \mathcal{D}$ may be called a *covariant functor* for emphasis. Of course, one could object, on a technicality, to this example of freedom in defining functors since the functors have different variances. Well, this doesn't reduce the choice when considering the powerset construction alone, but if you are not yet convinced, there is at least one other covariant powerset functor.

Exercise 3.1.1 Show that the following construction yields another covariant powerset functor.

$$\mathcal{P}_2 f(A_0) = \{b \in B \mid f(a) = b \text{ implies } a \in A_0\}$$

Let $\mathcal{C}$ be a category which has chosen binary products. Then the product construction takes a pair of objects, i.e. an object of (A, B) of $\mathcal{C}^2$ and produces an object $A \times B$ of $\mathcal{C}$. The existence fragment of the universal property of the product allows the extension of the construction to morphisms. If $f : A \rightarrow A'$ and $g : B \rightarrow B'$ then

$$f \times g = \langle f \circ \pi, g \circ \pi' \rangle : A \times B$$

Functoriality equations follow by the uniqueness part of the universal property.

Exercise 3.1.2 Prove functoriality of the product.

Let I be an object in a category $\mathcal{C}$. The *slice* $\mathcal{C}/I$ of $\mathcal{C}$ by I has as objects morphisms $\alpha : A \rightarrow I$ of $\mathcal{C}$ with codomain I. Given $\beta : B \rightarrow I$ then morphisms $\alpha \rightarrow \beta$ in $\mathcal{C}/I$ are morphisms $f : A \rightarrow B$ in $\mathcal{C}$ such that

$$\begin{tikzcd}
A \arrow[rr, "f"] \arrow[dr, "\alpha"'] & & B \arrow[dl, "\beta"] \\
& I &
\end{tikzcd}$$

commutes. When $\mathcal{C} = \mathbf{Sets}$ then an object of $\mathbf{Sets}/I$ corresponds to an I indexed family $\{A_i | i \in I\}$ of sets where $A_i = \alpha^{-1}(i)$. Conversely, given such a family of sets then define A to be their disjoint union and define $\alpha : A \rightarrow I$ to map A_i to i. Then a morphism in the slice is just a function which respects the indexing, i.e. a family of functions $f : A_i \rightarrow B_i$. Composition and identities are constructed as in $\mathcal{C}$. Hence there is a faithful functor $U_I : \mathcal{C}/I \rightarrow \mathcal{C}$.

If $\mathcal{C}$ also has finite products then there is also a canonical functor $F_I : \mathcal{C} \rightarrow \mathcal{C}/I$ which maps A to $\pi' : A \times I \rightarrow I$ and $f : A \rightarrow B$ to $f \times \mathtt{id} : A \times I \rightarrow B \times I$.

Let $\mathcal{C}$ and $\mathcal{D}$ be categories. Then for each object D of $\mathcal{D}$ there is a constant functor $K_D : \mathcal{C} \rightarrow \mathcal{D}$ which maps every object to D and every morphism to $\mathtt{id}_D$.

For each object A in a category $\mathcal{C}$ there are a pair of *hom-functors*, one of which is covariant

$$\mathcal{C}(A, -) : \mathcal{C} \rightarrow \mathbf{Sets}$$

while the other is contravariant

$$\mathcal{C}(-, A) : \mathcal{C}^{op} \rightarrow \mathbf{Sets}$$

The covariant one maps an object B to $\mathcal{C}(A, B)$ and a morphism $g : B \rightarrow C$ to

$$\begin{array}{ccc} \mathcal{C}(A, B) & \xrightarrow{\mathcal{C}(A, g)} & \mathcal{C}(A, C) \\ \\ f & \mapsto & g \circ f \end{array}$$

The contravariant one maps B to $\mathcal{C}(B, A)$ and $g : B \rightarrow C$ to $(-) \circ g : \mathcal{C}(C, A) \rightarrow \mathcal{C}(B, A)$.

Let $\mathcal{C}$ be a category which supports list datatypes, e.g. **Sets**. Then there is a functor $L : \mathcal{C} \rightarrow \mathcal{C}$ defined by $LA = A\ list$ and $Lf = map(f)$ for $f : A \rightarrow B$.

3.2 Natural Transformations

A functor $F : \mathcal{C} \rightarrow \mathcal{D}$ can be thought of as constructing an image of $\mathcal{C}$ in $\mathcal{D}$ or as a translation of the "language" of $\mathcal{C}$ into that of $\mathcal{D}$. Given another such translation (functor) G of $\mathcal{C}$ into $\mathcal{D}$ we may wish to compare them. This can be done by giving, for each object A in $\mathcal{C}$ a transformation $\alpha_A : FA \rightarrow GA$ which compares the two images of A. Not any morphism will do, however, as we would like the construction to be parametric in A, rather than ad hoc. Since A is an object of $\mathcal{C}$ while FA is in $\mathcal{D}$ we cannot link them by a morphism. Rather the goal is that the transformation should respect the "operations" i.e. the morphisms of $\mathcal{C}$. In other words, the following diagram should commute

$$\begin{tikzcd} FA \arrow[r, "\alpha_A"] \arrow[d, "Ff"'] & GA \arrow[d, "Gf"] \\ FB \arrow[r, "\alpha_B"'] & GB \end{tikzcd}$$

Thus the interpretations of f by F and G are compatible with the transformation under α.

Formally, a *natural transformation* $\alpha : F \Rightarrow G : \mathcal{C} \rightarrow \mathcal{D}$ consists of a family of morphisms $\alpha_A : FA \rightarrow GA$ in $\mathcal{D}$ called its *components* such that for all morphisms $f : A \rightarrow B$ in $\mathcal{C}$ the diagram above commutes.

Let me emphasise that a natural transformation has one component for each object of $\mathcal{C}$, which together satisfy one equation for each morphism of $\mathcal{C}$.

Consider the following examples:

Let $f, g : P \rightarrow Q$ be functors (order-preserving functions) between pre-orders (qua categories). Then there is a (necessarily unique) natural transformation $f \Rightarrow g$ denoted $f \leq g$ iff $f(x) \leq g(x)$ for all x in P. These latter inequalities are the components of the natural transformation. The commutativity conditions hold automatically.

Let $f, g : M \to N$ be monoid homomorphisms regarded as functors. Then a natural transformation $\alpha : f \Rightarrow g$ has just one component (also called α) which satisfies

$$\alpha f(x) = g(x)\alpha$$

for any x in M (and multiplication is given by concatenation). This condition will be more familiar to those who know a little group theory. In that context α has an inverse and $f(x) = \alpha^{-1} g(x)\alpha$ shows that $f = h \circ g$ where $h(x) = \alpha^{-1} x \alpha$ is an inner automorphism.

Let $\mathcal{C}$ be a category with finite products. Then the projections are natural, in the following sense. Define $F, G, H : \mathcal{C}^2 \to \mathcal{C}$ on objects (A, B) by $F(A, B) = A \times B$ and $G(A, B) = A$ and $H(A, B) = B$ with the obvious extensions to morphisms. Then $\pi : F \to G$ has components $\pi_{A,B} =: A \times B \to A$ and $\pi' : F \to H$ has components $\pi' : A \times B \to B$. Naturality of π and π' follow from the definition of $f \times g = \langle f \circ \pi, g \circ \pi' \rangle$.

Exercise 3.2.1 Show that the diagonal morphisms

$$\delta_A = \langle \mathtt{id}_A, \mathtt{id}_A \rangle : A \to A \times A$$

are natural for an appropriate choice of functors.

This exercise illustrates the typical situation. One begins with a parametric operation, and then tries to prove it natural. The founders of category theory, Eilenberg and Mac Lane said that in order to define naturality they were lead to functors, which lead them to categories.

Constructors for generic types are interpreted by coproduct conclusions, which are natural. For example, `nil` and `cons` are natural

$$\begin{aligned} \mathtt{nil} : K_1 &\Rightarrow \mathit{list} \\ \mathtt{cons} : \mathtt{id} \times \mathit{list} &\Rightarrow \mathit{list} \end{aligned}$$

since the following diagrams commute

$$\begin{tikzcd} 1 \arrow[r, "\mathtt{nil}"] \arrow[d, "\mathtt{id}"'] & \mathit{list}A \arrow[d, "\mathit{list} f"] \\ 1 \arrow[r, "\mathtt{nil}"'] & \mathit{list}B \end{tikzcd} \qquad \begin{tikzcd} A \times A\mathit{list} \arrow[r, "\mathtt{cons}"] \arrow[d, "f \times f\mathit{list}"'] & A\mathit{list} \arrow[d, "f\mathit{list}"] \\ B \times B\mathit{list} \arrow[r, "\mathtt{cons}"'] & B\mathit{list} \end{tikzcd}$$

It can be shown that folding preserves naturality, so that append = foldright(nil, cons) is natural, etc.

Of course, the natural transformations are closed under identities and composition. If $\alpha : F \Rightarrow G : \mathcal{C} \to \mathcal{D}$ and $\beta : G \Rightarrow H$ are natural transformations then $\beta \circ \alpha : F \Rightarrow H$ is natural with components $\beta_A \circ \alpha_A : FA \to HA$. Thus the functors from $\mathcal{C}$ to $\mathcal{D}$ and the natural transformations between them form a category $\mathbf{Cat}(\mathcal{C}, \mathcal{D}) = \mathcal{D}^{\mathcal{C}}$.

Let $f : A \to B$ be a morphism in $\mathcal{C}$. Then

$$\mathcal{C}(f, -) : \mathcal{C}(B, -) \Rightarrow \mathcal{C}(A, -)$$

is a natural transformation between hom-functors whose component at C is $\mathcal{C}(f, C) = (-) \circ f$ given by pre-composition with f i.e. maps $g : B \to C$ to $g \circ f : A \to C$. Naturality is assured by the associativity of composition. Dually, $\mathcal{C}(-, f) : \mathcal{C}(-, A) \Rightarrow \mathcal{C}(-, B)$ is defined by post-composition with f.

The mappings $A \mapsto \mathcal{C}(-,A)$ and $f \mapsto \mathcal{C}(-,f)$ describes the *Yoneda functor*

$$y : \mathcal{C} \to \mathbf{Sets}^{\mathcal{C}^{op}}$$

from $\mathcal{C}$ into its category of *presheaves*. Functoriality is again given by the associativity and unicity of composition. This functor is of great interest because it maps an arbitrary category into a presheaf category, which inherits most (some would say *all*) of the good properties of **Sets**. Thus information about y is tantamount to information about $\mathcal{C}$. Consider the following results.

Lemma 3.2.2 (Yoneda) *Let $F : \mathcal{C} \to \mathbf{Sets}$ be a functor and let A be an object of $\mathcal{C}$. Then there is a natural bijection between natural transformations $\mathcal{C}(A,-) \Rightarrow F$ and elements of FA.*

Proof Consider a natural transformation $\alpha : \mathcal{C}(A,-) \Rightarrow F$. It is determined by its components $\alpha_B : \mathcal{C}(A,B) \to FB$. Its naturality forces the commutativity of the following square

$$\begin{tikzcd}
\mathcal{C}(A,A) \arrow[r, "\alpha_A"] \arrow[d, "\mathcal{C}(A,f)"'] & FA \arrow[d, "Ff"] \\
\mathcal{C}(A,B) \arrow[r, "\alpha_B"'] & FB
\end{tikzcd}$$

for each morphism $f : A \to B$. Thus

$$\begin{aligned}
\alpha_B(f) &= \alpha_B(f \circ \mathtt{id}) \\
&= \alpha_B \circ \mathcal{C}(A,f)(\mathtt{id}_A) \\
&= Ff \circ \alpha_A(\mathtt{id}_A) \\
&= Ff(x)
\end{aligned}$$

where $x = \alpha_A(\mathtt{id}_A)$ is the desired element of FA. Thus α is determined by x. Conversely, every $x \in FA$ determines a transformation by $\alpha_B(f) = Ff(x)$. It is natural since for any $g : B \to C$ the commutativity of

$$\begin{tikzcd}
\mathcal{C}(A,B) \arrow[r, "\alpha_B"] \arrow[d, "\mathcal{C}(A,g)"'] & FB \arrow[d, "Fg"] \\
\mathcal{C}(A,C) \arrow[r, "\alpha_C"'] & FC
\end{tikzcd}$$

follows from $Fg(Ff(x)) = F(g \circ f)(x)$ for any $f : A \to B$.

Naturality in A follows directly. □

Of course, there is a dual version of the Yoneda Lemma obtained by replacing $\mathcal{C}$ by $\mathcal{C}^{op}$ which asserts that natural transformations $\mathcal{C}(-,A) \Rightarrow F : \mathcal{C}^{op} \to \mathbf{Sets}$ are in bijection with elements of FA.

Corollary 3.2.3 *The Yoneda functor is fully faithful.*

Proof Let A and B be objects of $\mathcal{C}$. Then the dual Yoneda Lemma shows that natural transformations $\mathcal{C}(-,A) \Rightarrow \mathcal{C}(-,B)$ are in bijection with elements of $\mathcal{C}(A,B)$. □

This corollary formalises the categorists' notion that an object is determined by knowing either the morphisms into it, or the morphisms out of it, since if there is a natural bijecion between morphisms out of A and morphisms out of B

$$\mathcal{C}(A,-)\cong\mathcal{C}(B,-)$$

then

$$A\cong B$$

Exercise 3.2.4 Verify this last statement.

The Yoneda Lemma has many other applications. Here is another example. One can prove that $A\times B$ and $B\times A$ are isomorphic by explicit constuction, but it is quicker to outline a proof using Yoneda, by providing a natural isomorphism between morphisms $X\to A\times B$ and $X\to B\times A$ as follows (where the horizontal lines denote a natural bijection between the data above and below)

$$\frac{\dfrac{\dfrac{X\to B\times A}{X\to A \qquad\qquad X\to B}}{X\to B \qquad\qquad X\to A}}{X\to B\times A}$$

Of course, one must understand what the naturality of each step means.

3.3 The Godement Calculus

Consider the following diagram of functors and natural transformations

$$\begin{tikzcd} \mathcal{A} \arrow[r, bend left, "F"] \arrow[r, bend right, "F'"'] \arrow[r, phantom, "\Downarrow\alpha"] & \mathcal{B} \arrow[r, bend left, "G"] \arrow[r, bend right, "G'"'] \arrow[r, phantom, "\Downarrow\beta"] & \mathcal{C} \end{tikzcd}$$

We want to define a composite natural transformation between them, but will proceed by stages. First, define $\beta_F : GF\to G'F$ to have components $(\beta_F)_A = \beta_{FA}$. Its naturality follows from that of β restricted to the image of F. Second, define $G\alpha : GF\to GF'$ to have components $(G\alpha)_A = G(\alpha_A)$. Its naturality follows from that of α and the fact that functors preserve commuting diagrams:

$$\begin{tikzcd} GFA \arrow[r, "G\alpha_A"] \arrow[d, "GFf"'] & GF'A \arrow[d, "GF'f"] \\ GFB \arrow[r, "G\alpha_B"'] & GF'B \end{tikzcd}$$

Now we have two choices for the composition of α and β, namely $\beta_{F'}\circ G\alpha$ or $G'\alpha\circ\beta_F$. Fortunately, they are both equal

$$\begin{tikzcd} GFA \arrow[r, "\beta_{FA}"] \arrow[d, "G\alpha_A"'] & G'FA \arrow[d, "G'\alpha_A"] \\ GF'A \arrow[r, "\beta_{F'A}"'] & G'F'A \end{tikzcd}$$

by the naturality of β. Define their common value to be the *horizontal composite* $\beta * \alpha$ of α and β.

Thus, in **Cat** we have hom-*categories* with (horizontal) composition defined on both the objects and the morphisms (the natural transformations). In fact, we have the following:

Theorem 3.3.1 *Horizontal composition in* **Cat** *is functorial, and is associative and unitary.*

Proof Consider the following diagram of functors and natural transformations:

$$\begin{tikzcd}
\mathcal{A} \arrow[r, shift left=4, "F"] \arrow[r, phantom, shift left=2, "\Downarrow\alpha"] \arrow[r, "F'" description] \arrow[r, phantom, shift right=2, "\Downarrow\alpha'"] \arrow[r, shift right=4, "F''"'] & \mathcal{B} \arrow[r, shift left=4, "G"] \arrow[r, phantom, shift left=2, "\Downarrow\beta"] \arrow[r, "G'" description] \arrow[r, phantom, shift right=2, "\Downarrow\beta'"] \arrow[r, shift right=4, "G''"'] & \mathcal{C}
\end{tikzcd}$$

That horizontal composition

$$\mathbf{Cat}(\mathcal{B},\mathcal{C})\times\mathbf{Cat}(\mathcal{A},\mathcal{B}) \longrightarrow \mathbf{Cat}(\mathcal{A},\mathcal{C})$$

preserves composition is the assertion that

$$(\beta'\beta) * (\alpha'\alpha) = (\beta' * \alpha')(\beta * \alpha)$$

more usually known as the *interchange law*. It follows since

$$\begin{aligned}
((\beta' * \alpha') \circ (\beta * \alpha))_A &= (\beta'_{F''A} \circ G'\alpha'_A) \circ (\beta_{F'A} \circ G\alpha_A) \\
&= \beta'_{F''A} \circ \beta_{F''A} \circ G\alpha'_A \circ G\alpha_A) \\
&= (\beta' \circ \beta) * (\alpha' \circ \alpha)_A
\end{aligned}$$

which follows by unwinding the definitions and the naturality of β. The other equations follow similarly. $\square$

Thus we may think of **Cat** as a category whose homs and composition are not mere sets and functions, but categories and functors, i.e. a *2-category*. (Aside: this is a fundamental example of an *enriched category*, i.e. a category whose homs are objects of some *base category* e.g. **Sets** or **Cat** or **Pos**, and whose composition is given by morphisms of the base category.) The resulting collection of equations and their manipulation is known as the Godement calculus.

Chapter 4

Algebraic Theories

A *single-sorted algebraic theory* is usually given by specifying a family of operations with given arities, and some equations expressed in a term algebra generated from variables and the application of operations of type n to n-tuples of terms. For example, the theory of monoids is given by two operations, namely a multiplication m of arity 2 and a unit e which is a constant (arity 0). The associativity and unitary laws are then given by the equations

$$\begin{aligned} m(m(x,y),z) &= m(x,m(y,z)) \\ m(e,x) &= x = m(x,e) \end{aligned}$$

Note that the equations are inseparable from the construction of the term algebra, and that *substitution* is the fundamental operation.

Lawvere clarified both these points by representing the operations as the morphisms of a category, whose objects are the arities. Thus the equations become equalities of operations without mention of variables, and substitution is broken into two processes, tupling (the cartesian product) and composition. Thus the category $\mathbf{T}$ which is the theory of a monoid has for objects the natural numbers and has morphisms generated by the basic operations $m : 2 \to 1$ and $e : 0 \to 1$ and closed under the product structure (projections and diagonals) and composition. These must satisfy the equations generated by

$$\begin{aligned} m \circ (m \times \mathtt{id}) &= m \circ (\mathtt{id} \times m) \\ m \circ \langle e, \mathtt{id} \rangle &= \mathtt{id} = m \circ \langle \mathtt{id}, e \rangle \end{aligned}$$

which is diagramatically represented by

$$\begin{tikzcd} 3 \arrow[r, "m\times\mathtt{id}"] \arrow[d, "\mathtt{id}\times m"'] & 2 \arrow[d, "m"] \\ 2 \arrow[r, "m"'] & 1 \end{tikzcd}$$

and

$$\begin{tikzcd} 1 \arrow[r, "\langle e,\mathtt{id}\rangle"] \arrow[dr, "\mathtt{id}"'] & 2 \arrow[d, "m"] & 1 \arrow[l, "\langle \mathtt{id},e\rangle"'] \arrow[dl, "\mathtt{id}"] \\ & 1 & \end{tikzcd}$$

In general, a *finite product theory* is simply a category with finite products. The single-sorted case we are focussing on here is a a category $\mathcal{T}$ whose objects are the natural numbers, with n being the product of n copies of 1. Thus we have projections $\pi_i : n \rightarrow 1$ and all the usual diagonals and symmetries etc. A morphism $\omega : n \rightarrow 1$ is an *operation* of arity n. A general morphism $f : m \rightarrow n$ is thus of the form $f = \langle \omega_1, \omega_2, \ldots, \omega_n \rangle$ by the product structure.

Starting from a collection of generating operations and equations as given above for monoids, one can construct the corresponding category. We will not go into this theory of *sketches* here, but the interested reader can pursue the details in Barr and Wells.

So, algebraic theories are categories. Of much deeper significance, however, is that models are functors. More precisely, a model of $\mathcal{T}$ in a category $\mathcal{C}$ with finite products is a product-preserving functor $F : \mathcal{T} \rightarrow \mathcal{C}$. For example, take $\mathcal{T}$ to be the theory of monoids and let $F : \mathcal{T} \rightarrow \mathbf{Sets}$ be a model. If $F1 = M$ then $Fn = (F1)^n = M^n$ and

$$\begin{array}{rcl} Fm : M^2 & \rightarrow & M \\ Fe : M^0 = 1 & \rightarrow & M \end{array}$$

are its multiplication and unit. The associativity and unicity of the abstract multiplication of $\mathcal{T}$ then imply, by functoriality, the same equations for the multiplication of M.

Perhaps you can see the next result coming: a natural transformation between models is exactly an algebra homomorphism, since it is a translation between the two interpretations of $\mathcal{T}$ in $\mathcal{C}$. Let $\alpha : F \rightarrow G$ be such a natural transformation. Let $F1 = A$ and $G1 = B$, and define $f = \alpha_1 A \rightarrow B$. Then the commutativity of

$$\begin{tikzcd} A^n \arrow[r, "\alpha_n"] \arrow[d, "\pi_i"'] & B^n \arrow[d, "\pi_i"] \\ A \arrow[r, "f"'] & B \end{tikzcd}$$

shows that the ith component of α_n is f whence $\alpha_n = f^n$. Thus, α is determined by its action at 1. Let $\omega : n \rightarrow 1$ be an operation in $\mathcal{T}$. Then naturality of α shows that

$$\begin{tikzcd} A^n \arrow[r, "f^n"] \arrow[d, "\omega_A"'] & B^n \arrow[d, "\omega_B"] \\ A \arrow[r, "f"'] & B \end{tikzcd}$$

commutes, i.e. f is an algebra homomorphism. The proof of the converse, that every algebra homomorphism yields a natural transformation, is direct. Note also that there is a forgetful functor $U : \mathcal{T} - Alg \rightarrow \mathcal{C}$ given by evaluation at 1, which amounts to throwing away the structure information (how the operations are interpreted).

To summarise: theories are categories, models are functors and homomorphisms are natural transformations, and we can define the category of algebras for the theory $\mathcal{T}$ to be the category $\mathcal{T} - Alg$ of product-preserving functors and all natural transformations between them.

Chapter 5

Monads

Monads can be defined in any 2-category, but we will restrict ourselves to their description in **Cat**.

Consider again an algebraic theory $\mathcal{T}$. Given a set X we can construct a term algebra TX which is generated by X and closed under application of the operations, subject to the equations of the theory. When $\mathcal{T}$ is the theory of monoids then TX consists of words in the alphabet X. In general this construction can be quite messy to explain fully, but the work of closing up under substitution and equations has already been done in defining $\mathcal{T}$ and need not be repeated here. Thus we can define a functor $T : \mathbf{Sets} \to \mathbf{Sets}$ as follows. Let

$$T_0X = \{\omega(x_1, x_2, \ldots, x_n) | \omega \in \mathcal{T}(n,1), x_i \in X\}$$

and if $f : X \to Y$ is a function then let

$$T_0f(\omega(x_1x_2\ldots x_n)) = \omega(f(x_1)f(x_2)\ldots f(x_n))$$

This set is too large since we want to identify, for example, the terms $\pi(x,y)$ and x which differ only in their context. Define

$$\begin{array}{l} \hat{\pi}_i = \langle \pi_1, \pi_2, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_n \rangle : \\ X_1 \times X_2 \times \ldots \times X_n \to X_1 \times X_2 \times \ldots X_{i-1} \times X_{i+1} \times \ldots \times X_n \end{array}$$

and let TX be the quotient of T_0X by the equivalence relation generated by

$$\omega \circ \hat{\pi}_i(x_1, x_2, \ldots, x_n) \equiv \omega(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

Clearly T_0f respects the equivalence relation, and so induces $Tf : TX \to TY$. We will abuse the notation and use elements of T_0X to represent their equivalence classes in TX.

Further, we have a pair of natural transformations $\eta : \mathtt{id} \Rightarrow T$ and $\mu : T^2 \Rightarrow T$ with components $\eta_X(x) = \mathtt{id}_1(x)$ and

$$\mu_X(\omega(\omega_1(x_{11}, x_{12}, \ldots, x_{1n_1}), \ldots, \omega_m(x_{m1}, \ldots, x_{mn_m}))) = \langle \omega_1, \ldots, \omega_m \rangle (x_{11}, \ldots, x_{mn_m})$$

Note that T^2X consists of words in the alphabet TX so that its elements have two kinds of multiplication (concatenation). The μ simply identifies them.

Further, these two natural transformations satisfy some equations, called *unicity* of μ

$$\begin{tikzcd} TX \arrow[r, "\eta_{TX}"] \arrow[dr, "\mathtt{id}"'] & T^2X \arrow[d, "\mu_X"] & TX \arrow[l, "T\eta_X"'] \arrow[dl, "\mathtt{id}"] \\ & TX & \end{tikzcd}$$

and its *associativity*

$$\begin{tikzcd} T^3X \arrow[r, "\mu_{TX}"] \arrow[d, "T\mu_X"'] & T^2X \arrow[d, "\mu_X"] \\ T^2X \arrow[r, "\mu_X"'] & TX \end{tikzcd}$$

This leads us to define a *monad* $\mathbf{T} = (T, \eta, \mu)$ on a category $\mathcal{C}$ (sometimes called a *triple*) to consist of a functor $T : \mathcal{C} \to \mathcal{C}$ equipped with a pair of natural transformations $\eta : \mathtt{id}_{\mathcal{C}} \Rightarrow T$ and $\mu : T^2 \to T$ satisfying equations (5) and (5).

Here are some examples.

Every single-sorted algebraic theory yields a monad on **Sets** whose functor maps a set X to its term algebra.

We can define a monad $\mathbf{T} = (T, \eta, \mu)$ on **Pos** where TP is $P_\perp$ the partial order obtained from P by adding a new element $\perp$ which is below every element of P. The inclusion of P in $P_\perp$ is η and $\mu_P : (P_\perp)_\perp \to P_\perp$ maps both of the introduced $\perp$ to $\perp$. The desired equations hold since all maps are identities on the image of P and all introduced elements are mapped to $\perp$ in $P_\perp$. This monad can be used to represent partial functions.

If $\mathcal{C}$ has products and (M, m, e) is a monoid in $\mathcal{C}$ then the functor $TX = M \times X$ yields a monad with

$$\begin{aligned} \eta_X = \langle e, \mathtt{id}_X \rangle &\quad : \quad X \to M \times X \\ \mu_X = m \times \mathtt{id} &\quad : \quad M \times M \times X \to M \times X \end{aligned}$$

The axioms are exactly the unicity and associativity of the monoid. This monad can be used to introduce a parameter or context.

If E is an object in $\mathcal{C}$ which has coproducts then the functor $TX = X + E$ yields a monad with unit given by inclusion and multiplication induced by the co-diagonal $[\mathtt{id}, \mathtt{id}] : E + E \to E$. This can be used to model exceptions.

Let P : **Sets**$\to$**Sets** be the direct image power functor. It has unit $A \to PA$ which maps $x \in A$ to $\{x\}$ and multiplication given by set-theoretic union. It can be used to model non-determinism.

Consider again a general monad $\mathbf{T}$ as above. A $\mathbf{T}$-algebra is given by an object A and a *structure morphism* $a : TA \to A$ which satisfies the following pair of equations

$$\begin{tikzcd} A \arrow[r, "\eta_A"] \arrow[dr, "\mathtt{id}"'] & TA \arrow[d, "a"] \\ & A \end{tikzcd} \qquad \begin{tikzcd} T^2A \arrow[r, "Ta"] \arrow[d, "\mu_A"'] & TA \arrow[d, "a"] \\ TA \arrow[r, "a"'] & A \end{tikzcd}$$

The structure morphism "interprets" the $\mathbf{T}$-structure inside A. An *algebra homomorphism* $f : (A, a) \to (B, b)$ between algebras is given by a morphism $f : A \to B$ in $\mathcal{C}$ which is compatible with the algebra structure, i.e.

$$\begin{tikzcd} TA \arrow[r, "Tf"] \arrow[d, "a"'] & TB \arrow[d, "b"] \\ A \arrow[r, "f"'] & B \end{tikzcd}$$

Clearly these are closed under the composition and identities of $\mathcal{C}$ and so form a category called $\mathcal{C}^{\mathbf{T}}$ which, when $\mathcal{C} = \mathbf{Sets}$, may be called $\mathbf{T}$-Alg. There is a forgetful functor $U^{\mathbf{T}} : \mathcal{C}^{\mathbf{T}} \rightarrow \mathcal{C}$ which maps a homomorphism $f : (A, a) \rightarrow (B, b)$ to $f : A \rightarrow B$.

There is also a free functor $F^{\mathbf{T}} : \mathcal{C} \rightarrow \mathcal{C}^{\mathbf{T}}$ which takes an object X in $\mathcal{C}$ to its *free* $\mathbf{T}$-algebra defined by

$$F^{\mathbf{T}} X = (TX, \mu_X)$$

For example the free monoid on X is the words on X with multiplication given by concatenation. We will consider the technical meaning of freeness in later lectures. If $f : X \rightarrow Y$ in $\mathcal{C}$ then $F^{\mathbf{T}} f = Tf$ is an algebra homomorphism by the naturality of μ.

Of course, the composite $U^{\mathbf{T}} F^{\mathbf{T}}$ is just T.

Thus if we begin with an algebraic theory $\mathcal{T}$ we have two categories of algebras, namely $\mathcal{T}$-Alg and $\mathbf{T}$-Alg.

Theorem 5.0.2 *Let $\mathcal{T}$ be a single-sorted algebraic theory. Then the categories $\mathcal{T}$-Alg and $\mathbf{T}$-Alg are equivalent.*

Proof We will prove is that there are functors $F : \mathcal{T} - Alg \rightarrow \mathbf{T} - Alg$ and $G : \mathbf{T} - Alg \rightarrow \mathcal{T} - Alg$ together with natural isomorphisms $\alpha : \mathtt{id} \Rightarrow GF$ and $\beta : FG \Rightarrow \mathtt{id}$. This is not quite enough to prove that the two categories are isomorphic, since this would imply that $GF = \mathtt{id}$ and $FG = \mathtt{id}$. However, there are many product-preserving functors $\mathcal{T} \rightarrow \mathbf{Sets}$ with the same value at the object 1, which are merely isomorphic. Thus we have a many-one relationship between the $\mathcal{T}$-algebras and the $\mathbf{T}$-Algebras.

Let $M : \mathcal{T} \rightarrow \mathbf{Sets}$ be a model with $M1 = A$. Define $a : TA \rightarrow A$ by

$$a(\omega(x_1, x_2, \ldots, x_n) = (M\omega)(x_1, x_2, \ldots, x_n) \tag{5.1}$$

Then $a \circ \eta(x) = a(\mathtt{id}(x)) = (M\mathtt{id})(x) = x$ and

$$\begin{aligned} a \circ Ta(\omega(\omega_i(x_1, x_2, \ldots, x_n))) &= a(\omega(M\omega_i(x_1, x_2, \ldots, x_n))) \\ &= (M\omega)(M\omega_i(x_1, x_2, \ldots, x_n)) \\ &= M(\omega\langle\omega_i\rangle)(x_1, x_2, \ldots, x_n) \\ &= a \circ \mu(\omega(\omega_i(x_1, x_2, \ldots, x_n))) \end{aligned}$$

Thus (A, a) is a $\mathbf{T}$-algebra.

Now if $\alpha : M \rightarrow N$ is a morphism of models, i.e. a natural transformation, then $\alpha_1 = f : A \rightarrow B$ is a $\mathbf{T}$-algebra homomorphism. Let $b : TB \rightarrow B$ be the induced structure map on B. Thus we have have

$$\begin{aligned} b \circ Tf(\omega(x_1, x_2, \ldots, x_n)) &= (N\omega)(fx_1, fx_2, \ldots, fx_n) \\ &= (N\omega \circ \alpha_n(x_1, x_2, \ldots, x_n) \\ &= \alpha_1(M\omega)(x_1, x_2, \ldots, x_n) \\ &= f \circ a(\omega(x_1, x_2, \ldots, x_n) \end{aligned}$$

which shows that f is a homomorphism. Functoriality of the construction F of $\mathbf{T}$-algebras (respectively homomorphisms) from $\mathcal{T}$-algebras (respectively homomorphisms) is immediate.

Conversely, given a $\mathbf{T}$-algebra (A, a) define a functor $G(A, a) = M : \mathcal{T} \rightarrow \mathbf{Sets}$ by setting $Mn = A^n$ and defining $M\omega$ by Equation (5.1). That M is a product-preserving functor follows directly. Given a $\mathbf{T}$-algebra homomorphism

$$f : (A, a) \rightarrow (B, b)$$

define a natural transformation $\alpha : M \Rightarrow N$ between the corresponding models by $\alpha_n = f^n$. Naturality follows directly.

The verification that $GF(M) \cong M$ and $FG(A,a) \cong (A,a)$ are left to the reader. $\square$

There is another category habitually associated to a monad $\mathbf{T}$ on $\mathcal{C}$ namely its Kleisli category $\mathcal{C}_{\mathbf{T}}$. It has the same objects as $\mathcal{C}$ but morphisms are given by

$$\mathcal{C}_{\mathbf{T}}(A,B) = \mathcal{C}(A,TB)$$

Composition of $f : A \to B$ and $g : B \to C$ in $\mathcal{C}_{\mathbf{T}}$ is given by

$$\begin{tikzcd}
 & & T^2C \arrow[d, "\mu"] \\
 & TB \arrow[ur, "Tg"] & TC \\
A \arrow[ur, "f"'] & B \arrow[ur, "g"'] & C
\end{tikzcd}$$

$\mu_C \circ Tg \circ f : A \to TC$ in $\mathcal{C}$. Associativity follows from that of μ while identity on A is given by the unit $\eta_A : A \to TA$

$$\begin{aligned}
\mu_B \circ Tf \circ \eta_A &= \mu_B \circ \eta_{TB} \circ f = f \\
\mu_B \circ T\eta_B \circ f &= f
\end{aligned}$$

The category of algebras $\mathcal{C}^{\mathbf{T}}$ (also known as the Eilenberg-Moore category) and the the Kleisli category were discovered simultaneously. It used to be thought that the algebras were central and that the Kleisli category was relatively useless, but recent uses of monads in computing place the Kleisli category in the central role.

For example, let $\mathbf{T}$ be the monad corresponding to an algebraic single-sorted theory. Then its Kleisli category is the category of substitutions. That is, a morphism $X \to Y$ maps elements of the set X to terms in the free algebra on Y. Kleisli composition exactly describes the way that substitutions are combined.

We have a forgetful functor $U_{\mathbf{T}} : \mathcal{C}_{\mathbf{T}} \to \mathcal{C}$ which maps an object A to TA and maps $f : A \to B$ in $\mathcal{C}_{\mathbf{T}}$ (i.e. $f : A \to TB$ in $\mathcal{C}$) to $f^\dagger = \mu_B \circ Tf : TA \to TB$. There is also a functor $F_{\mathbf{T}} : \mathcal{C} \to \mathcal{C}_{\mathbf{T}}$ in the opposite direction which maps an object A to itself, and maps $f : A \to B$ to $\eta_B \circ f$.

Exercise 5.0.3 Confirm that $U_{\mathbf{T}}$ and $F_{\mathbf{T}}$ are functors.

Note also that the composite $U_{\mathbf{T}}F_{\mathbf{T}}A = TA$ and $U_{\mathbf{T}}F_{\mathbf{T}}f = \mu \circ T(\eta \circ f) = Tf$ so that $U_{\mathbf{T}}F_{\mathbf{T}} = T$. Thus we have two decompositions of T as a composite of functors. These two are in fact related by yet another functor

$$\begin{tikzcd}
\mathcal{C}_{\mathbf{T}} \arrow[r, "\phi"] & \mathcal{C}^{\mathbf{T}}
\end{tikzcd}$$

$$\begin{tikzcd}
A \arrow[d, "f"'] & & TA \arrow[d, "f^\dagger"] \\
B & \mapsto & TB
\end{tikzcd}$$

which makes the following diagram commute:

$$\begin{tikzcd}
\mathcal{C}_{\mathbf{T}} \arrow[r, "\phi"] \arrow[d, shift left, "U_{\mathbf{T}}"] & \mathcal{C}^{\mathbf{T}} \arrow[d, shift left, "U^{\mathbf{T}}"] \\
\mathcal{C} \arrow[u, shift left, "F_{\mathbf{T}}"] \arrow[r, phantom, "="] & \mathcal{C} \arrow[u, shift left, "F^{\mathbf{T}}"]
\end{tikzcd}$$

Exercise: Show that ϕ is fully faithful. It follows that $\mathcal{C}_{\mathbf{T}}$ may be thought of as the sub-category $\mathcal{C}^{\mathbf{T}}$ of free algebras and all homomorphisms between them.

Chapter 6

Adjunctions

6.1 Universal Properties

Consider the construction of the free monoid FX on a set X. It is called the free monoid because for every function $f : X \to UM$ from X into the underlying set of a monoid, there is a unique monoid homomorphism $g : FX \to M$ making the following diagram commute:

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "f"'] & UFX \arrow[d, "Ug"] & FX \arrow[d, "g"] \\
& UM & M
\end{tikzcd}$$

namely $g(x_1 x_2 \ldots x_n) = f(x_1)f(x_2)\ldots f(x_n)$.

Thus the action of f determines the action of g on the whole of the monoid FX, i.e. the elements of X generate FX, and any such function will do, without restriction, i.e. FX is *freely* generated.

Let us generalise the situation as follows, while keeping our example to hand for reference. Consider a functor $G : \mathcal{B} \to \mathcal{A}$ and an object A of $\mathcal{A}$. A *universal arrow* from A to G is given by an object FA of $\mathcal{B}$ and a morphism $\eta_A : A \to GFA$ such that for every morphism $f : A \to GB$ in $\mathcal{A}$ there is a unique morphism $g : FA \to B$ in $\mathcal{B}$ such that the following diagram commutes

$$\begin{tikzcd}
A \arrow[r, "\eta_A"] \arrow[dr, "f"'] & GFA \arrow[d, "Gg"] & FA \arrow[d, "g"] \\
& GB & B
\end{tikzcd}$$

Assume now that every object in $\mathcal{A}$ has a universal arrow to G. This has many consequences. First, η is a natural transformation. More precisely, to be natural requires that the definition of F be extended to morphisms. The naturality of η

then asserts that for each morphism $f : A \to A'$ the following square commutes

$$\begin{tikzcd} A \arrow[r, "\eta_A"] \arrow[d, "f"'] & GFA \arrow[d, "GFf"] & & FA \arrow[d, "Ff"] \\ A' \arrow[r, "\eta_{A'}"'] & GFA' & & FA' \end{tikzcd}$$

The universal property of η_A determines Ff and also forces its functoriality.

In our example the induced functor is of course the free monoid functor, whose action on a function $f : X \to Y$ is

$$Ff(x_1 x_2 \ldots x_n) = f(x_1)f(x_2)\ldots f(x_n)$$

Of course the free construction of monoids is typical of a wealth of examples of free constructions of algebras, such as rings, vector spaces, posets, etc.

6.2 Natural bijections

We can reformulate universality as follows. Given a morphism $g \in \mathcal{B}(FA, B)$ we can always obtain a morphism $\phi_{A,B}(g) = Gg \circ \eta_A \in \mathcal{A}(A, GB)$. The universality of η asserts that $\phi_{A,B}$ is an isomorphism or, since we are in **Sets**, a bijection. This isomorphism is commonly expressed by the picture

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

(Note that the data above and below the lines are in different categories.) Further, the naturality of η implies that ϕ is natural in A and B. Hopefully, the naturality of ϕ is plausible before realising what the functors are. The full picture is:

$$\phi : \mathcal{B}(F-, ?) \Rightarrow \mathcal{A}(-, G?) : \mathcal{A}^{op} \times \mathcal{B} \to \mathbf{Sets}$$

Its naturality asserts the commutativity of

$$\begin{tikzcd} A' & \mathcal{B}(FA', B) \arrow[r, "\phi"] \arrow[d, "\mathcal{B}(Ff, g)"'] & \mathcal{A}(A', GB) \arrow[d, "\mathcal{A}(f, Gg)"] & B \arrow[d, "g"] \\ A \arrow[u, "f"] & \mathcal{B}(FA, B') \arrow[r, "\phi"'] & \mathcal{A}(A, GB') & B' \end{tikzcd}$$

That is, for $h : FA' \to B$

$$\begin{aligned} Gg \circ (Gh \circ \eta) \circ f &= Gg \circ Gh \circ GFf \circ \eta \text{ naturality of } \eta \\ &= G(g \circ h \circ Ff) \circ \eta \text{ functoriality of } G \end{aligned}$$

Exercise 6.2.1 Show that if $\alpha : H \Rightarrow K : \mathcal{A} \to \mathcal{B}$ is a natural transformation whose components are all isomorphisms then α has an inverse natural transformation.

Thus the inverse $\psi_{A,B} : \mathcal{A}(A, GB) \to \mathcal{B}(FA, B)$ of $\phi_{A,B}$ is also natural.

How is this natural bijection to be interpreted? The functors F and G are not inverses, since we can see from our example that neither GF nor FG need be isomorphic to an identity functor. One way of thinking about this is to recall the analogy between functors and translations and make it literal. Let $\mathcal{A}$ be the world of German speakers and $\mathcal{B}$ be that of French speakers. Let $F : \mathcal{A} \to \mathcal{B}$ translate into French from German and $G : \mathcal{B} \to \mathcal{A}$ translate into German from French. Now it is impossible to expect that translating from one language to another and back will leave the meaning unchanged. However, there remain two ways for a German A to communicate a message to a Frenchman B. Either the message is given in French with A translating, which we can represent as $FA \to B$, or the message is given in German with B translating, represented as $A \to GB$. In the first case B may think he is listening to a Frenchman, while in the second A may think he is speaking to a German. The natural bijection then corresponds to the assertion that these two distinct ways of communicating are equivalent.

Consider a category $\mathcal{C}$ which has chosen limits of all diagrams D of type I denoted by

$$\mathtt{lim}D \xrightarrow{\lambda_i} D_i$$

Thus

$$\frac{X \to \mathtt{lim}D}{X \to D}$$

is a natural bijection. Where are the functors? The cones from X to D are natural transformations $\Delta X \Rightarrow D$.

$$\begin{tikzcd}
\Delta i = X \arrow[r, "x_i"] \arrow[d, "\Delta u = 1"'] & Di \arrow[d, "Du"] \\
\Delta j = X \arrow[r, "x_j"'] & Dj
\end{tikzcd}$$

where ΔX is the functor which is constantly X. Thus Δ is a functor $\mathcal{C} \to \mathcal{C}^I$ and the cone x is a morphism $\Delta X \to D$ in $\mathcal{C}^I$. The limit construction is a functor $\mathtt{lim} : \mathcal{C}^I \to \mathcal{C}$ and the natural bijection is

$$\frac{X \to \mathtt{lim}D}{\Delta X \to D}$$

6.3 Triangle Laws

With ϕ as our starting point we can recover η by $\eta_A = \phi(\mathtt{id}_{FA})$. Dually, from ψ we obtain another natural transformation $\varepsilon : FG \Rightarrow \mathtt{id}$ whose components are given by

$$\varepsilon_B = \psi(\mathtt{id}_{GB}) : FGB \to B$$

Naturality of ε amounts to showing that for $g \in \mathcal{B}(B, B')$ we have $g \circ \varepsilon = \varepsilon \circ FGg$ which will follow by showing both sides equal to $\psi(Gg)$. Naturality of ψ in its second argument shows that

$$\begin{tikzcd}
\mathcal{A}(GB, GB) \arrow[r, "\psi"] \arrow[d, "Gg \circ -"'] & \mathcal{B}(FGB, B) \arrow[d, "g \circ -"] \\
\mathcal{A}(GB, GB') \arrow[r, "\psi"'] & \mathcal{B}(FGB, B')
\end{tikzcd}$$

which when applied to $\mathtt{id}_{GB}$ shows that $\psi(Gg) = g \circ \varepsilon$. The other half of the argument follows from the naturality of ψ in its first argument.

At this point we have functors $G : \mathcal{B} \rightarrow \mathcal{A}$ and $F : \mathcal{A} \rightarrow \mathcal{B}$ and natural transformations $\eta : \mathtt{id}_{\mathcal{A}} \Rightarrow GF$ and $\varepsilon : FG \Rightarrow \mathtt{id}_{\mathcal{B}}$. Using the calculus of functors and natural transformations, we can construct many diagrams from this data. Consider the following pair.

$$
\begin{tikzcd}
& GFGB \arrow[dr, "G\varepsilon_B"] & \\
GB \arrow[ur, "\eta_{GB}"] \arrow[rr, "\mathtt{id}"'] & & GB
\end{tikzcd}
\qquad
\begin{tikzcd}
& FGFA \arrow[dr, "\varepsilon_{FA}"] & \\
FA \arrow[ur, "F\eta_A"] \arrow[rr, "\mathtt{id}"'] & & FA
\end{tikzcd}
\tag{6.1}
$$

That ϕ is a natural bijection forces these diagrams to commute. They are called the *triangle laws*. They follow since

$$
\begin{aligned}
G\varepsilon_B \circ \eta_{GB} &= \phi(\psi(\mathtt{id}_{GB})) = \mathtt{id}_{GB} \\
\varepsilon_{FA} \circ F\eta_A &= \psi(\phi(\mathtt{id}_{FA})) = \mathtt{id}_{FA}
\end{aligned}
$$

The fact that the triangle laws can be expressed in the general calculus of functors and natural transformations leads us to base the definition of adjunctions on them.

Definition 6.3.1 *An* adjunction *is given by a pair of functors $F : \mathcal{A} \rightarrow \mathcal{B}$ and $G : \mathcal{B} \rightarrow \mathcal{A}$ (called the* left *and* right *adjoints respectively) and a pair of natural transformations $\eta : \mathtt{id}_{\mathcal{A}} \Rightarrow GF$ and $\varepsilon : FG \Rightarrow \mathtt{id}_{\mathcal{B}}$ (called respectively the* unit *and* counit*) which satisfy the two triangle laws (6.1). The data is represented by $(F, G, \eta, \varepsilon) : \mathcal{B} \rightarrow \mathcal{A}$ or, when the unit and counit are understood, by $F \dashv G$ or $G \vdash F$.*

We have seen that universal arrows yield hom-bijections which in turn give us adjunctions. In fact the unit of an adjunction gives us a family of universal arrows into G.

$$
\begin{tikzcd}
A \arrow[r, "\eta_A"] \arrow[dr, "f"'] & GFA \arrow[d, "Gg"] \\
& GB
\end{tikzcd}
\qquad
\begin{tikzcd}
FA \arrow[d, "g = \varepsilon_B \circ Ff"] \\
B
\end{tikzcd}
$$

If there is a morphism $h : FA \rightarrow B$ such that $Gh \circ \eta = f$ then

$$
\begin{aligned}
\varepsilon_B \circ Ff &= \varepsilon_B \circ F(Gh \circ \eta_) \\
&= \varepsilon_B \circ FGh \circ F\eta_A \\
&= h \circ \varepsilon_{FA} \circ F\eta_A \\
&= h
\end{aligned}
$$

by the naturality of ε and the triangle laws. Thus if there is such an h it is unique. The proof that g above does actually have this property follows similarly.

This completes a circle of argument showing the equivalence of all the concepts we have introduced, which is summarised in the following theorem.

Theorem 6.3.2 *Any one of the following items provides the data for an adjunction:*

- *$(F, G, \eta, \varepsilon) : \mathcal{B} \rightarrow \mathcal{A}$ with η and ε satisfying the triangle laws (6.1).*

- *A functor $G : \mathcal{B} \to \mathcal{A}$ and a family of universal arrows $\eta_A : A \to GFA$ from each object A of $\mathcal{A}$ to G.*
- *A pair of functors $F : \mathcal{A} \to \mathcal{B}$ and $G : \mathcal{B} \to \mathcal{A}$ and a natural bijection $\phi_{A,B}$: $\mathcal{B}(FA, B) \to \mathcal{A}(A, GB)$*

Corollary 6.3.3 *If a functor $G : \mathcal{B} \to \mathcal{A}$ has a left adjoint it is unique up to isomorphism. That is, if $F, F' : \mathcal{A} \to \mathcal{B}$ are both left adjoints with units η and η' respectively then there is a unique isomorphism $\alpha : F \to F'$ such that $\alpha \circ \eta = \eta'$.*
Proof

$$\begin{tikzcd} A \arrow[r, "\eta_A"] \arrow[dr, "\eta'_A"'] & GFA \arrow[d, "G\alpha_A"] \\ & GF'A \end{tikzcd}$$

Details are left to the reader. □

As if the three presentations of adjunctions above were not enough, the whole structure dualises, to provide three more characterisations.

Lemma 6.3.4 *If $F \dashv G : \mathcal{B} \to \mathcal{A}$ then $F^{op} \vdash G^{op} : \mathcal{B}^{op} \to \mathcal{A}^{op}$.*
Proof Dualising doesn't change the direction of functors, but does change that of natural transformations, whose components reverse direction. Thus we have $\varepsilon^{op} : \mathtt{id}_{\mathcal{B}^{op}} \Rightarrow F^{op}G^{op}$ and $\eta^{op} : G^{op}F^{op} \Rightarrow \mathtt{id}_{\mathcal{A}^{op}}$. Reversing the original triangle laws yields those for the dual adjunction. □

Thus a functor $F : \mathcal{A} \to \mathcal{B}$ has a right adjoint iff there is a family of universal arrows *from* F to each of the objects B in $\mathcal{B}$.

$$\begin{tikzcd} GB & & FGB \arrow[r, "\varepsilon_B"] & B \\ A \arrow[u, dashed, "f"] & & FA \arrow[u, "Ff"] \arrow[ur, "g"'] & \end{tikzcd}$$

Another class of examples is given by the inclusions of subcategories, often full, whose objects (and perhaps arrows) have some additional property not possessed by arbitrary things. For example, let $U : \mathbf{Pos} \to \mathbf{Pre}$ be the inclusion functor from posets to preorders. It has a left adjoint R which maps a pre-order P to its quotient under the equivalence relation $x \equiv y$ iff $x \leq y \leq x$. Now, let Q be a poset. A homomorphism $f : P \to UQ$ must map equivalent elements of P to the same element of Q, and so induces a unique map $g : RP \to Q$ in **Pos**. Thus the quotient map $P \to URP$ is universal from P to U, and we have an adjunction. Note that in this case the counit $Q \to RUQ$ is the identity. When this happens we say that U has a *reflection* R and that **Pos** is a *reflective subcategory* of **Pre**. Dually, if the unit is an isomorphism then we have a *co-reflection*.

Exercise 6.3.5 Show that the inclusion of the category $\mathbf{Pos}_\perp$ of posets with a least element (and $\perp$-preserving morphisms) into **Pos** has a left adjoint.

Exercise 6.3.6 Construct left and right adjoints to the forgetful functor $\mathbf{Pos} \to \mathbf{Sets}$.

6.4 Some Theorems

Theorem 6.4.1 *If* $(F, G, \eta, \varepsilon) : \mathcal{B} \to \mathcal{A}$ *is an adjunction then* $\mathbf{T} = (T, \eta, \mu)$ *is a monad on* $\mathcal{A}$ *where* $T = GF$ *and* $\mu = G\varepsilon F : GFGF \to GF$.
Proof Confirm that the following equations hold:

$$\begin{aligned}
\mu_A \circ T\eta_A &= G\varepsilon_{FA} \circ GF\eta_A \\
&= G(\varepsilon_{FA} \circ F\eta_A) = \mathtt{id} \\
\mu_A \circ \eta_{TA} &= G\varepsilon_{FA} \circ \eta_{GFA} = \mathtt{id} \\
\mu_A \circ T\mu_A &= G\varepsilon_{FA} \circ GFG\varepsilon_{FA} \\
&= G(\varepsilon_{FA} \circ FG\varepsilon_{FA} \\
&= G(\varepsilon_{FA} \circ \varepsilon_{FGFA}) \\
&= \mu_A \circ \mu_{TA}
\end{aligned}$$

□

Thus every adjunction induces a monad. What about the converse?

Proposition 6.4.2 *Let* $\mathbf{T} = (T, \eta, \mu)$ *be a monad on* $\mathcal{C}$. *Then* $F^{\mathbf{T}} \dashv U^{\mathbf{T}} : \mathcal{C}^{\mathbf{T}} \to \mathcal{C}$.
Proof Let $(B, b : TB \to B)$ be a $\mathbf{T}$-algebra. Given a morphism $f : A \to B$ it follows that $g = b \circ Tf : TA \to B$ is an algebra homomorphism from (TA, μ_A) to (B, b) (Exercise). Conversely, if g is such an algebra homomorphism then define $f = g \circ \eta_A : A \to B$. To see that this is a bijection, observe that

$$\begin{aligned}
b \circ Tf \circ \eta &= b \circ \eta \circ f = f \\
b \circ T(g \circ \eta) &= b \circ Tg \circ T\eta \\
&= g \circ \mu \circ T\eta = g
\end{aligned}$$

Now composition with η is natural, and we are done. □

It is not the case, however, that there is a unique adjunction which yields a given monad, as can be seen from the following

Proposition 6.4.3 *Let* $\mathbf{T} = (T, \eta, \mu)$ *be a monad on* $\mathcal{C}$. *Then* $F_{\mathbf{T}} \dashv U_{\mathbf{T}} : \mathcal{C}_{\mathbf{T}} \to \mathcal{C}$.
Proof The equality $\mathcal{C}(A, U_{\mathbf{T}}B) = \mathcal{C}_{\mathbf{T}}(A, B)$ is a natural isomorphism. □

Theorem 6.4.4 *Right adjoints preserve all limits that exist. Dually, left adjoints preserve all colimits.*
Proof Let $F \dashv G : \mathcal{B} \to \mathcal{A}$ and let $\lambda_i : L \to D_i$ be a limit cone for a diagram $D : I \to \mathcal{B}$. Then for any object X in $\mathcal{A}$ we have the following sequence of natural bijections

$$\frac{\dfrac{\dfrac{X \to GD_i}{FX \to D_i}}{FX \to L}}{X \to GL}$$

The dual follows by considering $F^{op} \vdash G^{op}$. □

Chapter 7

Cartesian Closed Categories and λ-calculus

Let $\mathcal{C}$ be a category with all finite products (including the empty one). We saw above that the product functor $\times : \mathcal{C}^2 \to \mathcal{C}$ (if it exists) is right adjoint to the diagonal (constant) functor K. Now $\mathcal{C}$ is *cartesian closed* if for every one of its objects B the functor $(-)\times B : \mathcal{C}\to\mathcal{C}$ has a right adjoint $B\to(-) : \mathcal{C}\to\mathcal{C}$

$$\begin{tikzcd} \mathcal{C} \arrow[r, shift left=1ex, "(-)\times B"] \arrow[r, phantom, "\perp"] & \mathcal{C} \arrow[l, shift left=1ex, "B\to(-)"] \end{tikzcd}$$

The object $B\to C$ is a *function type* which is often denoted by C^B and called the the *exponential*. The universal property is given by the bijection

$$\frac{A\times B\to C}{A\to C^B}$$

If $f : A\times B\to C$ then the corresponding morphism is its *transpose*, denoted $\check{f} : A\to C^B$ or its *currying*, perhaps denoted $\Lambda(f)$.

In **Sets** the exponential C^B is the set of functions $B\to C$ which is exactly the Bth power of C (e.g. $3^2 = 9$). In **Pos** the exponential is the poset of order-preserving functions. It often happens in concrete categories (categories of sets with structure) that the set of morphisms $B\to C$ inherits the structure of C pointwise, and can be made into an exponential object of the category.

Consider a poset H as a category with finite products, given by the meet. Then H is cartesian closed iff it is a Heyting algebra, i.e. we have the bi-implication

$$\frac{p\wedge q\leq r}{p\leq q\to r}$$

There are, however, many interesting categories which aren't cartesian closed, for example, the category of topological spaces and continuous functions.

Notice that since $(-)\times B$ is a left adjoint it preserves all colimits and in particular

$$(A+A')\times B \cong (A\times B) + (A'\times B)$$

The counit of the adjunction defining $B\to(-)$ is the *evaluation*

$$(B\to C)\times B \xrightarrow{\mathtt{eval}_{B,C}} C$$

Lemma 7.0.5 *The exponential construction $B \mapsto B \to (-)$ is functorial in B and in fact $(?) \to (-) : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{C}$ is a functor.*

Proof Exercise □

Rather than attempt further categorical motivation for cartesian closure, let us proceed to interpret λ-calculus in cartesian closed categories. In fact, there is a very tight correspondence between the following three settings

- types and terms
- propositions and proofs
- objects and morphisms

when correctly interpreted. This correspondence is not limited to a particular framework, and has become something of a slogan, often called the Curry-Howard correspondence, though many think Lambek should be included, at least for the connections with categories. We will interpret lambda-terms as proofs, and then interpret the proofs in a cartesian closed category.

Consider the simply-typed λ-calculus with surjective pairing and unit type generated by a collection of base types $X, Y, Z, \ldots$ and base operation symbols $f : X \to Y$ etc. i.e. a graph G. The types are given by

$$T ::= X \mid T_1 \to T_2 \mid T_1 \times T_2 \mid 1$$

where X is a base type and 1 is the *unit* or *terminal* type. There are infinitely many *variables* of each type. *Contexts* are lists of variables with their assigned types $\Gamma = \{x_i : X_i\}$. If x does not appear in Γ then the notation $\Gamma, x : X$ represents the consing of $x : X$to Γ. The terms will be defined by giving some proof theory whose theorems are assertions of the form

$$\Gamma \vdash t : T$$

which asert that in context Γ we have t is a term of type T. The axiom of the calculus is:

$$x : X \vdash x : X$$

The *logical rules* are:

$$\frac{\Gamma \vdash t : T}{\Gamma, x : X \vdash t : T}\ (Weak) \qquad \frac{\Gamma' \vdash t : T}{\Gamma \vdash t : T}\ (Ex)$$

$$\frac{\Gamma, x : A \vdash t : B \quad \Gamma \vdash a : A}{\Gamma \vdash t[a/x] : B}\ (Cut)$$

where Γ' is a permutation of Γ and $t[a/x]$ denotes the term obtained by replacing all free occurences of x in t by a. The remaining rules (and axiom) are:

$$\frac{\Gamma, x : X \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (\to I) \qquad \frac{\Gamma \vdash f : A \to B \quad \Gamma \vdash a : A}{\Gamma \vdash fa : B}\ (\to E)$$

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma \vdash t_2 : T_2}{\Gamma \vdash \langle t_1, t_2 \rangle : T_1 \times T_2}\ (\times I) \qquad \frac{\Gamma \vdash \pi_i t : T_i}{\Gamma \vdash t : T_1 \times T_2}\ (\times E_i)$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash f(t) : B}\ (opI) \qquad \vdash * : 1 (*I)$$

Now let $\mathcal{C}$ be a cartesian closed category with chosen products and exponentials is constructed as follows. A model $[| - |]$ of this calculus in $\mathcal{C}$ is generated by a graph morphism $G \to |\mathcal{C}|$, i.e. the base types are mapped to objects, and the function symbols to morphisms. General types are then interpreted by mapping product and exponential types to the corresponding constructions in $\mathcal{C}$. The interpretation of a context Γ is the product of the types of the variables appearing in it. The terms are not interpreted directly, but rather a proof $\Gamma \vdash t : T$ is interpreted by a morphism $[|t|] : [|\Gamma|] \to [|T|]$. For example, a variable $x : T$ may be interpreted by $\mathtt{id}_{[|T|]}$ if considered as the proof of the axiom $x : T \vdash x : T$ but will be interpreted as the second projection if considered as a proof of $\{y : S, x : T\} \vdash x : T$ by weakening. So, strictly, we are interpreting proofs P of $\Gamma \vdash t : T$ by morphisms $[|P|] : [|\Gamma|] \to [|T|]$. In the following table the left-hand side presents a proof rule and the right-hand side its interpretation.

$$\frac{P}{\Gamma, x : X \vdash t : T}\ (Weak) \qquad [|\Gamma|]\times[|X|] \xrightarrow{\ \pi\ } [|\Gamma|] \xrightarrow{[|P|]} [|T|]$$

$$\frac{P}{\Gamma' \vdash t : T}\ (Ex) \qquad [|\Gamma'|] \longrightarrow [|\Gamma|] \xrightarrow{[|P|]} [|T|]$$

$$\frac{P_1 \quad P_2}{\Gamma \vdash t[a/x] : B}\ (Cut) \qquad [|\Gamma|] \xrightarrow{\langle \mathtt{id}, [|P_2|]\rangle} [|\Gamma|]\times[|A|] \xrightarrow{[|P_1|]} [|B|]$$

$$\frac{\Gamma \vdash \lambda x.t : A \to B}{]}\ (\to I|_P \qquad [|\Gamma|] \xrightarrow{\qquad \Lambda[|P|] \qquad} [|A \to B|]$$

$$\frac{P_1 \quad P_2}{\Gamma \vdash fa : B}\ (\to E) \qquad [|\Gamma|] \xrightarrow{\langle [|P_1|], [|P_2|]\rangle} [|A \to B|]\times[|A|] \xrightarrow{\mathtt{eval}} [|B|]$$

$$\frac{P_1, P_2}{\Gamma \vdash \langle t_1, t_2\rangle}\ (\times I) \qquad [|\Gamma|] \xrightarrow{\qquad \langle [|P_1|], [|P_2|]\rangle \qquad} [|A \times B|]$$

$$\frac{P}{\Gamma \vdash \pi_i t}\ (\times E_i) \qquad [|\Gamma|] \xrightarrow{[|P|]} [|T_1 \times T_2|] \xrightarrow{\ \pi_i\ } [|T_i|]$$

$$\frac{P}{\Gamma \vdash a : A}\ (opI) \qquad [|\Gamma|] \xrightarrow{[|P|]} [|A|] \xrightarrow{[|f|]} [|B|]$$

$$\frac{}{\Gamma \vdash * : 1}\ (*I) \qquad [|\Gamma|] \longrightarrow 1$$

This interpretation respects β- and η-conversions. β-conversion asserts the equivalence of the following two proofs, and hence of their labels

$$\frac{\dfrac{P_1}{\Gamma \vdash \lambda x.t : A \to B} \quad \dfrac{P_2}{\Gamma \vdash a : A}}{\Gamma \vdash (\lambda x.t)a : B} \qquad\qquad \frac{P_1 \quad P_2}{\Gamma \vdash t[a/x] : B}$$

The interpretations of these proofs are morphisms corresponding to the upper and lower edges of the following diagram, where $[|P_1|] = f$ and $[|P_2|] = g$.

$$\begin{tikzcd}
X \arrow[r, "\langle \check{f}, g\rangle"] \arrow[dr, "\langle \mathtt{id}, g\rangle"'] & (A \to B)\times A \arrow[dr, "\mathtt{eval}"] & \\
& X \times A \arrow[u, "\check{f}\times\mathtt{id}"] \arrow[r, "f"'] & B
\end{tikzcd}$$

The diagram commutes since the right-hand triangle is one of the triangle laws for the product-exponential adjunction.

Exercise 7.0.6 η-conversion can be similarly formulated, and shown to be respected by the interpretation.

Thus we have shown how to construct an interpretation of the language generated by the graph G from a graph morphism $G \rightarrow |\mathcal{C}|$. This can be turned into a morphism of a category by freely constructing a cartesian closed category from the language, as follows. Objects are contexts and morphisms $\Gamma \rightarrow \Delta$ are tuples of proofs, one for each variable $y_j : Y_j$ of Δ. It follows that $\Delta = [y_j : Y_j]$ is the product of the contexts $[y_j : Y_j]$ which is important for constructing the exponential, the key case of which is now to construct $[x : X] \rightarrow [y : Y] = [u : X \rightarrow Y]$.

Exercise 7.0.7 Complete this construction.

Chapter 8

Suggested Reading

J. Adamek, H. Herrlich and G. Strecker, Abstract and Concrete Categories (Wiley, 1990). An updated version of the old Herrlich and Strecker, with good presentation of factorisation systems, and a chapter on partial maps.

M. Barr, C. Wells, Toposes, Triples and Theories (Springer). A good presentation of its topics, particularly exactness conditions, and sketches. Not really an introduction. Beware of typos, particularly in the first chapter.

M. Barr, C. Wells, Category Theory for Computing Science (Hoare's series). Probably the best currently available for CS.

P.T. Johnstone, Stone Spaces, (Cambridge Studies in Advanced Mathematics). A solid introduction to locales and their duality with algebra.

J. Lambek, P Scott, Introduction to Higher Order Categorical Logic. (Cambridge University Press). A detailed account of the correspondence between lambda-calculus and cartesian closed categories.

S. Mac Lane, Category Theory for the Working Mathematician (Springer). The classic book on the subject, for the mathematically trained beginner. Including monoidal categories, abelian categories, and a wider class of limits than most books cover.

B. C. Pierce, A Taste of Category Theory for Computer Scientists (MIT Press). This monograph contains useful bibliographic information on the works referred to below, and others.

R.F.C. Walters, Categories for Computer Science (Carslaw). Introduction to distributive categories, imperative programming, etc.