Contents

ii   Contents

Answers to Exercises   83

Bibliography   97

Index   102

Preface

This is the electronic supplement to **Category Theory for Computing Science, second edition, Prentice-Hall International, 1995, ISBN 0-13-323809-1**. It consists of (revisions of) ¯ve chapters from the ¯rst edition of that book, as well as continuing updates of the ¯rst and second editions.

The chapters included here are, as numbered in the ¯rst edition, Chapters 7 (Finite discrete sketches), 9 (More about sketches), 10 (The category of sketches), 11 (Fibrations) and 14 (Toposes).

References, both in the published book and here to material in this electronic supplement are pre¯xed by ` ES', while references to the printed book are unadorned. Thus a reference to 4.5 of Chapter 3 is a reference to Section 4.5 of Chapter 3 of the printed book, while references to Section 4.5 of Chapter ES 3 is a reference to Section 4.5 of Chapter 3 of this supplement. A reference without a chapter number refers, as usual, to the current chapter.

1
Finite discrete sketches

This chapter discusses FD (¯nite discrete) sketches, which are a more general kind of sketch than that described in Chapter 7. The sketches described here allow the speci¯cation of objects which are sums as well as products. Section ES 1.1 introduces these sketches. Section ES 1.2 gives a detailed description of the sketch for ¯elds as an example of an FD sketch. In Section ES 1.3, we show how to modify initial algebra construction so that it works for FD sketches. FD sketches have theories, too, but discussion of that is postponed until Chapter ES 2.

1.1 Sketches with sums

Using sum cocones as well as product cones in a sketch allows one to express alternatives as well as n-ary operations.

1.1.1 A **¯nite discrete sketch**, or **FD sketch**, $\mathscr{S} = (\mathscr{G}; \mathscr{D}; \mathscr{L}; \mathscr{K})$, consists of a ¯nite graph $\mathscr{G}$, a ¯nite set of ¯nite diagrams $\mathscr{D}$, a ¯nite set $\mathscr{L}$ of discrete cones with ¯nite bases, and a ¯nite set $\mathscr{K}$ of discrete cocones with ¯nite bases. A **model** of the FD sketch $\mathscr{S}$ in **Set** is a graph homomorphism M : $\mathscr{G}$ ¡! **Set** which takes the diagrams in $\mathscr{D}$ to commutative diagrams, the cones in $\mathscr{L}$ to product cones, and the cocones in $\mathscr{K}$ to sum cocones.

1.1.2 The sketch for lists The FD sketch for ¯nite lists illustrates how FD sketches can be used to specify a data type which has an exceptional case for which an operation is not de¯ned. (Compare 7.1.6.) The graph has nodes 1, d (the data), l (the lists) and l⁺ (the nonempty lists). There are no diagrams. There is a cone with empty base expressing the fact that 1 must become a terminal object in a model, another cone

and one cocone

1
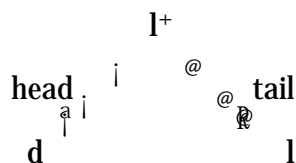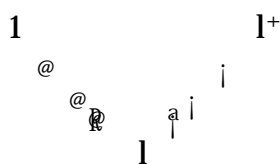
2   Finite discrete sketches

These can be summed up in the expressions $l^+ = d \times l$ (a nonempty list has a datum as head and a list as tail) and $l = 1 + l^+$: (a list is either the empty list { the single member of the terminal set { or a nonempty list). The fact that a sum becomes a disjoint union in a model in **Set** enables us to express alternatives in the way exhibited by this description. We thereby avoid the problem mentioned in 7.1.6: we can define head for only nonempty lists because we have a separate sort for nonempty lists.

   If S is any finite set, there is a model M of this sketch for which $M(d) = S$, $M(1)$ is the singleton set containing only the empty list hi, $M(l)$ is the set of finite lists of elements of S, and $M(l^+)$ is the set of finite nonempty lists of elements of S. $M(\text{head})$ and $M(\text{tail})$ pick out the head and tail of a nonempty list. This is an initial term algebra (to be defined in Section ES 1.3) for the sketch extended by one constant for each element of S.

1.1.3   Given a finite set S, the model M just mentioned is certainly not the only model of the sketch for which $M(d)$ is S. Another model has $M(l)$ all finite and infinite lists of elements of S, with $M(l^+)$ the nonempty lists and $M(\text{head})$ and $M(\text{tail})$ as before. This model contains elements inaccessible by applying operations to constants: you cannot get infinite lists by iterating operations starting with constants. They are junk as defined in 7.6.3.

1.1.4 The sketch for natural numbers   We will now modify the sketch 7.1.5 in a way which shows how FD sketches can model overflow.

   The graph contains nodes we will call 1, n, n_{over} and $n + n_{over}$, which we interpret, respectively, as a one-element set, a set of natural numbers (of which there may be only finitely many), an overflow element and the set of all natural numbers. The only cone is the one implicit in the name 1: the cone has 1 as the vertex and the empty base. There is similarly a (discrete) cocone implicit in the name $n + n_{over}$. There is a constant operation $zero : 1 \to n$ and a unary operation $succ : n \to n + n_{over}$. There are no diagrams.

   There are many models of this sketch in **Set**. Here are three of them. They are all term models.

1.1.5   The first model we consider of this sketch is the set of natural numbers. The value of zero is 0 and $succ(i) = i + 1$. The model takes n_{over} to the empty set. This model illustrates the fact that models are permitted to take the empty set as value on one or more nodes. Classical model theorists have not usually allowed sorts in a model to be empty.

1.1.6   The second model takes n to the set of integers up to and including an upper bound N. The value at n_{over} is a one- element set we will call 1 , although any other convenient label (including N + 1) could be used instead. As above,

the value of zero is 0. As for succ, it is de¯ned by succ(i) = i + 1 for i < N, while succ(N) = 1 . Note that the domain of succ does not include 1 .

1.1.7 The third model takes for the value of n also the set of integers up to a ¯xed bound N. The value of zero is again 0 and of n$_{over}$ is empty. The operation succ is de¯ned by succ(i) = i + 1, for i < N while succ(N) = 0. This is arithmetic modulo n.

This third model could be varied by letting succ(N) = N or, for that matter, any intermediate value. This shows that these models really have at least two parameters: the value of N and what happens to the successor of N.

1.1.8 Exercises

1. Construct an FD sketch whose only model is the two-element Boolean algebra.

2. Give an example of a sketch which has no models in **Set**. (Hint: In **Set**, 1 + 1 ⊖ 1.)

1.2 The sketch for ¯elds

Another example of an FD sketch is the sketch for the mathematical structure known as a **¯eld**. The concept of ¯eld abstracts the properties of the arithmetic of numbers. We will describe it in some detail here. This section is used later only as an example in Section ES 1.3. Rather than de¯ne `¯eld' we will describe the sketch and say that a ¯eld is a model of the sketch in sets.

The nodes are 1; u; f; f £ f and f £ f £ f. There are operations

FO{1 0 : 1 ¡! f.
FO{2 1 : 1 ¡! u.
FO{3 + : f £ f ¡! f.
FO{4 ¤ : f £ f ¡! f.
FO{5 ¡ : f ¡! f.
FO{6 ( )$^{¡1}$: u ¡! u.
FO{7 j : u ¡! f.

The reader should note that the two 1's in FO{2 above are, of course, di®erent. One of them is the name of a node and the other of an operation. This would normally be considered inexcusable. The reasons we do it anyway are (a) each usage is hallowed by long tradition and (b) because they are of di®erent type, there is never any possibility of actual clash. The reader may think of it as an early example of overloading.

The diagrams are:

FE{1 (associativity of +): + ± (id £ +) = + ± (+ £ id) : f £ f £ f ¡! f.

4 Finite discrete sketches

FE{2 (associativity of ¤): ¤ ± (id £¤) = ¤ ± (¤ £ id) : f £ f £ f ¡! f.

FE{3 (commutativity of +): + ± hp₂; p₁i = + : f £ f ¡! f.

FE{4 (commutativity of ¤): ¤ ± hp₂; p₁i = ¤ : f £ f ¡! f.

FE{5 (additive unit): + ± hid; 0 ± hii = + ± h0 ± hi; idi = id : f ¡! f

FE{6 (multiplicative unit): ¤ ± hj; j ± 1 ± hii = ¤ ± hj ± 1 ± hi; ji = j : u ¡! f

FE{7 (additive inverse): + ± hid; ¡ i = + ± h¡ ; idi = 0 ± hi : f ¡! f

FE{8 (multiplicative inverse): ¤ ± (j £ j) ± hid; ()⁻¹i = ¤ ± (j £ j) ± h()⁻¹; idi = 1 ±
    hi : u ¡! u

FE{9 (distributive): + ± (¤ £ ¤) ± hp₁; p₂; p₁; p₃i = ¤ ± (id £+);
    + ± (¤ £ ¤) ± hp₁; p₃; p₂; p₃i = ¤ ± (+ £ id) : f £ f £ f ¡! f.

There are cones de¯ned implicitly and one cocone:
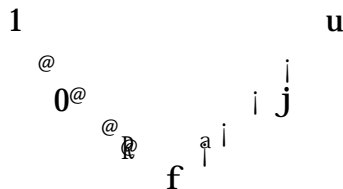
Intuitively, this says that each element of a ¯eld is either zero or else is an element
of u. u is interpreted as the set of multiplicatively invertible elements in a model.

1.2.1 An example of a ¯eld is the set of ordinary rational numbers. Expressed
as a model M of the sketch, the numbers $M(0)$ and $M(1)$ are the usual ones,
$M(f)$ and $M(u)$ are the sets of all rationals and nonzero rationals, respectively,
and $M(j)$ is the inclusion. The arithmetic operations are the usual ones. Other
familiar examples are the real and complex numbers.

1.2.2 In building a model of this sketch, we would have to start with the ele-
ments 0 and 1 and then begin adding and multiplying them to get new elements.
For example, we could let $2 = 1 + 1$. Now since $M(f) = M(u) + f0g$, we have
to decide whether $2 \, 2 \, M(u)$ or $2 \, 2 \, f0g$, that is, whether or not $2 = 0$. Quite
possibly, the only ¯elds the reader has seen have the property that $2 \, \ominus \, 0$. Even
so, one cannot exclude out of hand the possibility that $2 = 0$ and it is in fact
possible: there is a model of this sketch whose value at f is f0; 1g with the values
of the operations being given by the tables:

+	0	1		¤	0	1
0	0	1		0	0	0
1	1	0		1	0	1

(These tables are addition and multiplication $(\mathrm{mod}\, 2)$.) If, on the other hand,
$2 \, \ominus \, 0$, we can form $3 = 1 + 2$. The same question arises: is $3 = 0$ or not? It is not
hard to write down a ¯eld in which $3 = 0$.

Suppose that neither 2 nor 3 is 0? We would define $4 = 1 + 3$ (and prove, by the way, that $2 + 2 = 2 \times 2 = 4$) and ask whether $4 = 0$. It turns out that if $2 \neq 0$ then $4 \neq 0$. In fact, it is not hard to show that any product of elements of u also lies in u. Thus the first instance of an integer being 0 must happen at a prime. But, of course, it need never happen, since no number gotten by adding 1 to itself a number of times is zero in the rational, real or complex field.

1.2.3 A homomorphism between fields must preserve all operations. Since it preserves the operation $(\ )^{-1}$, no nonzero element can be taken to zero. Consequently, two distinct elements cannot go to the same element since their difference would be sent to zero. Thus field homomorphisms are injective.

From this, it follows that there can be no field that has a homomorphism both to the field of two elements and to the rationals, since in the first $1 + 1 = 0$, which is not true in the second. This implies that the category of fields and field homomorphisms does not have products.

1.2.4 Exercises

1. Prove that in a field, if two elements each have multiplicative inverses, then so does their product. Deduce that if $4 = 0$, then $2 = 0$.

2. Prove that the next to last sentence of ES 1.2.3 implies that the category of fields and field homomorphisms does not have products.

1.3 Term algebras for FD sketches

A complication arises in trying to extend the construction of initial term algebras to FD sketches. As we see from the examples of natural numbers and fields, an operation taking values in the vertex of a discrete cocone forces us to choose in which summand the result of any operation shall be. The choice, in general, leads to nonisomorphic term models which are nevertheless initial in a more general sense which we will make precise.

1.3.1 Daemons How to make the choice? Clearly there is no systematic way. One way of dealing with the problem is to take all choices, or at least to explore all choices. In the example in Section ES 1.2 of fields, we mentioned that not all choices are possible; once $2 \neq 0$, it followed that also $4 \neq 0$. (Recall that in that example, saying that something is zero is saying that it is in one of two summands.)

More generally, suppose we have an FD sketch and there is an operation $s : a \to b$ and a cocone expressing $b = b_1 + b_2 + \ldots + b_n$. If we are building a model M of this sketch and we have an element $x \in M(a)$, then $M(s)(x) \in M(b)$, which means that we must have a unique i between 1 and n for which $M(s)(x) \in M(b_i)$. (For simplicity, this notation assumes that $M(b)$ is the actual union of the $M(b_i)$.)

Now it may happen that there is some equation that forces it to be in one rather than another summand, but in general there is no such indication. For example, the result of a push operation on a stack may or may not be an over°ow, depending on the capacity of the machine or other considerations. Which one it is determines the particular term model we construct and it is these choices that determine which term model we will get.

Our solution is basically to try all possible sequences of choices; some such sequences will result in a model and others will abort. Thus as we explore all choices, some will eventually lead to a model; some will not. The theoretical tool we use to carry out this choice we call a **d½mon**. Just as a Maxwell D½mon chooses, for each molecule of a gas, whether it goes into one chamber or another, our d½mon chooses, for each term of a model, which summand it goes into. The following description spells this out precisely.

1.3.2 De¯nition   Let $\mathscr{S}$ be an FD sketch and suppose the maximum number of nodes in the base of any cocone is $\cdot$. A **d½mon** for $\mathscr{S}$ is a function d from the set of all strings in the alphabet $A_{\mathscr{S}}$ (see 7.6.5) of the underlying FP sketch (in other words, forget the cocones) to the initial segment $f1:: \cdot g$ of the positive integers.

1.3.3   We will use a d½mon this way. We assume that the nodes in the base of each cocone of $\mathscr{S}$ are indexed by $1, 2, \ldots, k$ where $k \cdot \cdot$ is the number of nodes in that cocone. In constructing an initial algebra, if a string w must be in a sort which is the vertex of a cocone (hence in the model it must be the disjoint union of no more than $\cdot$ sorts), we will choose to put it in the $D(w)$th summand. If $D(w) > k$, the construction aborts. We will make this formal.

1.3.4 Construction of initial term models for FD sketches   This construction includes the processes in 4.7.10 and 7.6.5; we repeat them here modi¯ed to include the e®ects of a d½mon D. The alphabet is the same as in 7.6.5.

If a node b is the vertex of a cocone with $k = k(b)$ summands, the summands will be systematically denoted $b^1, \ldots, b^k$ and the inclusion arrows $u^i : b^i \downarrow! \ b$ for $i = 1, \ldots, k$. If b is not the vertex of a cocone, then we take $k(b) = 1$, $b^1 = b$ and $u^1 = id_b$. We denote the congruence relation by » and the congruence class containing the element x by [x]. Rules FD{ES 3 through FD{ES 6 refer to a cone C in $\mathscr{L}$ of the form:

$$
\begin{array}{ccccc}
& & q & & \\
& {}_{\textstyle \dot{\imath}} & \big| {\scriptstyle @} & & \\
p_1 & {}_{\textstyle \dot{\imath}} & p_i & {\scriptstyle @} & p_n \\
{}_{\textstyle \dot{a}}{}^{\dot{\imath}} & & \big|{}^? & {\scriptstyle @}{}_{\textstyle \dot{a}}{}^{@} & \\
a_1 & ¢¢¢ & a_i & ¢¢¢ & a_n
\end{array}
$$

FD{1 If $u^i : a^i \downarrow! \ a$ is an inclusion in a cocone and $[x] \, 2 \, I(a^i)$, then $[u^i x] \, 2 \, a$ and $I(u^i)[x] = [u^i x]$. (Thus we ignore the wishes of the d½mon in this case.)

FD–2 Suppose $f : a \to b$ in $\mathcal{G}$, f is not an arrow of the form u^i, and $[x] \in I(a)$. Let $j = D(fx)$ (we ask the dæmon what to do). If $j > k(b)$, the construction aborts. Otherwise, we let $[fx] \in I(b^j)$ and $I(f)[x] = [u^j fx]$.

FD–3 For $i = 1,\ldots,n$, let $[x_i]$ be a term in $I(a_i)$. Let

$$j = D(C(x_1,\ldots,x_n))$$

If $j > k(q)$, the construction aborts. If not, put $[C(x_1,\ldots,x_n)]$ in $I(q^j)$.

FD–4 If for $i = 1,\ldots,n$, $[x_i]$ and $[y_i]$ are elements in $I(a_i)$ for which $[x_i] = [y_i]$, $i = 1,\ldots,n$, then

$$[C(x_1,\ldots,x_n)] = [C(y_1,\ldots,y_n)]$$

FD–5 For $i = 1,\ldots,n$, $I(p_i)([C(x_1,\ldots,x_n)]) = [x_i]$.

FD–6 For $x \in I(q)$,

$$[x] = [C(p_1 x,\ldots,p_n x)]$$

FD–7 If $\langle f_1,\ldots,f_m \rangle$ and $\langle g_1,\ldots,g_k \rangle$ are paths in a diagram in $\mathcal{D}$, both going from a node labeled a to a node labeled b, and $[x] \in I(a)$, then

$$(If_1 \circ If_2 \circ \cdots \circ If_m)[x] = (Ig_1 \circ Ig_2 \circ \cdots \circ Ig_k)[x]$$

in $I(b)$. If

$$D(f_1 f_2 \cdots f_m x) \neq D(g_1 \cdots g_k x)$$

(causing $(If_1 \circ If_2 \circ \cdots \circ If_m)[x]$ and $(Ig_1 \circ Ig_2 \circ \cdots \circ Ig_k)[x]$ to be in two different summands of $I(b)$), then the construction aborts.

This construction gives a term model if it does not abort. It is an initial model for only part of the category of models, however. To make this precise, we recall the definition of connected component from 4.3.10. It is easy to see that each connected component is a full subcategory of the whole category of models.

1.3.5 Proposition For each dæmon for which the construction in FD–1 to FD–6 does not abort, the construction is a recursive definition of a model I of $\mathcal{S}$. Each such model is the initial model of a connected component of the category of models of $\mathcal{S}$, and there is a dæmon giving the initial model for each connected component.

We will not prove this theorem here. However, we will indicate how each model determines a dæmon which produces the initial model for its component. Let M be a model of an FD sketch $\mathcal{S}$. Every string w which determines an element of a sort $I(a)$ in a term model as constructed above corresponds to an element of $M(a)$. That element must be in a unique summand of a; if it is the ith summand, then define $D(w) = i$. On strings not used in the construction of the term models, define $D(w) = 1$, not that it matters.

Our definition of d½mon shows that one can attempt a construction of an initial model without already knowing models. In concrete cases, of course, it will often be possible to characterize which choices give initial models and which do not.

1.3.6 Confusion maybe, junk no   The slogan, `No junk, no confusion' is only half true of the initial models for FD theories. The `No junk' half of the slogan expresses exactly what we mean when we say that every element is reachable. There are no extraneous elements. `No confusion' means no relations except those forced by the equations in the theory. As we will show by example it may happen that some initial models have confusion and others not. Later we give an example of a sketch that has more than one unconfused initial model and one that has no unconfused initial (or noninitial) model.

If there is just one unconfused initial model, that one may be thought of as a `generic' model. The others remain nonetheless interesting. In fact, it is likely that the generic model is the one that cannot be accurately modeled on a real machine.

1.3.7 Example   A typical example of a sketch with many initial models is the sketch for natural numbers with overflow. The generic model is easily seen to be the one in ES 1.1.5 in which the overflow state is empty. The models with overflow in ES 1.1.6 are all initial algebras for some component of the category of models, but they have confusion, since nothing in the sketch implies that the successor of any element can be the same element. None of these models have junk.

The modular arithmetic models of ES 1.1.7 are not initial models; in fact they are all in the same component as the natural numbers since the remainder map $(\bmod N)$ is a morphism of models. They also have no junk.

1.3.8 Example   Here is a simple sketch with no generic model. It has two initial models, each satisfying an equation the other one does not. There are five nodes $a = b + c$, d and 1. There is one constant x of type d, and a single operation $s : d \to a$. The initial models have one element { the constant { of type d. One of the initial models has an element of type b and the other an element of type c.

By modifying this example, we can get forced confusion. Add constants y and z of type b and c, respectively, and cones forcing b and c to be terminal. Now there are two initial models, one in which $s(x) = y$ and another in which $s(x) = z$. Since there is a model in which $s(x) \neq y$, there can be no equation that forces $s(x) = y$ and there is similarly no equation that forces $s(x) = z$. But one or the other equation must hold in any model.

1.3.9 Example   It is well known and proved in abstract algebra texts that the initial fields are (a) the rational numbers and (b) the integers mod p for each prime p. (The word for initial model in these texts is `prime field'.) A field is in

the component of the integers mod p if and only if $1 + 1 + ::: + 1$ (sum of p 1's) is zero. These ¯elds have confusion. Otherwise the ¯eld is in the component of the rational numbers, which have no confusion (nor junk). Duval and Reynaud [1994a, 1994b] show how to implement simultaneous computation in the initial algebras for a ¯nite discrete sketch for ¯elds.

The real numbers and the complex numbers form ¯elds with the usual operations. The irrational real numbers constitute junk.

1.3.10 Example   The example in ES 1.1.2 has only one component and hence a single initial model in which all sorts are empty except the singleton 1. If you add constants to d, the initial model is just the set of lists of ¯nite length of elements of d. In the model discussed there which also has all in¯nite sequences, the in¯nite sequences are junk.

1.3.11 Binary trees   We now describe a sketch for ordered rooted binary trees (called trees in this discussion). `Binary' means that each node has either no children or two children, and `ordered' means that the children are designated left and right. This is an example which uses cocones to treat exceptional cases, in this case the empty tree.

Trees are parametrized by the type of data that are stored in them. We will say nothing about this type of data, supposing only that it is a type for which there is an initial model. The way in which the parametrized data type is ¯lled in with a real one is described, for example, in Section ES 3.2. We would like to thank Adam D. Barr for helpful discussions on how binary trees operate (especially their error states) on real machines.

The sketch will have sorts 1; t; s; d. Informally, t stands for tree, s for nonempty tree and d for datum. We have the following operations:

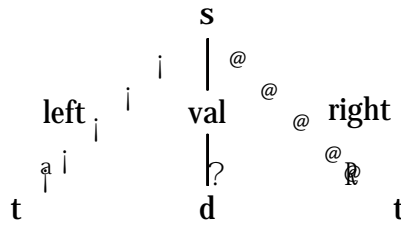empty : 1 ¡! t

incl : s ¡! t

val : s ¡! d

left : s ¡! t

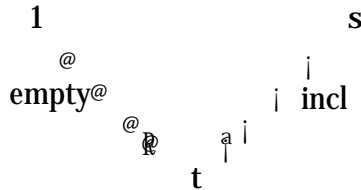right : s ¡! t

The intended meaning of these operations is as follows.

Empty hi is the empty tree; incl is the inclusion of the set of nonempty trees in the set of trees; val(S) is the datum stored at the root of S; left(S) and right(S) are the right and left branches (possibly empty) of the nonempty tree S, respectively.

We require that

$$
\begin{array}{ccc}
 & s & \\
\text{left} & \text{val} & \text{right} \\
t & d & t
\end{array}
$$

be a cone and that

$$
\begin{array}{ccc}
1 & & s \\
\text{empty} & & \text{incl} \\
 & t &
\end{array}
$$

be a cocone.

There are no diagrams.

The cocone says that every tree is either empty or nonempty. This cocone could be alternatively expressed t = s + femptyg. The cone says that every nonempty tree can be represented uniquely as a triplet (left(S); val(S); right(S)) and that every such triplet corresponds to a tree. Note that this implies that left, val and right become coordinate projections in a model.

Using this, we can de¯ne subsidiary operations on trees. For example, we can de¯ne an operation of left attachment, lat : t £ s ¡! s by letting

$$
\text{lat}(T; (\text{left}(S); \text{val}(S); \text{right}(S))) = (T; \text{val}(S); \text{right}(S))
$$

This can be done without elements: lat is de¯ned in any model as the unique arrow making the following diagram commute (note that the horizontal arrows are isomorphisms):

$$
\begin{array}{ccc}
M(t) £ M(s) & \xrightarrow{\ M(t) £ h\text{left}; \text{val}; \text{right}i\ } & M(t) £ M(t) £ M(d) £ M(t) \\
\text{lat} \downarrow & & \downarrow hp_1; p_3; p_4 i \\
M(s) & \xrightarrow{\ h\text{left}; \text{val}; \text{right}i\ } & M(t) £ M(d) £ M(t)
\end{array}
$$

In a similar way, we can de¯ne right attachment as well as the insertion of a datum at the root node as operations de¯nable in any tree. These operations are implicit in the sketch in the sense that they occur as arrows in the theory generated by the sketch (see 4.6.11), and therefore are present in every model.

1.3.12 Proposition   Supposing there is an initial algebra for the data type, then the category of binary trees of that type has an initial algebra. If the data type has (up to isomorphism) a unique initial algebra, then so does the corresponding category of binary trees.

Proof. We construct the initial algebra recursively according to the rules:

(i) The empty set is a tree;

(ii) If T_l and T_r are trees and D is element of the initial term algebra for the data type, then $(T_l; D; T_r)$ is a nonempty tree;

(iii) Nothing else is a tree.

This is a model M_0 defined by letting $M_0(s)$ be the set of nonempty trees, $M_0(t) = M_0(s) + f; g$ and $M_0(d)$ be the initial model of the data type. Here `+' denotes disjoint union. It is clear how to define the operations of the sketch in such a way that this becomes a model of the sketch.

Now let M be any model with the property that M(d) is a model for the data type. Then there is a unique morphism $f(d) : M_0(d) \to M(d)$ that preserves all the operations in the data type. We also define $f(t)f; g$ to be the value of $M(\text{empty}) : 1 \to M(t)$. Finally, we define

$$f(s)((T_l; D; T_r)) = (f(t)(T_l); f(d)(D); f(t)(T_r))$$

where $f(t)$ is defined recursively to agree with $f(s)$ on nonempty trees. It is immediately clear that this is a morphism of models and is unique. In particular, if the data type has, up to isomorphism, only one initial model then M_0 is also unique up to isomorphism.   □

1.3.13   In Pascal textbooks a definition for a tree type typically looks like this:

```
type TreePtr = bTree;
     Tree = record LeftTree, RightTree : TreePtr;
        Datum : integer
     end;
```

Note that from the point of view of the preceding sketch, this actually defines nonempty trees. The empty tree is referred to by a null pointer. This takes advantage of the fact that in such languages defining a pointer to a type D actually defines a pointer to what is in effect a variant record (union structure) which is either of type D or of `type' null.

1.3.14 Exercise

1.ʸ a. Show that if $\mathscr{S}$ is an FD sketch and $f : M \to N$ is a homomorphism between models in the category of sets, then the image of f is a submodel of N.

b. Show that every model in the category of sets of an FD sketch has a smallest submodel.

2

More about sketches

This chapter develops the concept of sketch in several ways. The ¯rst three sections describe a generalization of FP theories called FL theories which allow the use of equalizers, pullbacks and other limits in the description of a structure. These theories have expressive power which includes that of universal Horn theories.

The last section gives further generalizations of the concept of sketch. These generalizations are described without much detail since they do not appear to have many applications (yet!) in computer science.

This chapter is needed only for Chapter ES 3, except that the sketch for categories of Section ES 2.1.5 is required for Section ES 5.7.

2.1 Finite limit sketches

A cone is called ¯nite if its shape graph is ¯nite, meaning that it has only ¯nitely many nodes and arrows.

2.1.1 De¯nition   A **¯nite limit** or **FL sketch** $\mathscr{S} = (\mathscr{G}; \mathscr{D}; \mathscr{L})$ is a ¯nite graph together with a ¯nite set $\mathscr{D}$ of diagrams and a ¯nite set $\mathscr{L}$ of ¯nite cones. A **model** of $\mathscr{S}$ is a model of $\mathscr{G}$ that takes all the diagrams in $\mathscr{D}$ to commutative diagrams and all the cones in $\mathscr{L}$ to limit cones.

For historical reasons, FL sketches are also known as **left exact sketches** or **LE sketches**.

In 7.2.8 we described the way the choice of products and terminal objects in a model to represent the vertices of cones in a sketch were irrelevant but could result in the technicality that, for example, the set representing the product might not actually be a set of ordered pairs. The same sort of statement is true of models of FL theories. In particular, in a model the equalizer of parallel arrows need not be a subset of their common domain.

FL sketches allow the speci¯cation of structures or data types with sorts that include equationally speci¯ed subsorts, by using equalizers. More generally, you can specify an operation whose domain, in a model, will be an equationally de¯ned subobject of another sort. FL theories can express anything expressible by universal Horn theories, but in general FL theories are more powerful (see [Barr, 1989] and [Ad§mek and Rosicky, 1994], pages 209-210). For example, small categories and functors form the category of models of an FL theory (which we

give in ES 2.1.5 below) but not of a universal Horn theory. Categories of models of FL sketches can be described axiomatically as **locally ¯nitely presentable categories** (see [Gabriel and Ulmer, 1971] and [Ad¶mek and Rosicky, 1994]).

In practice it is generally su±cient to restrict the types of cones to a few simple types, products, pullbacks and equalizers. In principle, an FL sketch can always be replaced by one which has an equivalent category of models and which has only these three types of cones, but there might be some case in which this is not the most e±cient approach.

2.1.2 Other approaches   In the introduction to Chapter 7, we mentioned three approaches to formalization: logical theories, signatures and equations, and sketches. Systems equivalent to FL sketches have been developed for both the other approaches. Coste [1976], Cartmell [1986] and McLarty [1986] generalize logical systems and Reichel [1987] generalize signatures. The book by Reichel has many examples of applications to computer science.

2.1.3 Notation for FL sketches   We extend the notational conventions in Section 7.3 to cover products, pullbacks and equalizers.

First, the notation of N{2 in 7.3.1 using product projections is extended to cover the arrows from the limit to any of the nodes in the diagram: that is they are all denoted by an appropriate p; moreover, the notation of N{4 in 7.3.1 is extended to arrows into the limit in terms of the composite with the projections. See Example ES 2.1.5 below of the sketch of categories to see how this is used. Of course, it is sometimes necessary to be more explicit than this.

We add the following to the notational conventions of Section 7.3.

N{8 If we have a node labeled a £$_c$ b this implies the existence of a cone
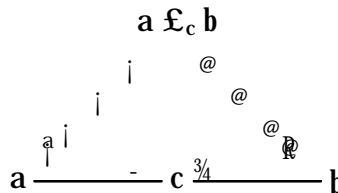
    Of course, this notation is not self-contained since it is necessary to specify the arrows a ¡! c Ã b. In many cases, these arrows are clear, but it may be necessary to specify them explicitly.

N{9 If we have an arrow labeled s : a)¡! b (recall from 2.8.2 that in a category this notation means an arrow that is a monomorphism), then we are implicitly including a cone

$$
\begin{array}{ccc}
 & a & \\
\text{id} & \downarrow s & \text{id} \\
a \xrightarrow{\;\;s\;\;} b & & \xrightarrow{\;\;s\;\;} a
\end{array}
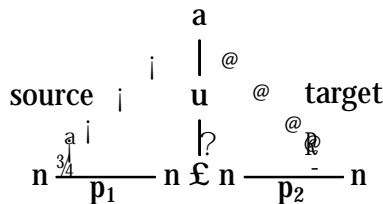$$
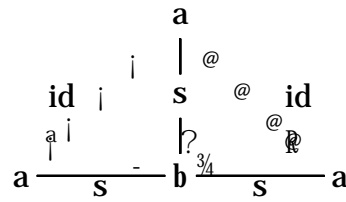
This notation makes sense because of Theorem 8.3.3.

We could go on and turn our notational conventions into a formal language, but have chosen not to do so because we do not know what it would add to the theory. For us, they remain a set of notational conventions; the real object of study is the whole sketch or else the theory it generates.

2.1.4 The sketch for simple graphs Here is a simple example that shows how the added power of FL sketches can be used to de¯ne a familiar type of structure. A **simple graph** is a graph with the property that for any pair $(a; b)$ of nodes there is no more than one arrow with source a and target b.

The sketch for graphs has the graph

$$
a \; \underset{\text{target}}{\overset{\text{source}}{\rightrightarrows}} \; n
$$

and no cones or diagrams. We can modify it to get a sketch for simple graphs by adding an object $n \pounds n$ and the necessary discrete cone to make it a formal product, an arrow $u : a)_{\textsubscript{i}}! \; n \pounds n$ (hence adding a formal pullback as in N{ES 9 making it formally monic) and also the diagram (not cone)

$$
\begin{array}{ccc}
 & a & \\
\text{source} & \downarrow u & \text{target} \\
n \xleftarrow{\;p_1\;} n \pounds n \xrightarrow{\;p_2\;} n
\end{array}
$$

The e®ect of this is to make the monic arrow u become $\langle \text{source}; \text{target}\rangle$ in a model, so that no two arrows can have the same source-target pair.

2.1.5 The sketch for categories We give here an FL sketch whose models in the category of sets are small categories and arrows between the models are functors. Models in an arbitrary category $\mathscr{C}$ with ¯nite limits are called **category objects** in $\mathscr{C}$. These are used in Section ES 5.7.

The sketch has nodes $c_0; c_1; c_2$ and c_3 which stand for the objects, the arrows, the composable pairs and the composable triples of arrows respectively. The arrows of the sketch are:

$$u : c_0 \;_{\textsubscript{i}}! \; c_1$$

$$s; t : c_1 \;_{\textsubscript{i}}! \; c_0$$

$p_1; p_2; c : c_2 \to c_1$

$p_1; p_2; p_3 : c_3 \to c_1$

$\langle p_1; p_2 \rangle; \langle p_2; p_3 \rangle; \langle p_1; c \rangle; \langle c; p_3 \rangle : c_3 \to c_2$

$\langle u \circ t; id \rangle; \langle id; u \circ s \rangle : c_1 \to c_2$

The intention is that in a **Set**-model, the arrows of the sketch will be interpreted as follows:

u is the unit function which assigns to each object its identity,

s and t are the source and target functions from the set of arrows to the set of objects, and

c is the function which takes a composable pair of arrows to its composite.

The remaining arrows of the sketch are projections from a limit or are interpreted as arrows to a limit with specified projections.

The diagrams are
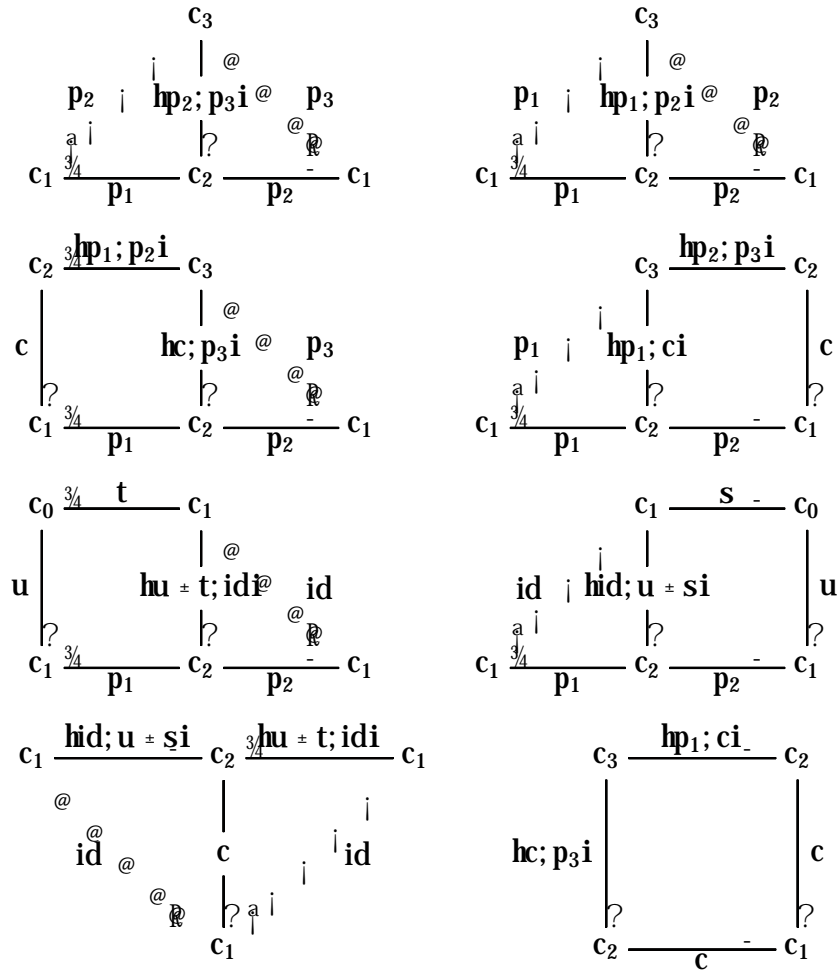
Finally, there are two cones that say that c_2 and c_3 are interpreted as the objects of composable pairs and triples or arrows, respectively.
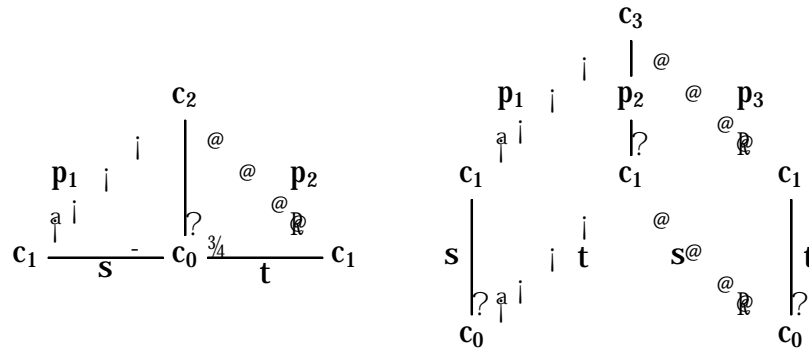
This sketch is the **sketch of categories** and is one of the simpler FL sketches around.

The category of models (in **Set**) of an FP sketch must be regular (see [Barr and Wells, 1985], Theorem 1 of Section 8.4) and it can be proved that the category of categories and functors is not regular (see Exercise 6 of Section 8.6). It follows that categories and functors cannot be the models of an FP sketch.

Lawvere [1966] described another structure whose models are categories. Let **1**; **2**; **3** and **4** denote the total orders with one, two, three and four elements, considered as categories in the usual way a poset is. Let $\mathscr{L}$ denote the opposite category. In that opposite category, it turns out that $\mathbf{3} = \mathbf{2} \pounds_1 \mathbf{2}$ and $\mathbf{4} = \mathbf{2} \pounds_1 \mathbf{2} \pounds_1 \mathbf{2}$. Then **Cat** is the category of limit-preserving functors of $\mathscr{L}$ into **Set** with natural transformations as arrows. It follows that the sketch whose objects and arrows are those of $\mathscr{L}$, whose diagrams are the commutative diagrams of $\mathscr{L}$ and cones are the limit cones of $\mathscr{L}$ is another sketch, closely related in fact to the one above, whose category of models is **Cat**.

We show how binary trees can be described as models of an FL sketch in the next section.

2.1.6 Exercises

1. A **groupoid** is a category in which every arrow is an isomorphism. Explain how to modify the sketch for categories to get a sketch for groupoids.

2. Show that in the category of sets, the de¯nition of homomorphism between two models of the sketch for categories gives the usual de¯nition of functor.

2.2 Initial term models of FL sketches

Like FP sketches, FL sketches always have initial models. The construction 7.6.5 produced an initial term algebra for each ¯nite FP sketch. A modi¯cation of the last three rules in 7.6.5 is su±cient to construct an initial term algebra for each ¯nite FL sketch. This will be described below.

A different construction is in Barr [1986b], where it is proved that in fact FL sketches have free algebras on any typed set X (see Section 9.2.) Volger [1987] gives a logic-based proof for the special case of Horn theories (see [Volger, 1988] for applications).

The modification of 7.6.5 is required by the fact that the base diagram D of a cone in $\mathscr{L}$ need not be discrete in the case of general limits; that is, the shape graph $\mathscr{I}$ of the diagram D in Section 8.2 may have nontrivial arrows $u : i \to j$. A limit of such a cone in the category of sets is not just any tuple of elements of the sets corresponding to the nodes of $\mathscr{I}$, but only tuples which are compatible with the arrows of $\mathscr{I}$.

2.2.1 Precisely, suppose $E : \mathscr{I} \to \mathbf{Set}$ is a finite diagram (we use the letter E instead of D to avoid confusion below). A **compatible family** of elements of E is a sequence $(x_1, \ldots, x_n)$ indexed by the nodes of $\mathscr{I}$ for which

C–1 $x_i \in E(i)$ for each node i.

C–2 If $u : i \to j$ in $\mathscr{I}$, then $E(u)(x_i) = x_j$.

An initial term algebra of an FL sketch $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L})$ is then the least model satisfying the following requirements.

FL–1 If $f : a \to b$ is an arrow of $\mathscr{G}$ and $[x]$ is an element of $I(a)$, then $[fx] \in I(b)$ and $I(f)[x] = [fx]$.

FL–2 If $(f_1, \ldots, f_m)$ and $(g_1, \ldots, g_k)$ are paths in a diagram in $\mathscr{D}$, both going from a node labeled a to a node labeled b, and $[x] \in I(a)$, then

$$(If_1 \circ If_2 \circ \cdots \circ If_m)[x] = (Ig_1 \circ Ig_2 \circ \cdots \circ Ig_k)[x]$$

in $I(b)$.

FL–3 If $D : \mathscr{I} \to \mathscr{G}$ is a diagram, $([x_1], \ldots, [x_n])$ is a compatible family of elements of $I \circ D$ and $p : q \to D$ is a cone over D in $\mathscr{L}$, then $[C(x_1, \ldots, x_n)]$ is an element of $I(q)$.

FL–4 If $p : q \to D$ and $([x_1], \ldots, [x_n])$ are as in FL–ES 3, and $x_1^0, \ldots, x_n^0$ is a compatible family of elements of $I \circ D$ for which $[x_i] = [x_i^0]$ for $i = 1, \ldots, n$, then

$$[C(x_1, \ldots, x_n)] = [C(x_1^0, \ldots, x_n^0)]$$

FL–5 If $p : q \to D$ and $([x_1], \ldots, [x_n])$ are as in FL–ES 3, then for $i = 1, \ldots, n$

$$[p_i C(x_1, \ldots, x_n)] = [x_i]$$

FL–6 If $p : q \to D$ is a cone over D in $\mathscr{L}$ and $x \in I(q)$, then

$$[x] = [C(p_1 x, \ldots, p_n x)]$$

18   More about sketches

Compatibility implies that for any arrow $u : i \to j$ of $\mathscr{I}$,

$$I(D(u))([x_i]) = [x_j]$$

As in 7.6.5, it follows that

$$I(p_i)([C(x_1; \ldots; x_n)]) = [x_i]$$

for each i.

2.2.2 Binary trees   We describe an FL sketch whose initial term algebra is the set of binary trees of integers. We gave an FD sketch for binary trees in ES 1.3.11; the sketch given here illustrates a different approach to the problem that operations such as taking the datum at the root or producing the left or right subtrees are not defined on the empty tree.

We have the following basic nodes in the sketch: 1; t; t^+; b; n. These should be thought of as representing the types of binary trees, nonempty binary trees, the Boolean algebra 2 and the natural numbers, respectively. We have the following operations:

empty : $1 \to t$	empty? : $t \to b$	incl : $t^+ \to t$
val : $t^+ \to n$	left : $t^+ \to t$	right : $t^+ \to t$
zero : $1 \to n$	succ : $n \to n$	true : $1 \to b$
and : $b \times b \to b$	not : $b \to b$	

The intended meaning of these operations is as follows: the constant empty⟨⟩ is the empty tree; empty? is the test for whether a tree is the empty tree; incl is the inclusion of the set of nonempty trees in the set of trees; val(T) is the datum stored at the root of the nonempty tree T; left(T) and right(T) are the right and left branches (possibly empty) of the nonempty tree T, respectively. The remaining operations are the standard operations appropriate to the natural numbers and the Boolean algebra 2.

We require that

and

$$t^+$$

$$@ \quad \text{incl}$$

$$1 \qquad\qquad t \qquad\qquad (2.1)$$

$$\text{not} \pm \text{true}^@ \qquad\qquad \text{empty?}$$

$$b$$

be cones and that

$$1 \xrightarrow{\;\text{empty}\;} t$$

$$\text{true}^@ \qquad \text{empty?}$$

$$b$$
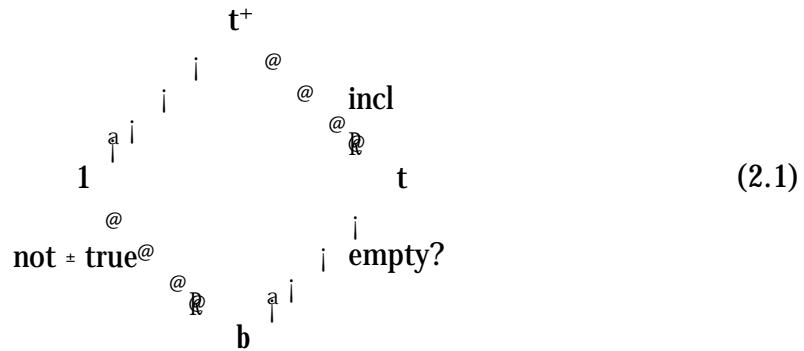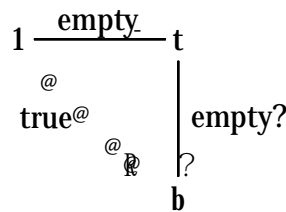
be a diagram.

In the cone (ES 2.1), there should be an arrow from the vertex to the node **b**. It will appear in a model as either of the two (necessarily equal) composites. Since its value is forced, it is customary to omit it from the cone; however, there actually does have to be such an arrow there to complete the cone (and the sketch). In omitting it, we have conformed to the standard convention of showing explicitly only what it is necessary to show.

As in ES 1.3.11, the existence of the ¯rst cone says that every nonempty tree can be represented uniquely as a triplet

$$(\text{left}(T); \text{val}(T); \text{right}(T))$$

The fact that (ES 2.1) is a cone requires that in a model M, $M(t^+)$ be exactly the subset of $M(t)$ of those elements which evaluate to false under $M(\text{empty?})$.

2.2.3 Exercise

1. In the sketch for binary trees, show that for any model M, $M(\text{incl})$ is an injective function. (Hint: use Diagram (ES 2.1).)

2.3 The theory of an FL sketch

Just as in the case of linear and FP sketches, every FL sketch generates a category with ¯nite limits in which it has a universal model.

2.3.1 Theorem   Given any FL sketch $\mathscr{S}$, there is a category $\mathbf{Th_{FL}}(\mathscr{S})$ with finite limits and a model $M_0 : \mathscr{S} \to \mathbf{Th_{FL}}(\mathscr{S})$ such that for any model $M : \mathscr{S} \to \mathscr{E}$ into a category with finite limits, there is a functor $F : \mathbf{Th_{FL}}(\mathscr{S}) \to \mathscr{E}$ that preserves finite limits for which

(i) $F \circ M_0 = M$, and

(ii) if $F' : \mathbf{Th_{FL}}(\mathscr{S}) \to \mathscr{E}$ is another functor that preserves finite limits for which $F' \circ M_0 = M$, then F and F' are naturally isomorphic.

$\mathbf{Th_{FL}}(\mathscr{S})$ is defined up to equivalence by the following properties:
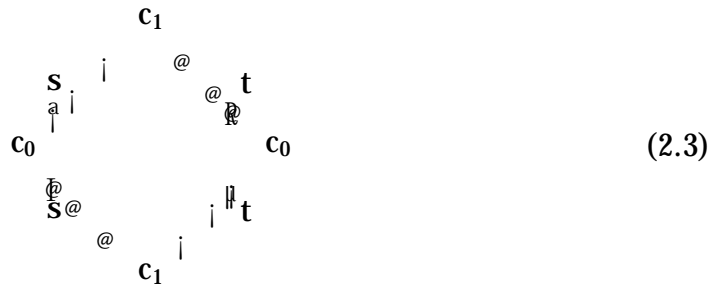
FLT–1 $\mathscr{T}$ has all finite limits.

FLT–2 M_0 takes every diagram of $\mathscr{S}$ to a commutative diagram in $\mathscr{T}$.

FLT–3 M_0 takes every cone of $\mathscr{S}$ to a limit cone of $\mathscr{T}$.

FLT–4 No proper subcategory of $\mathscr{T}$ includes the image of M_0 and satisfies FLT-1, FLT-2 and FLT-3.

The comments concerning Theorem 7.5.1 apply here too. See [Barr and Wells, 1985], Section 4.4, Theorem 2 (p. 156). The constructions in [Ehresmann, 1968] and [Bastiani and Ehresmann, 1972] are more general since they allow cones over infinite diagrams. Barr and Wells [1994] provide a construction of $\mathscr{T}$ that shows that every object of $\mathscr{T}$ is a limit of a finite diagram in the graph of $\mathscr{S}$.

2.3.2 Example   The sketch for semigroups (see Section 7.2) is an FP sketch and therefore an FL sketch whose cones all go to discrete diagrams. The FL theory of this sketch contains a node v that is the limit of this diagram:

$$s \xrightarrow{\quad c \quad} s \times s$$

$$\mathrm{id}_s \qquad\qquad c$$

$$s \tag{2.2}$$

Since M_0 is injective, we write s for $M_0(s)$, $s \times s$ for $M_0(s \times s)$, and so on.

A model M of the sketch induces a limit preserving functor F which must take v to a set. Since v is the equalizer of id_s and $c \circ c$, $F(v)$ must be the set of elements e of $M(s)$ that satisfy the requirement $ee = e$, that is, the set of idempotents in the semigroup $M(s)$. (This set is not in general a subsemigroup.)
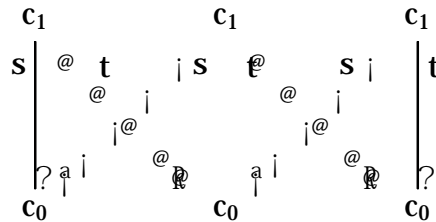
2.3.3 Example The FL theory of the sketch for categories (see ES 2.1.5) contains a node v that is the limit of the diagram

$$
\begin{array}{c}
c_1 \\
s \quad i \quad @ \quad t \\
a \, i \quad\quad @ \, \mathbb{R} \\
c_0 \quad\quad\quad c_0 \\
\mathbb{P} \\
s @ \quad\quad i \parallel t \\
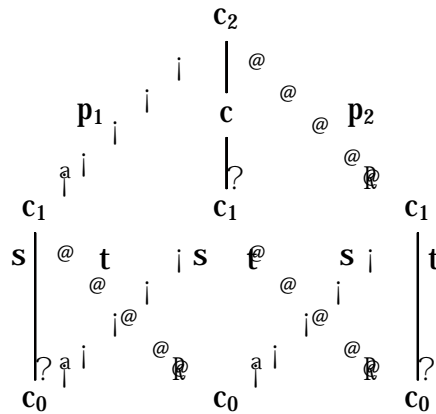@ \quad\quad i \\
c_1
\end{array}
\tag{2.3}
$$

If $\mathscr{C}$ is a small category, hence a model of the sketch, the induced limit-preserving functor F takes v to the set of all parallel pairs of arrows in $\mathscr{C}$.

2.3.4 Exercises

1. The following diagram in the sketch for categories has a limit v in the FL theory of that sketch. What is its value in a model?

$$
\begin{array}{ccc}
c_1 & c_1 & c_1 \\
s \; @ \; t \quad i \; s \; \mathbb{P} & s \; i & t \\
@ \; i \quad @ \; i \\
i @ \quad i @ \\
? \, a \, i \quad @ \, \mathbb{R} \quad a \, i \quad @ \, \mathbb{R} \; ? \\
c_0 & c_0 & c_0
\end{array}
$$

2. Same question as Exercise ES 1 for the following diagram:

$$
\begin{array}{ccc}
 & c_2 & \\
 & i \; \mid @ \; @ & \\
p_1 \; i & c & @ \; p_2 \\
a \, i & \; \mid ? & @ \, \mathbb{R} \\
c_1 & c_1 & c_1 \\
s \; @ \; t \quad i \; s \; \mathbb{P} & s \; i & t \\
@ \; i \quad @ \; i \\
i @ \quad i @ \\
? \, a \, i \quad @ \, \mathbb{R} \quad a \, i \quad @ \, \mathbb{R} \; ? \\
c_0 & c_0 & c_0
\end{array}
$$

3. Show that any model M of a sketch $\mathscr{S}$ is isomorphic to a model M^0 for which $M^0(c)$ and $M^0(d)$ have no elements in common if $c \neq d$.

2.4 General definition of sketch

Although we will not be using the most general notion of sketch, we give the definition here.

2.4.1 Definition   A **sketch** $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L}, \mathscr{K})$ consists of a graph $\mathscr{G}$, a set $\mathscr{D}$ of diagrams in $\mathscr{G}$, a set $\mathscr{L}$ of cones in $\mathscr{G}$ and a set $\mathscr{K}$ of cocones in $\mathscr{G}$.

2.4.2 Definition   A model M of a sketch $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L}, \mathscr{K})$ in a category $\mathscr{A}$ is a homomorphism from $\mathscr{G}$ to the underlying graph of $\mathscr{A}$ that takes every diagram in $\mathscr{D}$ to a commutative diagram, every cone in $\mathscr{L}$ to a limit cone and every cocone in $\mathscr{K}$ to a colimit cocone.

2.4.3 Definition   Let M, N be models of a sketch $\mathscr{S}$ in a category $\mathscr{A}$. A **homomorphism of models** $\circledR : M \to N$ is a natural transformation from M to N.

Sketches in general do not have initial algebras, or even families of initial algebras. What they do have is `locally free diagrams' as described by Guitart and Lair [1981, 1982].

2.4.4 Regular sketches   Chapter 8 of [Barr and Wells, 1985] describes special classes of sketches with cones and cocones that have in common the fact that their theories are embedded in a topos, and so inherit the nice properties of a topos. (Toposes are discussed in Chapter ES 5.) These are the regular sketches, coherent sketches and geometric sketches. We describe regular sketches here to illustrate the general pattern. (Note: The French school uses the phrase \regular sketch" for a sketch in which no node is the vertex of more than one cone.)

2.4.5   An arrow $f : A \to B$ is a regular epimorphism if and only if there is a cocone diagram

$$C \underset{h}{\overset{g}{\rightrightarrows}} A \xrightarrow{f} B$$

It follows that the predicate of being a regular epimorphism can be stated within the semantics of a finite limits sketch. Note that an arrow can be required to become an epimorphism (not necessarily regular) in a model by using the dual of Theorem 8.3.3.

Let us say that a **regular cocone** is one of the form

$$c \rightrightarrows a \to b$$

2.4.6 Definition   A **regular sketch** $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L}, \mathscr{K})$ consists of a graph, a set of diagrams, a set of finite cones and a set of regular cocones.

There is one undesirable feature to the definition above. The introduction of a regular sketch requires the introduction of a sort c and two arrows $c \rightrightarrows a$. A model will have to provide a value for c as well as the two arrows. This is generally irrelevant information that one would not normally want to have to provide. Worse, the definition of natural transformation is such that arrows

between models will have to preserve this additional information and that is
definitely undesirable. The way in which this is usually dealt with is by adding,
with each cocone

$$c \rightrightarrows^{d^0}_{d^1} a \xrightarrow{d} b$$

the cone

$$
\begin{array}{ccc}
c & \xrightarrow{d^0} & a \\
\scriptstyle d^1 \downarrow & & \downarrow \scriptstyle d \\
a & \xrightarrow{d} & b
\end{array}
$$

It follows from Exercise 5 of Section 8.4 that if d has a kernel pair, then d is
a regular epimorphism if and only if it is the coequalizer of that kernel pair.
The addition of this cone thus adds no new data, at least in the case of models
in a category with finite limits and coequalizers. The preservation of the kernel
pair and the two arrows by homomorphisms of models is automatic, given the
universal mapping properties of limits.

The point of these considerations is that a regular sketch can be described
as one in which one can specify any kind of finite limits and that any particular
arrow is a coequalizer of some pair, without having to specify (or preserve in an
arrow between models) what pair of arrows it coequalizes.

One of the things we want a sketch to do is minimize the number of items — or
amount of information — that has to be specified and maximize the amount that is
implicit in the sketch. Thus we want to be able to say that an arrow is a regular
epimorphism (that is, coequalizer of some pair of arrows) without specifying what
pair it coequalizes. In the preceding discussion, we described a way to do this: to
observe that a regular epi is a coequalizer of its own kernel pair and that the kernel
pair can be calculated. There is a price to pay in solving the problem in this way,
since this works only when you form models in a category that has that kernel pair.
An alternative approach would be to simply add regular epi to the list of primitives
that a sketch can make use of, so that one could require than an arrow be a regular
epi, without having to specify the arrows it coequalizes, in a category that may or
may not have pullbacks. This is an example of expanding the concept of sketch to
allow specification of constructions other than commutative diagrams, limits and
colimits. (References to generalized sketches are given at the end of this chapter.)

As an example, we construct a sketch for reflexive graphs — graphs with the
property that there is at least one loop on each node. The problem that arose
in 4.6.9 that homomorphisms had to take certain specific loops to other specific
loops does not arise in this construction.

We take the sketch for graphs with nodes n and a and arrows source, target : a → n. We add a new type r with a cone

$$r \rightarrow a \begin{array}{c} \text{source} \\ \downarrow\downarrow\downarrow\downarrow\downarrow \\ \text{target} \end{array} n$$

and a cocone to express that the composite r → a → n (the latter arrow being either source or target) is regular epic. In a model, r becomes the set of loops. The fact that the arrow r → a → n maps surjectively to the nodes implies that there is at least one loop at each node. A homomorphism from M to M^0 must take M(r) to $M^0(r)$ as a set, but there is no particular loop in M(r) that is distinguished by the construction.

2.4.7 Theories of sketches with colimits.   The reader may have noticed that in Chapter 7, we discussed FP sketches, their initial algebras and their theories, and in Chapter ES 1 we discussed FD sketches and their initial algebras. We did not discuss theories for FD sketches. In fact, theories exist for FD sketches, and indeed for all sketches. This is because sketches can themselves be modeled by an FL theory, and the theory of a sketch can then be realized as an initial algebra (see Section ES 2.2). See [Wells, 1990], [Bagchi and Wells, 1994].

Here, we state the universal properties for theories of FD sketches for models in categories with finite products and finite disjoint universal sums. The resulting theory has these properties, too, and is embedded in a topos. The proof by initial algebras just mentioned does not give such an embedding.

2.4.8 Theorem   Let $\mathscr{S}$ be an FD sketch. Then there is a category denoted $\mathbf{Th_{FD}}(\mathscr{S})$ with finite disjoint universal sums and a model $M_0 : \mathscr{S} \rightarrow \mathbf{Th_{FD}}(\mathscr{S})$ such that for any model $M : \mathscr{S} \rightarrow \mathscr{E}$ into a category with finite products and finite disjoint universal sums, there is a functor $F : \mathbf{Th_{FD}}(\mathscr{S}) \rightarrow \mathscr{E}$ that preserves finite products and finite sums for which

  (i) $F \circ M_0 = M$, and
  (ii) If $F^0 : \mathbf{Th_{FD}}(\mathscr{S}) \rightarrow \mathscr{E}$ is another functor that preserves finite products and finite sums for which $F^0 \circ M_0 = M$, then F and F^0 are naturally isomorphic.

A proof may be found in [Barr and Wells, 1985], Proposition 1 of Section 8.2. (FD sketches are called FS sketches there.)

The situation is similar for regular sketches.

2.4.9 Theorem   Let $\mathscr{S}$ be a regular sketch. Then there is a regular category denoted $\mathbf{Th_{Reg}}(\mathscr{S})$ and a model $M_0 : \mathscr{S} \rightarrow \mathbf{Th_{Reg}}(\mathscr{S})$ such that, for any model $M : \mathscr{S} \rightarrow \mathscr{E}$ into a regular category, there is a regular functor $F : \mathbf{Th_{Reg}}(\mathscr{S}) \rightarrow \mathscr{E}$ such that

  (i) $F \circ M_0 = M$, and

(ii) if $F': \mathbf{Th_{Reg}}(\mathscr{S}) \to \mathscr{E}$ is another regular functor for which $F' \cdot M_0 = M$, then F and F' are naturally isomorphic.

The theories of many types of sketches can be constructed as subcategories of toposes; this is done in [Barr and Wells, 1985], Chapters 4 and 8. Other constructions of theories for special kinds of sketches are given in [Ehresmann, 1968], [Bastiani and Ehresmann, 1972], [Peake and Peters, 1972], [Kelly, 1982], [Wells, 1990] and [Barr and Wells, 1994].

2.4.10 Categories of set-valued models of general sketches can be axiomatized as **accessible categories** (see [Makkai and Paré, 1990], [Adámek and Rosicky, 1994].) A precise statement of the relationship between categories of models of first order theories and categories of models of sketches is given in [Adámek and Rosicky, 1994] (Theorems 5.35 and 5.44).

Generalizations of the concept of sketch are given by Lair [1987], Wells [1990] (see also [Bagchi and Wells, 1994]), Power and Wells [1992], and Makkai [Makkai, 1994].

3

The category of sketches

The first section of this chapter defines the concept of homomorphism of sketches, yielding a category of sketches. In Section ES 3.2 we describe a formalism for defining parametrized data types using sketch homomorphisms. In Section ES 3.3 we develop the theory of sketches further, showing that a homomorphism of sketches induces a contravariant functor between the model categories and making contact with Goguen and Burstall's concept of institution.

Section ES 3.3 requires only Section ES 3.1 to read. Nothing in this chapter is needed later in the book.

Much more is known about the category of sketches than is mentioned here. It is cartesian closed, for example. A basic study in English of the category of sketches which is oriented toward computer science is given by Gray [1989].

3.1 Homomorphisms of sketches

3.1.1   Let $\mathscr{S} = (\mathscr{G}, \mathscr{D}, \mathscr{L}, \mathscr{K})$ and $\mathscr{S}^0 = (\mathscr{G}^0, \mathscr{D}^0, \mathscr{L}^0, \mathscr{K}^0)$ be sketches. A graph homomorphism $F : \mathscr{G} \to \mathscr{G}^0$ takes a diagram $D : \mathscr{I} \to \mathscr{G}$ to a diagram $F \cdot D : \mathscr{I} \to \mathscr{G}^0$ which is called the **image** of D in $\mathscr{G}^0$. It takes a cone $p : v \to D$ to a cone $F(p) : F(v) \to F \cdot D$ where, for each node a of the shape graph of D, $F(p)_a : F(v) \to F(D(a))$ is defined to be $F(p_a)$. It is defined on cocones similarly.

An arrow or **homomorphism of sketches** $F : \mathscr{S} \to \mathscr{S}^0$ is a graph homomorphism from the graph $\mathscr{G}$ to the graph $\mathscr{G}^0$ for which, if D is a diagram in $\mathscr{D}$ then $F \cdot D$ lies in $\mathscr{D}^0$; if $p : v \to D$ is a cone in $\mathscr{L}$ then $F(p)$ is a cone in $\mathscr{L}^0$; and if $c : D \to v$ is a cocone in $\mathscr{K}$ then $F(c)$ is a cocone in $\mathscr{K}^0$.

Note that if in the sketch $\mathscr{S}$ the sets of diagrams, cones and cocones are all empty, then an arrow from $\mathscr{S}$ to $\mathscr{S}^0$ is precisely a graph homomorphism from $\mathscr{G}$ to $\mathscr{G}^0$.

3.1.2 Example   Let $\mathscr{E}$ denote the impoverished sketch which has one node we will call e and no arrows, diagrams, cones or cocones. Any assignment of e to a node of any sketch is a morphism of sketches.

3.1.3 Example   The sketch for graphs has the graph

$$a \underset{\text{target}}{\overset{\text{source}}{\rightrightarrows}} n$$

and no cones or diagrams. There is a graph homomorphism from the graph of the sketch of 4.2.17 to the graph just given that takes 0 to a, 1 to n, and u to source. This is a sketch homomorphism since there are no diagrams, cones or cocones in either sketch. (Of course, there is another sketch homomorphism taking u to target.)

3.1.4 Example   In ES 2.1.4, we modifed the sketch for graphs by adding a cone making it the sketch for simple graphs. The inclusion of the sketch for graphs into the sketch for simple graphs is a homomorphism of sketches.

3.1.5 Example   A ¯eld has an associative binary operation of addition on it, so one would suspect that there is a homomorphism from the sketch for semigroups in 7.2.1 to the sketch for ¯elds in ES 1.2. That is the case. The homomorphism takes s to f, s £ s to f £ f and s £ s £ s to f £ f £ f, and the arrow c to the arrow +. The homomorphism must take projection arrows of cones to corresponding projection arrows, and of course the associativity diagram (7.8) then is mapped to the diagram implied by FE{1 of ES 1.2.

3.1.6 The sketch underlying a category   Let $\mathscr{C}$ be a category. There is an underlying sketch of $\mathscr{C}$, call it Sk($\mathscr{C}$), whose graph consists of the objects of $\mathscr{C}$ as nodes and the arrows of $\mathscr{C}$ as arrows. This underlying sketch is not in general ¯nite or even small. The commutative diagrams of this sketch are all diagrams that are commutative in $\mathscr{C}$. Similarly we take for cones all those that are limit cones in $\mathscr{C}$ and for cocones all those that are colimit cocones. An arrow from $\mathscr{S}$ to Sk($\mathscr{C}$) is then exactly what we have called a model of $\mathscr{S}$ in $\mathscr{C}$ in ES 2.4.2.

We note that although the composition in $\mathscr{C}$ has been forgotten, it can be completely recovered from the knowledge of which diagrams commute. For example, the information f ± g = h is equivalent to the information that

$$\xymatrix{ \mathsf{c} \ar[r]^{g} \ar[dr]_{h} & \mathsf{c} \ar[d]^{f} \\ & \mathsf{c} }$$

commutes so that we could recover the category $\mathscr{C}$ entirely from the underlying sketch (in fact, just from the graph and the commutative triangles).

3.1.7 The category of sketches   With the de¯nition of homomorphism of sketches given above, sketches themselves form a category which we will call **Sketch**. By restricting the shape graphs for the diagrams, cones and cocones one obtains many full subcategories of **Sketch**, for example the category of FP sketches, the category of FD sketches, and so on.

One consequence of this is that all constructions that we can carry out in any category make sense in the category of sketches. The particular construction that interests us here is the formation of colimits, particularly pushouts.

3.1.8 Exercise

1. Let $\mathscr{E}$ be the sketch with one node and no arrows, diagrams, cones or cocones. Show that the category of models of $\mathscr{E}$ in a category $\mathscr{C}$ is isomorphic to $\mathscr{C}$. In particular, the category of models of $\mathscr{E}$ in **Set** is isomorphic to **Set**.

3.2 Parametrized data types
as pushouts

One thing a theory of data types should do is give a way of saying how the data types of stacks of integers, say, is like the type of stacks of reals or for that matter of stacks of arrays of trees of characters. In other words, we need a way of talking about an abstract stack in a way that leaves it open to ¯ll in the blank corresponding to the thing that it is stacks of. The data type is the parameter.

The way we do this is to describe a sketch for stacks of d where d stands for an abstract data type and then use a pushout construction to identify d with a concrete data type in any particular application. The point is that pushouts are the general way we use to identify things. We give several illustrations.

3.2.1 Abstract stacks Consider the sketch whose graph consists of nodes s; t; d; d £ s and 1. The idea is that s stands for the set of stack con¯gurations, t the set of nonempty stack con¯gurations, and d the data. There are operations push : d £ s ¡! t, pop : t ¡! d £ s, empty : 1 ¡! s and incl : t ¡! s. In order to state the equations we also have two arrows $\mathrm{id}_{d £ s}$ and id_t which will be forced to be identity arrows.
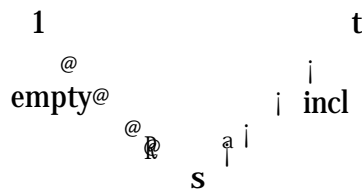
There are four equations (diagrams). Two of them, $\mathrm{id}_{d £ s} = ()_{d £ s}$ and $\mathrm{id}_t = ()_t$, force those arrows to be identities, and the following two, which express the essence of being a stack:

$$\mathrm{pop} \pm \mathrm{push} = \mathrm{id}_{d £ s}$$

and

$$\mathrm{push} \pm \mathrm{pop} = \mathrm{id}_t$$

There are the cones to express 1 as terminal and d £ s as a product, and one cocone

So far, this sketch is not very interesting, since the type of d is still undetermined. In fact its initial model I has $I(d) = I(t) = \emptyset$ and $I(s) = \{empty\}$. Note that $\{empty\}$ is not empty. It contains one element which we interpret as representing the empty stack. Since the type d of data is empty, the empty stack is the only kind of stack there is in the model I.

If there were constants in the data type d, then $I(s)$ would be the set of all possible configurations of a stack of data of type $I(d)$, and $I(t)$ would be the set of all possible configurations other than the empty stack.

3.2.2 Stacks of natural numbers   In 4.7.7, we described the sketch with two nodes, 1 and n, two operations zero : $1 \to n$ and succ : $n \to n$. There are no equations (diagrams), just the cone describing 1 as terminal and no cocones. This sketch, which we called **Nat**, has models that can be reasonably viewed as describing natural numbers. We described a more elaborate sketch in ES 1.1.4 which could also be used in what follows.

Let us consider the following diagram in the category of sketches, where $\mathscr{E}$ is the trivial sketch defined in Example ES 3.1.2 and the arrows F and G are defined by letting $Fe = n$ and $Ge = d$.

$$
\begin{array}{ccc}
\mathscr{E} & \xrightarrow{\ F\ } & \mathbf{Nat} \\[2pt]
G \downarrow \ \ & & \\[2pt]
\mathbf{Stack} & &
\end{array}
\qquad (3.1)
$$

A pushout of this diagram is a sketch **Stack**(**Nat**) made by forming the union of **Nat** and **Stack** and then identifying the node n of **Nat** with the node d of **Stack**.

From the definition of pushout, it follows that a model of this sketch is a model of the sketch for natural numbers that is simultaneously a model of the sketch for stacks whose value at the node d of **Stack** is the same as its value at the node n of **Nat**. Models of this sketch can be identified as stacks of natural numbers. The effect of this construction is to fill in the data parameter in the sketch for stacks with an actual data type. Of course, this data type of natural numbers could be replaced by any data type desired (including stacks of natural numbers!) by replacing $F : \mathscr{E} \to \mathbf{Nat}$ by an appropriate sketch homomorphism.

Although we think of this sketch as being that of stacks of natural numbers, the pushout construction is completely symmetric and any asymmetry is imposed by our way of looking at it. It is caused by our (quite reasonable) perception that the node d of **Stack** is an input parameter and the node s represents the actual data type.

3.2.3 Binary trees, revisited again   Here is a second example. In ES 1.3.11, we described a sketch called **BinTree** and then another sketch of the same name in ES 2.2.2. Here we describe yet another sketch we call **BinTree**. This third and ¯nal approach is fully parametrized and represents exactly what we mean by binary trees, no more and no less. We use nodes 1, t^+, t and d. Now the only operations we put in are empty; incl; val; left and right. We also need the cones and cocones necessary to say that $t^+ = 1 + t$ with inclusions empty and incl and that $t^+ = t \pounds d \pounds t$ with projections left, val and right.

Then if F : $\mathscr{E}$ ¡! **Nat** is as above and H : $\mathscr{E}$ ¡! **BinTree** is de¯ned by H (e) = d, then the pushout of

$$
\begin{array}{ccc}
\mathscr{E} & \xrightarrow{\quad F \quad} & \textbf{Nat} \\
{\scriptstyle H}\Big\downarrow{\scriptstyle ?} & & \Big\downarrow{\scriptstyle ?} \\
\textbf{BinTree} & \xrightarrow{\quad\quad} & \textbf{BinTree(Nat)}
\end{array}
$$

is a sketch whose models can be interpreted as binary trees of natural numbers.

3.2.4 Further combinators   This operation can be iterated. For example, we can form the pushout

$$
\begin{array}{ccc}
\mathscr{E} & \xrightarrow{\quad K \quad} & \textbf{BinTree(Nat)} \\
{\scriptstyle G}\Big\downarrow{\scriptstyle ?} & & \Big\downarrow{\scriptstyle ?} \\
\textbf{Stack} & \xrightarrow{\quad\quad} & \textbf{Stack(BinTree(Nat))}
\end{array}
$$

with K e = t or even

$$
\begin{array}{ccc}
\mathscr{E} & \xrightarrow{\quad L \quad} & \textbf{Stack(Nat)} \\
{\scriptstyle H}\Big\downarrow{\scriptstyle ?} & & \Big\downarrow{\scriptstyle ?} \\
\textbf{BinTree} & \xrightarrow{\quad\quad} & \textbf{BinTree(Stack(Nat))}
\end{array}
$$

with L e = s. Models of these types will be interpreted as stacks of binary trees of natural numbers, respectively binary trees of stacks of natural numbers. Clearly what is at issue here is a notion of a type having an input node and an output node. However, things are not quite so simple, as the following example shows.

3.2.5   The basic idea of this type is that of n-place records (Pascal) or structures (C). We leave aside here the use of variant records (which can be adequately handled by judicious use of ¯nite sums as C's union type shows). Another question we do not tackle is that of parametrizing n. At this point, we make no attempt to
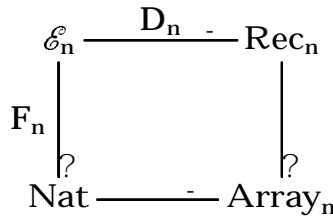
relate the sketches for two-place and three-place records, say, although it seems clear that in a mature theory this will be done. Parametrizing n is discussed in [Wagner, 1986] and [Gray, 1989].

So we describe a sketch we will call **Rec**$_n$. It has nodes r and $d_1; d_2; ::::; d_n$ and one cone
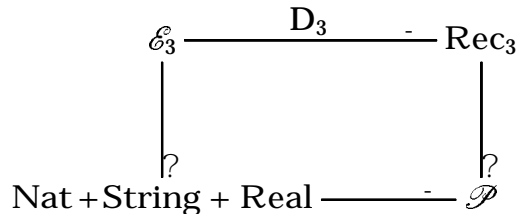
$$r$$

$$d_1 \qquad ¢¢¢ \qquad d_n$$
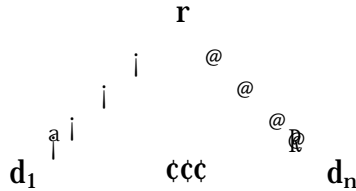
As it stands, the initial model of this sketch has all values empty. It is really a shell of a sketch. To give it content, we must ¯ll in values for the data. To do this we must ¯rst recognize that the sketch has n input types and these must all be speci¯ed. This could be done in n steps, but it is more coherent to do it all at once.

Let $\mathscr{E}_n$ denote the discrete graph with nodes $e_1; ::::; e_n$. There is an obvious arrow $D_n : \mathscr{E}_n ¡!$ **Rec**$_n$ given by $De_i = d_i$. If, for example, we formed the pushout

$$
\begin{array}{ccc}
\mathscr{E}_n & \xrightarrow{\ D_n\ } & \mathbf{Rec}_n \\
{\scriptstyle F_n}\Big\downarrow & & \Big\downarrow \\
\mathbf{Nat} & \xrightarrow{\quad} & \mathbf{Array}_n
\end{array}
$$

where $F_n e_i = n; \; i = 1; ::::; n$, the resultant sketch can be evidently interpreted as arrays of natural numbers. On the other hand, the pushout

$$
\begin{array}{ccc}
\mathscr{E}_3 & \xrightarrow{\ D_3\ } & \mathbf{Rec}_3 \\
\Big\downarrow & & \Big\downarrow \\
\mathbf{Nat} + \mathbf{String} + \mathbf{Real} & \xrightarrow{\quad} & \mathscr{P}
\end{array}
$$

(where we suppose that sketches for **String** and **Real** have already been de¯ned) is a sketch for three-place records, of which the ¯rst ¯eld is natural numbers, the second is strings and the third is °oating point reals.

The ideas in this section are discussed without the explicit use of sketches in [Thatcher, Wagner and Wright, 1982], [Ehrig et al., 1984] and [Wagner, 1986].

3.2.6 Parametrizing operations The sketch $\mathscr{E}$ in the upper left corner of the pushout diagrams can contain operations. For example, let **Nat**$^+$ denote the sketch for natural numbers with an added operation $+ : n \pounds n ¡!$ n with diagrams forcing it to be addition in the initial model, and **Nat**$^¤$ a similar sketch with an

added operation $\times : n \times n \to n$ forced to be multiplication there. Let **Diag** be the sketch whose graph contains operations

$$n \xrightarrow{\varsigma} n \times n \xrightarrow{m} n$$

a cone with arrows $p_i : n \times n \to n$, $i = 1, 2$, to force $n \times n$ to be the indicated product and a diagram to force ς to be the diagonal map in a model. Let **BinOp** be the sketch containing $n \times n \to n$ and similar data making $n \times n$ the indicated product. **BinOp** is included in both **Diag** and in each of **Nat**$^+$ and **Nat**$^\times$. A pushout

$$
\begin{array}{ccc}
\textbf{BinOp} & \longrightarrow & \textbf{N} \\
\downarrow & & \downarrow \\
\textbf{Diag} & \longrightarrow & \mathscr{P}
\end{array}
$$

with **N** = **Nat**$^+$ would add a doubling operator to **Nat**$^+$ and with **N** = **Nat**$^\times$ would add a squaring operator to **Nat**$^\times$.

3.2.7 Arithmetic   One last, more speculative example will show that we have barely begun to explore this idea. Let us suppose that we have de¯ned two sketches called **Real** and **Complex** that implement the arithmetic operations (addition, subtraction, multiplication and division) of real and complex numbers, respectively, as well as absolute value. While it is true that real numbers are a special case of complex numbers and therefore do not, in principle, have to be treated di®erently, it is also true that real arithmetic is much easier and faster and virtually every language that implements a complex number type treats them separately.

Suppose, further, there is a sketch **Trig** whose operations implement the trigonometric functions by means of power series or other approximations which have the same algorithm for real and for complex numbers. This sketch will use four formal arithmetic operations on an abstract data type d which admits the arithmetic operations and absolute value, but is not otherwise speci¯ed. These ¯ve operations are determined by a sketch **Arith** which has an arrow to all of **Real**, **Complex**, **Trig**.

If we now form the pushout

$$
\begin{array}{ccc}
\textbf{Arith} & \longrightarrow & \textbf{Real} \\
\downarrow & & \downarrow \\
\textbf{Trig} & \longrightarrow & \textbf{RealTrig}
\end{array}
$$

we get a sketch for the real type with trigonometric functions and if we replace **Real** by **Complex**, we get a sketch for complex trigonometry.

3.2.8 Exercise

1. If $\mathscr{S}$ is the pushout of Diagram (ES 3.1), describe precisely the nodes in $\mathscr{S}$ which must become a singleton in a model in **Set**.

3.3 The model category functor

Sketch is a category whose objects are sketches. If $\mathscr{C}$ is a fixed category, then each sketch produces a category of models of the sketch in $\mathscr{C}$ (see 7.4.4). The category of models in $\mathscr{C}$ of a sketch $\mathscr{S}$ we will call $\mathbf{Mod}_{\mathscr{C}}(\mathscr{S})$. ($\mathbf{Mod}_{\mathscr{C}}(\mathscr{S})$ might very well be the empty category.)

> Most of the examples in this book have had $\mathscr{C} = $ **Set**, which regrettably obscures one of the major advantages sketches have over standard logical theories: the fact that their models can be in any suitable category.

3.3.1 For each sketch homomorphism $F : \mathscr{S} \to \mathscr{T}$ we will now describe a functor $\mathbf{Mod}_{\mathscr{C}}(F) : \mathbf{Mod}_{\mathscr{C}}(\mathscr{T}) \to \mathbf{Mod}_{\mathscr{C}}(\mathscr{S})$ (note the reversal). $\mathbf{Mod}_{\mathscr{C}}(F)$ is also denoted F^{*}, and is called the **functor induced by** F. F^{*} is defined by MF–1 and MF–2 below. If M is a model of $\mathscr{T}$ then $F^{*}(M)$ is a model of $\mathscr{S}$.

MF–1 The object function of F^{*} is defined by $F^{*}(M) = M \circ F$. Thus if g is a node or arrow of the graph of $\mathscr{S}$ and M is a model of $\mathscr{T}$, then $F^{*}(M)(g) = M(F(g))$.

MF–2 If $\alpha : M \to N$ is a homomorphism of models of $\mathscr{T}$, define $F^{*}(\alpha)$ to be the natural transformation αF as defined in Section 4.4. Thus at a node g of the graph of $\mathscr{S}$,

$$F^{*}(\alpha)(g) = (\alpha F)g = \alpha F(g) : M(F(g)) \to N(F(g))$$

3.3.2 Proposition For each model M of $\mathscr{T}$ and homomorphism $F : \mathscr{S} \to \mathscr{T}$, $F^{*}(M)$ is a model of $\mathscr{S}$.

Proof. Since M is among other things a graph homomorphism and so is F, and the composite of graph homomorphisms is a graph homomorphism, MF–1 makes $F^{*}(M)$, which is $M \circ F$, respect the source and target of arrows of the graph, so that it is a graph homomorphism to $\mathscr{C}$ as a model should be.

If D is a diagram of $\mathscr{S}$, then because F is a homomorphism of sketches, $F \circ D$ is a diagram of $\mathscr{T}$. Since M is a model of $\mathscr{T}$, $F^{*}(M) \circ D = M \circ F \circ D$ commutes.

Suppose $v \to D$ is a cone of $\mathscr{S}$. Then $F(v) \to F \circ D$ is a cone of $\mathscr{T}$ which must be taken to a limit cone by M. Since $F^{*}(M)(v \to D) = M(F(v)) \to M \circ F \circ D$, $F^{*}(M)$ takes $v \to D$ to a limit cone. A similar argument deals with cocones. □

3.3.3 Proposition   For each homomorphism $\phi : M \to N$ of models of $\mathscr{T}$ and homomorphism $F : \mathscr{S} \to \mathscr{T}$ of sketches, $\mathbf{Mod}_{\mathscr{C}}(F)(\phi)$ as defined by MF–2 is a homomorphism of models of $\mathscr{S}$.

Proof. By MF–2, $F^*(\phi)$ is the natural transformation ϕF defined in 4.4.1. Since its domain and codomain are models, it is a homomorphism of models by definition.   □

For any sketch $\mathscr{S}$, let $\mathbf{Mod}_{\mathscr{C}}(\mathscr{S})$ be the category of models of $\mathscr{S}$ in $\mathscr{C}$ (we called this $\mathbf{Mod}(\mathscr{S}; \mathscr{C})$ in Section ES 2.1. If $F : \mathscr{S} \to \mathscr{T}$ is a homomorphism of sketches, we have defined a functor $\mathbf{Mod}_{\mathscr{C}}(F) : \mathbf{Mod}_{\mathscr{C}}(\mathscr{T}) \to \mathbf{Mod}_{\mathscr{C}}(\mathscr{S})$.

3.3.4 Proposition $\mathbf{Mod}_{\mathscr{C}} : \mathbf{Sketch}^{\mathrm{op}} \to \mathbf{Cat}$ is a functor.

The proof involves some simple checking and is left as an exercise.

3.3.5 Example   In Section 3.1, we gave several examples of underlying set functors $U : \mathscr{C} \to \mathbf{Set}$. In general, such functors are induced by a homomorphism of sketches from the trivial sketch $\mathscr{E}$ defined in ES 3.1.2 to a particular node of the sketch $\mathscr{C}$. For example, the underlying set functor $U : \mathbf{Sem} \to \mathbf{Set}$ (see 3.1.8) is induced by the unique sketch homomorphism from $\mathscr{E}$ to the sketch for semigroups that takes the only node of $\mathscr{E}$ to s. This follows directly from MF–1 and MF–2 and the fact that a model of $\mathscr{E}$ in $\mathbf{Set}$ is essentially a set (Exercise ES 1 of Section ES 3.2). Similarly the underlying arrow and node functors $A : \mathbf{Grf} \to \mathbf{Set}$ and $N : \mathbf{Grf} \to \mathbf{Set}$ (see 3.1.9) are induced by the sketch homomorphisms from $\mathscr{E}$ to the sketch for graphs that take e to a and to n respectively.

3.3.6 Example   The sketch homomorphism of Example ES 3.1.3 induces the functor from the category of graphs to the arrow category of $\mathbf{Set}$ that takes a graph to its source function. The homomorphism of Example ES 3.1.4 that includes the sketch for graphs into the sketch for simple graphs induces the underlying functor that forgets that a graph is simple. Similarly the functor of ES 3.1.5 that includes the sketch for semigroups into the sketch for fields induces the underlying functor from fields to semigroups that takes a field to its additive semigroup.

3.3.7 Example   Sketches usually include only the minimal information that is needed to describe a theory. There is a cost to this in that they may omit crucial information needed to define morphisms. Here is an uncontrived example that illustrates the point. We begin by observing that in any category, if 1 is a terminal object (vertex of a cone with empty base) and $S \times S$ is a product of two

copies of S, then the square

$$
\begin{array}{ccc}
S \times S & \xrightarrow{\quad} & S \\
\downarrow & & \downarrow \\
S & \xrightarrow{\quad} & 1
\end{array}
$$

is a pullback. Now consider the functor from the category **Mon** of monoids to the category **Cat** of small categories that takes a monoid to the corresponding category with one object (see 2.3.12). One expects that this functor would be induced by a sketch homomorphism from the sketch for categories given by ES 2.1.5 to the sketch for monoids given by 7.2.1 as augmented by 7.3.2. Since the underlying set of the monoid is the set of arrows of the corresponding category, the sketch homomorphism should take the node c_1 of ES 2.1.5 to the node s of 7.2.1, and the arrow c of ES 2.1.5 (which we will call comp here to avoid confusion) should go to the multiplication c of 7.2.1. Since comp has domain c_2 and c has domain $s \times s$, c_2 should go to $s \times s$. Unfortunately, $s \times s$ is the vertex of a discrete cone and c_2 is the vertex of a pullback cone. And besides, where should c_0 go?

The key is to map c_0 to the object 1 of the sketch for monoids, so that s and t are forced to go to the unique map from s to 1. This would force the pullback cone

$$
\begin{array}{ccc}
& & c_2 \\
& p_1 \nearrow & \downarrow \quad @ \\
& & \quad @ \quad p_2 \\
c_1 & \xrightarrow{\;s\;} c_0 \xrightarrow[\;t\;]{} & c_1
\end{array}
$$

to go to the cone

$$
\begin{array}{ccc}
& & s \times s \\
& p_1 \nearrow & \downarrow \quad @ \\
& & \quad @ \quad p_2 \\
s & \xrightarrow{\quad} 1 & s
\end{array}
$$

which unfortunately is not a cone in the sketch for monoids. However, it is a cone in the FL theory for the sketch for monoids, so it would appear that, although we cannot realize the functor in question as induced by a sketch homomorphism from the sketch for categories to the sketch for monoids, it is nevertheless induced by a homomorphism to a sketch { the underlying sketch of the theory of monoids.

It is clear that, as the preceding example illustrates, one would normally want to consider homomorphisms of theories (functors that preserve ⁻nite limits in this case) rather than homomorphisms of sketches. In practice, one would construct a homomorphism from a sketch $\mathscr{S}$ to the theory of a sketch $\mathscr{T}$ since

such a homomorphism would extend essentially uniquely to a mapping between theories. An alternative would be to adjoin just what you need to $\mathcal{T}$ to make the homomorphism work (the cone of Diagram (ES 3.3.7) — and others — in this case), but that seems excessively clumsy when the whole theory exists in any case.

3.3.8   It is almost immediate that Propositions ES 3.3.2, ES 3.3.3 and ES 3.3.4 remain true if the category **Sketch** is replaced by the category of sketches with diagrams based on a specific class of shape graphs, cones to specific class of shape graphs not necessarily the same as that of the diagrams, and cocones from a specific class of shape graphs not necessarily the same as either of the other two. For example, taking any shapes for the diagrams, finite discrete shapes for the cones, and no shapes for the cocones gives the category of FP sketches and homomorphisms between them, so all these propositions are true of FP sketches.

3.3.9 Sentences   For this subsection only, we will say a **sentence** in a sketch $\mathcal{S}$ is a diagram, cone or cocone in the graph of $\mathcal{S}$, not necessarily one in the specified set of diagrams, cones or cocones. The sentence is **satisfied** in a model M of $\mathcal{S}$ if it commutes when M is applied (if it is a diagram) or if it is a limit (co)cone when M is applied (if it is a (co)cone).

If $F : \mathcal{S} \to \mathcal{T}$ is a sketch homomorphism, and σ is a sentence of $\mathcal{S}$, then $F(\sigma)$ is a sentence of $\mathcal{T}$ where $F(\sigma)$ is defined by composition: this follows from the definition of sketch homomorphism.

3.3.10 Proposition   If $F : \mathcal{S} \to \mathcal{T}$ is a sketch homomorphism and σ is a sentence of $\mathcal{S}$, then $F(\sigma)$ is satisfied in a model M of $\mathcal{T}$ in a category $\mathcal{C}$ if and only if σ is satisfied in $\mathbf{Mod}_{\mathcal{C}}(F)(M)$.

This follows directly from the definitions and is left as an exercise.

Propositions ES 3.3.4 and ES 3.3.10 imply that the category of sketches with designated shape graphs for each of the diagrams, cones and cocones, together with sentences, form a `simple institution' in the sense of [Goguen and Burstall, 1986].

3.3.11   The results of this section show the way in which sketches are a very different method for describing mathematical structures from the theories of traditional mathematical logic. Of course, sketches use graphs, diagrams, cones and cocones instead of variables, symbols, expressions and formulas, but that difference, although significant, is not the biggest difference. When you use sketches, you factor the entire descriptive process differently. We have already discussed the differences to some extent for FP sketches in 7.2.8.

An example of the subtlety of the difference is exemplified by our word `sentence' above. Diagrams correspond to universally quantified sentences, and the correspondence, which is not entirely trivial (see Section 7.7), is nevertheless exact. But to use a cone as a sentence corresponds to a statement about a type in

multisorted logic, rather than statements about terms. (McLarty [1986] makes this explicit.) Such statements have not played a big role in classical logic. In classical logic, you do not say (with an axiom) that a type is (for example) a product type; the product type is implicit in the existence of some given term which has a sequence of variables of specific types.

On the other hand, our sentences do not provide a way of giving universal Horn clauses in (for example) an FL sketch. When using sketches, universal Horn clauses are usually implicit. Such a clause which must be satisfied in all models is given by an arrow built into the graph of the sketch which causes an operation to factor through some limit type. Thus if you wanted to require

$$a = b \Rightarrow g(f(a,b)) = h(f(a,b))$$

where a and b are of type A, f is an operation of type C and g and h are operations of type D, you build the graph of your sketch with arrows $f : A \times A \to C$, $g, h : C \to D$, $e : E \to C$, $u : A \to E$ and $d : A \to A \times A$, with cones requiring $A \times A$ to be the required product, e to be the equalizer of g and h, d to be the diagonal map, and this diagram

$$
\begin{array}{ccc}
A & \xrightarrow{\ d\ } & A \times A \\
\Big\downarrow{\scriptstyle u} & & \Big\downarrow{\scriptstyle f} \\
E & \xrightarrow[\ e\ ]{} & C
\end{array}
$$

which requires $f \circ d$ to factor through the equalizer of g and h. It is built into the graph in a way analogous to the way in classical logic you build the product types implicitly by incorporating specific terms.

There is presumably a theory of sentences built in this way which makes FL sketches an institution, but it will require work to produce it: it does not fit well with the ingredients of a sketch. In the same way, building an institution out of classical logic using a definition of sentence which allows you to state that a type is a certain equalizer or pullback requires work because it does not fit the traditional way of doing things. (It is not impossible, it merely does not fit well.) The classical approach and the sketch approach make different things easy.

3.3.12 Exercises

1. In Example 4.3.6 we defined a functor $U \times U : \textbf{Mon} \to \textbf{Set}$. Show that it is induced by a homomorphism of sketches.

2. Prove that MF{2 makes $F^\times$ a natural transformation.

3. Prove Proposition ES 3.3.4.

4. Prove Proposition ES 3.3.10.

4
Fibrations

A category is both a generalized poset and a generalized monoid. Many constructions in category theory can be understood in terms of the constructions in posets that they generalize, so that it is generally good advice when learning about a new categorical idea to see what it says about posets. Seeing what a construction says about monoids has not usually been so instructive.

However, certain concepts used to study the algebraic structure of monoids generalize to categories in a natural way, and often the theorems about them remain true. In addition, applications of monoids to the theory of automata have natural generalizations to categories, and some work has been done on these generalized ideas.

In this chapter we describe some aspects of categories as generalized monoids. We begin in Section ES 4.1 with the concept of fibration, which has been used in recent research on polymorphism. One way of constructing fibrations is by the Grothendieck construction, described in Section ES 4.2, which is a generalization of the semidirect product construction for monoids. Section ES 4.3 gives an equivalence between certain types of fibrations and category-valued functors. Section ES 4.4 describes the wreath product of categories, a generalization of the concept of the same name for monoids; some applications of the construction are mentioned.

The Grothendieck construction is used in Section ES 5.7. The rest of the material is not used elsewhere in the book.

4.1 Fibrations

In this section, we describe fibrations, which are special types of functors important in category theory and which have been proposed as useful in certain aspects of computer science.

The next section gives a way of constructing fibrations from set or category-valued functors.

4.1.1 Fibrations and opfibrations Let $P : \mathscr{E} \to \mathscr{C}$ be a functor between small categories, let $f : C \to D$ be an arrow of $\mathscr{C}$, and let $P(Y) = D$. An arrow $u : X \to Y$ of $\mathscr{E}$ is **cartesian** for f and Y if

CA–1 $P(u) = f$.

CA{2 For any arrow $v : Z \to Y$ of $\mathscr{E}$ and any arrow $h : P(Z) \to C$ of $\mathscr{C}$ for which $f \circ h = P(v)$, there is a unique $w : Z \to X$ in $\mathscr{E}$ such that $u \circ w = v$ and $P(w) = h$.

Similarly, if $f : C \to D$ and $P(X) = C$, then an arrow $u : X \to Y$ is **op-cartesian** for f and X if

OA{1 $P(u) = f$.

OA{2 For any arrow $v : X \to Z$ of $\mathscr{E}$ and any arrow $k : D \to P(Z)$ for which $k \circ f = P(v)$, there is a unique $w : Y \to Z$ in $\mathscr{E}$ for which $w \circ u = v$ and $P(w) = k$.

If $P : \mathscr{E} \to \mathscr{C}$ is a functor, categorists often think of $\mathscr{E}$ as being above $\mathscr{C}$. (This is also common for functions between spaces, which is what originally suggested the ideas in this section.) For example, if $P(Y) = D$, one says that Y **lies over** D. Similar terminology is used for arrows. Thus a cartesian arrow for f must lie over f (CA{1) and one refers to CA{2 as a `unique lifting' property.

4.1.2 Definition A functor $P : \mathscr{E} \to \mathscr{C}$ is a **fibration** if there is a cartesian arrow for every $f : C \to D$ in $\mathscr{C}$ and every object Y of $\mathscr{E}$ for which $P(Y) = D$. P is an **opfibration** if there is an opcartesian arrow for every $f : C \to D$ in $\mathscr{C}$ and every object X of $\mathscr{E}$ for which $P(X) = C$. It follows that $P : \mathscr{E} \to \mathscr{C}$ is a fibration if and only if $P^{op} : \mathscr{E}^{op} \to \mathscr{C}^{op}$ is an opfibration.

If $P : \mathscr{E} \to \mathscr{C}$ is a fibration, one also says that $\mathscr{E}$ is **fibered over** $\mathscr{C}$. In that case, $\mathscr{C}$ is the **base category** and $\mathscr{E}$ is the **total category** of the fibration.

4.1.3 Definition A **cleavage** for a fibration $P : \mathscr{E} \to \mathscr{C}$ is a function $°$ that takes an arrow $f : C \to D$ and object Y such that $P(Y) = D$ to an arrow $°(f;Y)$ of $\mathscr{E}$ that is cartesian for f and Y. Similarly an opcleavage $\cdot$ takes $f : C \to D$ and X such that $P(X) = C$ to an arrow $\cdot(f;X)$ that is opcartesian for f and X.

The cleavage $°$ is a **splitting** of the fibration if it satisfies the following two requirements.

SC{1 Let D be an object of $\mathscr{C}$ and let Y be an object of $\mathscr{E}$ for which $P(Y) = D$. Then $°(\mathrm{id}_D;Y) = \mathrm{id}_Y$.

SC{2 Suppose $f : C \to D$ and $g : D \to E$ in $\mathscr{C}$ and suppose Y and Z are objects of $\mathscr{E}$ for which $P(Z) = E$ and Y is the domain of $°(g;Z)$. Then

$$°(g;Z) \circ °(f;Y) = °(g \circ f;Z)$$

Note that under the assumptions in SC{2, $P(Y) = D$, so that $°(f;Y)$ and $°(g;Z) \circ °(f;Y)$ are defined.

A fibration is **split** if it has a splitting.

Similarly, an opcleavage $\cdot$ is a splitting of an op⁻bration $P : \mathscr{E} \to \mathscr{C}$ if $\cdot(\mathrm{id}_C; X) = \mathrm{id}_X$ whenever $P(X) = C$ and

$$\cdot(g; Y) \ast \cdot(f; X) = \cdot(g \ast f; X)$$

whenever $f : C \to D$ and $g : D \to E$ in $\mathscr{C}$, $P(X) = C$ and Y is the codomain of $\cdot(f; X)$. A split op⁻bration is again one which has a splitting.

4.1.4 Example Let $\mathscr{A}$ and $\mathscr{C}$ be any categories. Then the second projection $p_2 : \mathscr{A} \times \mathscr{C} \to \mathscr{C}$ is both a split ⁻bration and a split op⁻bration. To see that it is a ⁻bration, suppose that $f : C \to C^0$ in $\mathscr{C}$ and let $Y = (A; C^0)$ be an object of $\mathscr{A} \times \mathscr{C}$. Then we can take $^{\circ}(f; Y)$ to be the arrow $(\mathrm{id}_A; f) : (A; C) \to (A; C^0)$. If $(g; h) : (A^0; C^{00}) \to (A; C^0)$ and $u : C^{00} \to C$ satisfy $f \ast u = h$ (note that $p_2(g; h) = h$), then the unique arrow from $(A^0; C^{00})$ to $(A; C^0)$ required by CA{2 is $(g; u)$.

4.1.5 Example If $\mathscr{C}$ is a category, the **arrow category** of $\mathscr{C}$ (which we have already mentioned in 4.2.17) has as objects the arrows of $\mathscr{C}$. An arrow from $f : A \to B$ to $g : C \to D$ is a pair $(h; k)$ of arrows with $h : A \to C$, $k : B \to D$ for which

$$(4.1)$$

commutes.

If $\mathscr{A}$ is the arrow category of $\mathscr{C}$, there is a functor $P : \mathscr{A} \to \mathscr{C}$ which takes $f : A \to B$ to B and $(h; k) : f \to g$ to k. If $\mathscr{C}$ has pullbacks, this functor is a ⁻bration. For a given $f : C \to D$ in $\mathscr{C}$ and object $k : B \to D$ of $\mathscr{A}$, a cartesian arrow for f and k is any $(u; f)$ given by a pullback

$$(4.2)$$

The veri⁻cation is left as an exercise (Exercise ES 4).

4.1.6 Fibers For any functor $P : \mathscr{E} \to \mathscr{C}$, the **⁻ber over** an object C of $\mathscr{C}$ is the set of objects X for which $P(X) = C$ and arrows f for which $P(f) = \mathrm{id}_C$. It is easy to verify that this ⁻ber is a subcategory of $\mathscr{E}$ (Exercise ES 1).

In the case of Example ES 4.1.4, the fibers are all the same: each one is isomorphic to the category $\mathscr{A}$. This suggests thinking of an arbitrary fibration as a type of generalized product, in which the first coordinates come in general from varying sets depending on the second coordinate. This observation can also be made concerning the relationship between a set product $S \times T$ and a general T-indexed set.

On the other hand, the fiber of the fibration in Example ES 4.1.5 over an object A of $\mathscr{C}$ is the slice category $\mathscr{C}/A$. Since an object of $\mathscr{C}/A$ can be thought of as an indexed family of objects of $\mathscr{C}$, indexed by A, this example has been referred to as $\mathscr{C}$ `fibered over itself'.

4.1.7 Cleavages induce functors   If $\mathscr{E}$ is fibered over $\mathscr{C}$, then the fibers form an indexed set of categories (indexed by the objects). Given a cleavage, the arrows of $\mathscr{C}$ induce functors between the fibers. In this way fibrations or opfibrations give a concept like that of indexed sets, in which the indexing takes into account the arrows of the underlying categories as well as the objects. Propositions ES 4.1.8 and ES 4.1.9 below spell this out.

An alternative approach to these ideas which follows the indexed set analogy more explicitly is the concept of **indexed category** (see [Johnstone and Paré, 1978], [Tarlecki, Burstall and Goguen, 1991]). Rosebrugh and Wood [1992] apply indexed categories to relational databases and Cockett and Spencer [1992] use them in studying datatypes.

In the rest of this chapter, when $F : \mathscr{C} \to$ **Cat** or $F : \mathscr{C} \to$ **Set** is a functor, we will normally write Ff for $F(f)$.

Let $P : \mathscr{E} \to \mathscr{C}$ be an opfibration with opcleavage κ. Define $F : \mathscr{C} \to$ **Cat** by

FF–1 $F(C)$ is the fiber over C for each object C of $\mathscr{C}$.

FF–2 For $f : C \to D$ in $\mathscr{C}$ and X an object of $F(C)$, $Ff(X)$ is defined to be the codomain of the arrow $\kappa(f, X)$.

FF–3 For $f : C \to D$ in $\mathscr{C}$ and $u : X \to X'$ in $F(C)$, $Ff(u)$ is the unique arrow from $Ff(X)$ to $Ff(X')$ given by OA–2 for which

$$Ff(u) \circ \kappa(f, X) = \kappa(f, X') \circ u$$

4.1.8 Proposition   Let $P : \mathscr{E} \to \mathscr{C}$ be an opfibration with cleavage κ. For any arrow $f : C \to D$ in $\mathscr{C}$, $Ff : F(C) \to F(D)$ as defined by FF–1 through FF–3 is a functor. Moreover, if κ is a splitting, then F is a functor from $\mathscr{C}$ to **Cat**.

Proof. Let $u : X \to X'$ and $v : X' \to X''$ in $F(C)$. Then

$$
\begin{aligned}
Ff(v) \circ Ff(u) \circ \kappa(f, X) &= Ff(v) \circ \kappa(f, X') \circ u \\
&= \kappa(f, X'') \circ v \circ u
\end{aligned}
\tag{4.3}
$$

by two applications of FF–3. But then by the uniqueness part of FF–3, $Ff(v) \circ Ff(u)$ must be $Ff(v \circ u)$. This proves Ff preserves composition. We leave the preservation of identities to you.

Now suppose γ is a splitting. Let $f : C \to D$ and $g : D \to E$ in $\mathscr{C}$, and let $u : X \to X'$ in $F(C)$. Then $F(g \circ f)(u)$ is the unique arrow from $F(g \circ f)(X)$ (the codomain of $\gamma(g \circ f; X)$) to $F(g \circ f)(X')$ (the codomain of $\gamma(g \circ f; X')$) for which

$$F(g \circ f)(u) \circ \gamma(g \circ f; X) = \gamma(g \circ f; X') \circ u$$

Since γ is a splitting, this says

$$F(g \circ f)(u) \circ \gamma(g; Ff(X)) \circ \gamma(f; X) = \gamma(g; Ff(X')) \circ \gamma(f; X') \circ u$$

By FF–3, the right side is

$$\gamma(g; Ff(X')) \circ Ff(u) \circ \gamma(f; X)$$

Applying FF–3 with g and $Ff(u)$ instead of f and u, this is the same as

$$Fg[Ff(u)] \circ \gamma(g; Ff(X)) \circ \gamma(f; X)$$

which is

$$Fg[Ff(u)] \circ \gamma(g \circ f; X)$$

because γ is a splitting. Using the uniqueness requirement in FF–3, this means $F(g \circ f)(u) = Fg[Ff(u)]$, so that F preserves composition. Again, we leave preservation of the identity to you. $\qquad\square$

In a similar way, split ﬁbrations give functors $\mathscr{C}^{\mathrm{op}} \to \mathbf{Cat}$. Let $P : \mathscr{E} \to \mathscr{C}$ be a ﬁbration with cleavage χ. Deﬁne $F : \mathscr{C}^{\mathrm{op}} \to \mathbf{Cat}$ by

FF'–1 $F(C)$ is the ﬁber over C for each object C of $\mathscr{C}$.

FF'–2 For $f : C \to D$ in $\mathscr{C}$ and Y an object of $F(D)$, $Ff(Y)$ is deﬁned to be the domain of the arrow $\chi(f; Y)$.

FF'–3 For $f : C \to D$ in $\mathscr{C}$ and $u : Y \to Y'$ in $F(D)$, $Ff(u)$ is the unique arrow from $Ff(Y)$ to $Ff(Y')$ given by CA–2 for which

$$\chi(f; Y') \circ Ff(u) = u \circ \chi(f; Y)$$

4.1.9 Proposition Let $P : \mathscr{E} \to \mathscr{C}$ be a ﬁbration with cleavage γ. For any arrow $f : C \to D$ in $\mathscr{C}$, $Ff : F(C) \to F(D)$ as deﬁned by FF–1 through FF–3 is a functor. Moreover, if χ is a splitting, then F is a functor from $\mathscr{C}^{\mathrm{op}}$ to $\mathbf{Cat}$.

The proof is similar to those of Proposition ES 4.1.8 and is left as an exercise.

4.1.10 Exercises

1. Verify that for any functor $P : \mathscr{E} \to \mathscr{C}$ and object C of $\mathscr{C}$, the ﬁber over an object C is a subcategory of $\mathscr{E}$.

2. Prove Proposition ES 4.1.9.

3. Let $\varphi : \mathbf{Z}_4 \to \mathbf{Z}_2$ be the homomorphism defined in Exercise 2 of Section 2.9.

a. Show that the functor from $C(\mathbf{Z}_4)$ to $C(\mathbf{Z}_2)$ induced by φ is a fibration and an opfibration. (If you know about groups, this is an instance of the fact that every surjective group homomorphism is a fibration and an opfibration.)

b. Show that φ is not a split fibration or opfibration.

4. Let $\mathscr{C}$ be a category with pullbacks and $\mathscr{A}$ its arrow category. For an arrow $f : A \to B$ (object of $\mathscr{A}$) let $P(f) = B$. For an arrow $(h, k) : f \to g$ (where $g : C \to D$ in $\mathscr{C}$) in $\mathscr{A}$, let $P(h, k) = k$.

a. Show that $P : \mathscr{A} \to \mathscr{C}$ is a functor.

b. Show that P is a fibration.

4.2 The Grothendieck construction

The Grothendieck construction is a way of producing fibrations. It generalizes the semidirect product construction for monoids, which is defined here. Hyland and Pitts [1989] use the Grothendieck construction to construct categories that are models of the calculus of constructions, a system due to Coquand and Huet [1988] that provides a way of handling polymorphism essentially by quantifying over types. (See also [Coquand, 1988], [Ehrhard, 1988] and [Asperti and Martini, 1992].)

The construction can be applied to either set-valued functors or category-valued functors. Given such a functor $F : \mathscr{C} \to \mathbf{Set}$ or $F : \mathscr{C} \to \mathbf{Cat}$, it constructs a category $\mathbf{G}(\mathscr{C}, F)$ and a functor from $\mathbf{G}(\mathscr{C}, F)$ to $\mathscr{C}$. When F is set-valued we will write $\mathbf{G}_0$ instead of $\mathbf{G}$. We will look at the set-valued case first, since it is simpler.

4.2.1 Let $\mathscr{C}$ be a small category and let $F : \mathscr{C} \to \mathbf{Set}$ be a functor. For each object C of $\mathscr{C}$, $F(C)$ is a set, and for each arrow $f : C \to D$, $F(f) : F(C) \to f(D)$ is a set function. There is no set of all sets or set of all set functions, but, since $\mathscr{C}$ is small, there certainly is a set consisting of all the elements of all the sets $F(C)$, and similarly there is a set consisting of all the functions $F(f)$. In other words, although **Set** is large, the description of $F : \mathscr{C} \to \mathbf{Set}$ requires only a small amount of data. (`Small' and `large' are used here in the technical sense, referring to whether or not a set of data is involved. See 1.3.4.)
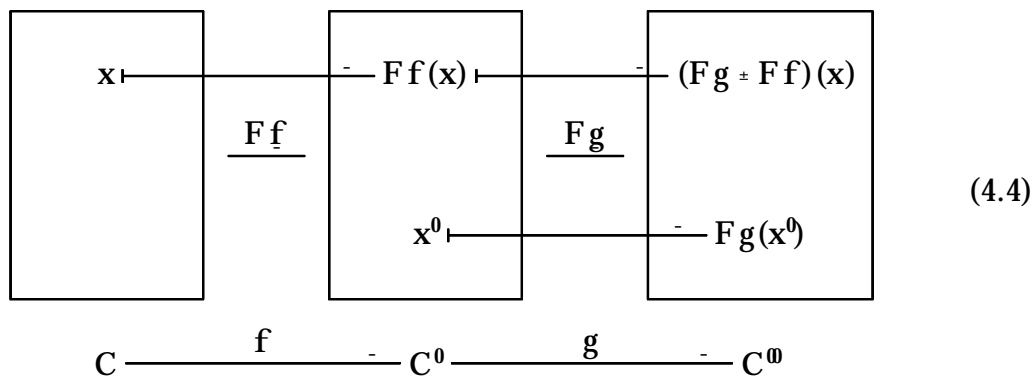
By contrast, to describe a functor $G : \mathbf{Set} \to \mathscr{C}$ would require a large amount of data – an object of $\mathscr{C}$ for each set, and so on.

We will formalize these observations about $F : \mathscr{C} \to \mathbf{Set}$ by taking the disjoint union of all the sets of the form $F(C)$ for all objects C of $\mathscr{C}$. The elements of this disjoint union can be represented as pairs (x, C) for all objects C of $\mathscr{C}$ and elements $x \in F(C)$. (Thus we construct the disjoint union of sets by labeling

the elements. The disjoint union is the construction in **Set** corresponding to the categorical concept of `sum', discussed in Section 5.4.)

We must do more than this to capture the functorial nature of F — what it does to arrows of $\mathscr{C}$. The category $\mathbf{G}_0(\mathscr{C};F)$ constructed by the Grothendieck construction does capture this structure, and its set of objects is the disjoint union just mentioned.

4.2.2 If we were to draw a picture to explain what F does, the result might be Diagram (ES 4.4), in which $f : C \to C'$ and $g : C' \to C''$ are arrows of $\mathscr{C}$ and x and x' are elements of $F(C)$ and $F(C')$ respectively. The box over each object C

$$
\begin{array}{ccccc}
\boxed{\begin{array}{c} x \mapsto \\[2em] {} \end{array}} & & \boxed{\begin{array}{c} Ff(x) \mapsto \\[1em] x' \mapsto \end{array}} & & \boxed{\begin{array}{c} (Fg \circ Ff)(x) \\[1em] Fg(x') \end{array}} \\[1em]
& \underline{Ff} & & \underline{Fg} & \\[1em]
C & \xrightarrow{\ f\ } & C' & \xrightarrow{\ g\ } & C''
\end{array}
\tag{4.4}
$$

of $\mathscr{C}$ represents the elements of $F(C)$. The arrows from x to $Ff(x)$ and from x' to $Fg(x')$ are there informally to illustrate what the set functions Ff and Fg do. These informal arrows become actual arrows in $\mathbf{G}_0(\mathscr{C};F)$.

4.2.3 Definition $\mathbf{G}_0(\mathscr{C};F)$ is the category defined as follows.

GS–1 An object of $\mathbf{G}_0(\mathscr{C};F)$ is a pair $(x;C)$ where C is an object of $\mathscr{C}$ and x is an element of $F(C)$ (as observed, the C occurs in the pair $(x;C)$ because we want the disjoint union of the values of F).

GS–2 An arrow is a pair of the form $(x;f) : (x;C) \to (x';C')$ where f is an arrow $f : C \to C'$ of $\mathscr{C}$ for which $Ff(x) = x'$.

GS–3 If $(x;f) : (x;C) \to (x';C')$ and $(x';g) : (x';C') \to (x'';C'')$, then $(x';g) \circ (x;f) : (x;C) \to (x'';C'')$ is defined by

$$(x';g) \circ (x;f) = (x;g \circ f)$$

Note in GS–3 that indeed $F(g \circ f)(x) = x''$ as required by GS–2.

The reason that we use the notation $(x;f)$ is the requirement that an arrow must determine its source and target. The source of $(x;f)$ is $(x;C)$, where C is the source of f and x is explicit, while its target is $(x';C')$, where C' is the target of f and $x' = Ff(x)$. In the literature, $(x;f)$ is often denoted simply f, so that

the same name f may refer to many different arrows — one for each element of $F(C)$. We used lacunary notation of this sort in defining slice categories in 2.6.10.

Projection on the second coordinate defines a functor

$$\mathbf{G}_0(F) : \mathbf{G}_0(\mathscr{C}; F) \to \mathscr{C}$$

$\mathbf{G}_0(\mathscr{C}; F)$ together with $\mathbf{G}_0(F)$ is called the **split discrete opfibration** induced by F, and $\mathscr{C}$ is the **base category** of the opfibration.

If C is an object of $\mathscr{C}$, the inverse image under $\mathbf{G}_0(F)$ of C is simply the set $F(C)$, although its elements are written as pairs so as to form a disjoint union.

This discrete opfibration is indeed an opfibration, in fact a split opfibration. If $f : C \to C^0$ in $\mathscr{C}$ and $(x; C)$ is an object of $\mathbf{G}_0(\mathscr{C}; F)$, then an opcartesian arrow is $(x; f) : (x; C) \to (Ff(x); C^0)$ (Exercise ES 2). The word `discrete' refers to the fact that the fibers are categories in which the only arrows are identity arrows; such categories are essentially the same as sets.

4.2.4 Semidirect products We now describe a more general version of the Grothendieck construction that has the semidirect product of monoids as a special case. We first define the semidirect product of monoids: it is constructed from two monoids, one of which acts on the other.

4.2.5 Definition If M and T are monoids, an **action** of M on T is a function $\circledR : M \times T \to T$ for which

MA–1 $\circledR(m; 1_T) = 1_T$ for all $m \in M$.

MA–2 $\circledR(m; tu) = \circledR(m; t)\circledR(m; u)$ for all $m \in M$ and $t; u \in T$.

MA–3 $\circledR(1_M; t) = t$ for all $t \in T$.

MA–4 $\circledR((mn); t) = \circledR(m; \circledR(n; t))$ for all $m; n \in M$ and $t \in T$.

If we curry $\circledR$ as in 6.1.2, we get a family of functions $\acute{A}(m) : T \to T$ with the properties listed in MA^0–1 through MA^0–4 below.

MA^0–1 $\acute{A}(m)(1_T) = 1_T$ for all $m \in M$.

MA^0–2 $\acute{A}(m)(tu) = \acute{A}(m)(t)\acute{A}(m)(u)$ for all $m \in M$ and $t; u \in T$.

MA^0–3 $\acute{A}(1_M)(t) = t$ for all $t \in T$.

MA^0–4 $\acute{A}(mn)(t) = \acute{A}(m)[\acute{A}(n)(t)]$ for all $m; n \in M$ and $t \in T$.

Thus we see that an alternative formulation of monoid action is that it is a monoid homomorphism $\acute{A} : M \to \mathbf{End}(T)$ ($\mathbf{End}(T)$ being the monoid of endomorphisms of T). MA^0–1 and MA^0–2 say that each function $\acute{A}(m)$ is an endomorphism of T, and MA^0–3 and MA^0–4 say that $\acute{A}$ is a monoid homomorphism.

4.2.6 Definition The **semidirect product** of M and T with the given action $\circledR$ as just defined is the monoid with underlying set T $\times$ M and multiplication defined by

$$(t; m)(t'; m') = (t\circledR(m; t'); mm')$$

To see the connection with the categorical version below you may wish to write this definition using the curried version of $\circledR$.

4.2.7 The categorical construction corresponding to a monoid acting on a monoid is a functor which takes values in **Cat** rather than in **Set**. A functor $F : \mathscr{C} \to$ **Cat** can be regarded as an action of $\mathscr{C}$ on a variable category which plays the role of T in the definition just given.

In the case of a monoid action defined by MA–1 through MA–4, the variable category is actually not varying: it is the category C (T) determined by the monoid T. The functor F in that case takes the single object of M to the single object of T, and, given an element $m \in M$, F (m) is the endomorphism of T which takes $t \in T$ to mt: in other words, $F(m)(t) = mt$. Thus F on the arrows is the curried form of the action $\circledR$.

> A set-valued functor is a special case of a category-valued functor, since a set can be regarded as a category with only identity arrows. Note that this is different from the monoid case: an action by a monoid on a set is not in general a special case of an action by the monoid on a monoid. It is, however, a special case of the action of a monoid on a category – a discrete category.

4.2.8 Given a functor $F : \mathscr{C} \to$ **Cat**, the Grothendieck construction in this more general setting constructs the opfibration induced by F, a category $\mathbf{G}(\mathscr{C}; F)$ defined as follows:

GC–1 An object of $\mathbf{G}(\mathscr{C}; F)$ is a pair (x; C) where C is an object of $\mathscr{C}$ and x is an object of F (C).

GC–2 An arrow $(u; f) : (x; C) \to (x'; C')$ has $f : C \to C'$ an arrow of $\mathscr{C}$ and $u : Ff(x) \to x'$ an arrow of F (C') (note that by definition Ff(x) is an object of F (C')).

GC–3 If $(u; f) : (x; C) \to (x'; C')$ and $(v; g) : (x'; C') \to (x''; C'')$, then $(v; g) \circ (u; f) : (x; C) \to (x''; C'')$ is defined by

$$(v; g) \circ (u; f) = (v \circ Fg(u); g \circ f)$$

4.2.9 Theorem Given a functor $F : \mathscr{C} \to$ **Cat**, $\mathbf{G}(\mathscr{C}; F)$ is a category and the second projection is a functor $P : \mathbf{G}(\mathscr{C}; F) \to \mathscr{C}$ which is a split opfibration with splitting

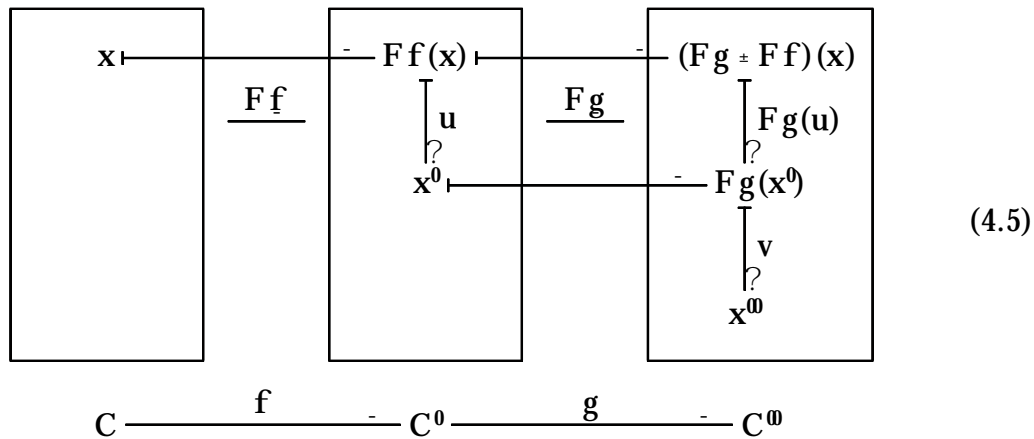$$\gamma(f; X) = (\mathrm{id}_{Ffx}; f) : (x; C) \to (Ffx; C')$$

for any arrow $f : C \to C'$ of $\mathscr{C}$ and object (x; C) of $\mathbf{G}(\mathscr{C}; F)$.

We omit the proof of this theorem. $\mathbf{G}(\mathscr{C}; F)$ is called the **crossed product** $\mathscr{C} \ltimes F$ by some authors.

It is instructive to compare this definition with the discrete opfibration constructed from a set-valued functor. In the case that F is set-valued, the first component u of an arrow $(u; f) : (x; C) \to (x'; C')$ has to be an identity arrow and it has to be $\mathrm{id}_{Ff(x)}$. Thus the only arrows are of the form $(\mathrm{id}_{Ff(x)}; f) : (x; C) \to (x'; C)$. Such an arrow is denoted $(x; f)$ in GS–1 through GS–3.

To visualize the **Cat**-valued Grothendieck construction, we can modify the picture in Diagram (ES 4.4) to get Diagram (ES 4.5). The arrows from inside one

$$
\begin{array}{ccccc}
x & \xrightarrow{\quad} & Ff(x) & \xrightarrow{\quad} & (Fg \circ Ff)(x) \\[4pt]
 & \underline{Ff} & \Big\downarrow u & \underline{Fg} & \Big\downarrow Fg(u) \\[4pt]
 & & x' & \xrightarrow{\quad} & Fg(x') \\[4pt]
 & & & & \Big\downarrow v \\[4pt]
 & & & & x''
\end{array}
\tag{4.5}
$$

$$
C \xrightarrow{\;\;f\;\;} C' \xrightarrow{\;\;g\;\;} C''
$$

box to inside another, such as the arrow from x to $Ff(x)$, are parts of arrows of $\mathbf{G}(f)$, which are now (in contrast to the discrete case) allowed to miss the target and be rescued by an internal arrow of the codomain category.

Thus in the picture above there is an arrow from x to $Ff(x)$ and $Ff(x)$ is not necessarily x'; the gap is filled by the arrow $u : Ff(x) \to x'$ of $F(C')$. The arrow $(u; f) : (x; C) \to (x'; C')$ of $\mathbf{G}(\mathscr{C}; F)$ may be pictured as the arrow from x to $Ff(x)$ followed by u. Observe that the definition of composition says that the square in the picture with corners $Ff(x)$, $(Fg \circ Ff)(x)$, x' and $Fg(x')$ 'commutes'.

As before, one writes $(x; C)$ for x and $(x'; C')$ for x' only to ensure that the union of all the categories of the form $F(C)$ is a disjoint union.

4.2.10   An analogous construction, also called the Grothendieck construction (in fact this is the original one), produces a split fibration $\mathbf{F}(\mathscr{C}; G)$ given a functor $G : \mathscr{C}^{\mathrm{op}} \to \mathbf{Cat}$.

FC–1 An object of $\mathbf{F}(\mathscr{C}; G)$ is a pair $(C; x)$ where C is an object of $\mathscr{C}$ and x is an object of $G(C)$.

FC–2 An arrow $(f; u) : (C; x) \to (C'; x')$ has $f : C \to C'$ an arrow of $\mathscr{C}$ and $u : x \to Gf(x')$ an arrow of $G(C)$.

FC{3 If $(f; u) : (C; x) \to (C'; x')$ and $(g; v) : (C'; x') \to (C''; x'')$, then $(g; v) \circ (f; u) : (C; x) \to (C''; x'')$ is de¯ned by

$$(g; v) \circ (f; u) = (g \circ f; Gf(v) \circ u)$$

4.2.11   In line with the concept that a category is a mathematical workspace, one could ask to construct objects in a suitably rich category which themselves are categories. The Grothendieck construction provides a way to describe functors from such a category object to the ambient category which is worked out in ES 5.7.2.

4.2.12 Exercises

1. Verify that GS{1 through GS{3 de¯ne a category.

2. Show that for any functor $F : \mathscr{C} \to \textbf{Set}$, $\textbf{G}_0(F) : \textbf{G}_0(\mathscr{C}; F) \to \mathscr{C}$ is a split op¯bration.

3. Verify that GC{1 through GC{3 de¯ne a category.

4. Show that for any functor $F : \mathscr{C} \to \textbf{Cat}$, $\textbf{G}(F) : \textbf{G}(\mathscr{C}; F) \to \mathscr{C}$ is a split op¯bration.

5. Verify that the de¯nition of the semidirect product in ES 4.2.6 makes $T \pounds M$ a monoid.

6. Let $F : \mathscr{C} \to \textbf{Cat}$ be a functor. Show that for each object C of $\mathscr{C}$, the arrows of the form $(u; \text{id}_C) : (x; C) \to (y; C)$ (for all arrows $u : x \to y$ of $F(C)$) (and their sources and targets) form a subcategory of the op¯bration $\textbf{G}(\mathscr{C}; F)$ which is isomorphic to $F(C)$.

4.3 An equivalence of categories

In this section, we describe how the construction of a functor from an op¯bration given in Proposition ES 4.1.8 (in one direction) produces an equivalence of categories (with the Grothendieck construction as pseudo-inverse) between a category of functors and a suitably de¯ned category of split op¯brations.

4.3.1 Cat-valued functors   For a category $\mathscr{C}$, $\textbf{Func}(\mathscr{C}; \textbf{Cat})$ is the category whose objects are functors from $\mathscr{C}$ to the category of categories, and whose arrows are natural transformations between them.

   If $F : \mathscr{C} \to \textbf{Cat}$ is such a functor and $f : C \to D$ is an arrow of $\mathscr{C}$, then $F(C)$ and $F(D)$ are categories and $Ff : F(C) \to F(D)$ is a functor. If also $G : \mathscr{C} \to \textbf{Cat}$ and $\circledR : F \to G$ is a natural transformation, then for each object of C,

$\circledR C : F(C) \to G(C)$ is a functor and the following diagram is a commutative diagram of categories and functors:

$$
\begin{array}{ccc}
F(C) & \xrightarrow{\circledR C} & G(C) \\
\Big\downarrow{\scriptstyle Ff} & & \Big\downarrow{\scriptstyle Gf} \\
F(D) & \xrightarrow[\circledR D]{} & G(D)
\end{array}
\tag{4.6}
$$

4.3.2 The category of split op¯brations of $\mathscr{C}$ Let $P : \mathscr{E} \to \mathscr{C}$ and $P^0 : \mathscr{E}^0 \to \mathscr{C}$ be two split op¯brations of the same category $\mathscr{C}$ with splittings $\cdot$ and $\cdot^0$ respectively. A **homomorphism of split op¯brations** is a functor $\Phi : \mathscr{E} \to \mathscr{E}^0$ for which

HSO{1 The diagram

$$
\begin{array}{ccc}
\mathscr{E} & \xrightarrow{\quad\Phi\quad} & \mathscr{E}^0 \\
& {\scriptstyle P}\searrow \quad \swarrow{\scriptstyle P^0} & \\
& \mathscr{C} &
\end{array}
\tag{4.7}
$$

commutes.

HSO{2 For any arrow $f : C \to D$ in $\mathscr{C}$ and object X of $\mathscr{E}$ such that $P(X) = C$,

$$\Phi(\cdot(f;X)) = \cdot^0(f;\Phi(X))$$

Thus a homomorphism of split ¯brations `takes ¯bers to ¯bers' and `preserves the splitting'.

4.3.3 De¯nition   Split op¯brations of $\mathscr{C}$ and homomorphisms between them form a category $\mathbf{SO}(\mathscr{C})$.

We will show that $\mathbf{SO}(\mathscr{C})$ is equivalent to $\mathbf{Func}(\mathscr{C}; \mathbf{Cat})$. We do this by de¯ning two functors

$$\mathbf{F} : \mathbf{SO}(\mathscr{C}) \to \mathbf{Func}(\mathscr{C}; \mathbf{Cat})$$

and

$$\mathbf{G} : \mathbf{Func}(\mathscr{C}; \mathbf{Cat}) \to \mathbf{SO}(\mathscr{C})$$

so that $\mathbf{F}$ is an equivalence with pseudo-inverse $\mathbf{G}$ as de¯ned in Section 3.4.

4.3.4 Definition
For a category $\mathscr{C}$, define the functor

$$\mathbf{F} : \mathbf{SO}(\mathscr{C}) \to \mathbf{Func}(\mathscr{C}; \mathbf{Cat})$$

as follows:

FI–1 If $P : \mathscr{E} \to \mathscr{C}$ is a split opfibration with splitting $\cdot$, then $\mathbf{F}(P; \cdot) : \mathscr{C} \to \mathbf{Cat}$ is the functor F satisfying FF–1 through FF–3 defined in ES 4.1.7.

FI–2 If $\varphi : (P; \cdot) \to (P'; \cdot')$ is a homomorphism of opfibrations, $\mathbf{F}\varphi : \mathbf{F}(P; \cdot) \to \mathbf{F}(P'; \cdot')$ is the natural transformation whose component at an object C of $\mathscr{C}$ is the functor φ restricted to $P^{-1}(C)$.

To show that $\mathbf{F}\varphi$ is a natural transformation, it is necessary to show that for every $f : C \to D$ in $\mathscr{C}$ the following diagram commutes:

$$
\begin{array}{ccc}
\mathbf{F}(P;\cdot)(C) & \xrightarrow{\ \mathbf{F}\varphi C\ } & \mathbf{F}(P';\cdot')(C) \\[4pt]
\Big\downarrow{\scriptstyle \mathbf{F}(P;\cdot)(f)} & & \Big\downarrow{\scriptstyle \mathbf{F}(P';\cdot')(f)} \\[4pt]
\mathbf{F}(P;\cdot)(D) & \xrightarrow[\ \mathbf{F}\varphi D\ ]{} & \mathbf{F}(P';\cdot')(D)
\end{array}
\tag{4.8}
$$

Let $u : X \to X'$ be in $\mathbf{F}(P; \cdot)(C)$. Note that u is an arrow in the inverse image $P^{-1}C$, so $Pu = \mathrm{id}_C$. Moreover, φu is an arrow for which $P'(\varphi u) = \mathrm{id}_C$.

By definition of cleavage, there are unique arrows $\mathbf{u}$ such that $P(\mathbf{u}) = \mathrm{id}_D$ and $\mathbf{h}$ such that $P'(\mathbf{h}) = \mathrm{id}_D$, for which

$$\mathbf{u} \circ \cdot(f; X) = \cdot(f; X') \circ u$$

and

$$\mathbf{h} \circ \cdot'(f; \varphi(X)) = \cdot(f; \varphi(X')) \circ \varphi u$$

The top route in Diagram (ES 4.8) takes u to $\mathbf{h}$ and the bottom route takes it to $\varphi\mathbf{u}$. The following calculation shows that the diagram commutes:

$$
\begin{aligned}
\mathbf{h} \circ \cdot'(f; \varphi(X))
&= \cdot'(f; \varphi(X')) \circ \varphi(u) \\
&= \varphi(\cdot\,(f; X')) \circ \varphi(u) \\
&= \varphi(\cdot\,(f; X') \circ u) \\
&= \varphi(\mathbf{u} \circ \cdot(f; X)) \\
&= \varphi(\mathbf{u}) \circ \varphi(\cdot\,(f; X)) \\
&= \varphi(\mathbf{u}) \circ \cdot'(f; \varphi(X))
\end{aligned}
\tag{4.9}
$$

so that $\mathbf{h} = \varphi(\mathbf{u})$ by the uniqueness requirement in the definition of $\mathbf{h}$.

4.3.5 The Grothendieck functor
To define the functor going the other way we extend the Grothendieck construction.

4.3.6 Definition For a category $\mathscr{C}$, define the functor

$$\mathbf{G} : \mathbf{Func}(\mathscr{C}; \mathbf{Cat}) \to \mathbf{SO}(\mathscr{C})$$

as follows:

GR–1 For $F : \mathscr{C} \to \mathbf{Cat}$, $\mathbf{G}(F) = \mathbf{G}(\mathscr{C}; F)$.

GR–2 For a natural transformation $\alpha : F \to G : \mathscr{C} \to \mathbf{Cat}$,

$$\mathbf{G}\alpha(x; C) = (\alpha Cx; C)$$

for $(x; C)$ an object of $\mathbf{G}(\mathscr{C}; F)$ (so that C is an object of $\mathscr{C}$ and x is an object of FC), and

$$\mathbf{G}\alpha(u; f) = (\alpha C'u; f)$$

for $(u; f)$ an arrow of $\mathbf{G}(\mathscr{C}; F)$ (so that $f : C \to C'$ in $\mathscr{C}$ and $u : Ffx \to x'$ in FC').

Note that in GR–2, $\alpha C'u$ has domain $\alpha C'(Ffx)$, which is $Gf(\alpha Cx)$ because α is a natural transformation. The verification that $\mathbf{G}\alpha$ is a functor is omitted.

4.3.7 Theorem The functor $\mathbf{F} : \mathbf{SO}(\mathscr{C}) \to \mathbf{Func}(\mathscr{C}; \mathbf{Cat})$ defined in ES 4.3.4 is an equivalence of categories with pseudo-inverse $\mathbf{G}$.

There is a similar equivalence of categories between split fibrations and contravariant functors. The details are in [Nico, 1983]. Moreover, the nonsplit case for both fibrations and opfibrations corresponds in a precise way to `pseudo-functors', which are like functors except that identities and composites are preserved only up to natural isomorphisms. See [Gray, 1966] (the terminology has evolved since that article).

4.3.8 Exercises

1. Verify that $\mathbf{G}\alpha$ as defined by GR–2 is a functor.

2. Verify that $\mathbf{G}$ as defined by GR–1 and GR–2 is a functor.

4.4 Wreath products

In this section, we introduce the idea of the wreath product of categories (and of functors), based on an old construction originating in group theory. In the monoid case, this construction allows a type of series-parallel decomposition of finite state machines (the Krohn–Rhodes Theorem). This section is not needed later.

4.4.1 Let $\mathscr{A}$ and $\mathscr{B}$ be small categories and $G : \mathscr{A} \to$ **Cat** a functor. With these data we define the **shape functor** $S(G; \mathscr{B}) : \mathscr{A}^{\mathrm{op}} \to$ **Cat** as follows. If A is an object of $\mathscr{A}$, then $S(G; \mathscr{B})(A)$ is the category of functors from the category $G(A)$ to $\mathscr{B}$ with natural transformations as arrows.

Thus an object of $S(G; \mathscr{B})(A)$ is a functor $P : G(A) \to \mathscr{B}$ and an arrow from $P : G(A) \to \mathscr{B}$ to $P' : G(A) \to \mathscr{B}$ is a natural transformation from P to P'. It is useful to think of $S(G; \mathscr{B})(A)$ as the category of diagrams of shape $G(A)$ (or models of $G(A)$) in $\mathscr{B}$; the arrows between them are homomorphisms of diagrams, in other words natural transformations.

> Embedding or modeling a certain shape (diagram, space, structure, etc.) into a certain workspace (category, topological space, etc.) in all possible ways is a tool used all over mathematics. In particular, what we have called the shape functor is very reminiscent of the singular simplex functors in algebraic topology.

We must say what $S(G; \mathscr{B})$ does to arrows of $\mathscr{A}^{\mathrm{op}}$. If $f : A \to A'$ is an arrow of $\mathscr{A}$, then $S(G; \mathscr{B})(f) : \mathbf{Func}(G(A'); \mathscr{B}) \to \mathbf{Func}(G(A); \mathscr{B})$ takes a functor $H : G(A') \to \mathscr{B}$ to $H \circ Gf : G(A) \to \mathscr{B}$ and it takes a natural transformation $\alpha : H \to H' : G(A') \to \mathscr{B}$ to the natural transformation $\alpha Gf : H \circ Gf \to H' \circ Gf : G(A) \to \mathscr{B}$ whose component at an object X of A is the component of α at $Gf(X)$. This is the usual action of a functor on diagrams in a category.

4.4.2 Since $S(G; \mathscr{B}) : \mathscr{A}^{\mathrm{op}} \to$ **Cat** is a category-valued functor, we can use the Grothendieck construction to form the split fibration of $\mathscr{A}$ by $S(G; \mathscr{B})$. This fibration consists of a category denoted $\mathscr{A} \, \mathrm{wr}^G \, \mathscr{B}$, called the **wreath product** of $\mathscr{A}$ by $\mathscr{B}$ with given action G, and a functor $| : \mathscr{A} \, \mathrm{wr}^G \, \mathscr{B} \to \mathscr{A}$.

We now unwind what this implies to give an elementary definition of the wreath product.

4.4.3 Definition Given small categories $\mathscr{A}$ and $\mathscr{B}$ and a functor $G : \mathscr{A} \to$ **Cat**, the wreath product $\mathscr{A} \, \mathrm{wr}^G \, \mathscr{B}$ is a category defined as follows:

WP–1 The objects of $\mathscr{A} \, \mathrm{wr}^G \, \mathscr{B}$ are pairs $(A; P)$, where A is an object of $\mathscr{A}$ and $P : G(A) \to \mathscr{B}$ is a functor.

WP–2 An arrow $(f; \gamma) : (A; P) \to (A'; P')$ of $\mathscr{A} \, \mathrm{wr}^G \, \mathscr{B}$ has $f : A \to A'$ an arrow of $\mathscr{A}$ and $\gamma : P \to P' \circ Gf$ a natural transformation.

WP–3 If $(f; \gamma) : (A; P) \to (A'; P')$ and $(g; \delta) : (A'; P') \to (A''; P'')$ are arrows of $\mathscr{A} \, \mathrm{wr}^G \, \mathscr{B}$, as in

$$
G(A) \xrightarrow{\;Gf\;} G(A') \xrightarrow{\;Gg\;} G(A'')
$$

then

$$(g; {}^1) \circ (f; {}_\lrcorner) = (g \circ f; {}^1{:}Gf \circ {}_\lrcorner) : (A; P) \to (A^{00}; P^{00})$$

To see the meaning of WP‐3, observe that ${}_\lrcorner : P \to P^0 \circ Gf$ and ${}^1 : P^0 \to P^{00} \circ Gg$ are natural transformations. Then

$$^1{:}Gf : P^0 \circ Gf \to P^{00} \circ Gg \circ Gf = P^{00} \circ G(g \circ f)$$

is the natural transformation whose component at an object x of G(A) is the component of ^{1} at $Gf(x)$ (this was described in Section 4.4). Then

$$^1{:}Gf \circ {}_\lrcorner : P \to P^{00} \circ G(g \circ f)$$

is the usual composite of natural transformations (see 4.2.11); it is the natural transformation whose component at an object x of G(A) is the composite of the components $(^1{:}Gf(x)) \circ {}_\lrcorner x$.

It follows from WP‐3 that there is a **projection functor**

$$\pi : \mathscr{A} \operatorname{wr}^G \mathscr{B} \to \mathscr{A}$$

taking $(A; P)$ to A and $(f; {}_\lrcorner)$ to f.

4.4.4 Special cases of the wreath product If the functor G in definition ES 4.4.3 is set-valued, then one obtains the **discrete wreath product** of $\mathscr{A}$ by $\mathscr{B}$ with action G. When $\mathscr{A}$ and $\mathscr{B}$ are both monoids, the discrete wreath product is also a monoid. (The general case need not be a monoid.)

4.4.5 Definition For any small category $\mathscr{C}$, the **right regular representation** of $\mathscr{C}$ is the functor $R_\mathscr{C} : \mathscr{C} \to$ **Set** defined as follows:

RR‐1 If C is an object of $\mathscr{C}$, then $R_\mathscr{C}(C)$ is the set of arrows of $\mathscr{C}$ with codomain C.

RR‐2 If $f : C \to C^0$ in $\mathscr{C}$ and $g \in R_\mathscr{C}(C)$, then $R_\mathscr{C}(f)(g) = f \circ g$.

For small categories $\mathscr{A}$ and $\mathscr{B}$, the **standard wreath product** $\mathscr{A} \operatorname{wr} \mathscr{B}$ is the wreath product $\mathscr{A} \operatorname{wr}^{R_\mathscr{A}} \mathscr{B}$. This is a generalization of what is called the standard wreath product for groups and monoids. It is the wreath product used in [Rhodes and Tilson, 1989]. They also have a two-sided version of the wreath product.

4.4.6 The action induced by a wreath product Given small categories $\mathscr{A}$ and $\mathscr{B}$ and functors $G : \mathscr{A} \to$ **Cat** and $H : \mathscr{B} \to$ **Cat**, there is an induced functor $G \operatorname{wr} H : \mathscr{A} \operatorname{wr}^G \mathscr{B} \to$ **Cat** defined as follows:

WF‐1 For an object $(A; P)$ of $\mathscr{A} \operatorname{wr}^G \mathscr{B}$, $(G \operatorname{wr} H)(A; P)$ is the split opfibration induced by $H \circ P : G(A) \to$ **Cat**.

WF{2 If $(h; \,)$ is an arrow of $\mathscr{A} \, \mathrm{wr}^G \mathscr{B}$ with domain $(A; P)$, and $(t; x)$ is an object of $(G \, \mathrm{wr} \, H)(A; P)$, so that x is an object of $G(A)$ and t is an object of $H(P(x))$, then

$$(G \, \mathrm{wr} \, H)(h; \,)(t; x) = (H \, \, x(t); Gh(x))$$

WF{3 If $(u; f) : (t; x) \, \rightarrow \, (t^0; x^0)$ is an arrow of $(G \, \mathrm{wr} \, H)(P; A)$, then

$$(G \, \mathrm{wr} \, H)(h; \,)(u; f) = (H(\, x^0)(u); Gh(f))$$

WF{1 can be perceived as saying that $G \, \mathrm{wr} \, H$ is obtained by composing the shapes given by G (see the discussion in ES 4.4.1) with H. Indeed, G. M. Kelly, who invented this concept [1974] called what we call the wreath product the `composite' of the categories. That is in some ways a better name: the word `product' suggests that the two factors are involved in the product in symmetric ways, which is not the case, as the next subsection describes.

4.4.7 The action $G \, \mathrm{wr} \, H$ of $\mathscr{A} \, \mathrm{wr}^G \mathscr{B}$ just de¯ned is said to be **triangular** because it is a precise generalization of the action of a triangular matrix. For example, the action

$$\begin{bmatrix} a & b \\ 0 & c \end{bmatrix} \text{\pounds} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cy \end{bmatrix}$$

can be described this way: the e®ect on the ¯rst coordinate depends on both the ¯rst and second coordinates, but the e®ect on the second coordinate depends only on the second coordinate.

The dependency of the action on the coordinates given in WF{2 and WF{3 is analogous to the dependency for the matrices in the example just given.

The wreath product can be generalized to many factors, using the following theorem, proved in [Kelly, 1974], Section 7. This theorem allows one to think of the wreath product as generalizing triangular matrices bigger than $2 \text{\pounds} 2$.

4.4.8 Proposition Let $G : \mathscr{A} \rightarrow \mathbf{Cat}$, $H : \mathscr{B} \rightarrow \mathbf{Cat}$ and $K : \mathscr{C} \rightarrow \mathbf{Cat}$ be functors. Then there is an isomorphism of categories I making this diagram commute.

$$\mathscr{A} \, \mathrm{wr}^G(\mathscr{B} \, \mathrm{wr}^H \mathscr{C}) \xrightarrow{\;\;I\;\;} (\mathscr{A} \, \mathrm{wr}^G \mathscr{B}) \, \mathrm{wr}^{G \, \mathrm{wr} \, H} \mathscr{C}$$

$$G \, \mathrm{wr}(H \, \mathrm{wr} \, K) \qquad \qquad (G \, \mathrm{wr} \, H) \, \mathrm{wr} \, K$$

$$\mathbf{Cat}$$

Note that the standard wreath product is not associative.

4.4.9 Applications of the wreath product   It is natural to wish to simulate complicated state transition systems using systems built up in some way from a small stock of simpler ones. This requires a precise notion of simulation. This is defined in various ways in the literature. In some cases one says a functor $F : \mathscr{C} \to \mathscr{D}$ is a simulation of $\mathscr{D}$ (the word `cover' is often used) if it has certain special properties. Other authors have the functor going the other way.

The Krohn–Rhodes Theorem for monoids says that every finite monoid action is simulated by an iterated wreath product of finite simple groups and certain very small monoids. The original Krohn–Rhodes Theorem was stated for semigroups. Discussions are in [Wells, 1976] and [Eilenberg, 1976]; the latter uses a different definition of cover. Wells [1980] proved a generalization of a weak form of the Krohn–Rhodes Theorem for finite categories and set-valued functors (which generalize the concept of action by a monoid, as discussed in Section 3.2). Wells [1988a, 1988b] describes how to use some of these decomposition techniques for category-valued functors.

Rhodes and Tilson use the idea of `division' of categories, which is related to the notion of cover, as the basic idea of an extensive study of varieties of semigroups and complexity. See See [Rhodes and Tilson, 1989, Rhodes and Weil, 1989].

Nico [1983] defines a category induced by any functor called the `kernel category' of the functor and proves a theorem which embeds the domain of the functor into the standard wreath product of the codomain and the kernel. This generalizes an old theorem of Kaloujnine and Krasner. It follows that every functor factors as a full and faithful functor which is injective on objects, followed by a fibration. Street and Walters [1973] have a related theorem.

In the case of groups, the kernel category of a homomorphism is a category equivalent, but not isomorphic, to the actual kernel of the homomorphism. In the case that $\mathscr{C}$ and $\mathscr{D}$ are monoids, the kernel category (which is not a monoid in general) is called the **derived category** of F. Rhodes and Tilson [1989] have a tighter definition obtained by imposing a congruence on the kernel. A theorem analogous to Nico's theorem is true for the tighter definition, as well. It is stated for semigroups and relations, not merely monoids and homomorphisms, so it is neither more nor less general than Nico's theorem. In the semigroup case, the `category' is replaced by a `semigroupoid', which is like a category but does not have to have identity arrows.

4.4.10 Exercises

1. Show that the discrete wreath product of two monoids is a monoid.

2. Show that the wreath product of two groups (monoids in which every element is invertible) regarded as categories is a category in which every arrow is an isomorphism.

5
Toposes

A topos is a cartesian closed category with some extra structure which produces an object of subobjects for each object. This structure makes toposes more like the category of sets than cartesian closed categories generally are.

Toposes, and certain subcategories of toposes, have proved attractive for the purpose of modeling computation. A particular reason for this is that in a topos, a subobject of an object need not have a complement. One of the fundamental facts of computation is that it may be possible to list the elements of a subset e®ectively, but not the elements of its complement (see [Lewis and Papadimitriou, 1981], Theorems 6.1.3 and 6.1.4.). Sets which cannot be listed e®ectively do not exist for computational purposes, and toposes provide a universe of objects and functions which has many nice set-like properties but which does not force complements to exist. We discuss one speci¯c subcategory of a topos, the category of modest sets, which has been of particular interest in the semantics of programming languages.

Toposes have interested mathematicians for other reasons. They are an abstraction of the concept of sheaf, which is important in pure mathematics. They allow the interpretation of second-order statements in the category in an extension of the language associated to cartesian closed categories in Chapter 6. This fact has resulted in toposes being proposed as an alternative to the category of sets for the foundations of mathematics. Toposes can also be interpreted as categories of sets with an internal system of truth values more general than the familiar two-valued system of classical logic; this allows an object in a topos to be thought of as a variable or time-dependent set, or as a set with various degrees of membership. In particular, most ways of de¯ning the category of fuzzy sets lead to a category which can be embedded in a topos.

Sections ES 5.1 and ES 5.2 describe the basic properties of toposes, for the most part without proof. Section ES 5.3 takes a closer look at an aspect of toposes which make many of them a better model of computation than, for example, **Set**.

Sections ES 5.4 and ES 5.5 describe a special case of categories of sheaves which makes the connection with sets with degrees of membership clear. The category of graphs is discussed as an example there. Section ES 5.6 describes the connection with fuzzy sets.

In Section ES 5.7 we describe category objects in a category, a notion that is needed in Section ES 5.8, which is a brief description of the realizability topos and modest sets.

56

This chapter depends on Chapters 1 through 6, Chapter 8, and Chapter 9. In addition, Section ES 5.7 needs ES 2.1.5, 11.1 and 11.2.

Sections 15.1 and 15.2 are needed in all the remaining sections. After that, the chapter consists of four independent units: Section 15.3, Sections 15.4 and 5, Section 15.6 and Sections 15.7 and 8.

Basic toposes are [Johnstone, 1977], [Barr and Wells, 1985], [Lambek and Scott, 1986], [Bell, 1988], [McLarty, 1992], [Mac Lane and Moerdijk, 1992]. None of these are aimed at applications to computer science. We do not discuss the language and logic corresponding to a topos in this book. The most accessible introduction to this is perhaps that of [McLarty, 1992], Chapter 11. Other discussions of the language and the relation with logic are in [Makkai and Reyes, 1977], [Fourman, 1977], [Fourman and Vickers, 1986], [Boileau and Joyal, 1981]. The texts [Makkai and Reyes, 1977] and [Freyd and Scedrov, 1990] discuss toposes and also more general classes of categories that have a rich logical structure. The use of toposes specifically for semantics is discussed in [Hyland, 1982], [Hyland and Pitts, 1989], [Vickers, 1992].

5.1 Definition of topos

5.1.1 The subobject functor   Recall from 2.8.11 that if C is an object of a category, a subobject of C is an equivalence class of monomorphisms $C_0 \rightarrowtail C$ where $f_0 : C_0 \rightarrowtail C$ is equivalent to $f_1 : C_1 \rightarrowtail C$ if and only if there are arrows (necessarily isomorphisms) $g : C_0 \rightarrow C_1$ and $h : C_1 \rightarrow C_0$ such that $f_1 \circ g = f_0$ and $f_0 \circ h = f_1$.

Assuming the ambient category $\mathscr{C}$ has pullbacks, the `set of subobjects' function is the object function of a functor $\mathrm{Sub} : \mathscr{C}^{\mathrm{op}} \rightarrow \mathbf{Set}$: precisely, for an object C, $\mathrm{Sub}(C)$ is the set of subobjects of C. We must define Sub on arrows.

If $k : C' \rightarrow C$ is an arrow and if $f_0 : C_0 \rightarrowtail C$ represents a subobject of C, then in a pullback

$$
\begin{array}{ccc}
C'_0 & \xrightarrow{\ k_0\ } & C_0 \\
{\scriptstyle f'_0}\Big\downarrow & & \Big\downarrow{\scriptstyle f_0} \\
C' & \xrightarrow{\ k\ } & C
\end{array}
\tag{5.1}
$$

the arrow f'_0 is also a monomorphism (see 8.3.4).

It is left as an exercise to prove, using the universal mapping property of pullbacks, that if the monomorphism $f_0 : C_0 \rightarrowtail C$ is equivalent to $f_1 : C_1 \rightarrowtail C$, then the pullbacks $f'_0 : C'_0 \rightarrowtail C'$ and $f'_1 : C'_1 \rightarrowtail C'$ are also equivalent. Thus not only is a pullback of a monomorphism a monomorphism, but also a pullback of a subobject is a subobject.

Thus we can define, for an arrow k as above,

$$\mathrm{Sub}(k) : \mathrm{Sub}(C) \to \mathrm{Sub}(C')$$

to be the function that sends the equivalence class containing f_0 to the equivalence class containing the pullback f_0'.

To show that this is a functor, we must show that the identity arrow induces the identity arrow on subobjects (exercise) and that if $k' : C'' \to C'$, then the diagram

$$
\begin{array}{ccc}
\mathrm{Sub}(C) & \xrightarrow{\;\mathrm{Sub}(k)\;} & \mathrm{Sub}(C') \\
 & & \\
\mathrm{Sub}(k \cdot k') & & \Big\downarrow \mathrm{Sub}(k') \\
 & & \\
 & & \mathrm{Sub}(C'')
\end{array}
$$

commutes. But the commutativity of this diagram at the subobject represented by f_0 is equivalent to the outer rectangle of the diagram

$$
\begin{array}{ccccc}
C_0'' & \xrightarrow{\;k_0'\;} & C_0' & \xrightarrow{\;k_0\;} & C_0 \\
f_0'' \downarrow & & f_0' \downarrow & & \downarrow f_0 \\
C'' & \xrightarrow{\;k'\;} & C' & \xrightarrow{\;k\;} & C
\end{array}
$$

being a pullback when the two smaller squares are, which is true by Exercise 10 of Section 8.3.

5.1.2 Definition   A **topos** is a category which

TOP{1 has finite limits;

TOP{2 is cartesian closed;

TOP{3 has a representable subobject functor.

We know that a functor is representable if and only if it has a universal element (see 4.5.9). A universal element of the subobject functor is an object, usually called Ω, and a subobject $\Omega_0 \subseteq \Omega$ such that for any object A and subobject $A_0 \subseteq A$, there is a unique arrow $\hat{A} : A \to \Omega$ such that there is a pullback

$$
\begin{array}{ccc}
A_0 & \longrightarrow & A \\
\downarrow & & \downarrow \hat{A} \\
\Omega_0 & \longrightarrow & \Omega
\end{array}
$$

It can be proved that Ω_0 is the terminal object and the left arrow is the unique arrow from A_0 ([Barr and Wells, 1985], Proposition 4 of Section 2.3).

The object Ω is called the **subobject classifier** and the arrow from $\Omega_0 = 1$ → Ω is usually denoted true. The arrow $\hat{A}$ corresponding to a subobject is called the **characteristic arrow** of the subobject.

The fact that the subobject functor is represented by Ω means precisely that there is a natural isomorphism

$$\phi : \mathrm{Sub}(\_\,) \to \mathrm{Hom}(\_\,; \Omega)$$

which takes a subobject to its characteristic function.

5.1.3 Example   The category of sets is a topos. It was shown in 6.1.9 that sets are a cartesian closed category. A two-element set, which we call 2, is a subobject classifier. In fact, call the two elements true and false. Given a set S and subset $S_0 \subseteq S$, define the characteristic function $\hat{A} : S \to \{\text{true}; \text{false}\}$ by

$$\hat{A}(x) = \begin{cases} \text{true} & \text{if } x \in S_0 \\ \text{false} & \text{if } x \notin S_0 \end{cases}$$

Then the following square (where the top arrow is inclusion) is a pullback:

Thus 2 is a subobject classifier.

5.1.4 Exercises

1. Referring to Diagram (ES 5.1), show that if the monomorphism $f_0 : C_0 \to C$ is equivalent to $f_1 : C_1 \to C$, then for any $g : C^0 \to C$, the pullbacks $f_0^0 : C_0^0 \to C^0$ and $f_1^0 : C_1^0 \to C^0$ are also equivalent.

2. Show that the identity arrow $C \to C$ induces the identity arrow $\mathrm{Sub}(C) \to \mathrm{Sub}(C)$.

3. Show that the category of finite sets and all functions between them is a topos.

4.[y] (Requires some knowledge of infinite cardinals.) Show that the category of finite and countably infinite sets and all functions between them has a subobject classifier but is not cartesian closed. (Hint: the set of subsets of a countable set is not countable.)

5.2 Properties of toposes

We list here some of the properties of toposes, without proof.

5.2.1 In the ﬁrst place, a topos is not only cartesian closed, it is locally cartesian closed (see Section 9.4). This is Corollary 1.43, p. 36 of [Johnstone, 1977], Corollary 7, p. 182 of [Barr and Wells, 1985] or section 17.2 of [McLarty, 1992].

5.2.2 Power objects In any topos, the object $[A \Rightarrow -]$ has the property that

$$\mathrm{Hom}(B; [A \Rightarrow -]) \cong \mathrm{Hom}(A \times B; -) \cong \mathrm{Sub}(A \times B) \tag{5.2}$$

These isomorphisms are natural when the functors are regarded as functors of either A or of B. The object $[A \Rightarrow -]$ is often called the **power object** of A and denoted $\mathscr{P}A$. It is the topos theoretic version of the powerset of a set. Theorem 1 of Section 5.4 of [Barr and Wells, 1985] implies that a category with ﬁnite limits is a topos if for each object A there is a power object that satisﬁes (ES 5.2).

The inverse image and universal image constructions in 9.2.7 for the powerset of a set can be made on $[A \Rightarrow -]$ for any object A in a topos. The left and right adjoints of the pullback functors (they exist because any topos is locally cartesian closed) are related to these images via the diagram in [Johnstone, 1977], Proposition 5.29; this diagram is called the `doctrinal diagram' and is the basis for introducing elementary (ﬁrst order) logic into a topos.

5.2.3 Eﬀective equivalence relations Let $d; e : E \rightrightarrows A$ be two arrows in a category. For any object B we have a single function

$$\langle \mathrm{Hom}(B; d); \mathrm{Hom}(B; e) \rangle : \mathrm{Hom}(B; E) \to \mathrm{Hom}(B; A) \times \mathrm{Hom}(B; A)$$

which sends f to the pair $(d \circ f; e \circ f)$. If this function is, for each object B, an isomorphism of $\mathrm{Hom}(B; E)$ with an equivalence relation on the set $\mathrm{Hom}(B; A)$, then we say that E is an **equivalence relation** on the object A. This means that the image of $\langle \mathrm{Hom}(B; d); \mathrm{Hom}(B; e) \rangle$ is actually an equivalence relation on $\mathrm{Hom}(B; A)$.

This can be thought of as embodying two separate conditions. First oﬀ, the function $\langle \mathrm{Hom}(B; d); \mathrm{Hom}(B; e) \rangle$ must be an injection, because we are supposing that it maps $\mathrm{Hom}(B; E)$ isomorphically to a subset of $\mathrm{Hom}(B; A) \times \mathrm{Hom}(B; A)$. Secondly, that subset must satisfy the usual reﬂexivity, symmetry and transitivity conditions of equivalence relations.

5.2.4 Kernel pairs Here is one case in which this condition is automatic. If $g : A \to C$ is an arrow, the pullback of the square

$$E \xrightarrow{\quad d \quad} A$$

with vertical arrows e on the left and g on the right, and bottom $A \xrightarrow{\quad g \quad} C$
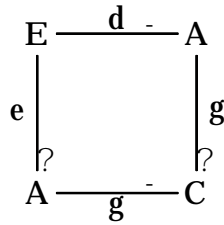
is called a **kernel pair** of g. Notation: we will write that

$$E \genfrac{}{}{0pt}{}{d}{\underset{e}{\rightrightarrows}} A \genfrac{}{}{0pt}{}{g}{\rightrightarrows} C$$

is a kernel pair. For an object B of $\mathscr{C}$, the definition of this limit is that there is a one to one correspondence between arrows $B \to E$ and pairs of arrows $(h; k)$ from B to A such that $g \circ h = g \circ k$ and that the correspondence is got by composing an arrow from B to E with d and e, resp. To put it in other words, $\mathrm{Hom}(B; E)$ is isomorphic to

$$f(h; k) \in \mathrm{Hom}(B; A) \times \mathrm{Hom}(B; A) \mid g \circ h = g \circ kg$$

which is an equivalence relation.

5.2.5 Suppose that $E \rightrightarrows A$ describes an equivalence relation. We say that the equivalence relation is **effective** if it is a kernel pair of some arrow from A. We say that a category has **effective equivalence relations** if every equivalence relation is effective. We give the following without proof. The interested reader may find the proof in [Barr and Wells, 1985] Theorem 7 of Section 2.3, [Johnstone, 1977], Proposition 1.23, p. 27, or [McLarty, 1992], Section 16.7.

5.2.6 Theorem In a topos, every equivalence relation is effective.

5.2.7 Example Equivalence relations in the categories **Set** and **Mon** are effective. An equivalence relation in **Set** is simply an equivalence relation, and the class map to the quotient set is a function that has the equivalence relation as kernel pair. An equivalence relation on a monoid M in **Mon** is a congruence relation on M (see Exercise 6 of Section 3.5); it is effective because a monoid multiplication can be defined on the quotient set of the congruence relation that makes the quotient map a homomorphism (Exercise ES 3).

There are many categories which lack effective equivalence relations. One is the category of partially ordered sets and monotone maps. Here is a simple example. Let C be a two-element chain $x < y$. Consider the subset E of $C \times C$ consisting of all four pairs $(x; x)$, $(x; y)$, $(y; x)$ and $(y; y)$. The only ordering is that $(x; x) \le (y; y)$. Then E is an equivalence relation, but is not the kernel pair of any arrow out of C. The least kernel pair that includes E has the same elements as E, but has the additional orderings $(x; x) \le (x; y) \le (y; y)$ and $(x; x) \le (y; x) \le (y; y)$.

Other important properties of toposes are contained in the following.

5.2.8 Theorem   Let $\mathscr{E}$ be a topos.

(a) $\mathscr{E}$ has finite colimits.

(b) $\mathscr{E}$ has finite disjoint and universal sums.

(c) Every epi in $\mathscr{E}$ is regular, and $\mathscr{E}$ is a regular category.

An early proof of the fact that a topos has finite colimits ([Mikkelson, 1976] mimicked the construction in sets. For example, the coequalizer of two arrows was constructed as a set of subsets, which can be identified as the set of equivalence classes mod the equivalence relation generated by the two arrows. However, the argument is difficult. The modern proof (due to Paré) is much easier, but it involves some legerdemain with triples and it is hard to see what it is actually doing. See [Barr and Wells, 1985], Corollary 2 of 5.1 for the latter proof. The rest is found in 5.5 of the same source.

5.2.9 The initial topos   There is an FL sketch whose models are toposes. (See [Barr and Wells, 1985], Section 4.4. In that book, FL sketches are called LE sketches.) It follows that there is an initial model of the topos axioms. This topos lacks a natural numbers object. It might be an interesting model for a rigidly finitistic model of computation, but would not allow the modeling of such things as recursion.

The phrase `initial topos' is usually reserved for the initial model of the axioms for toposes with a natural numbers object (Section 5.5). This category provides an interesting model for computation. The arrows from **N** to **N** in that category are, not surprisingly, the total recursive functions. In fact, all partial recursive functions are modeled in there as well, but as partial arrows, which we now describe.

5.2.10 Partial arrows   In 2.1.13 we discussed partial functions between sets. This concept can be extended to any category. Let A and B be objects. A partial arrow A to B consists of a subobject $A_0 \rightarrowtail A$ and an arrow $f : A_0 \rightarrow B$. This defines the partial arrow f in terms of a particular representative $A_0 \rightarrow A$ of the subobject, but given any other representative $A_0^0 \rightarrow A$, there is a unique arrow from A_0^0 to A_0 commuting with the inclusions which determines an arrow from A_0^0 to B by composition with f. The subobject determined by A_0 is called the **domain** of the partial arrow. If $g : A_1 \rightarrow B$ is another partial arrow on A we say the $f \leq g$ if $A_0 \rightarrowtail A_1$ and the restriction of g to A_0 is f. If we let $i : A_0 \rightarrow A_1$ denote the inclusion arrow, then the second condition means simply that $g \circ i = f$. We will say that f and g are the same partial arrow if both $f \leq g$ and $g \leq f$. This means that the domains of f and g are the same subobject of A and that f and g are equal on that domain.

We say that **partial arrows to** B **are representable** if there is an object $\tilde{B}$ and an embedding $B \rightarrow \tilde{B}$ such that there is a one to one correspondence between arrows $A \rightarrow \tilde{B}$ and partial arrows A to B, the correspondence given by pulling back:

$$A_0 \xrightarrow{\quad\quad} A$$

$$\downarrow\qquad\qquad\downarrow$$

$$B \xrightarrow{\quad\quad} \bar{B}$$
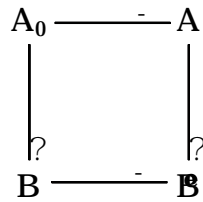
In a topos, the arrow true: $1 \to \bar{}$ represents partial functions to 1. The reason is that since each object has a unique arrow to 1, a partial arrow from A to 1 is equivalent to a subobject of A.

5.2.11 Theorem   In a topos, partial arrows into any object are representable.

See [Johnstone, 1977], Section 1.26 or [McLarty, 1992], Section 17.1 for the proof.

5.2.12 Exercises

1. Verify the isomorphisms of (ES 5.2) for **Set**.

2. Let S be a set and E be an equivalence relation on S. Then there are two arrows E to S, being the inclusion of E into $S \times S$, followed by the two projections on S. Show that E, together with these two arrows, is the kernel pair of the function that takes each element of S to the equivalence class containing it.

3. a. Let $d, e : E \to M$ be two monoid homomorphisms. Show that they form an equivalence relation in **Mon** if and only if the arrow

$$E \xrightarrow{\langle d, e \rangle} M \times M$$

is a monomorphism and the image of $\langle d, e \rangle$ is a congruence on M (see Exercise 6 of Section 3.5 with d, e the projections).

  b. Show that equivalence relations in **Mon** are e®ective.

4. Show that in any category with kernel pairs, if $f : C \to D$ is a coequalizer, then it is the coequalizer of its kernel pair.

5. Give an example in **Set** of a function with a kernel pair which is not the coequalizer of its kernel pair.

6. Show that in the category of sets, $\tilde{S}$ can be taken to be $S \cup \{\ast\}$, where $\ast$ is an element not in S.

7. Show that in a topos, the subobject classi¯er is $\tilde{1}$, the object that represents partial arrows into 1.

5.3 Is a two-element poset complete?

This discussion requires familiarity with the concept of recursive set (there is an algorithm that always halts that tells whether an element is in the set or not) and recursively enumerable set (there is an algorithm that generates all the elements of the set and no others).

   We consider a 2 element chain we denote by 2. The word `complete' in the question in the heading means that it has sups of all subsets. The question seems absurd at first, but it improves with age. In fact, we will show that both in a topos and in realistic models of computation, the answer is `no'. Moreover the answer is no in both cases for the same reason.

   The claim we want to make, and for which this discussion is evidence, is that topos semantics is an appropriate model for computation, or at least a more appropriate one than set theory. This will not mean that topos theory will tell you how to compute something that you could not compute without it. What happens is that in many toposes certain computationally meaningless constructions cannot be made at all, although they are all possible in **Set**.

5.3.1 Computation models   Of course, in a most naive sense, 2 is certainly complete as a poset, but we want to look at this in a more sophisticated way. What we really want to be true if we say that a poset P is complete is that for any other object A, the poset [A ¡! P], with the pointwise order, is complete. In the case of 2, this means that [A ¡! 2] is complete. To see why we want this to be true, recall that an element of a set is essentially the same thing as a function from a one-element set to that set. However, in categories thought of as workspaces, it is more useful to think of any arrow with codomain S as an element of S { a variable element of S. (A variable elements whose domain is a terminal object is then a constant element or global element (2.7.19)). Categories rich enough to be workspaces for a type of mathematics typically come with an **internal language**; statements in that internal language can often be perceived as statements about elements, but their truth is dependent on `elements' being interpreted as variable elements.

   In the case of ordinary set theory, it is still true that 2 is complete in this internal sense. For computational semantics, the situation is di®erent. Consider the case A = **N**, although any in¯nite set would do as well. A function **N** ¡! 2 is determined by and determines a subset of **N**. However, a computable function **N** ¡! 2 is determined by and determines a recursive subset of **N**. And it is well known that the set of recursive subsets of **N** is not complete. It is not even countably complete; in fact, the recursively enumerable subsets are characterized as the countable unions of recursive subsets. Of course, arbitrary unions of recursive subsets will be even worse.

5.3.2 Topos models   The situation in an arbitrary topos is similar. A topos has an object Ω with the property that for any object A, $\mathrm{Hom}(A, \Omega)$ is the set of subobjects of A. A topos also always has an object $2 = 1 + 1$, and an arrow $A \to 2$ does not represent an arbitrary subobject, but rather a complemented subobject (see 8.6.2). Such an arrow $A \to 2$ gives a decomposition of A as a sum $A_0 + A_1$ where A_0 and A_1 are the pullbacks shown below, where $\mathrm{true}, \mathrm{false} : 1 \to 2$ are the two injections:

$$
\begin{array}{ccc}
A_0 \longrightarrow A & \qquad & A_1 \longrightarrow A \\
\downarrow \qquad \downarrow & \qquad & \downarrow \qquad \downarrow \\
1 \xrightarrow{\ \mathrm{false}\ } 2 & \qquad & 1 \xrightarrow{\ \mathrm{true}\ } 2
\end{array}
$$
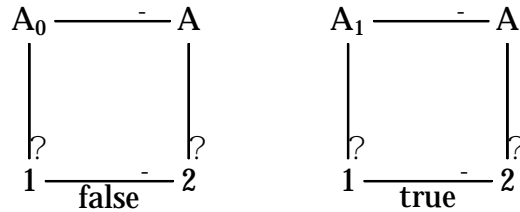
The fact that a topos has universal sums implies that from $2 = 1 + 1$ we can conclude that $A = A_0 + A_1$. On the other hand, if $A = A_0 + A_1$, there is a unique arrow $A \to 2$ whose restriction to A_0 is $\mathrm{false} \cdot \mathord{!}$ and to A_1 is $\mathrm{true} \cdot \mathord{!}$. Thus 2 is internally complete in a topos if and only if the supremum of complemented sets are complemented. It is not hard to show that this is not true in general. In fact, there is a topos in which the arrows from **N** to 2 are just the total recursive two-valued functions (and in fact, the arrows from **N** to itself are also the total recursive functions). In that topos a complemented subset of **N** is exactly a recursive subset (one which, with its complement, has a recursive characteristic function) and the fact that a union of recursive subsets is not recursive finishes the argument.

In 6.6.4 we constructed a semantics for an If loop by constructing a supremum in $[A \to D_?]$, where $D_?$ is a flat CPO (of which 2 is an example). However, the last paragraph shows that in general this semantics does not make computational sense. The reason is that, although you can write down the sup as a formal thing, it will not be guaranteed to give a terminating program. The sup of partial functions exists, but the domain of such a partial function is not generally computable.

In a topos model of computational semantics, in which all arrows $\mathbf{N} \to \mathbf{N}$ are given by recursive functions, a subobject of **N** is determined by an arrow into Ω. The maps into Ω cannot usually be computed. The one special case in which they can is that of an arrow that factors through the subobject $\langle \mathrm{true}, \mathrm{false} \rangle : 2 \to \Omega$. This corresponds to the traditional distinction between recursively enumerable and recursive subsets. In classical set theory, all subsets are classified by arrows into 2, but here only recursive subsets are.

5.4 Presheaves

5.4.1 Definition   Let $\mathscr{C}$ be a category. A functor $E : \mathscr{C}^{\mathrm{op}} \to$ **Set** is called a **presheaf** on $\mathscr{C}$. Thus a presheaf on $\mathscr{C}$ is a contravariant functor. The presheaves on $\mathscr{C}$ with natural transformations as arrows forms a category denoted **Psh**$(\mathscr{C})$.

We considered presheaves as actions in Section 3.2. They occur in other guises in the categorical and computer science literature, too. For example, a functor $F : A \to$ **Set**, where A is a set treated as a discrete category, is a `bag' of elements of A. If $a \in A$, the set $F(a)$ denotes the multiplicity to which a occurs in A. See [Taylor, 1989] for an application in computing science.

5.4.2 Proposition   The category of presheaves on a category $\mathscr{C}$ form a topos.

The proof may be found in [Barr and Wells, 1985] Section 2.1, Theorem 4. That proof uses a different, but equivalent, definition of topos.

5.4.3 Example   Consider the category we will denote by $0 \to 1$. It has two objects and four arrows, two being the identities. A contravariant set-valued functor on this category is a pair of objects G_0 and G_1 and a pair of arrows we will call source; target $: G_1 \to G_0$. The two identities are sent to identities. Thus the category of presheaves on this category is the category of graphs, which is thereby a topos. This category is the theory of the sketch for graphs given in 4.6.7.

We described the exponential object for graphs in 6.1.12. It is instructive to see what the subobject classifier is. We have to find a graph Ω and an arrow true $: 1 \to \Omega$ such that for any graph $\mathscr{G}$ and subgraph $\mathscr{G}_0 \subseteq \mathscr{G}$, there is a unique graph homomorphism $\hat{A} : \mathscr{G} \to \Omega$ such that the diagram

$$
\begin{array}{ccc}
\mathscr{G}_0 & \xrightarrow{\hspace{1.5cm}} & \mathscr{G} \\
\downarrow{?} & & \downarrow{\hat{A}} \\
1 & \xrightarrow[\text{true}]{\hspace{1.5cm}} & \Omega
\end{array}
$$

commutes.

We define the graph Ω as follows. It has five arrows we will call `all', `source', `target', `both' and `neither'. The reason for these names should become clear shortly. It has two nodes we will call `in' and `out'. The arrows `all' and `both' go from `in' to itself, `neither' goes from `out' to itself. The arrow `source' goes from `in' to `out' and `target' from `out' to `in'. The terminal graph, which has one arrow and one node, is embedded by the function true that takes the arrow to `all' and the node to `in'.

Now given a graph $\mathscr{G}$ and a subgraph $\mathscr{G}_0$ we define a function $\hat{A} : \mathscr{G} \to \Omega$ as follows. For a node n of $\mathscr{G}$, $\hat{A}(n)$ is `in' or `out', according to whether n is in $\mathscr{G}_0$ or not. For an arrow a, we let $\hat{A}(a)$ be `all' if $a \in \mathscr{G}_0$ (whence its source and

target are as well). If not, there are several possibilities. If the source but not the target of a belongs to $\mathscr{G}_0$, then $\hat{A}(a) = $ `source'. Similarly, if the target but not the source is in $\mathscr{G}_0$, it goes to `target'. If both the source and target are in it, then $\hat{A}(a) = $ `both' and if neither is, then it goes to `neither'.

5.4.4 Exercises

1. Show that the graphs No and Ar discussed in 6.1.12 are actually the contravariant set-valued functors on $0 \rightrightarrows 1$ represented by the objects 0 and 1, respectively.

2. Show that the object Ω in the category of graphs can be described as follows. The nodes are the subgraphs of No and the arrows are the subgraphs of Ar and the source and target are induced by s and t (de¯ned in 6.1.12), respectively.

3. Let $\mathscr{C}$ be a category, and let M and N be two objects of **Psh**$(\mathscr{C})$. Use the Yoneda Lemma and the adjunction de¯ning cartesian closed categories to show that for any object X of $\mathscr{C}$, $[M \to N](X)$ must be the set of natural transformations from $\text{Hom}(-;X) \times M$ to N, up to isomorphism. Note that this does not prove that **Psh**$(\mathscr{C})$ is cartesian closed. That is true, but requires ideas not given here (see [Mac Lane and Moerdijk, 1992], page 46).

4. Let $\mathscr{C}$ be a category and X an object of $\mathscr{C}$. Let Ω be the subobject classi¯er of **Psh**$(\mathscr{C})$. Show that $\Omega(X)$ is, up to isomorphism, the set of subfunctors (see Exercise 3 of Section 4.3) of $\text{Hom}(-;X)$.

5.5 Sheaves

The general de¯nition of sheaves requires a structure on the category called a **Grothendieck topology**. The most accessible and detailed discussion of Grothendieck topologies is that of [Mac Lane and Moerdijk, 1992]. Here we will discuss the special case of sheaves in which the category is a partial order.

5.5.1 Let P be a partially ordered set. From the preceding section, a presheaf E on P assigns to each element $x \in P$ a set $E(x)$ and whenever $x \leq y$ assigns a function we will denote $E(x;y) : E(y) \to E(x)$ (note the order; x precedes y, but the arrow is from $E(y)$ to $E(x)$). This is subject to two conditions. First, that $E(x;x)$ be the identity function on $E(x)$ and second that when $x \leq y \leq z$, that $E(x;y) \circ E(y;z) = E(x;z)$. The arrows $E(x;y)$ are called **restriction functions**.

5.5.2 Heyting algebras   We make the following supposition about P.

HA{1 There is a top element, denoted 1, in P.

HA{2 Each pair of elements $x; y \in P$ has an in¯mum, denoted $x \wedge y$.

HA{3 Every subset $\{x_i\}$ of elements of P has a supremum, denoted $\bigvee x_i$.

HA$\neg$4 For every element $x \in P$ and every subset $\{x_i\} \subseteq P$, $x \wedge (\bigvee x_i) = \bigvee (x \wedge x_i)$.

A poset that satisfies these conditions is called a **complete Heyting algebra**.

5.5.3 If $\{x_i\}$ is a subset with supremum x, and E is a presheaf, there is given a restriction function $e_i : E(x) \to E(x_i)$ for each i. The universal property of product gives a unique function $e : E(x) \to \prod_i E(x_i)$ such that $p_i \cdot e = e_i$. In addition, for each pair of indices i and j, there are functions $c_{ij} : E(x_i) \to E(x_i \wedge x_j)$ and $d_{ij} : E(x_j) \to E(x_i \wedge x_j)$ induced by the relations $x_i \geq x_i \wedge x_j$ and $x_j \geq x_i \wedge x_j$. This gives two functions $c, d : \prod_i E(x_i) \to \prod_{ij} E(x_i \wedge x_j)$ such that

$$
\begin{array}{ccc}
\prod_i E(x_i) & \xrightarrow{\ c\ } & \prod_{ij} E(x_i \wedge x_j) \\
\Big\downarrow{p_i} & & \Big\downarrow{p_{ij}} \\
E(x_i) & \xrightarrow[c_{ij}]{} & E(x_i \wedge x_j)
\end{array}
$$

and

$$
\begin{array}{ccc}
\prod_i E(x_i) & \xrightarrow{\ d\ } & \prod_{ij} E(x_i \wedge x_j) \\
\Big\downarrow{p_i} & & \Big\downarrow{p_{ij}} \\
E(x_i) & \xrightarrow[d_{ij}]{} & E(x_i \wedge x_j)
\end{array}
$$

commute.

5.5.4 Definition A presheaf is called a **sheaf** if it satisfies the following additional condition:

$$x = \bigvee x_i$$

implies

$$E(x) \xrightarrow{\ e\ } \prod_i E(x_i) \underset{d}{\overset{c}{\rightrightarrows}} \prod_{ij} E(x_i \wedge x_j)$$

is an equalizer.

5.5.5 Theorem The category of sheaves on a Heyting algebra is a topos.

As a matter of fact, the category of sheaves for any Grothendieck topology is a topos (see any of the texts [Johnstone, 1977], [Barr and Wells, 1985], [Mac Lane and Moerdijk, 1992], [McLarty, 1992]).

5.5.6 Constant sheaves A presheaf E is called **constant** if for all $x \in P$, $E(x)$ is always the same set and for all $x \leq y$, the function $E(y, x)$ is the identity function on that set.

The constant presheaf at a one-element set is always a sheaf. This is because the sheaf condition comes down to a diagram

$$1 \longrightarrow 1 \rightrightarrows 1$$

which is certainly an equalizer. No constant presheaf whose value is a set with other than one element can be a sheaf. In fact, the 0 (bottom) element of P is the supremum of the empty set and the product of the empty set of sets is a one-element set (see 5.3.6). Hence the sheaf condition on a presheaf E is that

$$E(0) \longrightarrow \prod_{\emptyset} \rightrightarrows \prod_{\emptyset}$$

which is

$$E(0) \longrightarrow 1 \rightrightarrows 1$$

and this is an equalizer if and only if $E(0) = 1$.

5.5.7 A presheaf is said to be **nearly constant** if whenever $0 < x \leq y$ in P, the restriction $E(y) \longrightarrow E(x)$ is an isomorphism. It is interesting to inquire when a nearly constant presheaf is a sheaf. It turns out that every nearly constant presheaf over P is a sheaf over P if and only if the meet of two nonzero elements of P is nonzero.

To see this, suppose E is a nearly constant presheaf whose value at any $x \neq 0$ is S and that $x = \bigvee x_i$. In the diagram

$$E(x) \longrightarrow \prod E(x_i) \rightrightarrows \prod E(x_i \wedge x_j)$$

every term in which $x_i = 0$ contributes nothing to the product since $1 \times Y \cong Y$. An element of the product is a string $\{s_i\}$ such that $s_i \in S$. The condition of being an element of the equalizer is the condition that the image of s_i under the induced function $E(x_i) \longrightarrow E(x_i \wedge x_j)$ is the same as the image of s_j under $E(x_j) \longrightarrow E(x_i \wedge x_j)$. But in a nearly constant sheaf, all these sets are the same and all the functions are the identity, so this condition is simply that $s_i = s_j$. But this means that an element of the equalizer must be the same in every coordinate, hence that diagram is an equalizer.

5.5.8 Interpretation of sheaves Let E be a sheaf on P. The reader will want to know how E is to be interpreted. Start by thinking of P as an algebra of truth values. Whereas propositions (assertions) in ordinary logic are either true or false (so that ordinary logic is often called 2-valued), in P-valued logic, the truth of an assertion is an element of P. In the next section, we will use this idea to discuss

time sheaves in which propositions can be true at some times and not at others. If p is some proposition, let us write $[\![p]\!]$ to denote the element of P that is the truth value.

A sheaf E is a set in this logic. For $x \in P$, the (ordinary) set $E(x)$ could, as a first approximation, be thought of as the set of all entities a for which $[\![a \in E]\!]$ is at least x. If $y < x$, then $[\![a \in E]\!] \geq x$ implies that $[\![a \in E]\!] \geq y$ so that $E(x) \subseteq E(y)$. This is only a first approximation and what we have described is actually a P-valued fuzzy set (see ES 5.6). The reason is that equality is also a predicate and it may happen, for example, that $[\![a = b]\!]$ could lie between x and y so that the entities a and b are not discernably equal at level x, but are equal at level y. The result is that rather than an inclusion, we have a restriction function $E(x) \longrightarrow E(y)$ for $y \leq x$.

5.5.9 Time sheaves, I   Here is a good example of a topos in which one can see that the restriction arrows should not be expected to be injective. Consider the partially ordered set whose elements are intervals on the real line with inclusion as ordering. It is helpful to think of these as time intervals.

Now consider any definition of a naive set. Some possible definitions will be time invariant, such as the set of mathematical statements about, say, natural numbers, that are true. Others of these `sets' change with time; one example might be the set of all books that are currently in print; another the set of statements currently known to be true about the natural numbers. These may conveniently be thought of as the presheaves whose value on some time interval is the set of books that were in print over the entire interval and the set of statements about natural numbers known to be true during that entire interval. The restriction to a subinterval is simply the inclusion of the set of books in print over the whole interval to that (larger) set of those in print over that subinterval or the restriction of the knowledge from the interval to the subinterval. In this example, the restrictions are injective.

Instead of books in print, we could take the example of businesses in operation. Because of the possibility of merger, divestment and the like, two businesses which are actually distinct over a large interval might coincide over a smaller subinterval. Thus for this example of a \set", the restriction function is not injective in general.

Another situation in which the restriction functions are not necessarily injective arises from the set of variables in a block-structured programming language. The presheaf is this: the set for a certain time interval during the running of the program is the quotient of the set of variables which exist over the whole time interval, modulo the equivalence relation that identifies two variables in an interval if they should happen to have the same value over the whole interval. Two variables may not be equivalent over a large interval, whereas they may be equivalent over a smaller one; in that case the restriction function would not be injective.

In general, any property describes the set of all entities that have that property over the entire interval. The restriction to a subinterval arises out of the observation that any entity that possesses a property over an interval possesses it over any subinterval.

The sheaf condition in this case reduces to this: if the interval I is a union of subintervals I_k (where k ranges over an index set which need not be countable) and an entity possesses that property over every one of the subintervals I_k, then it does over the whole interval. This condition puts a definite restriction on the kinds of properties that can be used. For example, the property of being **grue**, that is blue over the first half of the interval and green on the second half, is not allowed. The properties that are allowed are what might be called **local**, that is true when they are true in each sufficiently small interval surrounding the point in time. This statement is essentially a tautology, but it does give an idea of what the sheaf condition means.

5.5.10 Time sheaves, II   Here is another topos, rather different from the one above, that might also be considered to be time sheaves. Unlike the one above which is symmetric to forward and reverse time, this one definitely depends on which way time is flowing. This is not to say that one is better or worse, but they are different and have different purposes. In this one, the elements of the Heyting algebra are the times themselves. In other words, we are looking at time indexed sets, as opposed to sets indexed by time intervals.

We order this set by the reverse of the usual order. So a presheaf in this model is a family $\{X(t)\}$ of sets, t a real number, together with functions $f(s;t) : X(t) \to X(s)$ for $t \leq s$, subject to the condition that $f(t;t)$ is the identity and $f(r;s) \circ f(s;t) = f(r;t)$ for $t \leq s \leq r$. The sheaf condition of Definition ES 5.5.4 is a bit technical, but can easily be understood in the special case of a presheaf all of whose transition arrows $f(s;t)$ are inclusions. In that case, the condition is that when $t = \bigvee t_i$ (so that t is the greatest lower bound of the t_i), then $X(t) = \bigwedge X(t_i)$.

An example, which might be thought typical, of such a sheaf might be described as the `sheaf of states of knowledge'. At each time t we let $X(t)$ denote the set of all things known to the human race. On the hypothesis (which might be disputed) that knowledge is not lost, we have a function $X(t) \to X(s)$ for $t \leq s$. In common parlance, we might consider this function to be an inclusion, or at least injective, but it is possible to modify it in various ways that will render it not so. We might introduce an equivalence relation on knowledge, so that two bits of knowledge might be considered the same. In that case, if at some future time we discover two bits of knowledge the same, then bits not equal at one time become equal at a later time and the transition arrow is not injective.

For example, consider our knowledge of the set of complex numbers. There was a time in our history when all the numbers e, i, π and -1 were all known, but it was not known that $e^{i\pi} = -1$. In that case, the number $e^{i\pi}$ and -1 were

known separately, but not the fact that they were equal. See [Barr, McLarty and Wells, 1985]. The sheaf condition is this: if $\{t_i\}$ is a set of times and t is their infimum, then anything known at time t_i for every i is known at time t.

5.5.11 Exercise

1. Let $\mathcal{H}$ be a complete Heyting algebra. Define the binary operation $\Rightarrow : H \times H \to H$ by requiring that $a \Rightarrow b$ is the join of all elements c for which $a \wedge c \leq b$.

   a. Prove that $a \wedge c \leq b$ if and only if $c \leq a \Rightarrow b$.

   b. Prove that when $\mathcal{H}$ is regarded as a category in the usual way, it is cartesian closed with $\Rightarrow$ as internal hom.

5.6 Fuzzy sets

Fuzzy set theory is a more-or-less categorical idea that some claim has application to computer modeling. It appears to be closely related to topos theory. In fact, it appears to us that the interesting core of the subject is already implicit in topos theory. We will not go deeply into the subject, but only give a few definitions and point out the connections. More details can be found in [Barr, 1986a], [Pitts, 1982].

5.6.1   At this point, it would be appropriate to give some definitions. One of the problems with fuzzy sets is that the meaning of the term has been left vague (one might say fuzzy). Rather than attempt to give all possible definitions, we content ourselves with a definition that is common and for which the connection with topos theory is as simple as possible.

5.6.2 Definition   Let P be a complete Heyting algebra. A P-**valued set** is a pair (S, σ) consisting of a set S and a function $\sigma : S \to P$. A **category of fuzzy sets** is the category of P-valued sets (in other words, the slice category $\mathbf{Set}/P$) for a fixed complete Heyting algebra P.

   In practice, fuzzy sets are defined with P being the closed interval of real numbers from 0 to 1, which is a complete Heyting algebra with the usual ordering. Think of $\sigma(s)$ as being the degree of membership of s in the fuzzy set. If $\sigma(s) = 1$, then s is fully in the fuzzy set, while if $\sigma(s) = 0$, then s is not in the fuzzy set at all.

   Actually that last statement is not quite true; we will return to this point later; pretend for the moment that it is.

5.6.3   Let (S, σ) and (T, τ) be fuzzy sets. An **arrow** $f : (S, \sigma) \to (T, \tau)$ is an arrow $f : S \to T$ such that $\sigma \leq \tau \circ f$. Thus the degree of membership of s in (S, σ) cannot exceed that of $f(s)$ in (T, τ). With this definition and the obvious identity arrows, the fuzzy sets based on P form a category $\mathbf{Fuzz}(P)$.

The hypothesis actually made on P was that both P and the opposite order P^{op} were Heyting algebras ([Goguen, 1974]). The hypothesis on P^{op} plays no role in the theory and so we have omitted it.

5.6.4 Once we have de¯ned the category of fuzzy sets, the de¯nition of subset of a fuzzy set emerges. For $f : (S; \frac{3}{4}) \to (T; \dot{c})$ to be monic it is necessary that f be injective. In particular, we can think of a subset of $(T; \dot{c})$ as being a fuzzy set $(T_0; \dot{c}_0)$ where $T_0 \subseteq T$ and $\dot{c}jT_0 \cdot \dot{c}_0$.

5.6.5 More ado about nothing Consider the following two fuzzy subsets of $(S; \frac{3}{4})$. The ¯rst is the set $(;; \hbar i)$ and the second is the set $(S; 0)$ where $\hbar i$ is the unique function of $;$ to P and 0 stands for the function that is constantly zero. One is the empty set and the other is the set in which every element is not there. There is seemingly no di®erence between these two sets as neither actually contains any elements. In fact, in fuzzy set theory, these two sets (and sets in between) are not considered to be equal. This results in the class of fuzzy set theories being curiously restricted (see ES 5.6.10).

5.6.6 Fuzzy sets and sheaves The reader may suspect (from the title of this section, if nothing else) that there is a connection between fuzzy sets and toposes. Both are generalizations of set theory to introduce lattices more general than the two-element lattice as truth values.

One of the two di®erences has just been mentioned; the di®erent treatment of the null set. Actually, this di®erence is relatively minor. The second one is not. Suppose $(S; \frac{3}{4})$ is a fuzzy set. We can de¯ne a presheaf E by letting

$$E(x) = fs \, 2 \, S \, j \, \frac{3}{4}(s) \, , \, xg$$

as suggested in our informal discussion. Clearly, if $y \cdot x$, then $E(x) \subseteq E(y)$ and using these inclusions, we get a presheaf on P. It is almost never a sheaf, however. The essential reason for this is that $E(0) = S$, while we have seen in ES 5.5.6 that $E(0) = 1$ when E is a sheaf.

5.6.7 It turns out there is a very simple way to make E into a sheaf, but not on P. Let P^+ denote the poset constructed from P by adding a new bottom element. Let us call the new bottom element ? to distinguish it from the old one we called 0. Now given a P indexed fuzzy set, de¯ne a presheaf on P^+ by letting $E(x)$ be de¯ned as above for $x \, 2 \, P$ and $E(?) = 1$.

5.6.8 Proposition The presheaf E just de¯ned is a sheaf. It is a subsheaf of the near constant sheaf C de¯ned by $C(x) = S$ for $x \neq ?$ and $C(?) = 1$.

Proof. We ¯rst observe that P^+ obviously has the property that the meet of two nonzero elements is nonzero because P has ¯nite meets. Thus C is a sheaf. A diagram chase shows that if C is a sheaf and E a subpresheaf, then E is a sheaf if and only if for each $x = \bigvee x_i$, the diagram

$$E(x) \xrightarrow{\quad\cong\quad} E(x_i)$$
$$\downarrow \qquad\qquad\qquad \downarrow$$
$$C(x) \xrightarrow{\quad\cong\quad} C(x_i)$$

is a pullback. The vertical arrows are just the inclusions. As we saw in ES 5.5.7, the lower horizontal function is just the inclusion of $C(x)$ into the set of constant strings. It follows that this is essentially what the upper horizontal arrow is. Now in order that a string of elements $\{s_i\} \in \prod E(x_i)$ be constant, it is necessary and sufficient that all the s_i be the same element s and that $s \in E(x_i)$ for all i which means that $\mu(s) \geq x_i$ for all i. But this is just what is required to have $\mu(s) \geq x$ and $s \in E(x)$. $\qquad\square$

Continuing in this vein, it is possible to show the following.

5.6.9 Theorem   For any Heyting algebra P, the category of fuzzy sets based on P is equivalent to the full subcategory of the category of P^+ sheaves consisting of the sheaves that are subsheaves of the near constant sheaves.

5.6.10   The introduction of P^+ instead of P is directly traceable to the failure the two kinds of empty sets as mentioned in ES 5.6.5 to be the same. The fact that the sheaves are subsheaves of the near constant sheaves is really a reflection of the fact that in fuzzy set theory only one of the two predicates of set theory is made to take values in P (or P^+).

This shows up in the fact that in fuzzy set theory there is no fuzzy set of fuzzy subsets of a fuzzy set. In other words, the $\mathscr{P}$ construction is missing. Here's why. Suppose S is a set, considered as a fuzzy set with $\mu(s) = 1$ for all $s \in S$. (Such a fuzzy set is called a **crisp** set.) Let $x < y$ be two elements of P and consider the subsets $S_x = (S; \mu_x)$ and $S_y = (S; \mu_y)$, with μ_x and μ_y being the functions which are constant at x and y respectively. Then of course, $S_x \subseteq S_y$ (actually S_x is a proper subset of S_y), but it is clear that when looking only at degrees of membership at level x or below, the two subsets are equal. In fact, in the topos, the degree to which S_x equals S_y is just x. But this predicate cannot be stated in the language of fuzzy sets and the result is that there are not and cannot be power objects (unless P has just one element).

The point is that there are two predicates in set theory, membership and equality. In topos theory, both may be fuzzy, but in fuzzy set theory, only membership is allowed to be. But $\mathscr{P}$ converts membership into equality as explained in the preceding paragraph and so cannot be defined in fuzzy set theory. Thus fuzzy set theory, as currently implemented, lacks a certain conceptual consistency.

One can try to refine the definition of fuzzy set so as to allow fuzzy equality. The obvious way to proceed is to define as objects triplets $(S; \mu, \sim)$, with $(S; \mu)$ as above

and $\sigma : S \times S \to P$, interpreted as fuzzy equality. These must be subject to the condition that the degree to which two elements are equal cannot exceed the degree to which either one is defined. The resultant category is equivalent to the topos of sheaves on P^+.

5.7 External functors

5.7.1 Category objects in a topos One of the tools proposed for programming language semantics is the category of modest sets, which we will describe in the next section. The category of modest sets is not a category in the sense we have been using the word up until now: it is a category object in another category, called the effective topos.

Recall the sketch for categories that was described in detail in ES 2.1.5. A model of this sketch in the category of sets is, of course, a category. A model in a category $\mathscr{C}$ is called a **category object** in $\mathscr{C}$. A homomorphism between such category objects is called an **internal functor** between those category objects.

Referring to the sketch, we see that a category object consists of four objects C_0; C_1; C_2 and C_3 such that $C_2 \cong C_1 \times_{C_0} C_1$ and $C_3 \cong C_1 \times_{C_0} C_1 \times_{C_0} C_1$. There are arrows in $\mathscr{C}$ corresponding to unit, source, target and composition. The crucial commutative diagrams are:

We may denote such a category object by $\mathbf{C} = \langle C_0; C_1; u_{\mathbf{C}}; s_{\mathbf{C}}; t_{\mathbf{C}}; c_{\mathbf{C}} \rangle$ since the remaining data are determined by these. As a matter of convenience, we usually omit the indexes unless they are necessary for comprehension. If $\mathbf{D} = \langle D_0; D_1; u; s; t; c \rangle$ is another category object, then an internal functor $f : \mathbf{D} \to \mathbf{C}$ is given by homomorphisms $f_0 : D_0 \to C_0$ and $f_1 : D_1 \to C_1$ such that the following diagrams commute, where $f_2 : D_2 \to C_2$ is the unique arrow for which $p_i \circ f_2 = f_1 \circ p_i$; $i = 1; 2$.

(5.3)

5.7.2 External functors The notion of a functor from a category into the category of sets can be extended to describe a functor from a category in $\mathcal{E}$ to $\mathcal{E}$ itself. Such a functor is called an **external functor**. This construction is based on the construction in Section ES 4.2 for **Set**. Theorem ES 4.3.7 gives an equivalence of categories between external functors and the split opfibrations which are produced by the Grothendieck construction, and so justifies the representation by split opfibrations of external functors defined on a category object.

Essentially, what we will do is use objects and arrows representing sets and functions involved in the Grothendieck construction, find enough cones and commutative diagrams to characterize it, and take that as the definition of external functor in an arbitrary category.

5.7.3 Let $\mathcal{E}$ be a category, and let

$$\mathbf{C} = (C_0; C_1; \text{source}; \text{target}; \text{unit}; \text{comp})$$

be a category object in $\mathcal{E}$. An external functor $\mathbf{C} \to \mathcal{E}$ consists of data

$$(D_0; D_1; D_2; d^0; d^1; u; \pi_0; \pi_1; p_1; p_2; c)$$

for which the D_i are objects of $\mathcal{E}$ and the arrows have sources and targets as indicated:

$$
\begin{aligned}
d^i &: D_1 \to D_0; \quad i = 1; 2 \\
u &: D_0 \to D_1 \\
\pi_i &: D_i \to C_i; \quad i = 0; 1 \\
p_i &: D_2 \to D_1; \quad i = 1; 2 \\
c &: D_2 \to D_1:
\end{aligned}
$$

D_0 is the object corresponding to the disjoint union of the values of the functor. Note that we are not given a functor F as we were in Section ES 4.2 { we are being guided by Theorem ES 4.3.7 and the details of the Grothendieck construction to define an external functor, and π_0 represents the projection (taking $(x; C)$ to C in the case of the Grothendieck construction). D_1 is the object corresponding to the arrows of the category $\mathbf{G}(C; F)$ as defined in Section ES 4.2, and π_1 takes

$(x; f)$ to f. Thus γ_0 and γ_1 together describe the functor $\mathbf{G}(F)$ in the case of the Grothendieck construction.

d^0 and d^1 are the source and target maps of that category, c is the composition and u picks out the identities.

The data are subject to the requirements E{1 through E{4 below.

E{1 All three diagrams below must commute and (a) must be a pullback:

$$
\begin{array}{ccc}
\begin{array}{ccc}
D_1 & \xrightarrow{\ \gamma_1\ } & C_1 \\
\Big\downarrow{\scriptstyle d^0} & & \Big\downarrow{\scriptstyle \text{source}} \\
D_0 & \xrightarrow[\ \gamma_0\ ]{} & C_0 \\
(a)
\end{array}
&
\begin{array}{ccc}
D_1 & \xrightarrow{\ \gamma_1\ } & C_1 \\
\Big\downarrow{\scriptstyle d^1} & & \Big\downarrow{\scriptstyle \text{target}} \\
D_0 & \xrightarrow[\ \gamma_0\ ]{} & C_0 \\
(b)
\end{array}
&
\begin{array}{ccc}
D_1 & \xrightarrow{\ \gamma_0\ } & C_1 \\
\Big\uparrow{\scriptstyle u} & & \Big\uparrow{\scriptstyle \text{unit}} \\
D_0 & \xrightarrow[\ \gamma_1\ ]{} & C_0 \\
(c)
\end{array}
\end{array}
\qquad (5.4)
$$

That (a) is a pullback says in the case of the Grothendieck construction that, up to unique isomorphism, D_1 consists of elements of the form $(x; f)$ with f an arrow of $\mathscr{E}$ and $x \in F(C)$ where C is the source of f. This is part of GS{2 (see Section ES 4.2). The commutation of (b) says that $d_1(x; f) \in F(C^0)$ where C^0 is the target of F; that follows from GS{1 and GS{2 (there, x^0 must be in $F(C^0)$). That of (c) says that $u(x; C)$ must be (id_C). In the case of the Grothendieck construction that follows from the fact that F is given as a functor.

E{2 The following diagram is a pullback:

$$
\begin{array}{ccc}
D_2 & \xrightarrow{\ p_2\ } & D_1 \\
\Big\downarrow{\scriptstyle p_1} & & \Big\downarrow{\scriptstyle d^1} \\
D_1 & \xrightarrow[\ d^0\ ]{} & D_0 \\
& (d) &
\end{array}
$$

In the case of the Grothendieck construction this forces D_2 to be the set of composable pairs of arrows of $\mathbf{G}(C; F)$.

E{3 The following diagram must commute:

$$
\begin{array}{ccc}
D_1 & \xrightarrow{\;\phi_1\;} & C_1 \\
p_1 \downarrow & & \downarrow p_1 \\
D_2 & \xrightarrow{\;\phi_2\;} & C_2 \\
p_2 \downarrow & & \downarrow p_2 \\
D_1 & \xrightarrow{\;\phi_1\;} & C_1
\end{array}
$$

(e)

ϕ_2 is de¯ned by Diagram (ES 5.3) (called f_2 there). In the case of the Grothendieck construction this follows from GS{3: $(x; g^0)$ composes with $(x; f)$ only if g composes with f (but not conversely!).

E{4 The following diagrams must commute.

$$
\begin{array}{ccc}
D_2 \xrightarrow{\;\phi_2\;} C_2 \\
c \downarrow \quad\quad \downarrow \text{comp} \\
D_1 \xrightarrow{\;\phi_1\;} C_1
\end{array}
\qquad
\begin{array}{ccc}
D_2 \xrightarrow{\;p_1\;} D_1 \\
c \downarrow \quad\quad \downarrow d^0 \\
D_1 \xrightarrow{\;d^0\;} D_0
\end{array}
\qquad
\begin{array}{ccc}
D_2 \xrightarrow{\;p_2\;} D_1 \\
c \downarrow \quad\quad \downarrow d^1 \\
D_1 \xrightarrow{\;d^1\;} D_0
\end{array}
\qquad (5.5)
$$

(f)   (g)   (h)

In the case of the Grothendieck construction, (f) says that ϕ_1 preserves composition and (g) and (h) say that the composite has the correct source and target.

As we have presented these diagrams, we have observed that they are all true in the case of the Grothendieck construction in **Set**. In the case of the Grothendieck construction,

$$(D_0; D_1; \text{source}; \text{target}; \text{unit}; \text{comp})$$

is actually a category, hence a category object in **Set**. Moreover, $\phi = (\phi_0; \phi_1)$ is a functor, namely $\mathbf{G}(F)$. That is actually true in any category, as seen in the following.

5.7.4 Proposition   Let

$$(D_0; D_1; D_2; d^0; d^1; u; \phi_0; \phi_1; p_1; p_2; c)$$

be an external functor to a category object

$$\mathscr{C} = (C_0; C_1; \text{source}; \text{target}; \text{unit}; \text{comp})$$

in a category $\mathscr{E}$. Then

$$(D_0; D_1; \text{source}; \text{target}; \text{unit}; \text{comp})$$

is a category object in $\mathscr{E}$ and $(¼_0; ¼_1)$ is a functor in $\mathscr{E}$.

The proof is a lengthy series of diagram chases.

The converse is true, too: up to some technicalities, an external functor in **Set** arises from a functor to **Set** from a small category. That means we said enough about it in E{1 through E{4 to characterize it.

5.7.5 **Theorem**   Let

$$(\mathscr{D}_0; \mathscr{D}_1; \mathscr{D}_2; d^0; d^1; u; ¼_0; ¼_1; p_1; p_2; c)$$

be an external functor to a category

$$\mathscr{C} = (\mathscr{C}_0; \mathscr{C}_1; \text{source}; \text{target}; \text{unit}; \text{comp})$$

in **Set**. De¯ne $F : \mathscr{C} \;¡!\;$ **Set** by

F{1 If C is an object of $\mathscr{C}$, then $F(C) = ¼_1^{\;1}(C)$.

F{2 If $f : C \;¡!\; C^0$ is an arrow of $\mathscr{C}$ and $x \, 2 \, F(C)$, then $F(f)(x)$ is the target of the unique arrow $®$ of $\mathscr{D}$ for which $¼_1(®) = f$ and $d^0(®) = x$.

Then $F : \mathscr{C} \;¡!\;$ **Set** is a functor, and there is a unique isomorphism $^- : \mathscr{D} \;¡!\;$ $\mathbf{G}(\mathscr{C}; F)$ for which $\mathbf{G}(F) \pm \,^- = ¼$.

This theorem is essentially the discrete case of Theorem ES 4.3.7.

F{2 makes use of the fact that Diagram (ES 5.4)(a) is a pullback. A seasoned categorist will simply name $® \, 2 \, F^2$ as $(x; f)$, since he knows that between any two pullbacks de¯ned by (ES 5.4)(a) there is a unique isomorphism respecting the projections $¼_1$ and d^0.

5.8 The realizability topos

The category of modest sets has been proposed as a suitable model for the polymorphic ¸-calculus. It is a subcategory of a speci¯c topos, the realizability topos. Space limitations prevent us from giving a full exposition of this topic, Here we describe how the realizability topos and the subcategory of modest sets are constructed. Further references are [Carboni, Freyd and Scedrov, 1988], [Rosolini, 1990], [Gray, 1991].

5.8.1 Realizability sets In the discussion below, we will be writing $f(x)$ where f is a partial function. It will be understood that $f(x)$ is defined when we write this.

A **realizability set** is a pair $\mathbf{A} = (A; =_\mathbf{A})$ where A is a set called the **carrier** of $\mathbf{A}$ and $=_\mathbf{A}: A \times A \to \mathscr{P}(\mathbf{N})$ is a function to sets of natural numbers (thought of as reasons that two elements are equal; in fact, they can be thought of as the set of Gödel numbers of proofs that they are). We denote the value of this function at $(a_1; a_2) \in A \times A$ by $[\![a_1 =_\mathbf{A} a_2]\!]$, often abbreviated to $[\![a_1 = a_2]\!]$. This is subject to the following conditions:

REAL{1 There is a partial recursive function f such that for any a_1, $a_2 \in A$ and $n \in [\![a_1 = a_2]\!]$, $f(n) \in [\![a_2 = a_1]\!]$.

REAL{2 There is a partial recursive function of two variables g with the property that if $n \in [\![a_1 = a_2]\!]$ and $m \in [\![a_2 = a_3]\!]$, then $g(n; m) \in [\![a_1 = a_3]\!]$.

This is made into a category by defining an arrow from $\mathbf{A} = (A; =_\mathbf{A})$ to $\mathbf{B} = (B; =_\mathbf{B})$ to be an equivalence class of partial functions $\varphi : A \to B$ such that there is a partial recursive function f such that $n \in [\![a_1 = a_2]\!]$ implies that $f(n) \in [\![\varphi a_1 = \varphi a_2]\!]$. Two such arrows φ and ψ are equal if there is a partial recursive f such that $n \in [\![a = a]\!]$ implies that $f(n) \in [\![\varphi a = \psi a]\!]$.

Note that the last definition implies that f is defined on all elements $a \in A$ for which $[\![a = a]\!] \neq \emptyset$. The other elements of A are irrelevant.

5.8.2 This category is a topos. We will not prove this, but simply describe some of the constructions required. To find products, for example, first choose a recursive bijection $f : \mathbf{N} \times \mathbf{N} \to \mathbf{N}$. If $\mathbf{A} = (A; =_\mathbf{A})$ and $\mathbf{B} = (B; =_\mathbf{B})$ are realizability sets, the product is $(A \times B; =_\mathbf{A} \times =_\mathbf{B})$, where the latter is defined by

$$[\![(a_1; b_1) = (a_2; b_2)]\!] = \{f(n; m) \mid n \in [\![a_1 = a_2]\!]; m \in [\![b_1 = b_2]\!]\}$$

Notice that a choice of f is required to show that products exist. The products thereby exist, but, owing to the uniqueness of categorical products, do not depend in any way on the choice of the function.

Equalizers can be defined as follows. If $\varphi; \psi : \mathbf{A} \to \mathbf{B}$ are two arrows their equalizer is given by $(A; =_{\varphi; \psi})$ where $[\![a_1 =_{\varphi; \psi} a_2]\!] = [\![\varphi a_1 = \psi a_2]\!]$.

Finally power objects can be constructed as follows. Let f denote a pairing as above and also choose an enumeration g_e of all the partial recursive functions. The carrier of $\mathscr{P}\mathbf{A}$ is $[A \to \mathscr{P}\mathbf{N}]$, all functions from A to sets of natural numbers. If P and Q are two such functions, then we will say that $f(d; e) \in [\![P = Q]\!]$ if for all $n \in P(a)$ and $m \in [\![a = b]\!]$, we have $g_d(f(n; m)) \in Q(b)$ and for all $n \in Q(a)$ and $m \in [\![a = b]\!]$, we have $g_e(f(n; m)) \in P(b)$.

This topos is called the **realizability** or **effective** topos. It is due to Martin Hyland [1982], although the basic idea goes back to S. Kleene. See also [Rosolini, 1987].

5.8.3 Modest sets `Modest sets' is used more-or-less interchangeably to denote either a certain full subcategory of the category of realizability sets or an internal category object of that topos. A great deal of effort has been put into describing the connection between the two. We will begin with the former. The category of modest sets can be described directly and then embedded into the realizability topos.

A **modest set** consists of $\mathbf{A} = (\mathbf{N}; =_{\mathbf{A}})$ where as usual $\mathbf{N}$ denotes the natural numbers and $=_{\mathbf{A}}$ is a **partial equivalence relation** or **PER** on $\mathbf{N}$, which means it is symmetric and transitive, but not necessarily reflexive. We think of $\mathbf{A}$ as a quotient of a subobject of $\mathbf{N}$; the subobject is the set of n for which $n =_{\mathbf{A}} n$ modulo the relation $=_{\mathbf{A}}$.

The category of modest sets has arrows from $\mathbf{A}$ to $\mathbf{B}$ defined as partial recursive functions f defined on all n such that $n =_{\mathbf{A}} n$ and such that $n =_{\mathbf{A}} m$ implies that $f(n) =_{\mathbf{A}} f(m)$. Note that since the relation is symmetric and transitive, as soon as there is some m with $n =_{\mathbf{A}} m$, it is also the case that $n =_{\mathbf{A}} n$ and so $f(n)$ and $f(m)$ are defined.

The modest sets form a cartesian closed category. Choose, as above, an enumeration g_e of the partial recursive functions. Then $[\mathbf{A} \to \mathbf{B}]$ is the PER with relation given by $d =_{[\mathbf{A} \to \mathbf{B}]} e$ if and only if whenever $n =_{\mathbf{A}} m$ then $g_d(n) =_{\mathbf{B}} g_e(m)$. The modest sets do not form a topos.

We embed the modest sets into the realizability sets by choosing a pairing f and associating to the modest set $\mathbf{A} = (\mathbf{N}; =_{\mathbf{A}})$ the realizability set with carrier $\mathbf{N}$ and $e \in [\![n = m]\!]$ if and only if $n =_{\mathbf{A}} m$ and $e = f(n; m)$. Although this appears to depend on the choice of a pairing, it is easy to see that up to isomorphism it does not.

In fact it cannot. The pairing is used only to show that products exist. But their properties { in particular their isomorphism class { are independent of the particular construction used to prove their existence. Similar remarks apply to the cartesian closed structure, which does not depend on a particular enumeration of the partial recursive functions.

5.8.4 The internal category of modest sets This is a very brief glance at how one might internalize the description of modest sets to produce a category object inside the category of realizability sets. Except by way of motivation, this category object has no real connection with the category of modest sets. Nevertheless, we will call it the **internal category of modest sets**.

This internal category has the remarkable property that every internal diagram in it has a limit. This is not possible for ordinary categories (unless they are just posets), but the proof that it is not possible requires a property of set theory which is not valid for the realizability sets (namely the axiom of choice).

The construction is an exercise in internal expression. A modest set is a relation on $\mathbf{N}$ with certain properties. The natural numbers object in the category

of realizability sets is the object $(\mathbf{N}; =_\mathbf{N})$, where $e \in [\![n =_\mathbf{N} m]\!]$ if and only if $n = m = e$. That is $[\![n =_\mathbf{N} m]\!] = \{n\}$ if $n = m$ and is empty otherwise. We will denote it by $\mathbf{N}$. Then a modest set is a subset of $\mathbf{N} \times \mathbf{N}$ with certain properties.

The set of objects of the internal category of modest sets is then a certain set of subsets of $\mathbf{N} \times \mathbf{N}$, which is the same as a subset of $\mathscr{P}(\mathbf{N} \times \mathbf{N})$. We want to describe this subset as consisting of those relations that are symmetric and transitive and double negation closed. The trick is to define endoarrows s and t of $\mathscr{P}(\mathbf{N} \times \mathbf{N})$ that associate to a relation R the relations R^{op} and $R \cdot R$, respectively. For s, this is easy. There is an arrow $\langle p_2; p_1 \rangle : \mathbf{N} \times \mathbf{N} \to \mathbf{N} \times \mathbf{N}$ that switches the coordinates and we let $s = \mathscr{P}(\langle p_2; p_1 \rangle)$.

For t, it is a little harder. By Yoneda, an arrow $\mathscr{P}(\mathbf{N} \times \mathbf{N})$ to itself can be defined by describing a natural transformation $\mathrm{Hom}(-; \mathscr{P}(\mathbf{N} \times \mathbf{N}))$ to itself. An element of $\mathrm{Hom}(A; \mathscr{P}(\mathbf{N} \times \mathbf{N}))$ is, by the defining property of $\mathscr{P}$, a subobject $R \subseteq A \times \mathbf{N} \times \mathbf{N}$. Given such an R, we define R^0 to be the pullback in the following diagram:

$$
\begin{array}{ccc}
R^0 & \xrightarrow{\quad\ \ \ } & R \\
\downarrow{\scriptstyle ?} & & \downarrow{\scriptstyle ?} \\
R & \xrightarrow{\quad\ \ \ } & A \times \mathbf{N}
\end{array}
$$

The two maps from $R \to A \times \mathbf{N}$ are the inclusion into $A \times \mathbf{N} \times \mathbf{N}$ followed by $\langle p_1; p_2 \rangle$ and $\langle p_1; p_3 \rangle$, respectively. You should think of R as being a set of triples $(a; n_1; n_2)$ and then $R^0 \subseteq A \times \mathbf{N} \times \mathbf{N} \times \mathbf{N}$ is the set of 4-tuples $(a; n_1; n_2; n_3)$ such that $(a; n_1; n_2) \in R$ and $(a; n_2; n_3) \in R$. Then the inclusion of R^0 into $A \times \mathbf{N} \times \mathbf{N} \times \mathbf{N}$ followed by $\langle p_1; p_2; p_4 \rangle$ gives us an arrow $R^0 \to A \times \mathbf{N} \times \mathbf{N}$ whose image we denote by $t(R)$. It is interpreted as the set of $(a; n_1; n_3)$ such that there is an n_2 with both $(a; n_1; n_2) \in R$ and $(a; n_2; n_3) \in R$. We omit the proof that this construction from R to $t(R)$ is natural in A, but assuming that, it follows from the Yoneda Lemma that this results from an endomorphism t of $\mathscr{P}(\mathbf{N} \times \mathbf{N})$.

Finally, the object M_0 of objects is defined as the limit of

$$
\mathscr{P}(\mathbf{N} \times \mathbf{N}) \underset{t}{\overset{s}{\rightrightarrows}} \mathscr{P}(\mathbf{N} \times \mathbf{N})
$$

where the middle arrow is the identity. This is to be interpreted as the set of relations that are fixed under both s and t.

The object M_0 is the object of objects of the internal category of modest sets. The object of arrows is defined as a subobject of the object $[\mathbf{N} \to \mathbf{N}]_e$ of partial arrows of $\mathbf{N}$ to $\mathbf{N}$; namely those that satisfy the internal version of the definition of the ordinary of modest sets. This construction is tedious, but not difficult, and we omit it.

Answers to Exercises

Solutions for Chapter 1

Section 1.1

1. This should have two sorts 1 and b. There should be an operation true : 1 ¡! b, an operation false : 1 ¡! b and a cocone

$$
\begin{array}{ccc}
1 & & 1 \\
@ & & \vert \\
\text{true}^{@} & & \vert \;\; \text{false} \\
& @ & a \;\; \vert \\
& b &
\end{array}
$$
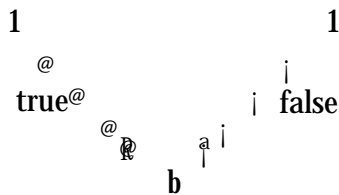
We need a cone with empty base to express that 1 is terminal. We should have operations _ : b £ b ¡! b, ^ : b £ b ¡! b and : : b ¡! b. We will need diagrams to express equations like

$$
\begin{array}{rcl}
: \text{true} & = & \text{false} \\
: \text{false} & = & \text{true} \\
: \; \wedge (x; y) & = & \_(x; y) \\
\_(0; 0) & = & 0 \\
\_(0; 1) & = & 1 \\
\_(1; 0) & = & 1 \\
\_(1; 1) & = & 1
\end{array}
$$

There are other equations, but they all follow from these. In particular, we do not, in this case, have to express the distributive laws since they follow, there being so few elements.

2. If we have one sort 1, one empty cone with vertex 1 and one cocone

$$
\begin{array}{ccc}
1 & & 1 \\
\text{id}^{@} & & \vert \;\; \text{id} \\
& @ & a \;\; \vert \\
& 1 &
\end{array}
$$
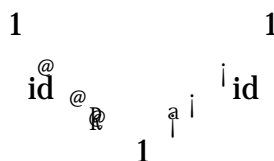
then there is no model in sets because the model must take 1 to the one-element set and that must satisfy that $1 = 1 + 1$, which is impossible. It follows from Exercise 4 of Section 8.6 that this sketch does have a model in the category of monoids.

Section 1.2

1. This is most readily done using elements, although it can all be done with diagrams. If $x ¤ x^{i\,1} = 1$ and $y ¤ y^{i\,1} = 1$, then

$$(x ¤ y) ¤ (y^{i\,1} ¤ x^{i\,1}) = x ¤ (y ¤ (y^{¤i\,1} ¤ x^{i\,1})) = x ¤ ((y ¤ y^{i\,1}) ¤ x^{i\,1})$$
$$= x ¤ (1 ¤ x^{i\,1}) = x ¤ x^{i\,1} = 1$$

If $2 ⊖ 0$ in a ⁻eld, then $2^{i\,1}$ exists, whence $(2 ¤ 2)^{i\,1} = 4^{i\,1}$ also exists. Hence $4 ⊖ 0$.

2. Let **Q** and F_2 denote the ⁻elds of rational numbers and the two-element ⁻eld respectively. In order to have a product of **Q** and F_2, we have to have a cone

$$F$$

$$i \qquad @$$

$$\overset{i}{a}{}^{i} \qquad @{}_{\mathbb{R}}$$

$$\mathbf{Q} \qquad\qquad F_2$$
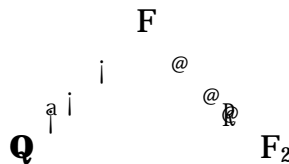
which is exactly what we do not have in the category of ⁻elds.

Section 1.3

1. a. Several things have to be shown. First that the image $N_0 \mu N$ is closed under the operations (that is if all the arguments of the operation are in the image, so does the value); second that the diagrams continue to commute; third that the cones remain products; and fourth that the cocones remain sums.

Let $u : s_1 \pounds s_2 \pounds ¢¢¢ \pounds s_n ¡!\ s$ be an operation. For $i = 1; :::; n$ let $x_i 2 M(s_i)$ and $x = u(x_1; :::; x_n)$. Then $f(x) = u(f(x_1); ::::; f(x_n))$, which demonstrates the ⁻rst point. Since the operations in N_0 are the restrictions of those in N any two paths that agree on N do so on N_0 (actually, whether or not they do on M). For the cones, we illustrate on binary cones. Suppose $s \tilde{A} r ¡!\ t$ is a cone in the sketch. Then in the diagram

$$M(r) \longrightarrow N_0(r) \longrightarrow N(r)$$

$$\Big\downarrow \qquad\qquad \Big\downarrow f \qquad\qquad \Big\downarrow$$

$$M(s) \pounds M(t) \longrightarrow N_0(s) \pounds N_0(t) \longrightarrow N(s) \pounds N(t)$$
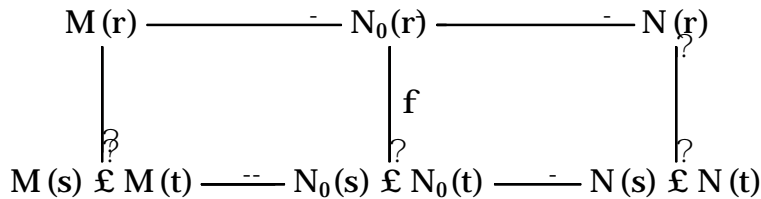
certain functions have been labeled as being surjective or injective (although others are and some are bijective). These use the facts that the product of two injective functions is injective and the product of two surjective functions is surjective. That f is surjective follows from the fact that the composite of two surjective functions is surjective and if the composite of two functions is surjective, so is the second one. Dually, the fact that f is injective uses the facts that the composite of two injective functions is injective and that if the composite of two functions is injective, the first one is.

The dual argument, using the fact that a sum of injectives is injective and a sum of surjectives is surjective, gives the corresponding result for discrete cocones.

It is worth noting that these arguments fail if either the cones or cocones fail to be discrete or if the category in which the models are taken is other than the category of sets. The reason is that even in the category of sets, the arrow induced between equalizers of epis is not necessarily epic and between coequalizers of monos will not be monic. (Equalizers and coequalizers are defined in chapter 8.) In categories other than sets, even the sum of epics will not generally be epic, nor the sums of monics monic.

b. There are two ways of doing this. One is to show that the intersection of submodels is a submodel. Then the intersection of all submodels is clearly the smallest submodel. Another is to take the image of the initial term model in the component of that model. Let M be the initial term model, N is the given model and N_0 this submodel. If $N_1 \subseteq N$ is any other submodel, there is an initial term model M^0 in the component of N_1 that has an arrow $M^0 \to N_1$. But $N_1 \subseteq N$ and so is in the same component as N. Thus $M^0 = M$ and the map $M \to N_1 \subseteq N$ is the original map $M \to N$. Since that factors through N_1, it follows that $N_0 \subseteq N_1$.

Solutions for Chapter 2

Section 2.1

1. Add a new unary operation $i : c_1 \to c_1$ and the following diagrams:

$$
\begin{array}{ccccc}
& & c_2 & & \\
& \overset{i}{\nearrow} & \big\downarrow @ & & \\
\mathrm{id} & \overset{i}{\cdot} & \mathrm{hid;i}\,i & \overset{@}{\cdot} & i \\
& \overset{a}{\nearrow}\, i & \big\downarrow ? & \overset{@}{\nwarrow}\, \text{®} & \\
c_1 & \xrightarrow[p_1]{\,¾\,} & c_2 & \xrightarrow[p_2]{\;-\;} & c_1
\end{array}
\qquad
\begin{array}{ccccc}
& & c_2 & & \\
& \overset{i}{\nearrow} & \big\downarrow @ & & \\
i & \overset{i}{\cdot} & \mathrm{hi;id}\,i & \overset{@}{\cdot} & \mathrm{id} \\
& \overset{a}{\nearrow}\, i & \big\downarrow ? & \overset{@}{\nwarrow}\, \text{®} & \\
c_1 & \xrightarrow[p_1]{\,¾\,} & c_2 & \xrightarrow[p_2]{\;-\;} & c_1
\end{array}
$$

$$c_1 \xrightarrow{\;h\langle id; i\rangle\;} c_2$$

$$c_1 \xrightarrow{\;h i; id\rangle\;} c_2$$

(diagrams with $id@$, c, c_1)
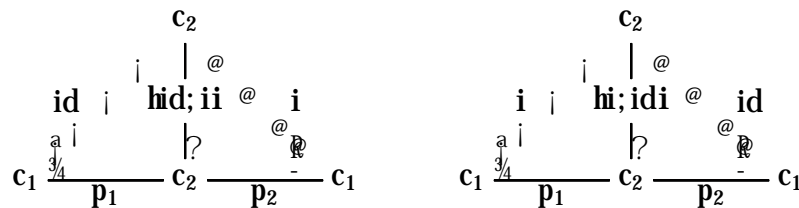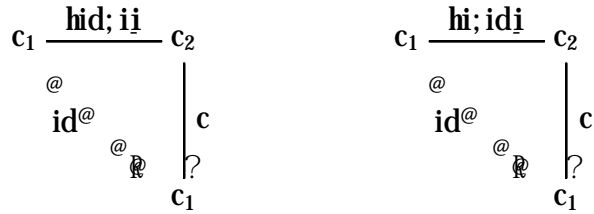
2. Let $(C_0; C_1; s; t; u; c)$ and $(D_0; D_1; s; t; u; c)$ be two models of the sketch for categories. Note that we have used the common convention of using the same letter to stand for the arrow in the sketch and in the model (in every model, in fact). A homomorphism consists of an arrow $F_0 : C_0 \to D_0$ and an arrow $F_1 : C_1 \to D_1$ that satisfy some conditions forced by the fact that a homomorphism is a natural transformation. First o®, we have $s \cdot F_1 = F_0 \cdot s$, $t \cdot F_1 = F_0 \cdot t$ and $u \cdot F_0 = F_1 \cdot u$. These mean that the homomorphism preserves source, target and identity arrows. These identities induce a unique arrow $F_2 : C_2 \to D_2$ such that $p_1 \cdot F_2 = F_1 \cdot p_1$ and $p_2 \cdot F_2 = F_1 \cdot p_2$. We further suppose that $c \cdot F_2 = F_1 \cdot c$. These conditions mean that the homomorphism preserves composition; thus the homomorphism is a functor.

Section 2.2

1. Referring to Diagram (ES 2.1), we see that incl is the arrow opposite $\Im$true $: 1 \to b$ in a cone, which means that in a model it is the arrow opposite an arrow from 1. But every arrow from 1 is monic (see Section 2.8, Exercise 7). Hence incl is monic.

Section 2.3

1. It is the set of all diagrams (not necessarily commutative) of the form

$$A \xrightarrow{\;f\;} B$$

with g, h, C.
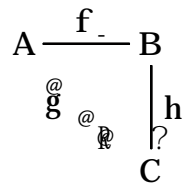
2. It is the set of all commutative diagrams of the form

$$A \xrightarrow{\;f\;} B$$

with g, h, C.
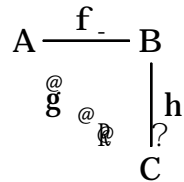
3. For each object C of $\mathscr{C}$, let $M^0(c) = f(x; c) \mid x \in M(c)g$. If $t : c \to d$ is an arrow in $\mathscr{S}$, let $t(x; c) = (t(x); d)$. Then the first projection is an isomorphism $M^0 \to M$ and it is clear that $M(c) \setminus M(d) = \emptyset$ for $c \ne d$.

Solutions for Chapter 3

Section 3.1

1. A model of $\mathcal{E}$ is a sketch homomorphisms $\mathcal{E} \to \mathcal{C}$. This assigns to the single node of $\mathcal{E}$ an object E and that is all. If we denote the single object of $\mathcal{E}$ by w, then the functor $M \mapsto M(e)$ is evidently an isomorphism between the objects of $\mathcal{C}$ and those of **Mod**$(\mathcal{E}; \mathcal{C})$. If $\alpha : M \to N$ is a natural transformation, the $\alpha e : M(e) \to N(e)$ is an arrow of $\mathcal{C}$ and is subject to no conditions. Thus means that **Nat**$(M; N) = \mathrm{Hom}(M(e); N(e))$ and so the functor is an isomorphism.

Section 3.2

1. Nat has one such node, denoted 1, as does **Stack**. The disjoint union has two such nodes, but when the coequalizer is formed they are identi¯ed to a single node in $\mathcal{S}$. This node, and only this, must become a singleton in any model.

Section 3.3

1. $U \pounds U$ is induced by the sketch homomorphism from $\mathcal{E}$ to the sketch for monoids (given in 7.2.1 and 7.3.2) that takes e to s $\pounds$ s.

2. For any $u : s_1 \to s_2$ in $\mathcal{S}$, we have $F(u) : F(s_1) \to F(s_2)$ which gives a commutative square

$$
\begin{array}{ccc}
M(F(s_1)) & \xrightarrow{M(F(u))} & M(F(s_2)) \\
\alpha F(s_1) \downarrow & & \downarrow \alpha F(s_2) \\
N(F(s_1)) & \xrightarrow[N(F(u))]{} & N(F(s_2))
\end{array}
$$

which is the same as

$$
\begin{array}{ccc}
F^{\mathrm{x}}(M)(s_1) & \xrightarrow{F^{\mathrm{x}}(M)u} & F^{\mathrm{x}}(M)(s_2) \\
F^{\mathrm{x}}(\alpha)s_2 \downarrow & & \downarrow F^{\mathrm{x}}(N)u \\
F^{\mathrm{x}}(N)(s_1) & \xrightarrow[F^{\mathrm{x}}(\alpha)s_2]{} & F^{\mathrm{x}}(N)(s_2)
\end{array}
$$

which is naturality of $F^{\mathrm{x}}(M)$.

3. We have for $\mathrm{id}_{\mathcal{S}} : \mathcal{S} \to \mathcal{S}$ that $(\mathrm{id}_{\mathcal{S}})^{\mathrm{x}}(M) = M \cdot \mathrm{id}_{\mathcal{S}} = M$ and for $\alpha : M \to N$, $(\mathrm{id}_{\mathcal{S}})^{\mathrm{x}}(\alpha) = \alpha\,\mathrm{id}_{\mathcal{S}} = \alpha$. Thus $(\mathrm{id}_{\mathcal{S}})^{\mathrm{x}}$ is the identity functor. If $G : R \to \mathcal{S}$ is another homomorphism of sketches, then $(F \cdot G)^{\mathrm{x}}(M) = M \cdot F \cdot G = G^{\mathrm{x}}(M \cdot F) = G^{\mathrm{x}}(F^{\mathrm{x}}(M)) = (G^{\mathrm{x}} \cdot F^{\mathrm{x}})(M)$ and similarly for $\alpha : M \to N$. Thus $(F \cdot G)^{\mathrm{x}} = G^{\mathrm{x}} \cdot F^{\mathrm{x}}$ which shows that **Mod**$_{\mathcal{C}}(\ )$ is a contravariant functor.

4. Suppose, say, that ¾ is a diagram. The other two possibilities are similar. Suppose ¾ says that

$$s_1 \pm ¢¢¢ \pm s_n = t_1 \pm ¢¢¢ \pm t_m$$

Then ¾ is satis¯ed in $F^¤(M)$ if and only if

$$F^¤(M)(s_1) \pm ¢¢¢ \pm F^¤(M)(s_n) = F^¤(M)(t_1) \pm ¢¢¢ \pm F^¤(M)(t_m)$$

which is the same as

$$M(F(s_1)) \pm ¢¢¢ \pm M(F(s_n)) = M(F(t_1)) \pm ¢¢¢ \pm M(F(t_m))$$

which is the same as the condition that M satisfy $F(¾)$.

Solutions for Chapter 4

Section 4.1

1. Let $\mathscr{E}_C$ be the ¯ber over an object C. If X is an object of $\mathscr{E}_C$, then $P(X) = C$ and $P(\mathrm{id}_X) = \mathrm{id}_C$ by de¯nition of functor It follows that id_X is an arrow of $\mathscr{E}_C$. If $f : X \; ¡! \; Y$ and $g : Y \; ¡! \; Z$ are arrows of $\mathscr{E}_C$, then $P(f) = P(g) = \mathrm{id}_C$, so $P(g \pm f) = P(g) \pm P(f) = \mathrm{id}_C \pm \mathrm{id}_C = \mathrm{id}_C$, so $g \pm f$ is an arrow of $\mathscr{E}_C$. Hence $\mathscr{E}_C$ is a subcategory.

2. Let $P : \mathscr{E} \; ¡! \; \mathscr{C}$ be a ¯bration with cleavage $°$. To see that Ff is a functor, note ¯rst that $°(f; Y) \pm \mathrm{id}_Y = \mathrm{id}_Y \pm °(f; Y)$ so that $Ff(\mathrm{id}_Y) = \mathrm{id}_{Ff_Y}$. Suppose $u^0 : Y^0 \; ¡! \; Y^{00}$. Then

$$°(f; Y^{00}) \pm Ff(u^0) \pm Ff(u) = u^0 \pm °(f; Y^0) \pm Ff(u) = u^0 \pm u \pm °(f; Y)$$

so by uniqueness $Ff(u^0 \pm u) = Ff(u^0) \pm Ff(u)$.

Now we show that F is a contravariant functor. If we show that $F(\mathrm{id}_C)$ is the identity on arrows it will have to be the identity on objects because functors preserve source and target. Let C be an object of $\mathscr{C}$ and u an arrow of $F(C)$. Then $F(\mathrm{id}_C)(u)$ is the unique arrow for which $°(\mathrm{id}_C; Y^0) \pm F(\mathrm{id}_C)(u) = u \pm °(\mathrm{id}_C; Y)$. By SC{1, this requirement becomes $\mathrm{id}_{Y^0} \pm F(\mathrm{id}_C)(u) = u \pm \mathrm{id}_Y$. Thus $F(\mathrm{id}_C)(u) = u$ as required.

Now suppose $f : C \; ¡! \; D$ and $g : D \; ¡! \; E$ in $\mathscr{C}$. Let $u : Y \; ¡! \; Y^0$ in $F(C)$. Then $F(g \pm f)(u)$ must be the unique arrow for which

$$°(g \pm f; Y^0) \pm F(g \pm f)(u) = u \pm °(g \pm f; Y)$$

But

$$
\begin{aligned}
u \pm °(g \pm f; Y) &= u \pm °(g; Y) \pm °(f; Ff(Y)) \\
&= °(g; Y^0) \pm Fg(u) \pm °(f; Ff(Y)) \\
&= °(g; Y^0) \pm °(f; Ff(Y^0)) \pm Ff(Fg(u))
\end{aligned}
$$

so by CA{2 and SC{2, $F(g \pm f) = F(f) \pm F(g)$.

3. a. We will also denote the functor by Á. It is a fibration if for every element x of $\mathbf{Z}_2$ there is an element u of $\mathbf{Z}_4$ such that $Á(u) = x$ (that is CA{1) and for every $v \in \mathbf{Z}_4$ and $h \in \mathbf{Z}_2$ such that $x + h = Á(v) \pmod 2$, there is a unique $w \in \mathbf{Z}_4$ for which $Á(w) = h$ and $u + w = v \pmod 4$ (which is CA{2). This amounts to saying that Á is surjective and every equation $m + y = n \pmod 4$ can be solved uniquely for y, which requires simple case checking (or knowing that $\mathbf{Z}_2$ and $\mathbf{Z}_4$ are groups). An analogous proof works for opfibration.

b. A splitting would be a monoid homomorphism (this follows immediately from SC{1 and SC{2) which would make Á a split epimorphism, which it is not by Exercise 2 of Section 2.9.

4. a. P preserves source and target by definition. Let $(h; k) : f \to f'$ and $(h'; k') : f' \to f''$. Then

$$P((h'; k') \circ (h; k)) = P(h' \circ h; k' \circ k) = k' \circ k = P(h'; k') \circ P(h; k)$$

Thus P preserves composition. The identity on an object $(A; B)$ is $(\mathrm{id}_A; \mathrm{id}_B)$, so it follows immediately that P preserves identities.

b. Let $f : C \to D$ in $\mathscr{C}$ and let $k : B \to D$ be an object of $\mathscr{A}$ lying over D. Let $°(f; k)$ be the arrow $(u; f)$ of $\mathscr{A}$ defined in ES 4.1.5. $P(°(f; k)) = f$ so CA{1 is satisfied. As for CA{2, let $(v; v') : z \to k$ in $\mathscr{A}$ and let h be an arrow of $\mathscr{C}$ such that $f \circ h = v'$. Let w be the unique arrow given by the pullback property for which $u \circ w = v$ and $u' \circ w = h \circ z$. The last equation says that $(w; h)$ is an arrow of $\mathscr{A}$, and $(u; f) \circ (w; h) = (u \circ w; f \circ h) = (v; v')$ as required. The uniqueness property of the pullback means that $(w; h)$ is the only such arrow.

Section 4.2

1. The identity arrow for $(x; C)$ is $(x; \mathrm{id}_C)$. Let $(x; f) : (x; C) \to (x'; C')$, $(x'; f') : (x'; C') \to (x''; C'')$ and $(x''; f'') : (x''; C'') \to (x'''; C''')$ be arrows of $\mathbf{G}_0(\mathscr{C}; F)$. Then $((x''; f'') \circ (x'; f')) \circ (x; f) = (x''; f'' \circ f') \circ (x; f) = (x''; (f'' \circ f') \circ f) = (x''; f'' \circ (f' \circ f)) = (x''; f'') \circ ((x'; f') \circ (x; f))$ so composition is associative.

2. Let $f : C \to C'$ in $\mathscr{C}$. Let $(x; C)$ be an object of $\mathbf{G}_0(\mathscr{C}; F)$ lying over C. Define $\cdot(f; (x; C))$ to be $(x; f) : (x; C) \to (x'; C')$ where $x' = Ff(x)$. Now suppose $(x; g) : (x; C) \to (y; C'')$ and suppose $k : C' \to C''$ has the property that $k \circ f = g$. The arrow required by OA{2 is $(x'; k) : (x'; C') \to (y; C'')$. This is well defined since $Fk(x') = Fk(Ff(x)) = F(k \circ f)(x) = Fg(x)$ which must be y since $(y; C'')$ is the given target of $(x; g)$. Moreover it satisfies OA{2 since $(x'; k) \circ (x; f) = (x; k \circ f) = (x; g)$. An arrow satisfying OA{2 must lie over k and have source $(x'; C')$ and target $(y; C'')$ so that it can compose with f to give g, so $(x'; k)$ is the only possible such arrow. To see that $\cdot$ is a splitting, suppose that $(x'; f') : (x'; C') \to (x''; C'')$. Then $\cdot(f'; (x'; C')) = (x'; f')$ and $(x'; f') \circ (x; f) = (x; f' \circ f) = \cdot(f' \circ f; (x; C))$ as required. The verification for identities is even easier.

3. The identity arrow is $(id_x; id_C) : (x; C) \to (x; C)$. It is well defined since $F(id_C)(x) = x$, so $id_x : F(id_C)(x) \to x$. Suppose that $(u; f)$ is an arrow from $(x; C)$ to $(x'; C')$, so that $u : Ff(x) \to x'$. Similarly, let $(u'; f') : (x'; C') \to (x''; C'')$ and $(u''; f'') : (x''; C'') \to (x'''; C''')$. Then

$$((u''; f'') \circ (u'; f')) \circ (u; f) = (u''; Ff''(u'); f'' \circ f') \circ (u; f)$$
$$= (u'' \circ Ff''(u') \circ F(f'' \circ f')(u); (f'' \circ f') \circ f)$$

and

$$(u''; f'') \circ ((u'; f') \circ (u; f)) = (u''; f'') \circ (u' \circ Ff(u); f' \circ f)$$
$$= (u'' \circ Ff''(u' \circ Ff(u)); f'' \circ (f' \circ f))$$

The result follow from the facts that F and Ff'' are functors and composition in $\mathscr{C}$ is associative.

4. Let $f : C \to C'$ and let $(x; C)$ lie over C. Define $\cdot(f; (x; C))$ to be

$$(id_{Ff(x)}; f) : (x; C) \to (Ff(x); C')$$

Let $(u; g) : (x; C) \to (y; C'')$ so that $u : Fg(x) \to y$. Let $k : C' \to C''$ satisfy $k \circ f = g$. Then $(u; k) : (Ff(x); C') \to (y; C'')$ because the domain of u is $Fg(x) = F(k \circ f)(x) = F(k)(Ff(x))$. Furthermore,

$$(u; k) \circ (id_{Ff(x)}; f) = (u \circ F(id_{Ff(x)}); k \circ f) = (u; g)$$

as required by OA{2. It follows as in the answer to the second problem that $(u; k)$ is the only possible arrow with this property.

5.

$$((t; m)(t'; m'))(t''; m'') = (t \otimes (m; t'); mm')(t''; m'')$$
$$= (t \otimes (m; t') \otimes (mm'; t''); mm'm'')$$

and

$$(t; m)((t'; m')(t''; m'')) = (t; m)(t; \otimes(m'; t''); m'm'')$$
$$= (t \otimes (m; t' \otimes (m'; t'')); mm'm'')$$

However, by MA{4,

$$\otimes(m; t') \otimes (mm'; t'') = \otimes(m; t') \otimes (m; \otimes(m'; t''))$$

and that is $\otimes(m; t' \otimes (m'; t''))$ by MA| 2.

6. Let ¯ take an object x of $F(C)$ to $(x; C)$ and an arrow u to $(u; id_C)$. By GC{3, if $v : y \to z$ then

$$(v; id_C) \circ (u; id_C) = (v \circ F(id_C)(u); id_C \circ id_C) = (v \circ u; id_C)$$

so ¯ preserves composition. It is clearly a bijection and preserves identities.

Section 4.3

1. Suppose that $® : F \;¡! \; G$ is a natural transformation. $\mathbf{G}®$ preserves identities because $\mathbf{G}®(\mathrm{id}_X ; \mathrm{id}_C) = ®C(\mathrm{id}_X ; \mathrm{id}_C) = (\mathrm{id}_{®C(x)} ; \mathrm{id}_C)$. Suppose that $(u ; f) : (x ; C) \;¡! \; (x^0 ; C^0)$ and $(u^0 ; f^0) : (x^0 ; C^0) \;¡! \; (x^{00} ; C^{00})$. Then

$$(u^0 ; f^0) \pm (u ; f) = (u^0 \pm F f^0(u) ; f^0 \pm f)$$

by GC{3 (Section ES 4.2). On the other hand,

$$
\begin{aligned}
\mathbf{G}®(u^0 ; f^0) \pm \mathbf{G}®(u ; f) &= (®C^{00}u^0 ; f^0) \pm (®C^0 u ; f) \\
&= (®C^{00}u^0 \pm G f^0(®C^0 u) ; f^0 \pm f) \\
&= (®C^{00}u^0 \pm ®C^{00}(F f^0(u)) ; f^0 \pm f) \\
&= (®C^{00}(u^0 \pm F f^0(u)) ; f^0 \pm f) \\
&= \mathbf{G}®(u^0 \pm F f^0(u) ; f^0 \pm f)
\end{aligned}
$$

where the third equality uses the naturality of $®$ and the fourth uses the fact that $®C^{00}$ is a functor. Thus $\mathbf{G}®$ preserves composition.

2. GR{2 implies that $\mathbf{G}$ preserves identity natural transformations, since any component of an identity transformation is an identity arrow. Let $® : F \;¡! \; F^0$ and $^- : F^0 \;¡! \; F^{00}$ be natural transformations, where F, F^0 and F^{00} are functors from $\mathscr{C}$ to **Cat**. Then for $(u ; f) : (x ; C) \;¡! \; (x ; C^0)$,

$$\mathbf{G}(^- \pm ®)(u ; f) = ((^- \pm ®)C^0 u ; f) = (^- C^0(®C^0 u) ; f)$$

and

$$(\mathbf{G}^-(\mathbf{G}®))(u ; f) = \mathbf{G}^-(®C^0 u ; f) = (^- C^0(®C^0 u) ; f)$$

so $\mathbf{G}$ preserves composition.

Section 4.4

1. If $\mathscr{A}$ and $\mathscr{B}$ are monoids, then each has only one object. By WP{1, an object of $\mathscr{A} \, \mathrm{wr}^G \, \mathscr{B}$ is a pair $(A ; P)$ where A is the only object of $\mathscr{A}$ and $P : G(A) \;¡! \; \mathscr{B}$ is a functor. But if $\mathscr{B}$ has only one object and $G(A)$ is discrete, there is only one functor from $G(A)$ to $\mathscr{B}$ { it must take all the objects of $G(A)$ to the only object of $\mathscr{B}$. Hence $\mathscr{A} \, \mathrm{wr}^G \, \mathscr{B}$ has only one object, so is a monoid.

2. Let $\mathscr{A}$ and $\mathscr{B}$ be groups. Since they have only one object each, we can simplify the notation in WP{1 through WP{3 and omit mention of the objects A, A^0 and A^{00}. The value of G at the only object of $\mathscr{A}$ is a category we will call $\mathscr{G}$. If f is an element of the group $\mathscr{A}$, then Gf is an automorphism of the category $\mathscr{G}$. An object of $\mathscr{A} \, \mathrm{wr}^G \, \mathscr{B}$ is a functor $P : \mathscr{G} \;¡! \; \mathscr{B}$. An arrow $(f ; ¸) : P \;¡! \; P^0$ consists of an element f of the group $\mathscr{A}$ and a natural transformation $¸ : P \;¡! \; P^0 \pm Gf$. Each component of $¸$ is an element of the group $\mathscr{B}$ so has an inverse; thus $¸$ is

an invertible natural transformation. Since f is also a group element, it has an inverse f^{-1}. Let τ be the natural transformation whose component at an object X of $\mathscr{G}$ is $(\sigma(Gf)^{-1}(X))^{-1}$. Then the inverse of the arrow $(f; \sigma)$ is $(F^{-1}; \tau)$. To verify this, we calculate

$$(F^{-1}; \tau) \circ (f; \sigma) = (f^{-1} \circ f; \tau Gf \circ \sigma)$$

Now $F^{-1} \circ f$ is the identity of $\mathscr{A}$ and for an object X of $\mathscr{G}$,

$$
\begin{aligned}
(\tau Gf \circ \sigma)X &= \tau(Gf(X)) \circ \sigma X \\
&= \sigma((Gf)^{-1}(Gf(X)))^{-1} \circ \sigma X \\
&= (\sigma X)^{-1} \circ \sigma X = \mathrm{id}_X
\end{aligned}
$$

so $(f; \sigma)$ is invertible.

Solutions for Chapter 5

Section 5.1

1. Let the pullbacks be

Let $u : C_0 \to C_1$ and $v : C_1 \to C_0$ be the inverse isomorphisms such that $f_0 \circ v = f_1$ and $f_1 \circ u = f_0$. Then $f_1 \circ u \circ g_0 = f_0 \circ g_0 = g \circ f_0^0$ so by the universal mapping property of the right hand pullback, there is a unique $u^0 : C_0^0 \to C_1^0$ such that $g_1 \circ u^0 = f_1 \circ u$ and $f_1^0 \circ u^0 = f_0^0$. Similarly, there is a unique $v^0 : C_1^0 \to C_0^0$ such that $g_0 \circ v^0 = f_0 \circ v$ and $f_0^0 \circ v^0 = f_1^0$. Thus f_0^0 and f_1^0 belong to the same subobject.

2. For $f : C_0 \to C$, the square

is a pullback. Thus the function induced by the identity of C assigns to each subobject of C the subobject itself (or an equivalent one).

3. If S and T are ¯nite, so is the set of functions between them. In fact if we let #S denote the number of elements, then $\#[T \to S] = \#(S)^{\#(T)}$. For if $T = \;$, then there is exactly one function from T to S no matter what S is. If $T \ni \;$ and $S = \;$, then there are no functions from T to S. If both are nonempty, we suppose by induction that the conclusion is true for sets with fewer elements than T, let $t \in T$ and $T_0 = T \i$ ftg. A function from T to S is determined by a function from T_0 to S plus an element of S for t to go to. Thus the number of such functions is

$$\#(S)^{\#(T_0)}\#(S) = \#(S)^{\#(T)\i 1}\#(S) = \#(S)^{\#(T)}$$

which is still ¯nite. Thus the category of ¯nite sets is cartesian closed. The two-element set is also ¯nite so the category of ¯nite sets has a subset classi¯er.

4. It is not quite su±cient to point out that when S is in¯nite, the set 2^S of subsets of S is not countable. The point to be made is that $\mathrm{Hom}(1; 2^S) \cong \mathrm{Hom}(S; 2)$ (where here 2 is the set f0; 1g) and the latter set of functions is not countable, while there is no countable or ¯nite set T for which $\mathrm{Hom}(1; T)$ is not countable or ¯nite. Thus there is no ¯nite or countable set that has the universal mapping property required to be the powerset 2^S.

Section 5.2

1. In the category of sets, $- = 2$ so we must verify that

$$\mathrm{Hom}(B; 2^A) \cong \mathrm{Hom}(A \pounds B; 2) \cong \mathrm{Sub}(A \pounds B)$$

If $f : B \to 2^A$ is a function, let $\acute{A}(f) : A \pounds B \to 2$ by $\acute{A}(f)(a; b) = \acute{A}(b)(a)$. If $g : A \pounds B \to 2$ is a function, let $\tilde{A}(g) = f(a; b) \j g(a; b) = 1g \mu A \pounds B$. If $U \mu A \pounds B$ is a subset, then let

$$\frac{1}{4}(U)(b)(a) = \begin{cases} 1 & \text{if } (a; b) \in U \\ 0 & \text{if } (a; b) \notin U \end{cases}$$

It is immediate that $\frac{1}{2} \pm \tilde{A} \pm \acute{A} = \mathrm{id}_{\mathrm{Hom}(B;2^A)}$ and similarly for the other two serial composites, from which it is immediate that each of them is invertible, with inverse the composite of the other two.

2. For $x \in S$, let [x] denote the equivalence class containing it. Then $[x] = [y]$ if and only if $(x; y) \in E$. Let $d^0; d^1 : E \to S$ be the two arrows mentioned above and let $d : S \to S{=}E$ denote the arrow $x \mapsto [x]$. Then for $(x; y) \in E$, $d \pm d^0(x; y) = d(x) = [x] = [y] = d(y) = d \pm d^1(x; y)$ so that d coequalizes d^0 and d^1.

Now suppose that $e^0; e^1 : T \to S$ such that $d \pm e^0 = d \pm e^1$. Then for all $x \in T$, $[e^0(x)] = [e^1(x)]$; equivalently $(e^0(x); e^1(x)) \in E$. Thus we can let $f : T \to E$ by de¯ning $f(x) = (e^0(x); e^1(x))$. Then $d^0 \pm f(x) = d^0(e^0(x); e^1(x)) = e^0(x)$ and similarly $d^1 \pm f(x) = e^1(x)$. Finally if $g : T \to E$ is such that $d^0 \pm g(x) = d^0(e^0(x); e^1(x)) = e^0(x)$ and $d^1 \pm g(x) = e^1(x)$, then $g(x) = (e^0(x); e^1(x)) = f(x)$.

3. a. Suppose that d, e form an equivalence relation. This means that for any monoid N, the pair

$$\mathrm{Hom}(N;d), \mathrm{Hom}(N;e) : \mathrm{Hom}(N;E) \to \mathrm{Hom}(N;M)$$

gives an equivalence relation on the latter. This is another way of saying that

$$\langle \mathrm{Hom}(N;d), \mathrm{Hom}(N;e)\rangle : \mathrm{Hom}(N;E) \to \mathrm{Hom}(N;M) \times \mathrm{Hom}(N;M)$$

is an equivalence relation. Since

$$\mathrm{Hom}(N;M) \times \mathrm{Hom}(N;M) \cong \mathrm{Hom}(N; M \times M)$$

we have that $\langle d;e\rangle : E \to M \times M$ is monic. Since the image of a monoid homomorphism is a submonoid, the image is a submonoid. By letting $N = \mathbf{N}$, whence $\mathrm{Hom}(N;E)$ and $\mathrm{Hom}(N;M)$ are the underlying sets of E and M respectively, we conclude that the image of $\langle d;e\rangle$ is also an equivalence relation on the set underlying M. Thus it is a congruence.

Conversely, let $d;e : E \to M$ have the given property. Then up to isomorphism, we may suppose that $E \subseteq M \times M$ is both a submonoid and an equivalence relation and that d and e are the inclusion followed by the product projections. Then for any monoid N, $\mathrm{Hom}(N;E)$ consists of those pairs of arrows $f;g : N \to E$ such that for all $x \in N$, $(f(x);g(x)) \in E$. Since E is an equivalence relation, it is immediate that this makes $\mathrm{Hom}(N;E)$ into an equivalence relation on $\mathrm{Hom}(N;M)$.

b. Let us suppose that $E \subseteq M \times M$ is a submonoid and simultaneously an equivalence relation. For an $x \in M$, let [x] denote the equivalence class containing x. If $x' \in [x]$ and $y' \in [y]$, then $(x;x') \in E$ and $(y;y') \in E$, whence so is $(x;x')(y;y') = (xy;x'y')$. Thus $x'y' \in [xy]$ which means there an unambiguous multiplication defined on the set M/E of equivalence classes by $[x][y] = [xy]$. The associativity is obvious as is $[1][x] = [x][1] = [x]$. Thus M/E is a monoid and evidently the arrow $p : M \to M/E$, $x \mapsto [x]$ is a monoid homomorphism. $p(x) = p(x')$ if and only if $x' \in [x]$ if and only if $(x;x') \in E$ so E is the kernel pair of p.

4. Let f be the coequalizer of the two arrows $e^0;e^1 : E \to C$ and let the kernel pair be $d^0;d^1 : K \to C$. Since $f \circ e^0 = f \circ e^1$, it follows from the universal mapping property of kernel pairs that there is a unique arrow $e : E \to K$ such that $d^0 \circ e = e^0$ and $d^1 \circ e = e^1$. Now $d^0 \circ f = d^1 \circ f$ is one of the defining properties of kernel pairs. If $g : C \to B$ is an arrow such that $g \circ d^0 = g \circ d^1$, then $g \circ e^0 = g \circ d^0 \circ e = g \circ d^1 \circ e = g \circ e^1$. Since f is the coequalizer of e^0 and e^1, there is a unique $h : D \to B$ such that $h \circ f = g$.

5. Let $S = \{a\}$, $T = \{a;b\}$ and $f : S \to T$ be the inclusion. The kernel pair d^0, d^1 of f is the equality relation on S, that is $d^0 = d^1$, but f is not the coequalizer.

For any set U with more than one element and any function $g : S \to U$ there are many functions $h : T \to U$ such that $h \circ f = g$, since $g(b)$ can be any element of U.

6. Let T be a set, $T_0 \subseteq T$ a subset and $f : T_0 \to S$ be an arbitrary function. Define $\bar{f} : T \to S \cup \{*\}$ by

$$\bar{f}(x) = \begin{cases} f(x) & \text{if } x \in T_0 \\ * & \text{otherwise} \end{cases}$$

Then $T_0 = \{x \in T \mid \bar{f}(x) \in S\}$, which is another way of saying that

$$
\begin{array}{ccc}
T_0 & \xrightarrow{\quad f \quad} & S \\
\downarrow{\scriptstyle ?} & & \downarrow{\scriptstyle ?} \\
T & \xrightarrow[\bar{f}]{\quad} & S \cup \{*\}
\end{array}
$$
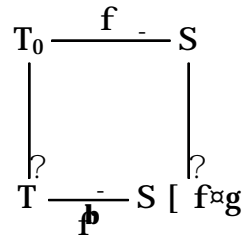
is a pullback. If g is a function with the same property, then $g(x) \in S$ if and only if $\bar{f}(x)$ is, which means that $g(x) = *$ if and only if $\bar{f}(x) = *$. Also, if $g(x) \neq *$, then $x \in T_0$ so that $g(x) = f(x) = \bar{f}(x)$. Thus $g = \bar{f}$ in all cases.

7. Since each object has a unique arrow to 1, there is a one to one correspondence between subobjects of an object A and partial arrows of A to 1. But partial arrows of A to 1 are in one to one correspondence with arrows $A \to \tilde{1}$. Thus $\text{Hom}(A; \tilde{1}) \cong \text{Sub}(A)$ which is the defining property of $-$.

Section 5.4

1. We are thinking of $\mathscr{C} = 0 \rightrightarrows 1$ as a category and 0 and 1 as two objects. The category of graphs is the category of contravariant functors $\mathscr{C} \to$ **Set**. In particular, the objects 0 and 1 represent functors and the question is which ones. Let us denote the two nonidentity arrows of $\mathscr{C}$ by s and t (mnemonic for `source' and `target'). A functor $F : \mathscr{C} \to$ **Set** determines and is determined by the two sets $F(1)$ and $F(0)$ and the two arrows $F(s) : F(1) \to F(0)$ and $f(t) : F(1) \to F(0)$. In the case that $F = \text{Hom}(\_ ; 0)$, the two sets are $\text{Hom}(0; 0)$ and $\text{Hom}(1; 0)$ which are a one-element set and the empty set, respectively, and the functions are the unique functions that exists in that case. Thus the contravariant functor represented by 0 is the graph **1** which we have also called No.

For take the functor represented by 1, the two sets are $\text{Hom}(1; 1) = \{\text{id}_1\}$ and $\text{Hom}(0; 1) = \{s, t\}$. The two arrows take the element of $\text{Hom}(1; 1)$ to s and t respectively. The graph **2** that we have also called No has just one arrow and two nodes which are the source and target of the arrow, respectively, and so is the graph represented by 1.

2. There are two ways of dealing with this question and similar ones. We already have an explicit description of the subobject classi¯er in this category and we could simply examine the set of nodes and set of arrows and see that their subfunctors have the desired structure. However, there is an easier way, once we know there is a subobject classi¯er. For the set of subobjects of the graph No is Hom(No; -) which, in turn, is isomorphic by the preceding problem, to the set of nodes of - . Similarly, the set of subobjects of the graph Ar is Hom(Ar; -), which is the set of arrows of - .

3. Let $NT(F; G)$ denote the set of natural transformations between functors F and G. The Yoneda lemma implies that if $[M \mathbin{\text{¡!}} N] = F$, then

$$F(X) \cong NT(Hom(\text{¡}\ ; X); F) \cong NT(Hom(\text{¡}\ ; X); [M \mathbin{\text{¡!}} N])$$
$$\cong NT(Hom(\text{¡}\ ; X) \pounds M; N)$$

4. We have that

$$Sub(Hom(\text{¡}\ ; X)) \cong NT(Hom(\text{¡}\ ; X); -) \cong - (X)$$

the latter by the Yoneda Lemma.

Section 5.5

1. a. Let $C = \text{fc} \mathbin{2} \mathscr{H} \text{ j a} \wedge \text{c} \cdot \text{bg}$. Then $a \Rightarrow b = \bigvee_{c2C} c$. Then

$$a \wedge (a \Rightarrow b) = a \wedge \bigvee_{c2C} c = \bigvee_{c2C} (a \wedge c) \cdot b$$

since a join of elements less than b is also less than b. Therefore, if $c \cdot a \Rightarrow b$ then $a \wedge c \cdot b$. On the other hand, if $a \wedge c \cdot b$, then $c \mathbin{2} C$ so that $c \cdot a \Rightarrow b$.

  b. The fact that $a \cdot b \wedge c$ if and only if $a \cdot b$ and $a \cdot c$ means that when the Heyting algebra is considered as a category, $Hom(a; b \wedge c) = Hom(a; b) \pounds Hom(a; c)$. This means that $b \wedge c$ is the categorical product of b and c. But then $a \wedge c \cdot b$ if and only if $c \cdot a \Rightarrow b$ means that $Hom(a \pounds c; b) = Hom(c; a \Rightarrow b)$, which means that $a \Rightarrow b$ is the internal hom.

Bibliography

At the end of each entry, the pages on which that entry is cited are listed in parentheses.

Adámek, J. and J. Rosicky (1994). Locally Presentable and Accessible Categories. Cambridge University Press. (12, 13, 25)

Asperti, A. and S. Martini (1992). `Categorical models of polymorphism'. Information and Computation, volume **99**, pages 1{79. (43)

Bagchi, A. and C. Wells (1994). `Graph-based logic and sketches I: The general framework'. Available by gopher from the site gopher.cwru.edu, path=/class/mans/math/pub/wells/. (24, 25)

Barr, M. and C. Wells (1985). Toposes, Triples and Theories, volume 278 of Grundlehren der mathematischen Wissenschaften. Springer-Verlag, New York. A list of corrections and additions is maintained in [Barr and Wells, 1993]. (16, 20, 22, 24, 25, 57, 59, 60, 61, 62, 66, 68)

Barr, M. and C. Wells (1994). `The categorical theory generated by a limit sketch'. Available by gopher from the site gopher.cwru.edu, path=/class/mans/math/pub/wells/. (20, 25)

Barr, M., C. McLarty, and C. Wells (1985). `Variable set theory'. Technical report, McGill University. (72)

Barr, M. (1986). `Fuzzy sets and topos theory'. Canadian Math. Bull., volume **24**, pages 501{508. (72)

Barr, M. (1986). `Models of sketches'. Cahiers de Topologie et Géométrie Différentielle Catégorique, volume **27**, pages 93{107. (17)

Barr, M. (1989). `Models of Horn theories'. Contemporary Mathematics, volume **92**, pages 1{7. (12)

Bastiani, A. and C. Ehresmann (1972). `Categories of sketched structures'. Cahiers de Topologie et Géométrie Différentielle, volume **13**, pages 104{213. (20, 25)

Bell, J. (1988). Toposes and Local Set Theories: An Introduction. Oxford Logic Guides. Oxford University Press. (57)

Boileau, A. and A. Joyal (1981). `La logique des topos'. Symbolic Logic, volume **46**, pages 6{16. (57)

Carboni, A., P. Freyd, and A. Scedrov (1988). `A categorical approach to realizability and polymorphic types'. In Mathematical Foundations of Programming Language Semantics, M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, volume 298 of Lecture Notes in Computer Science, pages 23{42. Springer-Verlag. (79)

Cartmell, J. (1986). `Generalized algebraic theories and contextual categories'. Annals Pure Applied Logic, volume **32**, pages 209{243. (13)

Cockett, J. R. B. and D. Spencer (1992). `Strong categorical datatypes i'. In Category Theory 1991, R. Seely, editor. American Mathematical Society. (41)

98   Bibliography

Coquand, T. and G. Huet (1988). `The calculus of constructions'. Information and Computation, volume **76**, pages 95{120.   (43)

Coquand, T. (1988). `Categories of embeddings'. In Proceedings of the Third Annual Symposium on Logic in Computer Science, Edinburgh, 1988, pages 256{263. Computer Society Press and IEEE.   (43)

Coste, M. (1976). `Une approche logique des théories définissables par limites projectives finies'. In Séminaire Bénabou. Université Paris-Nord.   (13)

Duval, D. and J.-C. Reynaud (1994). `Sketches and computation (1)'. Mathematical Structures in Computer Science, volume **4**.   (9)

Duval, D. and J.-C. Reynaud (1994). `Sketches and computation (2)'. Mathematical Structures in Computer Science, volume **4**.   (9)

Ehresmann, C. (1968). `Esquisses et types des structures algébriques'. Bul. Inst. Polit. Iaşi, volume **XIV**.   (20, 25)

Ehrhard, T. (1988). `A categorical semantics of constructions'. In Proceedings of the Third Annual Symposium on Logic in Computer Science, Edinburgh, 1988. Computer Society Press and IEEE.   (43)

Ehrig, H., H. Kreowski, J. Thatcher, E. Wagner, and J. Wright (1984). `Parameter passing in algebraic specification languages'. Theoretical Computer Science, volume **28**, pages 45{81.   (31)

Eilenberg, S. (1976). Automata, Languages and Machines, Vol. B. Academic Press.   (55)

Fourman, M. and S. Vickers (1986). `Theories as categories'. In Category Theory and Computer Programming, D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard, editors, volume 240 of Lecture Notes in Computer Science, pages 434{448. Springer-Verlag.   (57)

Fourman, M. (1977). `The logic of topoi'. In Handbook of Mathematical Logic, J. Barwise et al., editors. North-Holland.   (57)

Freyd, P. and A. Scedrov (1990). Categories, Allegories, volume 39 of North-Holland Mathematical Library. North-Holland.   (57)

Gabriel, P. and F. Ulmer (1971). Lokal Präsentierbare Kategorien, volume 221 of Lecture Notes in Mathematics. Springer-Verlag.   (13)

Goguen, J. A. and R. M. Burstall (1986). `A study in the foundations of programming methodology: Specifications, institutions, charters and parchments'. In Category Theory and Computer Programming, D. Pitt, S. Abramsky, A. Poigné, and D. Rydeheard, editors, volume 240 of Lecture Notes in Computer Science, pages 313{333. Springer-Verlag.   (36)

Goguen, J. A. (1974). `Concept representation in natural and artificial languages: Axioms, extensions and applications for fuzzy sets'. Int. J. Man-machine Studies, volume **6**, pages 513{564.   (73)

Gray, J. W. (1966). `Fibred and cofibred categories'. In Proc. La Jolla Conference on Categorical Algebra, S. Eilenberg et al., editors. Springer-Verlag.   (51)

Gray, J. W. (1989). `The category of sketches as a model for algebraic semantics'. In Categories in Computer Science and Logic, J. W. Gray and A. Scedrov, editors, volume 92 of Contemporary Mathematics, pages 109{135. American Mathematical Society.   (26, 31)

Gray, J. W. (1991). `Products in PER: an elementary treatment of the semantics of the polymorphic lambda calculus'. In Category Theory at Work (Bremen, 1990), volume 18 of Res. Exp. Math., pages 325{340. Heldermann.   (79)

Guitart, R. and C. Lair (1981). `Existence de diagrammes localement libres 1'. Diagrammes, volume **6**.   (22)

Guitart, R. and C. Lair (1982). `Existence de diagrammes localement libres 2'. Diagrammes, volume **7**.   (22)

Hyland, J. and A. Pitts (1989). `The theory of constructions: Categorical semantics and topos-theoretic models'. In Categories in Computer Science and Logic, J. Gray and A. Scedrov, editors, volume 92 of Contemporary Mathematics, pages 137{199. American Mathematical Society.   (43, 57)

Hyland, J. (1982). `The e®ective topos'. In The L.E.J. Brouwer Centenary Symposium, pages 165{216. North-Holland.   (57, 80)

Johnstone, P. T. and R. Paré (1978). Indexed Categories and their Applications. Lecture Notes in Mathematics. Springer-Verlag.   (41)

Johnstone, P. T. (1977). Topos Theory. Academic Press.   (57, 60, 61, 63, 68)

Kelly, G. (1974). `On clubs and doctrines'. In Proceedings Sydney Category Theory Seminar 1972/1973, G. Kelly, editor, volume 420 of Lecture Notes in Mathematics. Springer-Verlag.   (54)

Kelly, G. (1982). `On the essentially-algebraic theory generated by a sketch'. Bulletin of the Australian Mathematical Society, volume **26**, pages 45{56.   (25)

Lair, C. (1987). `Trames et semantiques catégoriques des systàmes de trames'. Diagrammes, volume **18**.   (25)

Lambek, J. and P. Scott (1986). Introduction to Higher Order Categorical Logic, volume 7 of Cambridge Studies in Advanced Mathematics. Cambridge University Press.   (57)

Lawvere, F. (1966). `The category of categories as a foundations for mathematics'. In Proceedings of the Conference on Categorial Algebra at La Jolla. Springer-Verlag.   (16)

Lewis, H. R. and C. H. Papadimitriou (1981). Elements of the Theory of Computation. Prentice-Hall.   (56)

Mac Lane, S. and I. Moerdijk (1992). Sheaves in Geometry and Logic. Universitext. Springer-Verlag.   (57, 67, 68)

Makkai, M. and R. Paré (1990). Accessible Categories: the Foundations of Categorical Model Theory, volume 104 of Contemporary Mathematics. American Mathematical Society.   (25)

Makkai, M. and G. Reyes (1977). First Order Categorical Logic, volume 611 of Lecture Notes in Mathematics. Springer-Verlag.   (57)

Makkai, M. (1994). `Generalized sketches as a framework for completeness theorems'. To appear in Journal of Pure and Applied Algebra.   (25)

McLarty, C. (1986). `Left exact logic'. J. Pure Applied Algebra, volume **41**, pages 63{66.   (13, 37)

McLarty, C. (1992). Elementary Categories, Elementary Toposes, volume 21 of Oxford Logic Guides. Clarendon Press.   (57, 60, 61, 63, 68)

Mikkelson, C. (1976). Lattice Theoretic and Logical Aspects of Elementary Topoi, volume 25 of Aarhus University Various Publications Series. Aarhus University.   (62)

Nico, W. (1983). `Wreath products and extensions'. Houston J. Math., volume **9**, pages 71{99.   (51, 55)

Peake, E. J. and G. R. Peters (1972). `Extension of algebraic theories'. Proceedings of the American Mathematical Society, volume **32**, pages 358{362.   (25)

Pitts, A. (1982). `Fuzzy sets do not form a topos'. Fuzzy Sets and Systems, volume **8**, pages 101{104.   (72)

Power, A. and C. Wells (1992). `A formalism for the speciˉcation of essentially algebraic structures in 2-categories'. Mathematical Structures in Computer Science, volume **2**, pages 1{28.   (25)

Reichel, H. (1987). Initial Computability, Algebraic Speciˉcations and Partial Algebras. Clarendon Press.   (13)

Rhodes, J. and B. Tilson (1989). `The kernel of monoid homomorphisms'. J. Pure Applied Algebra, volume **62**, pages 227{268.   (53, 55)

Rhodes, J. and P. Weil (1989). `Decomposition techniques for ˉnite semigroups i, ii'. J. Pure Applied Algebra, volume **62**, pages 269{284.   (55)

Rosebrugh, R. and R. Wood (1992). `Relational databases and indexed categories'. In Category Theory 1991, R. Seely, editor. American Mathematical Society.   (41)

Rosolini, G. (1987). `Categories and e®ective computation'. In Category Theory and Computer Science, volume 283 of Lecture Notes in Computer Science. Springer-Verlag.   (80)

Rosolini, G. (1990). `About modest sets.'. International Journal of Foundations of Computer Science, volume **3**, pages 341{353.   (79)

Street, R. and R. Walters (1973). `The comprehensive factorization of a functor'. Bull. Amer. Math. Soc., volume **79**, pages 936{941.   (55)

Tarlecki, A., R. M. Burstall, and J. A. Goguen (1991). `Some fundamental algebraic tools for the semantics of computation III: Indexed categories'. Theoretical Computer Science, volume **91**, pages 239{264.   (41)

Taylor, P. (1989). `Quantitative domains, groupoids and linear logic'. In Category Theory and Computer Science, D. Pitt, D. Rydeheard, P. Dybjer, A. Pitts, and A. Poignē, editors, volume 389 of Lecture Notes in Computer Science. Springer-Verlag.   (66)

Thatcher, J., E. Wagner, and J. Wright (1982). `Data type speciˉcation: Parametrization and the power of speciˉcation techniques'. ACM Transactions of Programming Languages and Systems, volume **4**, pages 711{732.   (31)

Vickers, S. (1992). `Geometric theories and databases'. In Applications of Categories in Computer Science (Durham, 1991), volume 177 of London Math. Soc. Lecture Note Series, pages 288{314. Cambridge Univ. Press.   (57)

Volger, H. (1987). `On theories which admit initial structures'. Technical report, Universität Passau.   (17)

Volger, H. (1988). `Model theory of deductive databases'. In CSL '87. First Workshop on Computer Science Logic. Proceedings, volume 329 of Lecture Notes in Computer Science. Springer-Verlag.   (17)

Wagner, E. (1986). `Categories, data types and imperative languages'. In Category Theory and Computer Programming, D. Pitt, S. Abramsky, A. Poignē, and D. Rydeheard, editors, volume 240 of Lecture Notes in Computer Science, pages 143{162. Springer-Verlag.   (31)

Wells, C. (1976). `Some applications of the wreath product construction'. Amer. Math. Monthly, volume **83**, pages 317{338.   (55)

Wells, C. (1980). `A Krohn-Rhodes theorem for categories'. J. Algebra, volume **64**, pages 37{45.   (55)

Wells, C. (1988a). `Wreath product decomposition of categories I'. Acta Sci. Math. Szeged, volume **52**, pages 307{319.   (55)

Wells, C. (1988b). `Wreath product decomposition of categories II'. Acta Sci. Math. Szeged, volume **52**, pages 321{324.   (55)

Wells, C. (1990). `A generalization of the concept of sketch'. Theoretical Computer Science, volume **70**, pages 159{178.   (24, 25)

Index