Category Theory
Definitions and Examples

Yellamraju V. Srinivas
Department of Information and Computer Science
University of California, Irvine, USA
srinivas@ics.uci.edu

Technical Report 90-14
18 February 1990

Abstract

Category theory was invented as an abstract language for describing certain structures and constructions which repeatedly occur in many branches of mathematics, such as topology, algebra, and logic. In recent years, it has found several applications in computer science, e.g., algebraic specification, type theory, and programming language semantics. In this paper, we collect definitions and examples of the basic concepts in category theory: categories, functors, natural transformations, universal properties, limits, and adjoints.

Contents

Introduction

Category theory was invented as an abstract language for describing certain structures and constructions which repeatedly occur in many branches of mathematics, such as topology, algebra, and logic. As opposed to set theory which is based on the membership relation and thus leads to the study of the internal structure of abstract entities, category theory takes morphisms or arrows as fundamental. Thus, in category theory, one studies the external properties of objects. To define an object, it is necessary and sufficient to describe its interaction (via morphisms) with all other objects.

As an example, consider the definition of the Cartesian product of two sets. In set theory, the product is defined by specifying its internal structure: the product of two sets is a set consisting of ordered pairs as elements, with the first component of each pair coming from one of the sets and the second component from the other. In category theory, we concentrate on the external properties of products: a product comes equipped with two special functions, projections, which map ordered pairs onto their first and second components. The latter definition, generalized by replacing sets by objects and functions by arrows, uniformly describes products of sets, functions, graphs, data types, groups, topological spaces, and so forth.

The use of category theory results in parsimonious descriptions of entities, because of its abstract nature, and its focus on essential external properties. In recent years, it has found several applications in computer science, e.g., algebraic specification, type theory, programming language semantics, graph rewriting, automata theory, and even abstract machines based on categorial primitives.

In this paper, we provide a rapid introduction to category theory. For a more detailed description, we refer the reader to the literature. A computer science oriented introduction to category theory is provided by Rydeheard and Burstall [RB88], Pierce [Pie88], and Pitt et al. [PAPR85]. Other, very readable, introductions are by Goldblatt [Gol84, Chapters 2 and 3], and Herrlich and Strecker [HS73]. The book by Mac Lane, one of the founders of category theory, is an uncluttered, succinct, and precise treatment [Mac71]. Schubert's book [Sch72] is terse but comprehensive, and thus suitable as a reference book. Goguen [Gog88] introduces concepts from category theory while addressing substitution and unification. The series of papers by Goguen et al. [GB84a, GB84b, TBG88] is a description of concepts especially useful for understanding the theory underlying algebraic specification. Lambek and Scott [LS86] provide a brief introduction to category theory while exploring the relationship between category theory and logic.

1 Categories: definition and examples

A category is an abstract mathematical structure defined axiomatically as shown below. The definition is taken from [Gol84].

DEFINITION 1.1: *Category.* A category $\mathcal{C}$ comprises

1. a collection of things called *$\mathcal{C}$-objects;*
2. a collection of things called *$\mathcal{C}$-arrows;*
3. operations assigning to each $\mathcal{C}$-arrow f a $\mathcal{C}$-object $\operatorname{dom} f$ (the "domain" of f) and a $\mathcal{C}$-object $\operatorname{cod} f$ (the "codomain" of f). If $a = \operatorname{dom} f$ and $b = \operatorname{cod} f$ we display this as
$$f: a \to b \quad \text{or} \quad a \xrightarrow{f} b$$
4. an operation, "$\circ$", called *composition,* assigning to each pair $\langle g, f \rangle$ of $\mathcal{C}$-arrows with $\operatorname{dom} g = \operatorname{cod} f$, a $\mathcal{C}$-arrow $g \circ f: \operatorname{dom} f \to \operatorname{cod} g$, the *composite of f and g*, such we have the

   Associative Law: Given the configuration
   $$a \xrightarrow{f} b \xrightarrow{g} c \xrightarrow{h} d$$
   of $\mathcal{C}$-objects and $\mathcal{C}$-arrows, we have
   $$h \circ (g \circ f) = (h \circ g) \circ f.$$
5. an assignment to each $\mathcal{C}$-object b of a $\mathcal{C}$-arrow $\mathrm{id}_b: b \to b$, called *the identity arrow on b,* such that we have the

   Identity Law: For any $\mathcal{C}$-arrows $f: a \to b$ and $g: b \to c$
   $$\mathrm{id}_b \circ f = f \quad \text{and} \quad g \circ \mathrm{id}_b = g.$$

□

Notation. We drop the prefix "$\mathcal{C}$-" when referring to the objects and arrows of a specific category whose name is clear from the context. The notation for composition follows usage in mathematics: $g \circ f$ means that f is applied first, then g. Some authors prefer to write the composite in diagrammatic order (left to right), as fg or as $f; g$.

EXAMPLE 1.2: *Trivial examples.* Trivial examples of categories include the category **0** with no objects and no arrows, the category **1** with exactly one object and the identity arrow, and the category **2** with two objects and one non-identity arrow: $\bullet \to \bullet$.

Any set can be made into a *discrete* category by treating each element as an object with an identity arrow, there being no other (non-identity) arrows. □

EXAMPLE 1.3: *The category of sets and functions.* The prototypical example of a category is the category **Set** whose objects are all sets and whose arrows are all functions between sets.[1] Composition is the familiar composition of functions defined as $(g \circ f)(x) = g(f(x))$. Associated with each set A is an identity function id_A, defined as $\mathrm{id}_A(x) = x$. Evidently, the associative and identity laws are satisfied. □

EXAMPLE 1.4: *Pre-orders as categories.* A binary relation R on a set P (i.e., $R \subseteq P \times P$) is called a *pre-order* if it is reflexive (i.e., for each $p \in P$ we have $p\ R\ p$) and transitive (i.e., whenever $p\ R\ q$ and $q\ R\ r$, we have $p\ R\ r$). Such a pre-order can be considered to be a category as follows. The objects are the elements of the set P. The arrows are the pairs $\langle p, q\rangle$ for which $p\ R\ q$, with $\mathrm{dom}\langle p, q\rangle = p$ and $\mathrm{cod}\langle p, q\rangle = q$. Given a composable pair of arrows

$$p \xrightarrow{\langle p,q\rangle} q \xrightarrow{\langle q,r\rangle} r,$$

we define

$$\langle q, r\rangle \circ \langle p, q\rangle = \langle p, r\rangle.$$

The arrow $\langle p, r\rangle$ exists because the relation R is transitive. Since R is reflexive, the arrow $\langle p, p\rangle$ always exists. We define $\mathrm{id}_p = \langle p, p\rangle$. The associative and identity laws are satisfied by virtue of transitivity. □

EXAMPLE 1.5: *Graphs and graph morphisms.* A *graph* (unlabelled, directed, multigraph) is a 4-tuple $\langle N, E, s, t\rangle$ of two sets called nodes (N) and edges (E) and two functions called source (s) and target (t) assigning nodes to edges. The nodes and edges of a graph G are denoted by $N(G)$ and $E(G)$, respectively.

A morphism of graphs $f: G \to H$ is a pair of functions

$$\langle f_N: N(G) \to N(H), f_E: E(G) \to E(H)\rangle,$$

mapping nodes to nodes and edges to edges, such that the pair of functions is compatible with the source and target maps, i.e., for all edges $e \in E(G)$

$$s_H(f_E(e)) = f_N(s_G(e))$$
$$\text{and } t_H(f_E(e)) = f_N(t_G(e)).$$

Graphs and graph morphisms form a category, **Graph**, with composition and identities the evident extensions from the category of sets. □

[1] Strictly speaking, the arrows in this category are triples $\langle a, f, b\rangle$ consisting of a function along with its domain and codomain. This is so that all the arrows in the category are distinct, and to differentiate between inclusion functions, $a \hookrightarrow b$, and identity functions, id_b.

EXAMPLE 1.6: *Signatures and signature morphisms.* A signature $\Sigma = \langle S, \Omega \rangle$, consists of a set S of *sorts* and a set Ω of *operation symbols.* Associated with each operation symbol is a sequence of sorts called its *rank.* For example, $f: s_1, s_2, \ldots, s_n \rightarrow s$ indicates that f is the name of an n-ary function, taking arguments of sorts $s_1, s_2, \ldots, s_n$ and producing a result of sort s. A nullary operation symbol, $c: \rightarrow s$, is called a *constant* of sort s.

Given two signatures $\Sigma = \langle S, \Omega \rangle$ and $\Sigma' = \langle S', \Omega' \rangle$, a signature morphism $\sigma: \Sigma \rightarrow \Sigma'$ is a pair of functions $\langle \sigma_S: S \rightarrow S', \sigma_\Omega: \Omega \rightarrow \Omega' \rangle$, mapping sorts to sorts and operations to operations, such that the sort map is compatible with the ranks of the operations, i.e.,

for all operation symbols $f: s_1, s_2, \ldots, s_n \rightarrow s$ in Ω,

the operation symbol $\sigma_\Omega(f): \sigma_S(s_1), \sigma_S(s_2), \ldots, \sigma_S(s_n) \rightarrow \sigma_S(s)$ is in Ω'.

The composition of two signature morphisms, obtained by composing the functions comprising the signature morphisms, is also a signature morphism. The identity signature morphism on a signature maps each sort and each operation onto itself. Signatures and signature morphisms form a category, **Sign.** □

EXAMPLE 1.7: *Σ-algebras and Σ-homomorphisms.* Given a signature $\Sigma = \langle S, \Omega \rangle$, a Σ-algebra $A = \langle A_S, F_A \rangle$ consists of two families:

1. a collection of sets, called the *carriers* of the algebra, $A_S = \{ A_s \mid s \in S \}$; and
2. a collection of (total) functions, $F_A = \{ f_A \mid f \in \Omega \}$, such that, if the rank of f is $s_1, s_2, \ldots, s_n \rightarrow s$, then f_A is a function from $A_{s_1} \times A_{s_2} \times \cdots \times A_{s_n}$ to A_s. The symbol $\times$ indicates the cartesian product of sets here.

Given a signature $\Sigma = \langle S, \Omega \rangle$ and two Σ-algebras A and B, a Σ-homomorphism $h: A \rightarrow B$ is a family of functions $\{ h_s: A_s \rightarrow B_s \mid s \in S \}$ between the carriers which are compatible with the operations, i.e., for all operation symbols $f: s_1, s_2, \ldots, s_n \rightarrow s$, and for all $a_1 \in A_{s_1}, a_2 \in A_{s_2}, \ldots, a_n \in A_{s_n}$,

$$h_s(f_A(a_1, a_2, \ldots, a_n)) = f_B(h_{s_1}(a_1), h_{s_2}(a_2), \ldots, h_{s_n}(a_n)).$$

The identity homomorphism maps an algebra onto itself. The composition of two homomorphisms is also a homomorphism. Σ-algebras and Σ-homomorphisms form a category, denoted $\mathbf{Alg}(\Sigma)$. □

EXAMPLE 1.8: *Sets with structure.* The previous example is an instance of a wide variety of categories found in mathematics: the objects are sets with structure, and the arrows are structure-preserving maps. Algebraic structures such as monoids, groups, rings, etc., along with homomorphisms form categories, with composition and identities being the evident ones. Similarly, partial orders and monotone maps, and topological spaces and continuous maps, form categories. □

EXAMPLE 1.9: *Types and terms.* We now define a category in which the arrows are a little unusual. Let $\Sigma = \langle S, \Omega \rangle$ be a signature. Let $X = \{ X_s \mid s \in S \}$ and $Y = \{ Y_s \mid s \in S \}$ be sets of variables indexed by the sort set S. Let $T_\Sigma(X)$ be the set of terms generated from Σ using variables in X (see examples 2.10 and 8.3 for a precise definition). A *substitution* of the variables in Y with terms generated using the variables X, is a map[2] $\sigma: Y \to T_\Sigma(X)$.

We can form a category, **Terms**(Σ), using these ingredients as follows. The objects are sorted sets of variables. An arrow from the object X to the object Y is a substitution $\sigma: Y \to T_\Sigma(X)$.[3] Identity arrows are trivial substitutions which replace variables by themselves.

To define composition of arrows, we need the following fact: a substitution $\sigma: Y \to T_\Sigma(X)$ can be uniquely extended to a function $\bar{\sigma}: T_\Sigma(Y) \to T_\Sigma(X)$.[4] Now, the composition of the arrows $\sigma: X \to Y$ and $\gamma: Y \to Z$ (with corresponding substitutions $\sigma: Y \to T_\Sigma(X)$ and $\gamma: Z \to T_\Sigma(Y)$) is defined as the arrow $\gamma \circ \sigma: X \to Z$ arising from the substitution $\bar{\sigma} \circ \gamma: Z \to T_\Sigma(X)$. The associativity of this operation is proved in [Gog88]. □

Remarks

These examples show the diversity of the concepts "object" and "arrow". The axioms for categories are quite weak, in the sense that a large number of mathematical structures satisfy them. Much of the utility of category theory comes from defining more complex constructions using arrows. Properties determined and proved in the general setting then specialize to any category in which the construction is possible.

2 Using arrows

As an example of the bias of category theory towards defining concepts using arrows, we define the concept of isomorphism (compare with the definition of bijective functions on sets).

DEFINITION 2.1: *Isomorphism.* An arrow $f: a \to b$ in a category $\mathcal{C}$ is said to be an isomorphism if there is a $\mathcal{C}$-arrow $g: b \to a$ such that

$$g \circ f = \mathrm{id}_a \quad \text{and} \quad f \circ g = \mathrm{id}_b .$$

□

The arrow g defined above is unique and is denoted by f^{-1}. Two objects a and b are said to be isomorphic, denoted $a \cong b$, if there exists an isomorphism, $f: a \to b$, between them. One of the themes in category theory is that isomorphic objects are "abstractly the same". At least, they cannot be distinguished within the theory. Thus, all the constructions outlined below are only defined up to isomorphism.

EXAMPLE 2.2. Isomorphisms in **Set** are bijective maps, i.e., isomorphic objects have the same cardinal number. □

[2] Actually an indexed collection of maps.

[3] Note the change in direction. Arrows in categories are formal entities; their direction is relevant only to the extent that it defines the domain and codomain.

[4] This follows from the freeness of $T_\Sigma(X)$, see example 8.3 below.

EXAMPLE 2.3. Isomorphisms in **Sign** are bijective renamings of sorts and operations. Isomorphic signatures are abstractly the same because what matters is the structure of a signature, and not the particular names attached to sorts and operations. □

Here are more instances of "arrows only" definitions, the categorial versions of injective (one-one) and surjective (onto) functions.

DEFINITION 2.4: *Monomorphism.* An arrow $f: a \to b$ in a category $\mathcal{C}$ is said to be a monomorphism (or simply, monic) if for any parallel pair g, h

$$c \underset{h}{\overset{g}{\rightrightarrows}} a \xrightarrow{f} b$$

of $\mathcal{C}$-arrows, the equality $f \circ g = f \circ h$ implies $g = h$. A monic arrow is denoted by $f: a \rightarrowtail b$. □

DEFINITION 2.5: *Epimorphism.* An arrow $f: a \to b$ in a category $\mathcal{C}$ is said to be an epimorphism (or simply, epic) if for any parallel pair g, h

$$a \xrightarrow{f} b \underset{h}{\overset{g}{\rightrightarrows}} c$$

of $\mathcal{C}$-arrows, the equality $g \circ f = h \circ f$ implies $g = h$. An epic arrow is denoted by $f: a \twoheadrightarrow b$. □

An interesting special case of monic arrows is given by the inclusion arrows in **Set**: if $A \subseteq B$, then the inclusion function $A \hookrightarrow B$ sends each element in A to itself (in B). Generalizing this (and treating isomorphic objects as the same), we define a *sub-object* of an object b to be a monic $f: a \rightarrowtail b$. Note that a sub-object is really an arrow, and not an object.

We now turn to the definition of extremal objects in a category.

DEFINITION 2.6: *Initial object.* An object a is said to be initial in a category $\mathcal{C}$ if there is exactly one $\mathcal{C}$-arrow from a to every other object in the category. □

DEFINITION 2.7: *Terminal object.* An object a is said to be terminal (also called final) in a category $\mathcal{C}$ if there is exactly one $\mathcal{C}$-arrow to a from every other object in the category. □

Notation. We will show below (section 3) that all initial objects and all terminal objects in a category are isomorphic. This justifies the use of the definite article, "the", when referring to these objects. Initial and final objects are usually denoted by 0 and 1, respectively. Given an object c, the unique arrow from the initial object and the unique arrow to the terminal object are denoted by $0_c: 0 \to c$ and $1_c: c \to 1$, respectively. Sometimes, unique arrows are simply denoted by !, as in $!: 0 \to c$.

EXAMPLE 2.8. The category **Set** has the empty set $\emptyset$ as the initial object and any one-element set as a terminal object. □

EXAMPLE 2.9. In the category **Sign**, the initial object is the empty signature, containing no sorts and no operations. The category **Sign** does not have a terminal object. □

EXAMPLE 2.10. In the category $\mathbf{Alg}(\Sigma)$, the terminal objects are trivial algebras, algebras all of whose carriers are one-element sets, and all of whose functions are defined in the only way possible.

An initial object in $\mathbf{Alg}(\Sigma)$ is the *ground term algebra,* denoted by $\mathcal{T}_\Sigma$, and defined as below. The carriers of $\mathcal{T}_\Sigma$, denoted by T_Σ, form a family of sets of ground terms $\{ T_{\Sigma,s} \mid s \in S \}$ defined inductively as

1. if $c\colon \rightarrow s$ is a constant of sort s, then the term $c \in T_{\Sigma,s}$;
2. if $f\colon s_1, s_2, \ldots, s_n \rightarrow s$ is an operation of sort s, and $t_1, t_2, \ldots, t_n$ are terms in T_{Σ,s_1}, $T_{\Sigma,s_2}, \ldots, T_{\Sigma,s_n}$ respectively, then the term $f(t_1, t_2, \ldots, t_n) \in T_{\Sigma,s}$.

The operations of $\mathcal{T}_\Sigma$ are denoted by the same names as in Σ. They are defined by (assuming f as above)

$$f(t_1, t_2, \ldots, t_n) \mapsto f(t_1, t_2, \ldots, t_n)$$

i.e., each operation carries its arguments onto the corresponding term formed by its application to those arguments (note that the expression on the left above denotes function application, while the one on the right is just a formal term).

A proof of initiality of $\mathcal{T}_\Sigma$ can be found in [EM85, Theorem 3.3] or [GTW78, Theorem 2]. □

3 Duality

On closer examination, it can be seen that the definitions of monic and epic arrows, and initial and terminal objects, are very similar; the only difference is the direction of the arrows used in the definitions. This is an instance of a more general principle in category theory called *duality.*

DEFINITION 3.1: *Opposite category.* Given a category $\mathcal{C}$, the opposite category $\mathcal{C}^{\mathrm{op}}$, has the same objects and arrows as $\mathcal{C}$ except that the domain and codomain relations are interchanged, i.e., for every $\mathcal{C}$-arrow $f\colon a \rightarrow b$, we have a $\mathcal{C}^{\mathrm{op}}$-arrow $f^{\mathrm{op}}\colon b \rightarrow a$. Identity arrows remain unchanged, and the composition $g \circ f$ becomes the composition $f^{\mathrm{op}} \circ g^{\mathrm{op}}$. □

The axioms for categories are invariant under the changes described above, from which we have the following duality principle.

DEFINITION 3.2: *Principle of duality.* If a statement S is true in a category $\mathcal{C}$, then the dual statement S^*, obtained by interchanging domains and codomains of arrows and by substituting $f^{\mathrm{op}} \circ g^{\mathrm{op}}$ for $g \circ f$, is true in the opposite category $\mathcal{C}^{\mathrm{op}}$. Consequently, if a statement S is true in all categories, then the statement S^* is also true in all categories. □

For a precise statement of this principle, and a proof, see [Hat82]. In view of this, the opposite category is also called the dual category. We will illustrate the principle of duality using the following theorem.

THEOREM. Any two initial objects in a category are isomorphic.

PROOF. Let a and a' be two initial objects in a category $\mathcal{C}$. Since both a and a' are initial, there are unique arrows

$$f: a \to a' \quad \text{and} \quad g: a' \to a.$$

Now, $g \circ f$ is an arrow from a to a. But the only arrow from a to a is the identity arrow id_a (a being an initial object). Thus the two arrows must be equal, i.e., $g \circ f = \mathrm{id}_a$. Similarly, $f \circ g = \mathrm{id}_{a'}$. Hence f is an isomorphism and $a \cong a'$. □

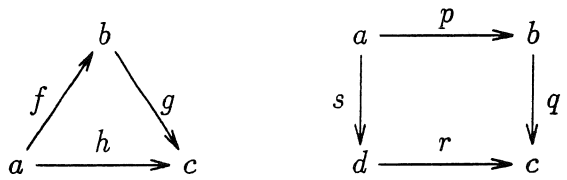
By the principle of duality, we can immediately conclude

THEOREM. Any two final objects in a category are isomorphic.

4 Commutative diagrams

In category theory, we frequently use diagrams to represent relationships between arrows. Strictly speaking, all such diagrams can be replaced by collections of equations which specify that certain pairs of arrows are equal. Diagrams have the advantage that domains and codomains of arrows can be read off at a glance. Moreover, representing composition by chaining together edges in a diagram allows fast[5] inferencing about equalities between arrows, an activity called "diagram chasing."

A diagram is a directed graph with vertices labelled by objects in a category and edges labelled by arrows in the same category. The source of an edge represents the domain of the arrow corresponding to the edge; the target represents the codomain. Such a diagram is said to be *commutative* if, for each pair of vertices v and v', any two paths formed by directed edges leading from v to v' yield, by composition of labels, equal arrows from v to v'. For example, the following two diagrams commute if $h = g \circ f$ and $q \circ p = r \circ s$.



An exception is diagrams in which there is more than one arrow between two objects, such as $a \rightrightarrows b$; these two arrows need not be equal, unless explicitly indicated.

In a commutative diagram, if an arrow is dashed, by convention, that arrow is expected to be uniquely defined.

5 Universal properties and constructions

A typical way of defining entities in category theory is that of specifying a so-called "universal" property.[6] Such a definition contains two parts. The first part specifies a property to

[5] After some practice, of course!

[6] This is an informal discussion of universal properties. A formal definition of "universal" needs more machinery, and can be found in section 8.

be satisfied by the entity. Of all the entities satisfying this property, the second part picks out a special one,[7] in some sense the entity which "minimally" satisfies the property. This special entity is said to be "universal" among the set of entities satisfying the property. As an example, we consider the definition of a least upperbound in a pre-order.

EXAMPLE 5.1: *Least upperbound.* Given a pre-order $\langle P, \leq \rangle$, i.e., a set P equipped with a reflexive and transitive relation $\leq$ on it, an *upperbound* of a subset $S \subseteq P$ is defined to be an element $u \in P$ such that

$$\forall s \in S \cdot s \leq u. \tag{i}$$

A *least upperbound* of S is an element $l \in P$ such that

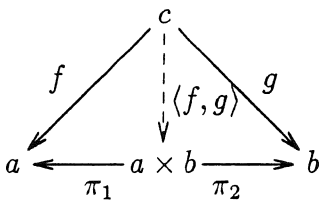
$$l \text{ is an upperbound, and, if } u \text{ is any other upperbound, then } l \leq u. \tag{ii}$$

Condition (i) specifies the property to be satisfied. Condition (ii) specifies "minimality". □

Using a universal property to describe an entity is a non-procedural way of defining that entity and is thus a powerful specification tool. Universal properties have been used to define parameterization, semantics of programming languages, unification of expressions, and minimal realizations of behaviour.

We now define some universal constructions[8] in category theory. Here are generalizations of cartesian product and disjoint union of sets. Note the "arrows only" feature of these definitions.

DEFINITION 5.2: *Product.* A product in a category $\mathcal{C}$ of two objects a and b is a $\mathcal{C}$-object $a \times b$ together with a pair of arrows $a \xleftarrow{\pi_1} a \times b \xrightarrow{\pi_2} b$ (called *projections*) such that to any other pair of arrows $a \xleftarrow{f} c \xrightarrow{g} b$, there is a unique arrow $\langle f, g \rangle : c \to a \times b$ such that the following diagram commutes,

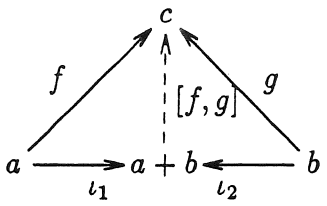


i.e., $\pi_1 \circ \langle f, g \rangle = f$ and $\pi_2 \circ \langle f, g \rangle = g$. □

[7] As usual, up to isomorphism.

[8] The term "construction" is a misnomer, although it is widely used. A universal construction is the specification of some object(s) or arrow(s) satisfying a universal property. The entities so defined are not "constructed"; they are already present in the category.

DEFINITION 5.3: *Coproduct.* A coproduct in a category $\mathcal{C}$ of two objects a and b is a $\mathcal{C}$-object $a + b$ together with a pair of arrows $a \xrightarrow{\iota_1} a + b \xleftarrow{\iota_2} b$ (called *injections*) such that to any other pair of arrows $a \xrightarrow{f} c \xleftarrow{g} b$, there is a unique arrow $[f, g]: a + b \to c$ such that the following diagram commutes,



i.e., $[f, g] \circ \iota_1 = f$ and $[f, g] \circ \iota_2 = g$. □

EXAMPLE 5.4. In **Set**, a product of two sets A and B is the set $A \times B$ consisting of ordered pairs $\langle x, y \rangle$ with $x \in A$ and $y \in B$. The projections are defined by $\pi_1(\langle x, y \rangle) = x$ and $\pi_2(\langle x, y \rangle) = y$. Products are only defined up to isomorphism; thus $B \times A$ is also a product of A and B.

A coproduct of A and B is the disjoint union of A and B, which can be defined as $(\{0\} \times A) \cup (\{1\} \times B)$, where each element is "labelled" to indicate the set it comes from. The injections are defined by $\iota_1(x) = \langle 0, x \rangle$ and $\iota_2(y) = \langle 1, y \rangle$. □

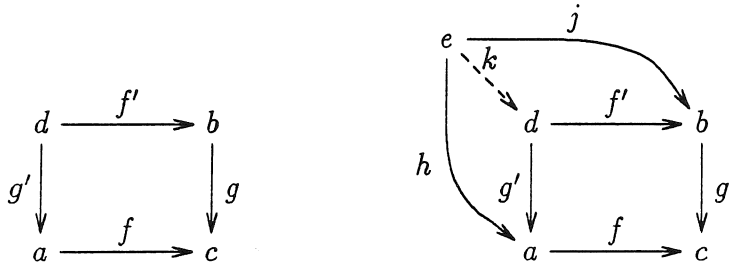
EXAMPLE 5.5. Products and coproducts in **Graph** can be constructed by extending the constructions in **Set**. For example, a product of the graphs G and H would consist of nodes $\langle a, b \rangle$ with $a \in N(G)$ and $b \in N(H)$, and edges $\langle a, b \rangle \xrightarrow{\langle p, q \rangle} \langle c, d \rangle$ with $a \xrightarrow{p} c \in E(G)$ and $b \xrightarrow{q} d \in E(H)$. □

EXAMPLE 5.6. In pre-orders regarded as categories, a least upperbound is a coproduct and a greatest lowerbound is a product. □

EXAMPLE 5.7. Products in the category $\mathbf{Terms}(\Sigma)$ are more like coproducts in **Set** (because of the reversal of arrows, see example 1.9). A product of the variable sets X and Y is their disjoint union $X \uplus Y$. The projection π_1 is the substitution which takes each variable in X to the same variable (possibly renamed to avoid clashes) in $X \uplus Y$. □

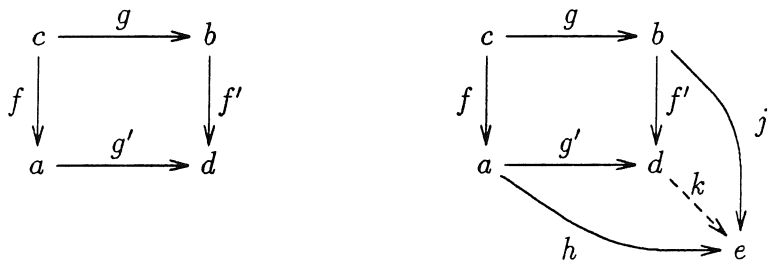
Products and coproducts can be defined for arbitrary families of objects. We will see how to do this in section 9. Now, we describe generalizations of products and coproducts which take into account "sharing", pullbacks and pushouts, also called "fibred products" and "amalgamated sums".

DEFINITION 5.8: *Pullback.* Given a pair of $\mathcal{C}$-arrows $a \xrightarrow{f} c \xleftarrow{g} b$ with common codomain, their pullback is defined to be a commutative square (shown on the left below) formed with a pair of $\mathcal{C}$-arrows $a \xleftarrow{g'} d \xrightarrow{f'} b$ (i.e., $f \circ g' = g \circ f'$) such that for any other pair of arrows $a \xleftarrow{h} e \xrightarrow{j} b$ which also form a commutative square (i.e., $f \circ h = g \circ j$) there is a unique $\mathcal{C}$-arrow $e \xrightarrow{k} d$ such that the diagram on the right below is commutative,



i.e., $g' \circ k = h$ and $f' \circ k = j$. □

DEFINITION 5.9: *Pushout.* Given a pair of $\mathcal{C}$-arrows $a \xleftarrow{f} c \xrightarrow{g} b$ with common domain, their pushout is defined to be a commutative square (shown on the left below) formed with a pair of $\mathcal{C}$-arrows $a \xrightarrow{g'} d \xleftarrow{f'} b$ (i.e., $g' \circ f = f' \circ g$) such that for any other pair of arrows $a \xrightarrow{h} e \xleftarrow{j} b$ which also form a commutative square (i.e., $h \circ f = j \circ g$) there is a unique $\mathcal{C}$-arrow $d \xrightarrow{k} e$ such that the diagram on the right below is commutative,



i.e., $k \circ g' = h$ and $k \circ f' = j$. □

Convention. When we say "pullback" or "pushout", we usually mean the *object* c at the corner of the squares in the definitions above.

EXAMPLE 5.10. In **Set**, a pullback of two functions $f: A \to C$ and $g: B \to C$ is given by $\{ \langle a, b \rangle \mid a \in A, b \in B, \text{ and } f(a) = g(b) \}$. The arrows into A and B are the projections π_1 and π_2. As a special case, the pullback of two subsets $A \hookrightarrow C$ and $B \hookrightarrow C$ is their intersection $A \cap B$ (up to isomorphism).

Dually, a pushout of $f: C \to A$ and $g: C \to B$ is obtained by taking the disjoint union $A \uplus B$ and coalescing pairs of elements $f(x)$ and $g(x)$ for each $x \in C$. Note that if inclusion arrows are used to indicate sharing, a pushout will contain only one copy of the shared entity. □

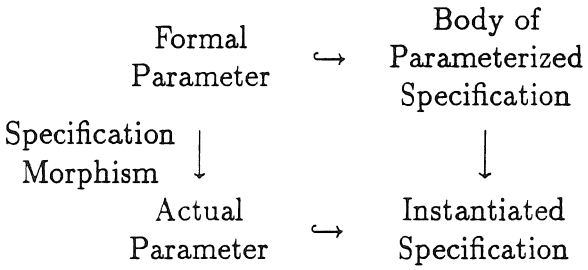
EXAMPLE 5.11. In **Set**, the pullback of a function $f: A \to B$ along an inclusion $C \hookrightarrow B$ (a subset $C \subseteq B$) produces the inverse image of C under f, defined by $f^{-1}(C) = \{ x \in A \mid f(x) \in C \}$. □

EXAMPLE 5.12. Products and coproducts can be constructed using pullbacks and pushouts as follows (0 and 1 are the initial and terminal objects, respectively):

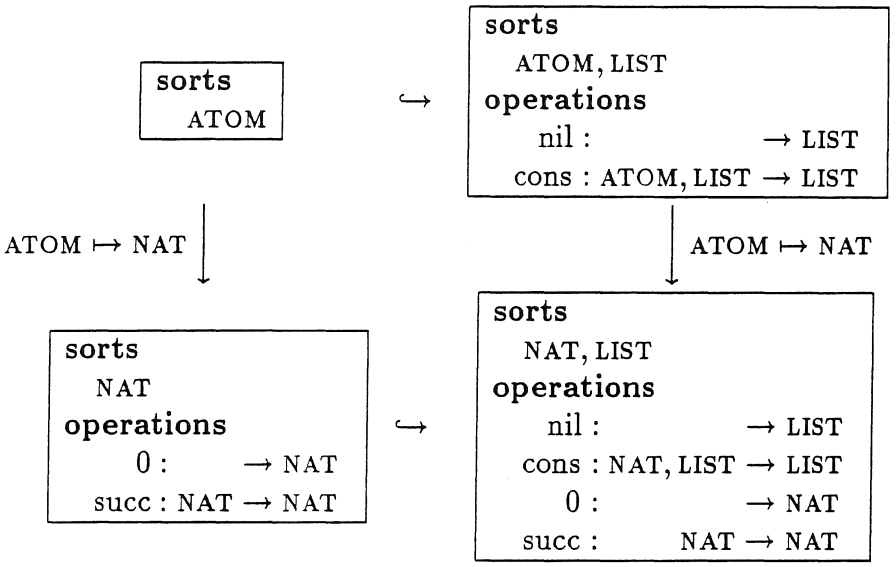
$$\begin{array}{ccc} a \times b & \xrightarrow{\pi_2} & b \\ {\scriptstyle \pi_1}\downarrow & & \downarrow{\scriptstyle 1_b} \\ a & \xrightarrow{1_a} & 1 \end{array} \qquad \begin{array}{ccc} 0 & \xrightarrow{0_b} & b \\ {\scriptstyle 0_a}\downarrow & & \downarrow{\scriptstyle \iota_2} \\ a & \xrightarrow{\iota_1} & a + b \end{array}$$

□

EXAMPLE 5.13. Pushouts in **Sign** are useful for instantiating parameters in a signature as shown below:[9]

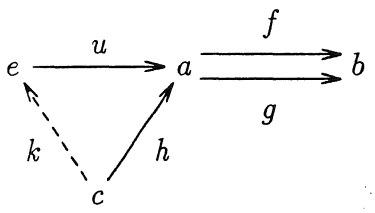


Here is a concrete example of parameterized lists, LIST(ATOM), instantiated with the specification of natural numbers, NAT, to obtain lists of natural numbers, LIST(NAT).



□

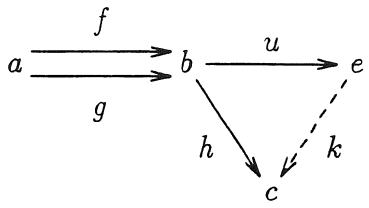
DEFINITION 5.14: *Equalizer.* Given a pair of $\mathcal{C}$-arrows $f, g: a \rightrightarrows b$ with the same domain and codomain, their equalizer is defined to be a $\mathcal{C}$-object e along with a $\mathcal{C}$-arrow $u: e \to a$ such that $f \circ u = g \circ u$, and for any $\mathcal{C}$-arrow $h: c \to a$ with $f \circ h = g \circ h$, there is a unique $\mathcal{C}$-arrow $k: c \to e$ such that the following diagram commutes,



i.e., $u \circ k = h$. □

[9]See [EM85, Chapter 8] for details.

DEFINITION 5.15: *Coequalizer.* Given a pair of $\mathcal{C}$-arrows $f, g: a \rightrightarrows b$ with the same domain and codomain, their coequalizer is defined to be a $\mathcal{C}$-object e along with a $\mathcal{C}$-arrow $u: b \to e$ such that $u \circ f = u \circ g$, and for any $\mathcal{C}$-arrow $h: b \to c$ with $h \circ f = h \circ g$, there is a unique $\mathcal{C}$-arrow $k: e \to c$ such that the following diagram commutes,



i.e., $k \circ u = h$. □

EXAMPLE 5.16. In the category **Set**, the equalizer of two functions $f, g: A \rightrightarrows B$ is the largest subset $E \hookrightarrow A$ of A on which the two functions f and g are equal (hence the name "equalizer"), i.e., $E = \{ x \in A \mid f(x) = g(x) \}$. Universality guarantees that E is the largest subset of A on which f and g are equal.

Dually, the coequalizer is the quotient of B by the smallest equivalence relation, $\sim$, which contains all pairs $\langle f(x), g(x) \rangle$ for $x \in A$. The coequalizing map $u: B \to B/\sim$ is given by $b \mapsto [b]$ for all $b \in B$, which takes each element of B to its equivalence class. □

EXAMPLE 5.17. In the category **Terms**(Σ), the most general unifier of two terms can be considered as an equalizer as follows. Let t_1 and t_2 be two Σ-terms of sort s with free variables in X. Let I be a singleton set consisting of exactly one variable i of sort s. The two terms can then be represented as a parallel pair of arrows

$$X \underset{t_2}{\overset{t_1}{\rightrightarrows}} I$$

given by the substitutions $i \mapsto t_1$ and $i \mapsto t_2$. A unifier of these two terms is a substitution from X into a collection of variables U, represented by the arrow $u: U \to X$, such that after this substitution, the two terms are equal, i.e., $t_1 \circ u = t_2 \circ u$. A most general unifier is a unifier such that every other unifier is an instance (via some substitution) of it; this defines it to be an equalizer. □

6 Functors

The next step in abstraction is to treat a category itself as an object. What is an appropriate notion of morphism for categories? Something that maps objects to objects and arrows to arrows, in a compatible way, just like homomorphisms of algbras. Such a morphism of categories is called a *functor.*

DEFINITION 6.1: *Functor.* A functor F from a category $\mathcal{C}$ to a category $\mathcal{D}$ is a function which assigns to each $\mathcal{C}$-object a, a $\mathcal{D}$-object $F(a)$, and to each $\mathcal{C}$-arrow $f: a \to b$, a $\mathcal{D}$-arrow $F(f): F(a) \to F(b)$, such that identity arrows and composites are preserved, i.e.,

$$F(\mathrm{id}_a) = \mathrm{id}_{F(a)}, \text{ for all } \mathcal{C}\text{-objects } a, \text{ and}$$
$$F(g \circ f) = F(g) \circ F(f), \text{ whenever } g \circ f \text{ is defined in } \mathcal{C}.$$

□

EXAMPLE 6.2: *Identity functors.* Given a category $\mathcal{C}$, the identity functor $\mathrm{id}_{\mathcal{C}}: \mathcal{C} \to \mathcal{C}$ is defined by $\mathrm{id}_{\mathcal{C}}(a) = a$ and $\mathrm{id}_{\mathcal{C}}(f) = f$ for all objects a and arrows f in $\mathcal{C}$. □

EXAMPLE 6.3: *Forgetful functors.* Let **Group** be the category of groups and group homomorphisms. The forgetful (or underlying) functor U: **Group** $\to$ **Set** maps each group onto its carrier set and each homomorphism onto its underlying function. There is an obvious extension to many-sorted algebras, e.g., the forgetful functor between $\mathbf{Alg}(\Sigma)$ and S-sorted sets and functions. □

EXAMPLE 6.4: *Reduct functors.* Given a signature morphism $\sigma: \Sigma \to \Sigma'$, we can define the σ-reduct functor $\_|_\sigma: \mathbf{Alg}(\Sigma') \to \mathbf{Alg}(\Sigma)$ which maps Σ'-algebras to Σ-algebras and Σ'-homomorphisms to Σ-homomorphisms (note that the reduct functor goes in the opposite direction to the signature morphism which induces it).

The σ-reduct of a Σ'-algebra A', denoted by $A'|_\sigma$, is the Σ-algebra $A = \langle A_S, F_A \rangle$ defined as follows: (with $\Sigma = \langle S, \Omega \rangle$)

$$A_s = A'_{\sigma(s)}, \text{ for } s \in S, \text{ and } f_A = \sigma(f)_{A'}, \text{ for } f \in \Omega.$$

Given a Σ'-homomorphism $h': A' \to B'$ between two Σ'-algebras A' and B', the σ-reduct of h' is a Σ-homomorphism $h: A'|_\sigma \to B'|_\sigma$, denoted by $h'|_\sigma$, and defined by the family of functions $h_s = h'_{\sigma(s)}$, for $s \in S$. □

EXAMPLE 6.5: *Monotone functions.* A functor F between two pre-orders P and Q treated as categories, is just a monotone function, i.e., $a \leq_P b$ implies $F(a) \leq_Q F(b)$. □

EXAMPLE 6.6: *The category of all categories.* The category of all categories, denoted **Cat**, contains categories as objects and functors as arrows. Identity functors form the identity arrows, and functors are composed in the obvious way.

Note that this category leads to foundational difficulties, because we can construct the set of *all* sets from it, which leads to Russell's paradox. To avoid this, we can use the set/class distinction and consider as objects of **Cat**, only those categories whose collection of arrows are sets (and not proper classes). Since the class of all sets is not a set, **Cat** does not contain itself, and we avoid a paradox. Alternatively, we can consider sets which are *small* with respect to some universe, as in [Mac71, section 1.6]. □

The functors discussed above preserve the direction of the arrows. Such functors are called *covariant.* A *contravariant* functor is one which reverses the direction of arrows.

DEFINITION 6.7: *Contravariant functor.* A contravariant functor F from a category $\mathcal{C}$ to a category $\mathcal{D}$ is a function which assigns to each $\mathcal{C}$-object a, a $\mathcal{D}$-object $F(a)$, and to each $\mathcal{C}$-arrow $f: a \to b$, a $\mathcal{D}$-arrow $F(f): F(b) \to F(a)$ (note the reversal of direction), such that identity arrows and composites (reversed) are preserved, i.e.,

$$F(\mathrm{id}_a) = \mathrm{id}_{F(a)}, \text{ for all } \mathcal{C}\text{-objects } a, \text{ and}$$
$$F(g \circ f) = F(f) \circ F(g), \text{ whenever } g \circ f \text{ is defined in } \mathcal{C}.$$

□

EXAMPLE 6.8: *Model functor.* The functor **Alg: Sign** $\to$ **Cat** which assigns to each signature Σ, the category of Σ-algebras and to each signature morphism σ the σ-reduct functor, is contravariant. □

EXAMPLE 6.9: *Powerset functor.* The contravariant powerset functor $\mathcal{P}$: **Set** $\to$ **Set** takes each set A to its powerset $\mathcal{P}(A)$ and each function $f: A \to B$ to a function $\mathcal{P}(f): \mathcal{P}(B) \to \mathcal{P}(A)$ which assigns to each subset $X \subseteq B$ its inverse image $f^{-1}(X) \subseteq A$. □

Contravariant functors can be made covariant by considering the opposite category for either the domain or codomain of the functor. Thus the model functor above is usually written as **Alg: Sign** $\to$ **Cat**$^{\mathrm{op}}$. By convention, the word "functor" used alone means "covariant functor".

In any category $\mathcal{C}$ let $\hom_{\mathcal{C}}(a, b)$ (called "hom-set") denote the collection of $\mathcal{C}$-arrows with domain a and codomain b. Observe that a functor $F: \mathcal{C} \to \mathcal{D}$ induces functions on hom-sets (one for each pair a, b of $\mathcal{C}$-objects) defined by

$$F_{a,b}: \hom_{\mathcal{C}}(a, b) \to \hom_{\mathcal{D}}(F(a), F(b)), \qquad f \mapsto F(f).$$

DEFINITION 6.10: *Full, faithful.* A functor $F: \mathcal{C} \to \mathcal{D}$ is said to be full if each function $F_{a,b}$ on the hom-sets is surjective, and faithful when each is injective. □

DEFINITION 6.11: *Subcategory.* A subcategory $\mathcal{S}$ of a category $\mathcal{C}$ is a collection of some $\mathcal{C}$-objects and some $\mathcal{C}$-arrows such that $\mathcal{S}$ forms a category (i.e., for each arrow f, the objects $\mathrm{dom}\, f$ and $\mathrm{cod}\, f$ are in $\mathcal{S}$, for each object a, the arrow id_a is in $\mathcal{S}$, for each pair of composable arrows, the composite arrow is in $\mathcal{S}$). □

Corresponding to a subcategory $\mathcal{S}$ of $\mathcal{C}$, we have an inclusion functor $\mathcal{S} \hookrightarrow \mathcal{C}$ which sends each object and each arrow of $\mathcal{S}$ to itself. This functor is faithful by definition. If it is also full, then $\mathcal{S}$ is said to be a *full subcategory* of $\mathcal{C}$.

EXAMPLE 6.12. The category **FinSet**, consisting of all finite sets and all functions between them, is a (full) subcategory of **Set**. □

EXAMPLE 6.13. Let $SP = \langle \Sigma, \Phi \rangle$ be a specification consisting of a signature Σ and a set of equations Φ. A model of SP is a Σ-algebra which satisfies all the equations in Φ.[10] The models of SP and the Σ-homomorphisms between them form a category $Alg[SP]$ which is a (full) subcategory of $\mathbf{Alg}(\Sigma)$. □

[10] See [EM85, Definition 1.13] for a definition of equation and satisfaction.

7 Natural transformations

Now that we have treated categories as objects and functors as arrows, the next step in abstraction is to treat functors themselves as objects.[11] We now define a[12] notion of morphism for functors.

DEFINITION 7.1: *Natural transformation.* Given two functors $F, G: \mathcal{C} \to \mathcal{D}$, a natural transformation $\tau: F \dot{\to} G$ is an assignment to each $\mathcal{C}$-object a of a $\mathcal{D}$-arrow $\tau_a: F(a) \to G(a)$ such that for any $\mathcal{C}$-arrow $f: a \to b$, the following diagram commutes.

$$\begin{array}{ccc} a & & F(a) \xrightarrow{\tau_a} G(a) \\ f \downarrow & \qquad & F(f) \downarrow \qquad\qquad \downarrow G(f) \\ b & & F(b) \xrightarrow{\tau_b} G(b) \end{array}$$

i.e., $G(f) \circ \tau_a = \tau_b \circ F(f)$. □

When the condition above holds, we say that the assignment of τ_a to a is *natural* in a (i.e., it preserves categorial structure). We can think of a natural transformation as translating the image of the functor F in the category $\mathcal{D}$ into the image of the functor G. The arrows τ_a, τ_b, τ_c, etc., are called the *components* of the natural transformation τ.

[11] Obviously, we can carry this process further and try to treat natural transformations as objects. However, such concepts do not seem to be necessary in mathematical practice. Moreover, natural transformations behave somewhat like functors, in that they can be composed with functors on the left and the right.

[12] There are other notions of morphisms of functors. For example, in the arrow category $\mathbf{Cat}^{\to}$, objects are functors, and a morphism from an object $F: A \to B$ to an object $G: C \to D$ is a pair of functors $X: A \to C$ and $Y: B \to D$ such that $Y \circ F = G \circ X$.

EXAMPLE 7.2. We noted in example 5.4 that the products $A \times B$ and $B \times A$ are isomorphic in the category **Set**. The intuitive notion that this is true for all sets B is captured in the definition of a natural transformation. We first define two functors:[13] (here, $f: X \to Y$ is a arrow in **Set**)

$$\begin{array}{lll} A \times \_ : \mathbf{Set} \to \mathbf{Set} & & \_ \times A : \mathbf{Set} \to \mathbf{Set} \\ X \mapsto A \times X & & X \mapsto X \times A \\ f \mapsto \mathrm{id}_A \times f & & f \mapsto f \times \mathrm{id}_A \end{array}$$

We will define a natural transformation $tw: A \times \_ \dot{\to} \_ \times A$ (tw for "twist"). For each object X in **Set**, let $tw_X: A \times X \to X \times A$ be defined by $tw_X(\langle a, x \rangle) = \langle x, a \rangle$. For any arrow $f: X \to Y$ in **Set**, the following diagram commutes.

$$\begin{array}{ccc} X & & A \times X \xrightarrow{tw_X} X \times A \\ f \downarrow & & \mathrm{id}_A \times f \downarrow \qquad\qquad \downarrow f \times \mathrm{id}_A \\ Y & & A \times Y \xrightarrow{tw_Y} Y \times A \end{array}$$

Thus the maps tw_X form the components of a natural transformation.

A natural transformation, such as tw, for which each of the components is an isomorphism, is called a *natural isomorphism* or *natural equivalence.* □

EXAMPLE 7.3: *Functor categories.* Given two categories $\mathcal{C}$ and $\mathcal{D}$ we can define the category of all functors and natural transformations between them as shown below. This category is called a functor category and is denoted $\mathcal{D}^{\mathcal{C}}$.

The objects of $\mathcal{D}^{\mathcal{C}}$ are functors $F: \mathcal{C} \to \mathcal{D}$. The arrows are natural transformations $\tau: F \dot{\to} G$. The identity arrows, $\mathrm{id}_F: F \dot{\to} F$ are identity natural transformations defined for each $\mathcal{C}$-object c by $c \mapsto \mathrm{id}_{F(c)}$ (identity arrows in $\mathcal{D}$). Two natural transformations $\tau: F \dot{\to} G$ and $\mu: G \dot{\to} H$ can be composed, denoted $\mu \cdot \tau$ ("vertical" composition[14]), by composing their components: $(\mu \cdot \tau)_c = \mu_c \circ \tau_c$ (in $\mathcal{D}$). It can be shown by straightforward diagram chasing that $\mu \cdot \tau$ is natural, and that this composition is associative. □

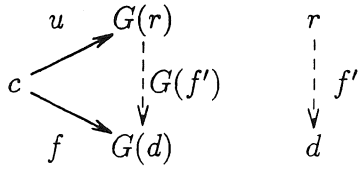
[13] Given arrows $f: A \to C$ and $g: B \to D$, the arrow $f \times g: A \times B \to C \times D$ is defined as $\langle f \circ \pi_1, g \circ \pi_2 \rangle$, where the notation $\langle \_, \_ \rangle$ indicates the unique arrow given by the definition of product.

[14] There is also a "horizontal" composition of natural transformations, see [Mac71, section II.5].

8 Universal arrows

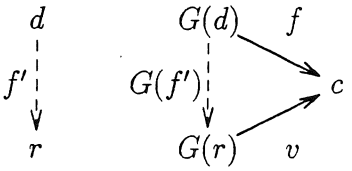
The concept of "universal property" introduced in section 5 above can be precisely formulated in terms of functors as shown below.

DEFINITION 8.1: *Universal arrow.* Given a functor $G: \mathcal{D} \to \mathcal{C}$ and a $\mathcal{C}$-object c, a universal arrow *from* c *to* G is a pair[15] $\langle r, u \rangle$ consisting of a $\mathcal{D}$-object r and a $\mathcal{C}$-arrow $u: c \to G(r)$ such that for every pair $\langle d, f \rangle$ of a $\mathcal{D}$-object d and a $\mathcal{C}$-arrow $f: c \to G(d)$, there is a unique $\mathcal{D}$-arrow $f': r \to d$ such that the following diagram commutes



i.e., $G(f') \circ u = f$.

Dually, a universal arrow *from* G *to* c[16] is a pair $\langle r, v \rangle$ consisting of a $\mathcal{D}$-object r and a $\mathcal{C}$-arrow $v: G(r) \to c$ such that for every pair $\langle d, f \rangle$ of a $\mathcal{D}$-object d and a $\mathcal{C}$-arrow $f: G(d) \to c$, there is a unique $\mathcal{D}$-arrow $f': d \to r$ such that the following diagram commutes



i.e., $v \circ G(f') = f$. □

[15]The pair $\langle r, u \rangle$ is called a universal arrow because the essential information is contained in the arrow $u: c \to G(r)$. Arrows of this form are objects in the comma category $(c \downarrow G)$; a morphism in this category from $u: c \to G(r)$ to $v: c \to G(s)$ is a $\mathcal{D}$-arrow $f: r \to s$ such that $G(f) \circ u = v$. The definition of a universal arrow from c to G is equivalent to the statement that $u: c \to G(r)$ is an initial object in the comma category $(c \downarrow G)$.

[16]We follow Mac Lane [Mac71] in using the terminology "universal arrow from G to c" rather than "couniversal arrow," which is preferred by other authors.

EXAMPLE 8.2: *Free categories.* For any category $\mathcal{C}$, its underlying graph is given by

$$\langle \mathit{Objects}(\mathcal{C}), \mathit{Arrows}(\mathcal{C}), \mathrm{dom}, \mathrm{cod} \rangle,$$

with the objects as nodes, arrows as edges, domain and codomain as source and target. Similarly, underlying every functor is a graph morphism. Thus we have a forgetful functor $U\colon \mathbf{Cat} \to \mathbf{Graph}$.

A universal arrow from a graph G to the functor U is a pair $\langle C_G, P\colon G \to U(C_G) \rangle$ consisting of the *free category* C_G (also called the *path category*) generated by the graph G and a graph morphism embedding G into the underlying graph of C_G.

The free category C_G is constructed as follows. The objects are the nodes of the graph G. The arrows are finite (directed) paths in G, i.e., a sequence of edges $e_1, e_2, \ldots, e_n$ with $s(e_i) = t(e_{i-1})$, for $i = 2, \ldots, n$. Pictorially, this path can be depicted as

$$v_1 \xrightarrow{e_1} v_2 \xrightarrow{e_2} \cdots \xrightarrow{e_n} v_{n+1}.$$

The domain of this arrow is defined to be v_1 and the codomain to be v_{n+1}. Composition of arrows is the concatenation of paths (which is obviously associative), and the identities are null paths which start and end at the same node.

A proof that the arrow $P\colon G \to U(C_G)$ is universal can be found in [Mac71, Theorem II.1]. □

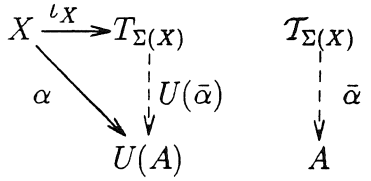
EXAMPLE 8.3: *Free algebras.* Given a signature $\Sigma = \langle S, \Omega \rangle$, let $\mathbf{Set}(\Sigma)$ denote the subcategory of $\mathbf{Set}$ consisting of S-sorted sets, i.e., families $\{ X_s \mid s \in S \}$, and S-sorted functions, i.e., families $\{ f_s : X_s \to Y_s \mid s \in S \}$.[17] The forgetful functor $U : \mathbf{Alg}(\Sigma) \to \mathbf{Set}(\Sigma)$ sends each Σ-algebra to its family of carrier sets and each Σ-homomorphism to itself, considered as a family of functions.

A universal arrow from a $\mathbf{Set}(\Sigma)$-object X to U is a pair $\langle \mathcal{T}_\Sigma(X), \iota_X : X \hookrightarrow T_\Sigma(X) \rangle$ consisting of the *free* Σ*-algebra* generated by X and the inclusion of X into the carriers of this free algebra.

The free algebra $\mathcal{T}_\Sigma(X)$ is obtained just like the ground term algebra $\mathcal{T}_\Sigma$ (see example 2.10), but with the additional rule stating that every variable is also a term:

> if $x \in X_s$ is a variable of sort s, then the term x is in $T_{\Sigma,s}(X)$.

$T_\Sigma(X)$ is thus the collection of all terms that can be generated from the signature Σ using free variables from X. The free algebra $\mathcal{T}_{\Sigma(X)}$ has the universal property that, given any Σ-algebra A, and an assignment $\alpha : X \to U(A)$ assigning "values" in the carriers of A to variables in X, the assignment can be uniquely extended to a Σ-homomorphism $\bar{\alpha} : \mathcal{T}_{\Sigma(X)} \to A$, such that the following diagram commutes,



i.e., $U(\bar{\alpha}) \circ \iota_X = \alpha$. The map $U(\bar{\alpha})$ corresponds to "evaluating" the terms in $T_{\Sigma(X)}$ with values for variables given by the assignment α.[18]

A proof that the arrow $\iota_X : X \hookrightarrow T_{\Sigma(X)}$ is universal can be found in [EM85, Theorem 3.3]. □

EXAMPLE 8.4: *Products.* Given a category $\mathcal{C}$, consider the *diagonal* functor $\Delta : \mathcal{C} \to \mathcal{C} \times \mathcal{C}$, defined by

$$\begin{array}{rcl} c & \mapsto & \langle c, c \rangle \\ f : a \to b & \mapsto & \langle f, f \rangle : \langle a, a \rangle \to \langle b, b \rangle \end{array}$$

A universal arrow from Δ to an object $\langle a, b \rangle$ of $\mathcal{C} \times \mathcal{C}$ is a pair $\langle a \times b, \langle \pi_1, \pi_2 \rangle \rangle$ consisting of the product of the $\mathcal{C}$-objects a and b together with the pair of projections. The definition of product is just a rephrasing of the universality of this arrow. □

[17] The family of functions $\{ f_s : X_s \to Y_s \mid s \in S \}$ can be considered as the coproduct arrow $\coprod_{s \in S} f_s : \coprod_{s \in S} X_s \to \coprod_{s \in S} Y_s$ in the category $\mathbf{Set}$.

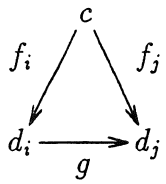
[18] Note that in algebra, the distinction between the algebra A and its underlying carriers $U(A)$ is fuzzed; similarly the distinction between $\bar{\alpha}$ and $U(\bar{\alpha})$. Thus, only the left triangle in the commutative diagram is shown.

9 Limits and colimits

The interpretation of the product as a universal arrow can be generalized to other constructions such as terminal objects, pullbacks, and equalizers. All these constructions are called *limits* and arise as universal arrows for various "diagrams". We first give a definition which captures the uniformity in the definition of various limits. Later, we see how limits are universal arrows, and thus are limits for certain functors.

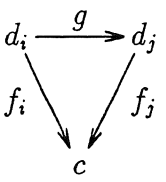
DEFINITION 9.1: *Diagram.* A diagram in a category $\mathcal{C}$ is a collection of $\mathcal{C}$-objects and a collection of $\mathcal{C}$-arrows between these objects. □

DEFINITION 9.2: *Cone.* Given a diagram D in a category $\mathcal{C}$ and a $\mathcal{C}$-object c, a cone *from the vertex c to the base D* is a collection of $\mathcal{C}$-arrows $\{ f_i : c \to d_i \mid d_i \in D \}$, one for each object d_i in the diagram D, such that for any arrow $g : d_i \to d_j$ in D, the following triangle commutes



i.e., we have $g \circ f_i = f_j$.

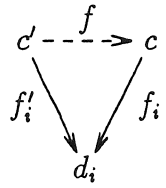
Dually, a cone[19] *from the base D to the vertex c* is a collection of $\mathcal{C}$-arrows $\{ f_i : d_i \to c \mid d_i \in D \}$, one for each object d_i in the diagram D, such that for any arrow $g : d_i \to d_j$ in D, the following triangle commutes



i.e., we have $f_j \circ g = f_i$. □

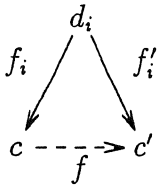
[19] We follow Mac Lane [Mac71] in the terminology here. Some authors use the name "co-cone".

DEFINITION 9.3: *Limit.* A limit for a diagram D in a category $\mathcal{C}$ is a $\mathcal{C}$-object c along with a cone $\{ f_i: c \to d_i \mid d_i \in D \}$ from c to D such that for any other cone $\{ f'_i: c' \to d_i \mid d_i \in D \}$ from a vertex c' to D, there is a unique $\mathcal{C}$-arrow $f: c' \to c$ such that for every object d_i in D, the following diagram commutes



i.e., $f_i \circ f = f'_i$. □

DEFINITION 9.4: *Colimit.* A colimit for a diagram D in a category $\mathcal{C}$ is a $\mathcal{C}$-object c along with a cocone $\{ f_i: d_i \to c \mid d_i \in D \}$ from D to c such that for any other cocone $\{ f'_i: d_i \to c' \mid d_i \in D \}$ from D to a vertex c', there is a unique $\mathcal{C}$-arrow $f: c \to c'$ such that for every object d_i in D, the following diagram commutes



i.e., $f \circ f_i = f'_i$. □

EXAMPLE 9.5. The following table lists limits and colimits for various diagrams. Observe that we can now define the products and coproducts of arbitrary families of objects.

Diagram	Limit	Colimit
empty	1 (terminal object)	0 (initial object)
$a \quad b$	$a \times b$ (product)	$a + b$ (coproduct)
$\{ a_x \mid x \in X \}$	$\prod_{x \in X} a_x$	$\coprod_{x \in X} a_x$
$\cdot \leftarrow \cdot \rightarrow \cdot$		pushout
$\cdot \rightarrow \cdot \leftarrow \cdot$	pullback	
$\cdot \rightrightarrows \cdot$	equalizer	coequalizer

□

We now precisely formulate limits and colimits as universal arrows. Let $\mathcal{C}$ and $\mathcal{J}$ be two categories, $\mathcal{J}$ being the "index" category (usually finite or small). Then, the image of a

functor $F: \mathcal{J} \to \mathcal{C}$ forms a diagram[20] (of "shape" $\mathcal{J}$) in $\mathcal{C}$. A limit for this diagram is a limit for the functor F.

Consider the functor category $\mathcal{C}^{\mathcal{J}}$. Corresponding to any $\mathcal{C}$-object c, we have the constant functor Δc which maps all $\mathcal{J}$-objects onto c and all $\mathcal{J}$-arrows onto id_c. A cone from c to the diagram generated by the functor $F: \mathcal{J} \to \mathcal{C}$ is then a natural transformation $\Delta c \stackrel{\cdot}{\to} F$. A limit for F is a universal such cone.

DEFINITION 9.6: *Diagonal functor.* Given two categories $\mathcal{C}$ and $\mathcal{J}$ the diagonal functor $\Delta: \mathcal{C} \to \mathcal{C}^{\mathcal{J}}$ is defined by (for any $\mathcal{C}$-objects c and c', $\mathcal{C}$-arrow $f: c \to c'$, and $\mathcal{J}$-objects j, k)

$$\begin{array}{rcl} \Delta c: \mathcal{J} & \to & \mathcal{C} \\ j & \mapsto & c \\ j \to k & \mapsto & \mathrm{id}_c \end{array} \quad \text{and} \quad \begin{array}{rcl} \Delta f: \Delta c & \stackrel{\cdot}{\to} & \Delta c' \\ j & \mapsto & f \end{array}$$

□

DEFINITION 9.7: *Limit, Colimit.* Given two categories $\mathcal{C}$ and $\mathcal{J}$, a limit (also called inverse limit, or projective limit) for a functor $F: \mathcal{J} \to \mathcal{C}$, denoted by $\varprojlim F$, is a universal arrow from the diagonal functor $\Delta: \mathcal{C} \to \mathcal{C}^{\mathcal{J}}$ to F. Dually, a colimit (also called direct limit, or inductive limit) for F, denoted by $\varinjlim F$, is a universal arrow from F to Δ. □

10 Completeness

It is useful to know if all limits (or all colimits) exist in a category. For example, in algebraic specification, colimits are used to "put together" specifications and theories from smaller ones; colimits have to always exist for these building operations to be well-defined.

DEFINITION 10.1: *Completeness.* A category is said to be complete (*finitely* complete) if all diagrams (finite diagrams) have limits in the category.

Dually, a category is said to be cocomplete (*finitely* cocomplete) if all diagrams (finite diagrams) have colimits in the category. □

EXAMPLE 10.2. The category **Set** is both complete and cocomplete. The category **Sign** of signatures and signature morphisms is finitely cocomplete. Pre-order categories are complete and cocomplete when they form a complete lattice. □

The following theorem shows that all finite limits can be constructed from a few basic ones.

THEOREM. For any category $\mathcal{C}$, the following conditions are equivalent:

1. $\mathcal{C}$ is finitely complete.
2. $\mathcal{C}$ has a terminal object, products of all pairs of objects, and equalizers of all parallel pairs of arrows.
3. $\mathcal{C}$ has a terminal object and all pullbacks.

[20] Some authors (e.g., Schubert [Sch72]) define a diagram to be a graph morphism from a graph J to the underlying graph of the category $\mathcal{C}$. This has the advantage that composition of arrows need not be defined in the graph J. The two approaches are, however, equivalent; it is not the graph J or the category $\mathcal{J}$ which matter, it is the *image* of the graph morphism or functor which matters.

For a proof, see [Mac71, Chapter V] or [HS73].

Limits in one category can be transferred to another category. For example, a functor which has a left adjoint (see section 11) preserves limits (i.e., maps limit cones to limit cones). The completeness of a functor category $\mathcal{C}^{\mathcal{D}}$ follows from that of $\mathcal{C}$ (all constructions in $\mathcal{C}$ can be "lifted" to constructions in the functor category).

11 Adjoints

Consider again the free category generated by a graph X (example 8.2). It is obtained as a universal arrow from X to the forgetful functor U: **Cat** $\to$ **Graph**. It so happens that we can find such a universal arrow for every graph. Now, let $\gamma: X \to Y$ be a graph morphism and let $\langle C_X, \eta_X: X \to U(C_X)\rangle$ and $\langle C_Y, \eta_Y: Y \to U(C_Y)\rangle$ be the universal arrows corresponding to X and Y. We thus have an arrow $\eta_Y \circ \gamma: X \to U(C_Y)$, and from the universal property of η_X, it follows that we have a unique arrow $C_\gamma: C_X \to C_Y$ such that the following square commutes

$$\begin{array}{ccc} X & \xrightarrow{\eta_X} & U(C_X) \\ {\scriptstyle\gamma}\downarrow & & \downarrow{\scriptstyle U(C_\gamma)} \\ Y & \xrightarrow[\eta_Y]{} & U(C_Y) \end{array} \qquad \begin{array}{c} C_X \\ \vdots \\ \downarrow \\ C_Y \end{array}$$

i.e., $\eta_Y \circ \gamma = U(C_\gamma) \circ \eta_X$.

The assignments $X \mapsto C_X$ and $\gamma \mapsto C_\gamma$ give us a functor C: **Graph** $\to$ **Cat** which maps each graph to the free category generated by it, and extends each graph morphism to a functor. The universal arrows in the picture above also provide us with the following bijection between hom-sets (for any graph X and any category D):

$$\mathbf{Graph}(X, U(D)) \cong \mathbf{Cat}(C(X), D).$$

The functors C and U are called adjoints, and this situation arises whenever there are "free" constructions.

DEFINITION 11.1: *Adjunction.* Let $\mathcal{C}$ and $\mathcal{D}$ be two categories. An adjunction from $\mathcal{C}$ to $\mathcal{D}$ is a triple $\langle F, G, \varphi\rangle: \mathcal{C} \rightharpoonup \mathcal{D}$ where F and G are functors

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

and φ is a function which assigns, to each pair consisting of a $\mathcal{C}$-object c and a $\mathcal{D}$-object d, a bijection

$$\varphi_{c,d}: \mathcal{C}(c, G(d)) \cong \mathcal{D}(F(c), d)$$

which is natural in c and d. □

The phrase "natural in c and d" used above means that the bijection φ preserves categorial structure while c and d vary (i.e., it is defined "uniformly" for all c and d). A precise formulation of this condition makes φ a natural transformation between two functors from $\mathcal{C}^{\mathrm{op}} \times \mathcal{D} \to \mathbf{Set}$ (see, for example, [Mac71, Section IV.1]).

There are several equivalent ways of defining adjoints. As we have already seen in the case of free categories generated by graphs, given an adjunction $\langle F, G, \varphi \rangle : \mathcal{C} \rightharpoonup \mathcal{D}$, there is a universal arrow $\langle F(c), \eta_c : c \to G(F(c)) \rangle$ from each $\mathcal{C}$-object c to the functor G. Moreover, the maps η_c provided by these universal arrows form the components of a natural transformation $\eta : \mathrm{id}_{\mathcal{C}} \dot{\to} G \circ F$, called the *unit* of the adjunction. Similarly, the adjunction determines (and is determined by) universal arrows $\langle G(d), \varepsilon_d : F(G(d)) \to d \rangle$ from F to each $\mathcal{D}$-object d. The maps ε_d form the components of a natural transformation $\varepsilon : F \circ G \dot{\to} \mathrm{id}_{\mathcal{D}}$, called the *counit* of the adjunction.

Notation. If there is an adjunction $\langle F, G, \varphi \rangle : \mathcal{C} \rightharpoonup \mathcal{D}$, then the functor F is said to be *left adjoint* to the functor G, written $F \dashv G$; the functor G is said to be *right adjoint* to F, written $G \vdash F$.

EXAMPLE 11.2: *Free functors.* Free functors are left adjoint to forgetful functors. For example, the forgetful functor $U : \mathbf{Alg}(\Sigma) \to \mathbf{Set}(\Sigma)$ (defined in example 8.3) has a left adjoint $\mathcal{F} : \mathbf{Set}(\Sigma) \to \mathbf{Alg}(\Sigma)$ which maps each S-sorted set to the free Σ-algebra generated by it and extends each S-sorted function to a Σ-homomorphism.

The functor $C : \mathbf{Graph} \to \mathbf{Cat}$ which produces free categories from graphs (see example 8.2) and extends graph morphisms to functors is left adjoint to the forgetful functor $U : \mathbf{Cat} \to \mathbf{Graph}$.

Corresponding to the reduct functors $\_|_\sigma : \mathbf{Alg}(\Sigma') \to \mathbf{Alg}(\Sigma)$, for each signature morphism σ, there are free functors $\mathcal{F}_\sigma : \mathbf{Alg}(\Sigma) \to \mathbf{Alg}(\Sigma')$ which freely extend any Σ-algebra to a Σ'-algebra (see, for example, [EM85, Theorem 7.16] or [GTW78]). These free functors provide an important way of incrementally building algebraic specifications. □

EXAMPLE 11.3: *Limits and colimits.* Suppose the category $\mathcal{C}$ has all products. Then, the functor $\prod : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ which assigns to each pair of objects $\langle a, b \rangle$ their product $a \times b$, is right adjoint to the diagonal functor $\Delta : \mathcal{C} \to \mathcal{C} \times \mathcal{C}$ defined by $c \mapsto \langle c, c \rangle$ (see example 8.4 for details). Dually, if the category $\mathcal{C}$ had all coproducts, then the coproduct functor $\coprod : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ is left adjoint to Δ.

If all limits or colimits of a certain kind exist in a category, then they arise from a right or left adjoint, respectively, to the diagonal functor $\Delta : \mathcal{C} \to \mathcal{C}^{\mathcal{J}}$, where $\mathcal{J}$ is the "shape" of the diagram for which limits and colimits are being considered. □

EXAMPLE 11.4: *Images of functions.* For any set A, the powerset $\mathcal{P}(A)$ consisting of all subsets of A is a pre-order under inclusion of subsets, and therefore a category. Now, given a function $f: A \to B$, we can define two order-preserving functions (or functors), $f_*: \mathcal{P}(A) \to \mathcal{P}(B)$ and $f^*: \mathcal{P}(B) \to \mathcal{P}(A)$, called the direct image function and the inverse image function, respectively:

$$\begin{aligned} f_*(X) &= \{ f(x) \mid x \in X \}, \\ f^*(Y) &= \{ x \mid f(x) = y \text{ for some } y \in Y \}. \end{aligned}$$

It is easy to see that $f_*(X) \subseteq Y$ if and only if $X \subseteq f^*(Y)$. Thus the functor f_* is left adjoint to f^*.

The inverse image functor f^* also has a right adjoint f^+ defined by $f^+(X) = \{ y \in Y \mid f^*\{y\} \subseteq X \}$. Here $f^*\{y\}$ is the inverse image of the element y. □

EXAMPLE 11.5: *Exponentiation.* In the category **Set**, let the object C^B denote the collection of functions from B to C. Multi-argument functions, say from $f: A \times B$ to C can be reduced to single-argument functions by "Currying":[21]

$$f \equiv \lambda x \cdot (\lambda y \cdot f(\langle x, y \rangle)).$$

We thus have a bijection of hom-sets

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

This bijection in turn produces an adjunction

$$\mathbf{Set} \underset{(\_)^B}{\overset{\_ \times B}{\rightleftarrows}} \mathbf{Set}$$

The counit of this adjunction provides "evaluation" arrows, $eval_{B,C}: C^B \times B \to C$, which when given a function $f: B \to C$ and an argument $b \in B$, evaluate the function to produce $f(b) \in C$. □

[21] We use λ-expressions to define functions.

References

[EM85] EHRIG, H., AND MAHR, B. *Fundamentals of Algebraic Specification 1: Equational and Initial Semantics, EATCS Monographs on Theoretical Computer Science,* Vol. 6. Springer-Verlag, Berlin, 1985.

[GB84a] GOGUEN, J. A., AND BURSTALL, R. M. Some fundamental algebraic tools for the semantics of computation, Part 1: Comma categories, colimits, signatures and theories. *Theoretical Comput. Sci. 31,* 2 (1984), 175–209.

[GB84b] GOGUEN, J. A., AND BURSTALL, R. M. Some fundamental algebraic tools for the semantics of computation, Part 2: Signed and abstract theories. *Theoretical Comput. Sci. 31,* 3 (1984), 263–295.

[Gog88] GOGUEN, J. A. What is unification? A categorical view of substitution, equation and solution. Tech. Rep. CSLI-88-124, CSLI, Stanford University, Apr. 1988.

[Gol84] GOLDBLATT, R. *Topoi: The Categorial Analysis of Logic.* North-Holland, Amsterdam, 1984.

[GTW78] GOGUEN, J. A., THATCHER, J. W., AND WAGNER, E. G. An initial algebra approach to the specification, correctness, and implementation of abstract data types. In *Data Structuring*, R. T. Yeh, Ed., *Current Trends in Programming Methodology,* Vol. IV. Prentice-Hall, Englewood Cliffs, NJ, 1978, pp. 80–149.

[Hat82] HATCHER, W. S. *The Logical Foundations of Mathematics.* Pergamon Press, Oxford, 1982.

[HS73] HERRLICH, H., AND STRECKER, G. E. *Category Theory: An Introduction.* Allyn and Bacon, Boston, 1973.

[LS86] LAMBEK, J., AND SCOTT, P. J. *Introduction to Higher Order Categorical Logic.* Cambridge University Press, Cambridge, 1986.

[Mac71] MAC LANE, S. *Categories for the Working Mathematician.* Springer-Verlag, New York, 1971.

[PAPR85] PITT, D., ABRAMSKY, S., POIGNÉ, A., AND RYDEHEARD, D., Eds. *Category Theory and Computer Programming, Tutorial and Workshop* (Guildford, UK, Sept. 1985), *Lecture Notes in Computer Science,* Vol. 240, Springer-Verlag.

[Pie88] PIERCE, B. C. A taste of category theory for computer scientists. Tech. Rep. CMU-CS-88-203, Computer Science Dept, Carnegie Mellon University, Pittsburgh, 1988.

[RB88] RYDEHEARD, D., AND BURSTALL, R. M. *Computational Category Theory.* Prentice-Hall, 1988.

[Sch72] SCHUBERT, H. *Categories.* Springer-Verlag, Berlin, 1972.

[TBG88] TARLECKI, A., BURSTALL, R. M., AND GOGUEN, J. A. Some fundamental algebraic tools for the semantics of computation, Part 3: Indexed categories. Tech. Rep. ECS-LFCS-88-60, University of Edinburgh, July 1988.

Index

The numbers in this index, such as 8.1, refer to either definition 8.1 or to example 8.1. Numbers in italics indicate the point of definition. Numbers in boldface refer to sections. Numbers without a decimal point are page numbers. Numbers followed by "n" refer to footnotes on the indicated page.