CATEGORIES TYPES AND STRUCTURES

An Introduction to Category Theory for the working computer scientist

Andrea Asperti

Giuseppe Longo

FOUNDATIONS OF COMPUTING SERIES

M.I.T. PRESS, 1991

INTRODUCTION

The main methodological connection between programming language theory and category theory is the fact that both theories are essentially "theories of functions." A crucial point, though, is that the categorical notion of morphism generalizes the set-theoretical description of function in a very broad sense, which provides a unified understanding of various aspects of the theory of programs. This is one of the reasons for the increasing role of category theory in the semantic investigation of programs if compared, say, to the set-theoretic approach. However, the influence of this mathematical discipline on computer science goes beyond the methodological issue, as the categorical approach to mathematical formalization seems to be suitable for focusing concerns in many different areas of computer science, such as software engineering and artificial intelligence, as well as automata theory and other theoretical aspects of computation.

This book is mostly inspired by this specific methodological connection and its applications to the theory of programming languages. More precisely, as expressed by the subtitle, it aims at a self-contained introduction to general category theory (part I) and at a categorical understanding of the mathematical structures that constituted, in the last twenty or so years, the theoretical background of relevant areas of language design (part II). The impact on functional programming, for example, of the mathematical tools described in part II, is well known, as it ranges from the early dialects of Lisp, to Edinburgh ML, to the current work in polymorphisms and modularity. Recent applications, such as CAML, which will be described, use categorical formalization for the purposes of implementation.

In addition to its direct relevance to theoretical knowledge and current applications, category theory is often used as an (implicit) mathematical jargon rather than for its explicit notions and results. Indeed, category theory may prove useful in construction of a sound, unifying mathematical environment, one of the purposes of theoretical investigation. As we have all probably experienced, it is good to know in which "category" one is working, i.e., which are the acceptable morphisms and constructions, and the language of categories may provide a powerful standardization of methods and language. In other words, many different formalisms and structures may be proposed for what is essentially the same concept; the categorical language and approach may simplify through abstraction, display the generality of concepts, and help to formulate uniform definitions. This has been the case, for example, in the early applications of category theory to algebraic geometry.

The first part of this book should encourage even the reader with no specific interest in programming language theory to acquire at least some familiarity with the categorical way of looking at formal descriptions. The explicit use of deeper facts is a further step, which becomes easier with access to this information. Part II and some chapters in part I are meant to take this further step, at

least in one of the possible directions, namely the mathematical semantics of data types and programs as objects and morphisms of categories.

We were urged to write the general introduction contained in part I, since most available books in category theory are written for the "working mathematician" and, as the subject is greatly indebted to algebraic geometry and related disciplines, the examples and motivations can be understood only by readers with some acquaintance with nontrivial facts in algebra or geometry. For most computer scientists, it is not much help in the understanding of "natural transformations" to see an involved example based on tensor products in categories of sheaves. Thus our examples will be based on elementary mathematical notions, such as the definition of monoid, group, or topological space, say, and on structures familiar for readers with some acquaintance with the tools in programming language semantics. In particular, partial orders and the various categories of domains for denotational semantics will often be mentioned or introduced, as well as basic results from computability theory. For example, we will try to present the fundamental operation of "currying" for cartesian closed categories with reference to the connection between the universal function and the g del- numbering of the partial recursive functions. Partial morphisms will be presented as a generalization of a common notion in theory of computation.

Category theory may be presented in a very abstract way: as a pure game of arrows and diagrams. It is useful to reach the point where acquaintance with the formal (essentially, equational) approach is so firm that it makes sense independently of any "structural" understanding. In this book, though, we will stress the role of structures, and we will always try to give an independent meaning to abstract notions and results. Each definition and fact will be exemplified, or even derived, from applications or structures in some way indebted to computing. However, in order to stress the role of the purely equational view, the last chapters of each part (essentially chapters 7 and 11) will be largely based on a formal, computational approach. Indeed, even if mathematically very abstract, the equational arguments turn out to be particularly relevant from a computer science perspective.

The early versions of this book grew out of two graduate courses taught by Longo in Pisa, in 1984/85, and at Carnegie Mellon University, in 1987/88. Then the book was entirely revised under the influence of Asperti's work for his Ph.D. dissertation. In particular, chapters 7 and 11, the technically most difficult, are part of his dissertation.

We are indebted to several people. The joint work with Simone Martini and Eugenio Moggi in several papers directly influenced many chapters. Moreover, Eugenio suggested, in handwritten notes and electronic mail messages, the basic ideas for the categorical understanding of polymorphism via internal categories and realizability toposes. Their mathematical insights and suggestions also influenced other parts of the book.

We must acknowledge the influence on our approach of the ideas and work of Dana Scott and Gordon Plotkin, who also encouraged us and made comments on early drafts. Pino Rosolini helped

us with comments and many suggestions. Jean Yves Girard and Yves Lafont brought to our attention the tidy categorical meaning of linear logic and its applications to computing. Roberto Amadio and many students helped us by detecting errors and incompleteness in the presentation. We are looking forward to aknowledge the readers who will detect the remaining errors.

The first draft of this book was completed while the authors were visiting Carnegie Mellon University, in 1987/88. Longo would like to thank the Computer Science Dept. of CMU for its very generous hospitality while he was teaching there that academic year. The circulation of the draft, its complete revision, and the writing of the final version of the book have been made possible by the Joint Collaboration Contract ST2J-0374-C (EDB) of the European Economic Community and by the Italian CNR "Stanford-grant" #89.00002.26. The authors would like to thank INRIA, Rocquencourt, for a postdoc granted to Asperti while completing this work and l'Ecole Normale Supérieure, Paris, for inviting Longo to teach a graduate course in 1989/90 based partly on this book.

TABLE OF CONTENTS

PART I: Categories and Structures

Chapter 1

CATEGORIES

Category Theory studies "objects" and "morphisms" between them. These concepts are both primitive in Category Theory: objects are not collections of "elements," and morphisms do not need to be functions between sets (thus morphisms cannot be applied to "elements" but only composed with other morphisms). Any immediate access to the internal structure of objects is prevented: all properties of objects must be specified by properties of morphisms (existence of particular morphisms, their unicity, validity of some equations among them, and so on). This is quite similar to considering objects as "abstract data types," that is, data specifications that are independent of any particular implementation. The relevance of Category Theory for programming languages comes from the previous consideration: it offers a highly formalized language especially suited for stating abstract properties of structures. Thus, it relates to widely used programming methodologies and provides as well a formal setting for the mathematical investigation of the semantics of programming languages.

1.1 Category: Definition and Examples

As we have mentioned, Category Theory is a theory of functions, and the only basic operation is composition. The concept of **Category** embodies some abstract properties of the composition operator "$\circ$" for functions that "reasonably" must be guaranteed. In particular, if g: a$\rightarrow$b and h: b$\rightarrow$c, then there exist h $\circ$ g: a$\rightarrow$c; moreover, composition must be associative and an identity must exist for all objects.
This is the formal definition:

1.1.1 Definition *A **category** **C** is*

- *a collection $Ob_{\mathbf{C}}$ of **objects**, denoted by a, b . . . A, B . . .*
- *a collection $Mor_{\mathbf{C}}$ of **morphisms** (**arrows**), denoted by f, g . . . ,*
- *two operations **dom**, **cod** assigning to each arrow f two objects respectively called **domain** (**source**) and **codomain** (**target**) of f*
- *an operation **id** assigning to each object b a morphism id_b (the **identity** of b) such that $dom(id_b) = cod(id_b) = b$*
- *an operation " $\circ$ " (**composition**) assigning to each pair f, g of arrows with dom(f) = cod(g) an arrow f $\circ$ g such that $dom(f \circ g) = dom(g)$, $cod(f \circ g) = cod(f)$*
- *identity and composition, moreover, must satisfy the following conditions:*

***identity law**: for any arrows f, g such that cod(f) = b = dom(g)*

$id_b \circ f = f$

$g \circ id_b = g$

***associative law:** for any arrows f, g, h such that dom(f) = cod(g) and dom(g) = cod(h)*

$(f \circ g) \circ h = f \circ (g \circ h)$

We write f: a → b to denote a morphism whose source and target are respectively a and b. Given two objects a and b, the collection of all morphisms f such that f: a → b is denoted by **C**[a,b]; the writing f∈**C**[a,b] is thus a third way to express the fact that dom(f) = a, and cod(f) = b. For the moment we shall use one notation or the other indifferently

The following table lists some common categories by specifying their objects and arrows, letting the definition of their operators as an exercise for the reader:

Category	**Objects**	**Morphisms**
Set	sets	functions
Top	topological spaces	continuous functions
Vect	vector spaces	linear transformations
Grp	groups	group homomorphisms
PO	partially ordered sets	monotone functions

The intuition of the notion of "category" suggested by the previous examples is to consider the objects as a collection of "structured" sets and the morphisms as the "associated" or "acceptable" functions with respect to the structure. This is too restrictive, though, since no requirement is made in the definition which may force the morphisms to be "single valued" or to be functions in extenso: a simple example is the category **Rel** with sets as objects and relations as morphisms.

The simplest category has only one object and one arrow (the identity for that object): this category is usually called **1**. Note that, by definition, if **C** is a category, then every object b of **C** has an identity id_b: b→b. The identity is unique, since if id_b' is another identity for b, then for the *identity law*, $id_b' = id_b \circ id_b' = id_b$. A category is called **discrete** if every arrow is the identity of some object: in this case a category is fully determined by the collection of its objects. **1** is a discrete category.

A category is called a **preorder** if for every pair of objects a, b there is *at most* one morphism f: a→b. The reason for the name is that a preorder category is fully determined by a preordering relation among its objects. Indeed, in a preorder **C**, there is only one way that composition may be defined; thus **C** is known when the collection of morphisms $Mor_{\mathbf{C}}$ and the operations dom and cod are known. But every arrow f: a→b may be identified with the pair (a,b), since once the source and

target are known there is no choice about what the arrow is to be; thus, all the information about the category **C** is given by the relation $R_{\mathbf{C}} = \{(a,b) / \text{there is an arrow } f \in \mathbf{C}[a,b]\}$, that is, by a preorder relation.(Exercise: prove that the relation $R_{\mathbf{C}}$ is a preorder for every category **C**).

Every discrete category is a preorder. The simplest nondiscrete category which is a preorder is the category **2**, which has two objects, let us call them 0 and 1, and three arrows: the two identities id_0, id_1 and an arrow (0,1): $0 \to 1$. In a similar way we can define for each natural number n a preorder category **n**, from the usual ordering on the set {0,1, . . . n-1}. Preorder categories have a common property: they may have plenty of objects, but given two objects, there exists at most one morphism between them.

A dual situation is given by monoids, viewed as categories. A **monoid** is a set having an associative binary operation and an identity element. A category with just one object yields a monoid, where composition of morphisms is the binary operation. Conversely, any monoid $(A, \cdot)$ is a category with just one object. For example, the category with the set of natural numbers as unique object and the recursive functions as morphisms yields the monoid of the recursive functions.

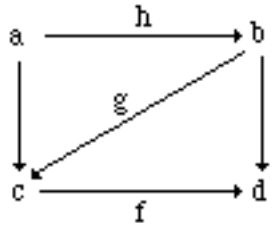
As well as preorders, another example where objects are not necessarily understood as "structured sets" is given by **deductive systems** as categories. In these categories propositions are objects and each morphism $f : a \to b$ corresponds to (a suitable equivalence class of) a proof of $a \vdash b$ (a **entails** b). Observe that a category is obtained easily in the presence of the identical entailment $i_a : a \to a$ and the associative composition of proofs

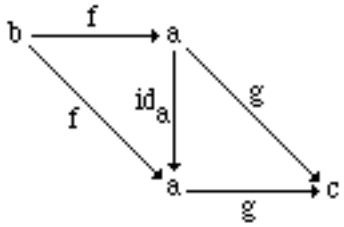
$$\frac{f : a \to b \qquad g : b \to c}{g \circ f : a \to c}$$

This approach to deduction is very relevant in the categorical understanding of logics of a constructive nature, such as Intuitionistic Logic, where the intended interpretation of proofs is given by (effective) operations. It will be the main paradigm for understanding the relation between types and objects investigated in the second part of this book.

1.2 Diagrams

An important tool in the practice of Category Theory is the use of **diagrams** for representing equations. In a diagram a morphism $f \in \mathbf{C}[a,b]$ is drawn as an arrow from a to b labeled f. A diagram **commutes** if the composition of the morphism along any path between two fixed objects is equal. For example, the *associative* and *identity laws* of the definition of "category" may be nicely visualized by the following commuting diagrams:





Diagrams are a typical way, in Category Theory, to describe equational reasoning and turn out to be particularly effective when dealing with several equations at a time. In particular, assertions such as "if $diagram_1$ and . . . $diagram_n$ commute, then $diagram$ commutes" express conditional statements about equalities.

We hope that the reader, while using this book, will acquire some familiarity with diagrams and will learn how to go back and forth from diagrams to equations. Our extended use of equations in this book comes from our desire to stress the "computational" nature of most categorical reasoning.

1.3 Categories out of Categories

A main feature of Category Theory is the facility to define new, more structured categories out of simpler ones. In this section we consider only a few simple constructions; a number of other examples occur throughout the book.

1.3.1 Definition *A category* $\mathbf{D}$ *is a* ***subcategory*** *of a category* $\mathbf{C}$*, if*

1. $Ob_{\mathbf{D}} \subseteq Ob_{\mathbf{C}}$;
2. *for all* a, b *in* $Ob_{\mathbf{D}}$, $\mathbf{D}[a,b] \subseteq \mathbf{C}[a,b]$;
3. *composition and identities in* $\mathbf{D}$ *coincide with those of* $\mathbf{C}$.

A subcategory is ***full*** *if for all* a, b *in* $Ob_{\mathbf{D}}$ $\mathbf{D}[a,b] = \mathbf{C}[a,b]$.

A full subcategory is fully determined by its collection of objects.

1.3.2 Definition *The* ***dual*** *category* $\mathbf{C}^{\mathbf{op}}$ *of a category* $\mathbf{C}$ *has the same objects and the same morphisms of* $\mathbf{C}$, $id^{op}_b = id_b$, $dom^{op}(f) = cod(f)$, $cod^{op}(f) = dom(f)$, *and* $f \circ^{op} g = g \circ f$.

Note that $\mathbf{C^{op}}[b,a] = \mathbf{C}[a,b]$ and $(\mathbf{C^{op}})^{\mathbf{op}} = \mathbf{C}$.

Exercise $\mathbf{Set^{op}}$ is a subcategory of **Rel**, but not of **Set**. Is it a full subcategory?

Duality is a very powerful technique of Category Theory. If P is a generic proposition expressed in the language of Category Theory, the dual of P (P^{op}) is the statement obtained by replacing the

word "dom" by "cod," "cod" by "dom," "$g \circ h$" by "$h \circ g$." If P is true in a category **C**, then P^{op} is true in $\mathbf{C}^{op}$; if P is true in every category, then also P^{op} is, since every category is the dual of its dual.

Duality may be applied to diagrams as well: given a diagram in a category **C**, the dual diagram in $\mathbf{C}^{op}$ is obtained by simply reverting the arrows; of course, a dual diagram commutes if and only if the original one does.

1.3.3 Definition *Given two Categories* ***C*** *and* ***D****, the* ***product category C×D*** *has for objects the pairs (a,b) where a and b are respectively objects of* ***C*** *and* ***D****, and for morphisms pairs (f,g):* $(a,b) \to (a',b')$ *where* $f: a \to a'$ *and* $g: b \to b'$ *are respectively morphisms of* ***C*** *and* ***D****. Finally,* $id_{(a,b)} = (id_a, id_b)$ *and* $(f,g) \circ (f',g') = (f \circ f', g \circ g')$.

1.3.4 Definition *Given a category* ***C*** *and an object* a *in* $Ob_{\mathbf{C}}$*, the* ***category*** $\mathbf{C}\downarrow a$ ***of objects over*** a *is so defined:* $Ob_{\mathbf{C}\downarrow a} = \{f \in Mor_{\mathbf{C}} \mid cod(f) = a\}$*; given two objects* $f: b \to a$, $g: c \to a$*, a morphism with source* f *and target* g *is an arrow* $h \in \mathbf{C}[b, c]$ *such that* $g \circ h = f$*. Identities and composition in* $\mathbf{C}\downarrow a$ *are inherited from* ***C****.*

In case **C** is **Set** in the above definition, it is useful to think of an object $g: B \to A$ in $\mathbf{Set}\downarrow A$ as an A-indexed family of disjoint sets, namely, $\{g^{-1}(a)\}_{a \in A}$ (these sets are the inverse images of elements in A under g). Then $h: B \to B'$ is a morphism from $g: B \to A$ to $g': B' \to A$ if and only if it is consistent with the "decomposition" of B and C induced by g and g', i.e., if and only if (iff) $\forall b \; b \in g^{-1}(a) \Rightarrow h(b) \in g'^{-1}(a)$.

Since the intended meaning behind the construction of a category $\mathbf{C}\downarrow I$ is that to consider an object $g: A \to I$ as a collection $\{\{i\} \times g^{-1}(i)\}_{i \in I}$, it is usual to call $\mathbf{C}\downarrow I$ a **slice category** over I (denoted **C**/I). An object $g: A \to I$ of the slice category is then called a **generalized object** of **C** at stage I. A **section** of $g: A \to I$ is a function $s: I \to A$ such that $g \circ s = id_I$; the idea is that s gives, for each index $i \in I$, an element $s(i) \in g^{-1}(i)$.

Exercise Define the dual notion, that is, the category $\mathbf{C}\uparrow a$ of objects*under* a, whose objects are the arrows with source a.

1.4 Monic, Epic, and Principal Morphisms

A function f between two sets A and B is called "injective" when, for all $a, a' \in A$, if $f(a) = f(a')$ then $a = a'$. In particular, given any two functions $g, h : C \to A$, if for all $c \in C$ $f(g(c)) = f(h(c))$, then for all $c \in C$ $g(c) = h(c)$ or, also, if $f \circ g = f \circ h$ then $g = h$. Thus, every injective function behaves like a left identity (it is left cancellable). The converse is also true: given $f: A \to B$, if

for any pair of functions $g, h : C \to A$, $f \circ g = f \circ h$ implies $g = h$, then f is injective. For suppose otherwise: then there are a and a' such that $f(a) = f(a')$ but $a \neq a'$; define then g and h by $g(c) = a$ for all $c \in C$, and $h(c) = a'$ for all $c \in C$; of course $f \circ g = f \circ h$ but $g \neq h$, that is, a contradiction.

We have proved thus that a function f is injective if and only if $f \circ g = f \circ h$ implies $g = h$. In a similar way it is not difficult to prove that f is surjective if and only if $g \circ f = h \circ f$ implies $g = h$. These considerations motivate the following definitions.

1.4.1 Definition. *Let* $\boldsymbol{C}$ *be a category and* $a, b \in Ob_{\boldsymbol{C}}$. *Then*

i. an arrow $h \in \boldsymbol{C}[a,b]$ *is* ***epic*** *(is an* ***epimorphism****) iff*

$$g \circ h = f \circ h \Rightarrow g = f;$$

ii. an arrow $h \in \boldsymbol{C}[a,b]$ *is* ***monic*** *(is a* ***monomorphism****) iff*

$$h \circ g = h \circ f \Rightarrow g = f;$$

iii. an arrow $h \in \boldsymbol{C}[a,b]$ *is* ***iso*** *(is an* ***isomorphism****) iff there exists* $g \in \boldsymbol{C}[b,a]$ *such that*

$$g \circ h = id \text{ and } h \circ g = id.$$

Two objects a and b are **isomorphic** ($a \cong b$) if there exists an isomorphism $h \in \mathbf{C}[a,b]$. Clearly, any isomorphism is monic and epic; the converse, though, does not need to be true (see the example and the exercises below).

A monic (or epic) $h \in \mathbf{C}[a,b]$ (or $h' \in \mathbf{C}[a,b]$) is **split** if there exist $g \in \mathbf{C}[b,a]$ (or $g' \in \mathbf{C}[b,a]$) such that $g \circ h = id$ ($h' \circ g' = id$).

Although the intuition of regarding mono- and epimorphisms as injective and surjective maps is correct for many interesting categories, sometimes it can be misleading. Consider, say, the category **Mon** of monoids and the inclusion inc from ω, the positive integers, into $\mathbf{z}$, the relative ones. Clearly mono, inc is also epi, though. As a matter of fact, take $g,h \in \mathbf{Mon}[\mathbf{z},a]$ for some monoid a, and write $\backslash g(n)$ for $g(-n)$. Then $g \circ inc = h \circ inc$ implies $g = h$ for $g(-n) = \backslash g(n) = \backslash h(n) = h(-n)$ (that is, the behavior of the monoids' homomorphism g or h on $\mathbf{z}$ is entirely determined by their behavior on ω). As a side consequence, we may also conclude that not every arrow that is both monic and epic is an isomorphism: this is clearly in contrast to the set-theoretic intuition.

Exercises

1. Give an epi which is not surjective in **Top**.
2. Find a counterexample for the following assertion: let $\mathbf{C}$ be a category; if $f \in \mathbf{C}[a,b]$ and $g \in \mathbf{C}[b,a]$ are mono, then a is isomorphic to b. (Note that the assertion is true in **Set.**)
3. Prove that a split monic is an iso.

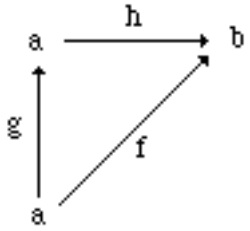
1.4.2 Definition *Let* $\boldsymbol{C}$ *be a category and* $a, b \in Ob_{\boldsymbol{C}}$. *Then*

i. an arrow $h \in \boldsymbol{C}[a,b]$ *is a* ***principal*** *morphism iff*

$\forall f \in \mathbf{C}[a,b]\ \exists g \in \mathbf{C}[a,a]\ \ f = h \circ g$;

ii. a pair of arrows $f \in \mathbf{C}[a,b]$ *and* $g \in \mathbf{C}[b,a]$ *is a* ***retraction pair*** *iff* $g \circ f = id$. *Then, a is called a* ***retract*** *of b* $(a<b)$ *via the retraction pair (f,g).*

By diagrams, h is principal iff for all f there is a g such that



Principal morphisms have been inspired by recursion theory; the idea they are based on essentially corresponds to a classical notion of reducibility (see the category **EN** in section 2.2 below).

1.4.3 Proposition *Let* $\mathbf{C}$ *be a category and* $a, b \in Ob_{\mathbf{C}}$. *Then*

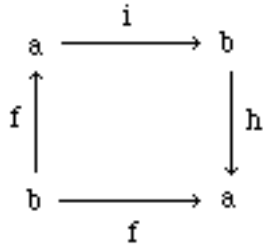
1. if $a<b$ *via* (i,h), *then h is epi and principal, i is mono;*

2. if $h \in \mathbf{C}[a,b]$ *is principal and there exists an epi* $k \in \mathbf{C}[a,b,]$, *then h is epi;*

3. if $a<b$ *and* $f \in \mathbf{C}[b,a]$ *is principal, then there exists* $g \in \mathbf{C}[a,b]$ *such that* $a<b$ *via* (g,f).

Proof 1. $g \circ h = f \circ h \Rightarrow g \circ h \circ i = f \circ h \circ i \Rightarrow g = f$, for $h \circ i = id$.

The proof that h is principal is a simple diagram chase:

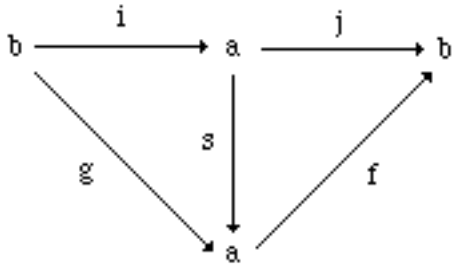


That is, $\forall f\ \exists g\ f = h \circ g$. Just take $g = i \circ f$; then $h \circ g = h \circ i \circ f = f$.

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

2. $g \circ h = f \circ h \Rightarrow g \circ k = g \circ h \circ g' = f \circ h \circ g' = f \circ k$ (for a suitable g') $\Rightarrow g = f$.

3. Let $a<b$ via (j,i). Since f is principal, $\exists s \in \mathbf{C}[b,b]\ \ j = f \circ s$. Then, for $g = s \circ i$, one has $f \circ g = j \circ i = id_a$. As a diagram,



◆

Exercises

1. Characterize retractions in terms of split monos and epis.

2. Show that, given a category **C,** one can define a category $\mathbf{C}^{\mathbf{Ret}}$ whose objects are the same of **C** and whose morphisms are retraction pairs in **C**, that is $F \in \mathbf{C}^{\mathbf{Ret}}[a,b]$ iff F=(f,g) and a<b via (f,g) in **C**.

If $(f:a \to b, g:b \to a)$ is a retraction pair, then the function $h = f \circ g: b \to b$ is idempotent, that is, $h \circ h = h$. Indeed, $h \circ h = (f \circ g) \circ (f \circ g) = f \circ (g \circ f) \circ g = f \circ g = h$. This property suggests the following definition:

1.4.4 Definition *Given a category* ***C*** *and an object* $b \in Ob_{\mathbf{C}}$*, the* ***category of idempotents on b*** $(\mathbf{Ret_b})$ *is so defined:*

$$Ob_{\mathbf{Ret_b}} = \{ f \in \mathbf{C}[b,b] \,/\, f \circ f = f \}$$
$$Mor_{\mathbf{Ret_b}} = \{ (f, k, g) \,/\, f, g \in Ob_{\mathbf{Ret_b}},\ k \in \mathbf{C}[b,b],\ k = g \circ k \circ f \}$$
$$dom((f, k, g)) = f,\ cod((f, k, g)) = g$$
$$id_f = (f, f, f)$$
$$(f, k, g) \circ (g', k', f) = (g', k \circ k', g)$$

We leave as an exercise for the reader to check the identity and associative laws for the previous category. $\mathbf{Ret_b}$ will be used in several places because of its relevance to this book.

1.5 Subobjects

The concept of *subobject* is the categorical version of the set-theoretical *subset* . The main idea is to regard a subset A of a given object B as a monomorphism $f: D \to B$ (intuitively, a monomorphism f such that "f(D) = A"). Of course, many different monic arrows may define the same subset; thus, it is necessary to introduce a reasonable equivalence relation, and define subobjects up to this equivalence.

Let **C** be a category. If $f: b \to a$ and $g: c \to a$ are two monic arrows with common target a, then we say $f \leq g$ if and only if there exists $h: b \to c$ such that $g \circ h = f$. Note that in this case, the unique h must be monic too, indeed $h \circ k = h \circ k' \Rightarrow g \circ h \circ k = g \circ h \circ k' \Rightarrow f \circ k = f \circ k' \Rightarrow k = k'$.

Exercise Prove that the preorder ≤ is the full subcategory of $\mathbf{C}\downarrow a$ determined by monomorphisms only.

When $f \leq g$ and $g \leq f$ we write $f \cong g$. Then $\cong$ is an equivalence relation among the monomorphisms with common target a (prove it as an exercise); the equivalence classes of this equivalence relation are called subobjects of a.

1.5.1 Definition *Let a be an object of a category* $\boldsymbol{C}$. *A* ***subobject*** *[f] of a is an equivalence class of a monomorphism* $f: b \rightarrow a$, *with respect to the equivalence relation* $\cong$ *defined above.*

Very often, we shall make no distinction between equivalence classes and their representatives, and we shall denote a subobject with a single monomorphism.

It should be clear that the categorical approach to "subsets" carries more information than the set-theoretic one. Monomorphisms, like all morphisms, preserve the structural information of the category. For example, in the category **Grp** of groups subobjects are subgroups: (mono)morphisms must take the identity to the identity and preserve the group operation. Similarly consider the category of p.o.sets (partially ordered sets) with a bottom element. A subobject of one such p.o.set must be a structured subset as well, and it must contain an element smaller than all the others.

References: Any book in Category Theory, such as MacLane (1971) Herrlich and Strecker (1973), Arbib and Manes (1975), Barr and Wells (1985), Rydeheard and Burstall (1988). The specific notions and categories introduced (such as retractions) will be used later in more structured settings, with the appropriate references.

Chapter 2

CONSTRUCTIONS

In this chapter we consider some fundamental categorical constructions, i. e., particular objects (and morphisms) that satisfy a given set of axioms described in the language of Category Theory. Since in this language there is no way to look at the internal membership structure of objects, all the concepts must be defined by their relations with other objects, and these relations are established by the existence and the equality of particular morphisms. This property of the categorical language, if compared to the traditional set-theoretic jargon, may be well understood by an analogy with computer science; namely, as we already mentioned, the categorical description corresponds to an abstract data specification, while the traditional set-theoretic approach is more similar to a concrete implementation.

2.1 Initial and Terminal Objects

2.1.1 Definition *Let* $\mathbf{C}$ *be a category. An object* 0 *is* ***initial*** *iff for any* $b \in Ob_{\mathbf{C}}$ *there is a unique* $f \in C[0,b]$.

The typical example of an initial object is the empty set $\emptyset$ in **Set**; indeed the empty function (i.e., the function whose graph is empty) is the unique arrow with $\emptyset$ for source.

A more interesting example is the following. Let $\mathbf{\Sigma}$ be a signature. The class $Alg_{\mathbf{\Sigma}}$ of $\mathbf{\Sigma}$-algebras with $\mathbf{\Sigma}$-homomorphisms as arrows forms a category. $Alg_{\mathbf{\Sigma}}$ has an initial object $T_{\mathbf{\Sigma}}$ which is called $\mathbf{\Sigma}$-word-algebras, or also Herbrand Universe for $\mathbf{\Sigma}$. The set $T_{\mathbf{\Sigma},s}$ (the carrier of $T_{\mathbf{\Sigma}}$ of sort s) is just the set of all well-formed expressions of sort s. If $\mathbf{\Sigma}$ is derived by a context free grammar (that is: sorts are nonterminals and operator symbols are productions of the grammar), then $T_{\mathbf{\Sigma},s}$ is the set of all parse trees for derivations in the grammar from the nonterminal s. In general the initial $\mathbf{\Sigma}$-algebra $T_{\mathbf{\Sigma}}$ corresponds to the syntax of a language of signature $\mathbf{\Sigma}$. Any other $\mathbf{\Sigma}$-algebras A in $Alg_{\mathbf{\Sigma}}$ is a possible semantic domain; the semantic function (interpretation) is the unique homomorphism from $T_{\mathbf{\Sigma}}$ to A.

Initiality is the simplest *universal* notion in Category Theory, since it is given by the existence and unicity of morphisms satisfying certain properties. This method is used everywhere in Category Theory.

2.1.2 Proposition *If* 0 *and* $0'$ *are two initial objects in a category* $\mathbf{C}$*, then they are isomorphic.*

Proof. Let $i: 0 \to 0'$, $j: 0' \to 0$ the morphisms respectively given by the initiality of 0 and $0'$. Then $j \circ i: 0 \to 0$, but also $id_0: 0 \to 0$, and since by initiality of 0, there is exactly one morphism in $\mathbf{C}[0,0]$, then $j \circ i = id_0$; in the same way, by initiality of $0'$ we have $i \circ j = id_{0'}$.♦

We will now show how *duality* can be used to define new concepts and to prove new assertions. Let P(c) be the property "for any $b \in Ob_{\mathbf{C}}$ there is a unique f, such that dom(f) = c, cod(f) = b." By definition c is initial iff P(c) holds; that is, P defines initiality. The dual statement of P is $P^{op}(c)$ = "for any $b \in Ob_{\mathbf{C}}$ there is a unique f, such that cod(f) = c, dom(f) = b." Usually the dual Q^{op} of a property Q defines a concept named by prefixing "co-" to the name of the property Q. In our case, we say that P^{op} defines coinitiality. An object c such that $P^{op}(c)$ holds, is called **co-initial**. Anyway it is common practice to assign to every coentity an independent name which better expresses its properties; for example, a coinitial object is known as **terminal** object. Note that an initial object is coterminal.

Terminal objects are usually represented with the number 1 or with the letter t. The unique morphism from an object a to the terminal object t is usually written $!a: a \to t$.

Any singleton set is terminal in **Set.** In the category **2** one object is initial and the other one is terminal. If c is initial in **C**, then it is terminal in the dual category $\mathbf{C}^{\mathbf{op}}$.

Consider now the statement P_1 = "*If 0 and 0' are two initial objects, then they are isomorphic*." Its dual is: P_1^{op} = "*If 0 and 0' are two terminal objects, then they are* isomorphic." (the property to be an isomorphism is the dual of itself: prove it as an exercise.) By our discussion of duality in chapter 1 and, since by proposition 2.1.2 P_1 holds in every category, P_1^{op} also does. We conclude the following:

2.1.3 Proposition *If 0 and 0' are two terminal objects in a category* ***C****, then they are isomorphic.*

Proof By duality and by proposition 2.1.2.♦

An object c in a category **C** may be both initial and terminal. An example is the unit group in **Grp**; in this case, it is called a **zero** object.

In **Set**, a morphism from the singleton {*} to a set A defines an element of A. For this reason an arrow from a terminal object t to an object a in a generic category **C** is usually called an **element** or a **point** of a. In this case, however, the set-theoretic intuition must be used very carefully, because it is quite common to work in categories where the categorical notion does not reflect the behavior of elements in **Set**. For example the set-theoretic intuition would suggest that every non-initial object must have at least one element: but consider the partial order category **3** which has three object $0 \leq 1 \leq 2$; clearly 0 is initial and 2 is terminal, 1 is non-initial but has no elements. Similarly, in **Set** two arrows are equal iff they coincide on all points, or, more formally, given functions f and g,

one has $f \neq g$ iff there is an element x of their domain such that $f \circ x \neq g \circ x$. However, in a generic category **C** with terminal object t, this is not necessarily true.

2.1.4 Definition. *Let* $\mathbf{C}$ *be a category.* $t \in Ob_{\mathbf{C}}$ *is a* ***generator*** *iff for all* $a,b \in Ob_{\mathbf{C}}$ *and all* $f,g \in \mathbf{C}[a,b]$, *one has:* $f \neq g \Rightarrow \exists h \in \mathbf{C}[t,a]\ f \circ h \neq g \circ h$.

$\mathbf{C}$ *has* ***enough points*** *(or is* ***well pointed****), if there exists a generator* t *that is terminal in the given category.*

In short, a category has enough points when the arrows from the terminal object allow to discriminate between morphisms, similarly as for elements over **Set**. Of course, it is not a surprise that the set-theoretic notions of "element" and of "extensionality" are somewhat awkward to deal with in the language of Category Theory.

2.2 Products and Coproducts

The categorical product is merely a "structural" generalization of the notion of Cartesian product of sets. Given two sets A and B, their cartesian product is:

$$A \times B = \{\langle x,y\rangle / x \in A, y \in B\}$$

Associated with this set there are two special maps $p_A: A \times B \to A$, $p_B: A \times B \to B$ called projections, such that for every $\langle x,y\rangle$ in $A \times B$ $p_A(\langle x,y\rangle) = x$, $p_B(\langle x,y\rangle) = y$. Note that for every c in $A \times B$, $\langle p_A(c), p_B(c)\rangle = c$.

Let C be another set, and $f: C \to A$, $g: C \to B$. Define $\langle f,g\rangle: C \to A \times B$ by $\langle f,g\rangle(c) = \langle f(c),g(c)\rangle$ for every $c \in C$. Then, for every $c \in C$, $p_A(\langle f,g\rangle(c)) = p_A(\langle f(c),g(c)\rangle) = f(c)$, that is, $p_A \circ \langle f,g\rangle = f$. In the same way, we obtain $p_B \circ \langle f,g\rangle = g$. Conversely, let $h: C \to A \times B$. Then for every $c \in C$, $\langle p_A \circ h, p_B \circ h\rangle(c) = \langle p_A(h(c)), p_B(h(c))\rangle = h(c)$, that is, $\langle p_A \circ h, p_B \circ h\rangle = h$.

The previous consideration suggests the following definition:

2.2.1 Definition *Let* $\mathbf{C}$ *be a category, and* $a,b \in Ob_{\mathbf{C}}$. *The* ***categorical product*** *of* a *and* b *is an object* $a \times b$ *together with two morphisms* $p_a: a \times b \to a$, $p_b: a \times b \to b$, *and for every object c an operation* $\langle\ ,\ \rangle_c : \mathbf{C}[c,a] \times \mathbf{C}[c,b] \to \mathbf{C}[c,a \times b]$ *such that for all morphisms* $f: c \to a$, $g: c \to b$, $h: c \to a \times b$, *the following equations hold:*

ia. $p_a \circ \langle f,g\rangle_c = f$;

ib. $p_b \circ \langle f,g\rangle_c = g$;

ii. $\langle p_a \circ h, p_b \circ h\rangle_c = h$.

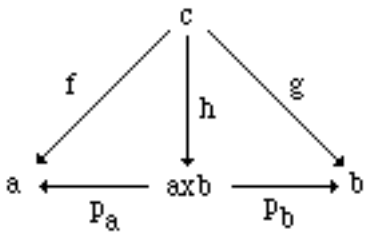
It is common practice to omit the subscript c in $\langle\ ,\ \rangle_c$ when its meaning is clear from the context.

The operation $< , > : \mathbf{C}[c,a]\times\mathbf{C}[c,b]\to\mathbf{C}[c,a\times b]$ of a categorical product is a bijection: its inverse is the operation that takes every arrow $h\in\mathbf{C}[c,a\times b]$ to the pair $(p_a \circ h, p_b \circ h)\in\mathbf{C}[c,a]\times\mathbf{C}[c,b]$.

The proof that these operations are inverse of each other is stated above in definition 2.2.1. Conversely, given a bijective operation $< , > : \mathbf{C}[c,a]\times\mathbf{C}[c,b]\to\mathbf{C}[c,a\times b]$ which satisfies (ia) and (ib), then (ii) is necessarily true. Indeed, let $h\in\mathbf{C}[c,a\times b]$. Then, since $< , >$ is bijective, there is a pair $(f,g)\in\mathbf{C}[c,a]\times\mathbf{C}[c,b]$ such that $h = <f,g>$; but $f = p_a \circ <f,g> = p_a \circ h$ and analogously, $g = p_b \circ <f,g> = p_b \circ h$; thus, $h = <p_a \circ h, p_b \circ h>$.

The last consideration leads us to a more compact but equivalent definition of a categorical product.

2.2.2 Definition *Let* $\mathbf{C}$ *be a category, and* $a,b\in Ob_{\mathbf{C}}$. *The* ***categorical product*** *of* a *and* b *is an object* $a\times b$ *together with two morphisms* $p_a: a\times b\to a$, $p_b: a\times b\to b$, *such that, for any* $f\in\mathbf{C}[c,a]$ *and* $g\in\mathbf{C}[c,b]$, *there exists exactly one* $h\in\mathbf{C}[c,a\times b]$ *such that the following diagram commutes*



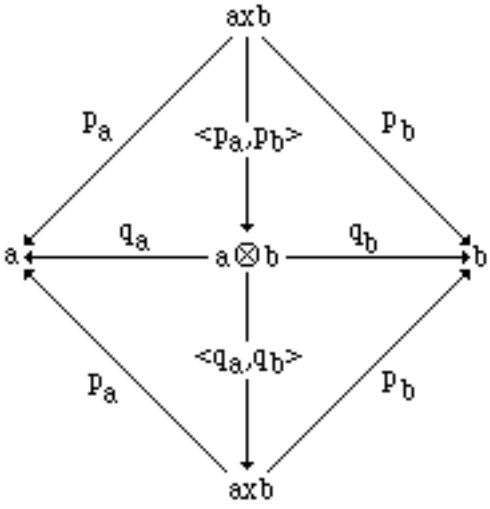
2.2.3 Definition *For* $f\in\mathbf{C}[a,c]$ *and* $g\in\mathbf{C}[b,d]$, *set* $f\times g = <f\circ p_a, g\circ p_b> : a\times b\to c\times d$.

Exercise Prove that for all arrows $h: e\to a$ and $k: e\to b$, $f\times g\circ<h,k> = <f\circ h, g\circ k>$.

2.2.4 Proposition *In a category, the product is unique (up to isomorphisms), if it exists.*

Proof Let $a\otimes b$ be an alternative product with projections q_a and q_b.

Then $<q_a,q_b> \circ <p_a,p_b>$ is the unique morphism such that the following diagram commutes:



Since $id_{a\times b}$ also does the same job, $id_{a\times b} = \langle p_a,p_b\rangle\circ\langle q_a,q_b\rangle$.
By symmetry, one also has $\langle p_a,p_b\rangle \circ \langle q_a,q_b\rangle = id_{a\otimes b}$.♦

Exercise. Prove the following facts:
1. $a \cong a'$ and $b \cong b'$ imply $a\times b \cong a'\times b'$.
2. $a\times b \cong b\times a$.

2.2.5 Definition *A category* ***C*** *is* ***Cartesian*** *(**C** is a CC) iff*

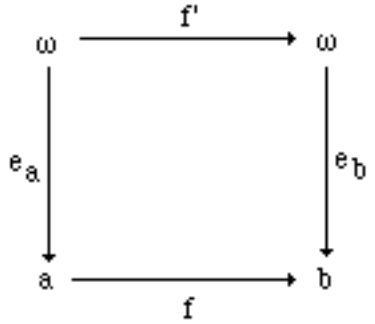
i. it contains a terminal object t;

ii. every pair $a,b\in Ob_{\mathbf{C}}$ *has a categorical product* $(a\times b, p_{a,b,1}: a\times b\to a, p_{a,b,2}: a\times b\to b)$

Exercises

1. Generalize the definition of a product of two objects to arbitrary products.
2. Prove that a Cartesian category **C** always contains all finite products.
3. Let **C** be a CC and let t be its terminal object. Prove that for all b in $Ob_{\mathbf{C}}$, $b \cong t\times b \cong b\times t$.

Examples The categories **Set**, **Top**, **Grp** are all Cartesian.
An interesting Cartesian category in Computability Theory is the category **EN** of numbered sets. Objects in **EN** are pairs $\underline{a} = (a,e_a)$, where a is a countable set and $e_a: \omega\to a$ is an onto map (an *enumeration* of a). $f\in$**EN**$[\underline{a},\underline{b}]$ iff for some total recursive f' the following diagram commutes:



We say that f' **represents** f. The product is easily obtained by using any effective pairing of ω^2, $[,]: \omega\times\omega\to\omega$.

A typical numbered set which is worth studying is $\underline{PR} = (PR,\phi)$, the partial recursive functions with a Goedel numbering $\phi: \omega\to PR$. Then **EN**$[\underline{PR},\underline{PR}]$ are exactly the type two recursive functionals. Of course, this is also a countable set. It is not trivial, though, to construct an "acceptable" enumeration of it. This will be an important issue in the sequel.

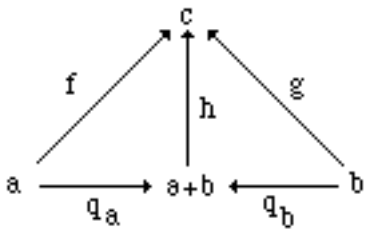
Exercises

1. Let $\underline{\omega} = (\omega,id)$ in **EN**. Then $f\in$**EN**$[\underline{\omega},\underline{PR}]$ iff $\exists f'\in PR$ $f'([x,y]) = f(x)(y)$. Moreover, $g\in$ **EN**$[\underline{\omega},\underline{PR}]$ is principal iff g is an acceptable Goedel numbering of PR, in the sense of classical recursion theory.

2. Let **C** be a CC, and V be an object such that $V \times V < V$. Then the category of retractions on V (see definition 1.4.4) is a CC.

The dual of the notion of a product is the **coproduct** a+b with embeddings q_1, q_2.

2.2.6 Definition. *Let **C** be a category, and $a,b \in Ob_{\mathbf{C}}$. The **coproduct** of a and b is an object a+b together with two morphisms $q_a: a \to a+b$, $q_b: b \to a+b$ such that, for any $f \in \mathbf{C}[a,c]$ and $g \in \mathbf{C}[b,c]$, there exists exactly one $h \in \mathbf{C}[a+b,c]$ such that the following diagram commutes*



By duality, the coproduct is unique (up to isomorphisms).

Examples

1. In **Set** the coproduct is the disjoint union.
2. In a preorder **P** the product is the greatest lower bound, if it exists. The coproduct is the least upper bound, if it exists.
3. Let **CPO** be the category of complete partial orders with continuous functions with respect to the order or Scott topology. **CPOS** is the subcategory with only strict functions, i.e., morphisms always take the least element $\perp$ to the least element of the target space. It is easy to see that both categories are Cartesian. The coproduct in **CPOS** is given by the coalesced sum, i.e., the disjoint union except for the identification of the two least elements. On the other hand, there is no coproduct in **CPO**. This may be seen by observing that in **CPO** one may have $f(\perp) \neq g(\perp)$, by which the coalesced sum fails to give a coproduct; an extra common least element (disjoint sum) may give more than one extension of the required $\langle f,g \rangle^{op}$.

2.3 Exponentials

In the connection we mentioned between Category Theory and Computation Theory, as "theories of functions," a fundamental aspect still has to be taken care of. In either case, we may be interested in computing with procedures as arguments. That is, we may need to describe higher type functions.

So far we have only become familiar with Cartesian categories, where the object a×b, representing the product, is defined. Thus, the notion of morphism taking morphisms as arguments doesn't yet make sense. What we first need, then, is a further closure property, namely, the existence within the

category of an object b^a which suitably represents the set of morphisms from b to a. With an informal reference to typing in programming, the key property of the objects, which represent the sets of morphisms, provides an interpretation to a common construct in actual programming, namely, the identification of types such as $A\times B\to C$ and $A\to(B\to C)$. This corresponds to the following important uniformity property of programs of several arguments, which is directly inherited from classical Recursion Theory.

Let $\{\phi_i\}_{i\in\omega} = PR$ be an acceptable Gödel numbering of the partial recursive functions and $[,]: \omega\times\omega\to\omega$ be an effective pairing. Define then, as usual, $f: \omega\times\omega\to\omega$ is a binary partial recursive function iff $\exists f'\in PR\ f(x,y) = f'([x,y])$ (similarly, for n-ary functions, $n \geq 2$). By this and by the s-m-n iteration theorem one immediately has $f: \omega\times\omega\to\omega$ is partial recursive iff $\exists s\in R\ \phi_{s(x)}(y) = f(x,y)$.

Thus, a two-(or more) argument function f is computable iff it is computable in each argument and the function $x \mapsto f(x,\_)$ is also "computable," i.e., $\exists s\in R\ \phi_{s(x)} = f(x,\_)$. In other words, in computability theory, f is in $\omega\times\omega\to\omega$ iff $x \mapsto f(x,\_)$ is in $\omega\to(\omega\to\omega)$. Similarly, the category-theoretic closure property we need concerns the existence, for any $f: c\times a\to b$, of a morphism within the category, which does the same job as s or $x \mapsto f(x,\_)$ in recursion theory. We will call it $\Lambda(f)$.

Exercise For $n \geq 2$, not every n-ary function which is computable in each argument needs to be computable. (Hint: take g total nonrecursive and set $f(x,y) = g(\min\{x,y\})$).

2.3.1 Definition *Let* $\mathbf{C}$ *be a Cartesian category, and* $a,b\in Ob_{\mathbf{C}}$. *The* ***exponent*** *of a and b is an object* b^a *together with a morphism* $eval_{a,b}: b^a\times a\to b$ *(evaluation map), and for every object c an operation* $\Lambda_c : \mathbf{C}[c\times a,b]\to\mathbf{C}[c,b^a]$ *such that for all morphisms* $f: c\times a\to b$, $h: c\to b^a$, *the following equations hold:*

$$\beta).\quad eval_{a,b}\circ(\Lambda(f)\times id_a) = f\ ;$$

$$\eta).\quad \Lambda_c(eval_{a,b}\circ(h\times id_a)) = h\ .$$

(We may omit the indices when unambiguous, as usual.)

In **Set** the exponent set of A and B is $B^A = \{f \mid f \text{ is a function from } A \text{ to } B\}$, thus $B^A = \mathbf{Set}[A,B]$. The function $eval: B^A\times A\to B$ is given by the rule: $eval(\langle f,x\rangle) = f(x)$.

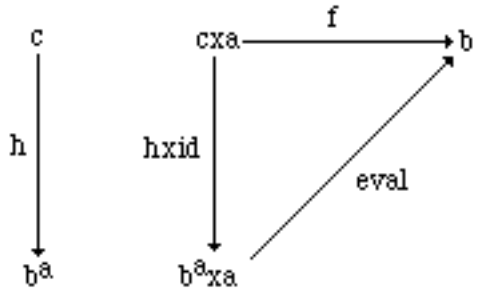
$\Lambda: \mathbf{Set}[C\times A,B]\to\mathbf{Set}[C,B^A]$ takes every function $f: C\times A\to B$ to the function $\Lambda(f): C\to B^A$ defined by $\Lambda(f)(c) = \lambda a.f(c,a)$, where $\lambda a.f(c,a)\in B^A=\mathbf{Set}[A,B]$ is the function which takes $a\in A$ to $f(c,a)\in B$. The proof of (β) and (η) is almost immediate.

As in the case of the product, observe that in general the operation $\Lambda: \mathbf{C}[c\times a,b]\to\mathbf{C}[c,b^a]$ in definition 2.3.1 is a bijection. Indeed, by (β) and (η), Λ^{-1} is the operation which takes every $h\in\mathbf{C}[c,b^a]$ to $eval_{a,b}\circ(h\times id_a)\in\mathbf{C}[c\times a,b]$.

Conversely, if $\Lambda : \mathbf{C}[c\times a,b]\to\mathbf{C}[c,b^a]$ is a bijection and (β) holds, then (η) is necessarily true. Indeed, let $h\in\mathbf{C}[c,b^a]$ and take $f\in\mathbf{C}[c\times a,b]$ such that $h = \Lambda(f)$; then $\Lambda(eval_{a,b}\circ(h\times id_a)) = \Lambda(eval_{a,b}\circ(\Lambda(f)\times id_a)) = \Lambda(f) = h$.

The following is thus an equivalent definition of "exponent":

2.3.2 Definition *Let* ***C*** *be a Cartesian category and* $a,b\in Ob_{\mathbf{C}}$. *The* ***exponent*** *of* a *and* b *is an object* b^a *together with a morphism* $eval_{a,b}: b^a\times a\to b$, *such that for all morphisms* $f: c\times a\to b$, *there exists one and only one* $h: c\to b^a$ *such that the following diagram commutes:*



Exercise By setting $\Lambda(f) = h$, give the details of the equivalence proof between the two definitions.

The previous diagram should suggest in which sense b^a "represents" $\mathbf{C}[a,b]$. The eval morphism generalizes the set-theoretic evaluation function $eval(f,x) = f(x)$. Moreover, take $c = t$, the terminal object. Then $\mathbf{C}[t,b^a] \cong \mathbf{C}[t\times a,b] \cong \mathbf{C}[a,b]$ as sets. This is particularly significant if **C** has enough points (why?).

2.3.3 Definition. ***C*** *is a* ***Cartesian closed category*** *(CCC) iff*

1. ***C*** *is cartesian,*

2. for every pair $a,b\in Ob_{\mathbf{C}}$, *there is an exponent.*

Set is a CCC: the previous definition of exponents in **Set** clearly holds for every pair of sets. Another simple CCC is **CPO**, the category of complete partial orders and continuous maps. As well-known, given c.p.o.'s a and b, $\mathbf{CPO}[a,b]$ is also a c.p.o., with respect to the pointwise ordering. Moreover, both eval and $\Lambda(f)$, defined as for **Set** by using continuous functions, are continuous and satisfy the required conditions. Note that the proof uses the well-known fact that in **CPO** a function is continuous iff it is so in each argument and the map $x \mapsto \lambda y.f(x,y)$ is continuous. Actually, even Λ is continuous.

Among the various examples of categories mentioned in these notes, an important one does not satisfy Cartesian closedness: the category **EN** in section.2.2. Consider, say, $\underline{\omega} = (\omega,id)$. Then $\mathbf{EN}[\underline{\omega},\underline{\omega}]$ (= R, the recursive functions) is surely countable. However, if a numbered set (ω^ω,φ) and

a morphism eval with the above properties existed, then $u(x,y) = eval(\varphi(x),y)$ would be a universal function for R.

Also, the ω-algebraic c.p.o.'s, that is, the c.p.o.'s with a countable collection of compact elements approximating all the others (see Scott domains below) and continuous maps as morphisms do not form a CCC. They contain, though, some fundamental subCCC's for the purposes of denotational semantics of programming languages and higher type Recursion Theory. They will be explored in the examples below.

Given a CCC **D**, it may be interesting to consider specific "structures of types" in it. That is, for a collection A of objects in **D**, let $\mathbf{D}_A$ be the full **sub CCC generated by** A in **D**, i.e., the least full sub category such that $A \subseteq \mathbf{D}_A$ and $a,b \in \mathbf{D}_A \Rightarrow a\times b, a^b \in \mathbf{D}_A$.

Exercise Prove that in any CCC one has $a^{b\times c} \cong (a^b)^c$.

In definition 1.4.2, we introduced the notion of "retract": in a category **C**, $a<b$ via the retraction (i,j) iff $j \circ i = id_a$. In these assumptions, i turns out to be mono and j epic. Thus, a retract a of b is a subobject of b in the sense of section 1.5. In the case of **Set**, nonempty subsets and retracts happen to coincide, as surjections from a set to a subset are always possible. In more structured categories this reinforcement of the idea of subset, given by retractions, turns out to be very informative. In particular, we will discuss categories with nontrivial objects a such that $a^a<a$. This is clearly impossible in **Set** because, by Cantor's theorem, the cardinality of the exponent a^a, when a is not a singleton, is strictly bigger than the cardinality of a. In short, we will put together retracts and exponents, in a nontrivial way, in order to discuss one of the early relevant applications of categorical notions to computer science, namely the invention of mathematical (categorical, to be precise) models of type-free languages. In these languages, programs are viewed as data or, semantically, exponents may be retracted into (source and target) objects. It is convenient to prove, in general, some basic properties of exponents and retractions for their relevance and simplicity as well as for some preliminary training on equational reasononing, which will turn out to be useful to the reader in the sequel.

2.3.4 Proposition *Let **C** be a CCC. If $a<a'$ (via $in_a: a\to a'$, $out_a: a'\to a$), and $b<b'$ (via $in_b: b\to b'$, $out_b: b'\to b$), then $b^a<b'^{a'}$, via $\Lambda(in_b\circ eval\circ(id\times out_a)): b^a\to b'^{a'}$, $\Lambda(out_b\circ eval\circ(id\times in_a)): b'^{a'}\to b^a$.*

Proof. $\Lambda(out_b\circ eval\circ(id\times in_a)) \circ \Lambda(in_b\circ eval\circ(id\times out_a)) =$

$= \Lambda(out_b\circ eval\circ(id\times in_a)\circ\Lambda(in_b\circ eval\circ(id\times out_a))\times id)$

$= \Lambda(out_b\circ eval\circ\Lambda(in_b\circ eval\circ(id\times out_a))\times id\circ(id\times in_a))$

$= \Lambda(out_b\circ(in_b\circ eval\circ id\times out_a)\circ(id\times in_a))$

$= \Lambda(eval \circ id\times(out_a\circ in_a))$

$= \Lambda(\text{eval} \circ \text{id}\times\text{id})$

$= \text{id}$. ♦

2.3.5 Definition *Let* ***C*** *be a CCC. An object* V *of* ***C*** *is* ***reflexive*** *iff* $V^V < V$.

Before we get to see some reflexive objects in relevant CCC's in the following sections, it is worth proving two simple, but general, properties of reflexive objects (see chap.8 for applications).

2.3.6 Proposition *Let* ***C*** *be a CCC, and* V *a reflexive object. Then* $t < V$ *and* $V\times V < V$.

Proof. Let ($\mathbf{in}$: $V^V \to V$, $\mathbf{out}$: $V \to V^V$) the retraction pair between V^V and V. In order to prove that $t<V$ we must only prove the existence of a morphism from t to V (why?). Let then p_1: $t\times V \to V$ be the projection; $\Lambda(p_1)$: $t \to V^V$, and thus $\mathbf{in} \circ \Lambda(p_1)$: $t \to V$.

The proof that $V\times V<V$ is much more complex; we prove that $V\times V<V^V$; then $V\times V<V$ follows by composition.

Let app = eval$\circ$($\mathbf{out}\times\text{id}_V$): $V\times V \to V$, and let $\alpha_{a,b,c}$ be the isomorphism $\alpha_{a,b,c}$: $(b\times c)\times a \to (a\times b)\times c$. Then: app $\circ$ (app$\times$id) $\circ$ $\alpha_{V,V,V}$: $(V\times V)\times V \to V$

and $\mathbf{in_1} = \Lambda(\text{app} \circ (\text{app}\times\text{id}) \circ \alpha) : (V\times V) \to V^V$.

By proposition 2.3.4 one has $(V^V)^V < V^V$ via

$\mathbf{in_2} = \Lambda(\mathbf{in} \circ \text{eval} \circ (\text{id}\times\text{id})) : (V^V)^V \to V^V$

$\mathbf{out_2} = \Lambda(\mathbf{out} \circ \text{eval} \circ (\text{id}\times\text{id})) : V^V \to (V^V)^V$.

Let p_2: $t\times V \to V$ and pr_1:$V\times V \to V$, pr_2: $V\times V \to V$ be the projections respectively associated with the products $t\times V$ and $V\times V$. Then, for i =1,2, $\Lambda(pr_i)$: $V \to V^V$ and, thus, for $\Lambda(\Lambda(pr_i) \circ p_2)$: $t \to (V^V)^V$, $\mathbf{p_i} = \mathbf{in} \circ \mathbf{in_2} \circ \Lambda(\Lambda(pr_i) \circ p_2)$: $t \to V$.

Define, then, $\mathbf{out_1} = \langle \text{eval} \circ \langle \text{id}, \mathbf{p_1} \circ !V^V \rangle , \text{eval} \circ \langle \text{id}, \mathbf{p_2} \circ !V^V \rangle \rangle : V^V \to V\times V$.

We must prove that $\mathbf{out_1} \circ \mathbf{in_1} = \text{id}_{V\times V}$, or equivalently that for i = 1,2, $pr_i \circ \mathbf{out_1} \circ \mathbf{in_1} = pr_i$.

$pr_i \circ \mathbf{out_1} \circ \mathbf{in_1} = \text{eval} \circ \langle \text{id}, \mathbf{p_i} \circ !V^V \rangle \circ \mathbf{in_1}$

$= \text{eval} \circ \langle \Lambda(\text{app} \circ (\text{app}\times\text{id}) \circ \alpha), \mathbf{p_i} \circ !V\times V \rangle$

$= \text{app} \circ (\text{app}\times\text{id}) \circ \alpha \circ \langle \text{id}_{V\times V}, \mathbf{p_i} \circ !V\times V \rangle$

$= \text{app} \circ \langle \text{app} \circ \langle \mathbf{p_i} \circ !V\times V, pr_1 \rangle , pr_2 \rangle$

$= \text{app} \circ \langle \text{eval}\circ(\mathbf{out}\times\text{id}_V) \circ \langle \mathbf{in}\circ\mathbf{in_2}\circ\Lambda(\Lambda(pr_i) \circ p_2) \circ !V\times V, pr_1 \rangle, pr_2 \rangle$

$= \text{app} \circ \langle \text{eval}\circ\langle\mathbf{in_2}\circ\Lambda(\Lambda(pr_i) \circ p_2) \circ !V\times V, pr_1 \rangle, pr_2 \rangle$

$= \text{app} \circ \langle \text{eval}\circ\langle \Lambda(\mathbf{in}\circ\text{eval}\circ(\text{id}\times\text{id}))\circ\Lambda(\Lambda(pr_i) \circ p_2) \circ !V\times V, pr_1 \rangle, pr_2 \rangle$

$= \text{app} \circ \langle \text{eval}\circ\langle \Lambda(\mathbf{in}\circ\text{eval}\circ(\Lambda(\Lambda(pr_i) \circ p_2)\times\text{id}))\circ !V\times V, pr_1 \rangle, pr_2 \rangle$

$= \text{app} \circ \langle \text{eval}\circ\langle \Lambda(\mathbf{in} \circ \Lambda(pr_i) \circ p_2)\circ !V\times V, pr_1 \rangle, pr_2 \rangle$

$= \text{app} \circ \langle \mathbf{in} \circ \Lambda(pr_i) \circ pr_1, pr_2 \rangle$

$= \text{eval}\circ(\mathbf{out}\times\text{id}_V) \circ (\mathbf{in} \circ \Lambda(pr_i))\times\text{id}_V$

$= \mathrm{eval} \circ (\Lambda(pr_i) \times id_V)$

$= pr_i$. ♦

Exercises For the following exercises, assume that **C** is a CCC.

1. Let V be a reflexive object of **C**. Prove that the collection $\mathbf{Ret_V}$ of all retracts of V in **C** is a CCC.

2. (Difficult, see section 8.8) Let b be an object of **C.** A **fixpoint operator** for b is a morphism $Fix_b: b^b \to b$ such that $Fix_b = eval_{b,b} \circ \langle id, Fix_b \rangle$. Let $V^V < V$ via (in,out). Let also

$F = eval \circ \langle id, in \rangle : V^V \to V$;

$H = \Lambda(eval \circ (id \times (F \circ out))) : V^V \to V^V$.

Prove that F∘H is a fixpoint operator for V. Define a fixpoint operator for all objects in $\mathbf{Ret_V}$.

3. Let **C** be a CCC and suppose that for all a, b in $Ob_{\mathbf{C}}$ there exists the coproduct a+b (with embedding $in_a: a \to a+b$, $in_b: b \to a+b$). Prove that, for all c in $Ob_{\mathbf{C}}$, (a×c)+(b×c) is isomorphic to (a+b)×c, and define explicitly the isomorphism.

Result: (a×c)+(b×c) ≅ (a+b)×c via

$i_1 = (in_a \times id_c)+(in_a \times id_c): (a \times c)+(b \times c) \to (a+b) \times c$

$i_2 = \Lambda^{-1}(\Lambda(in_{a \times c})+\Lambda(in_{b \times c})): (a+b) \times c \to (a \times c)+(b \times c)$

Proving $i_2 \circ i_1 = id$ is easy. For $i_1 \circ i_2 = id$, note first that $g \circ \Lambda^{-1}(f) = \Lambda^{-1}(\Lambda(g \circ eval) \circ f)$.

Then, in a few steps, one obtains

$i_1 \circ i_2 = \Lambda^{-1}(\Lambda(in_a \times id_c)+\Lambda(in_b \times id_c))$

$= \Lambda^{-1}(\Lambda(id) \circ in_a + \Lambda(id) \circ in_b)$

$= \Lambda^{-1}(\Lambda(id))$

$= id$

2.4 Examples of CCC's

2.4.1 Scott Domains

In this section and in the following one we introduced two fundamental examples of CCC's, namely, Scott domains and coherent domains. We define only the exponent object and the eval function and check that they are respectively an object and an arrow of the category. We leave the problem of defining the isomorphism Λ and checking (βcat) and (ηcat) as an (easy) exercise for the reader.

2.4.1.1 Definition *Let $(X,\leq)$ be a partiallly ordered set (po-set).*
*$D \subseteq X$ is **directed** iff it is nonempty and, for any $i,j \in D$, there is $k \in D$ such that $i \leq k$, $j \leq k$. A p.o.set $(X,\leq)$ is **complete** (is a **CPO**) iff every directed subset $D \subseteq X$ has a least upper bound $\cup D$ (the least element $\perp$ is the least upper bound of the empty directed set).*

*A point $x \in X$ is **compact** (**finite**) if for every directed D such that $x \leq \cup D$, there is an element $y \in D$ such that $x \leq y$. Let X_0 denote the collection of compact elements of X.*
*The c.p.o. $(X, \leq)$ is **algebraic** if for every $x \in X$ the set $x\downarrow = \{x_0 \in X_0 \mid x_0 \leq x\}$ is directed and $\cup(x\downarrow) = x$.*
*A c.p.o. $(X, \leq)$ is **bounded complete** if every bounded subset of X has a least upper bound. A **Scott Domain** is a bounded complete algebraic c.p.o..*

Exercises

1. Check that $\{\{y \mid x_0 \leq y\} \mid x_0 \in X_0\}$ is a basis for a T_0 topology on a Scott Domain. This topology is usually called **Scott topology**.
2. Prove that the least upper bound of a finite set of finite elements is always finite, if it exists.
3. (Nontrivial) Find counterexamples for the following assertions:
   i. if x_0 is compact then the set $\{y \mid y \leq x_0\}$ is finite;
   ii. if x_0 is compact and $y \leq x_0$ then y is compact.

2.4.1.2 Definition *Let $(X, \leq_X)$, $(Y, \leq_Y)$ be c.p.o.'s. A function $f: X \to Y$ is **monotonic** if it is order preserving, i.e., $i \leq_X j$ implies $f(i) \leq_Y f(j)$. (We will often omit the subscript X in $\leq_X$.)*
*A function $f: X \to Y$ is **continuous** if for every directed $D \subseteq X$, $f(\cup D) = \cup_{d \in D} f(d)$.*

Exercise Let $(X, \leq_X)$, $(Y, \leq_Y)$ be Scott domains. Prove that a function $f: X \to Y$ is continuous according to the previous definition iff it is continuos with respect to the Scott topology.

2.4.1.3 Definition *The category **D** has Scott domains for objects and continuous functions for morphisms. Let X, Y be objects of **D**. Y^X is just the collection of the continuous functions from X to Y ordered pointwise.*

Of course Y^X is a c.p.o.. We have to prove that it is bounded complete and algebraic.
In order to show that Y^X is bounded complete, assume that $\{f_i\}_{i \in I}$ has an upper bound g. Define then h by $h(x) = \cup_{i \in I}\{f_i(x)\}$. The function h is well defined since the set $\{f_i(x)\}_{i \in I}$ is bounded by g(x) and, thus, it has a least upper bound in Y. Moreover, h is continuous because for every directed set D in X one has:

$$\begin{aligned} h(\cup D) &= \cup_{i \in I}\{f_i(\cup D)\} \\ &= \cup_{i \in I} \cup_{x \in D} \{f_i(x)\} && \text{by the continuity of } f_i \\ &= \cup_{x \in D} \cup_{i \in I} \{f_i(x)\} \\ &= \cup_{x \in D} \{h(x)\} \end{aligned}$$

It easy to check that h is a least upper bound for $\{f_i\}_{i \in I}$.
To show that Y^X is algebraic, we explicitly define the set $(Y^X)_0$ of its compact elements.

2.4.1.4 Definition *A **step funcion** from X to Y is a function step-a,b where $a \in X_0$, $b \in Y_0$, defined by: step-a,b (x) = if $a \leq x$ then b else $\perp$.*

We claim that the compact element of Y^X are exactly the least upper bound of finite bounded sets of step functions. In other words, for every $f_0 \in (Y^X)_0$, (*) $f_0 = \cup_{i \in I}\{\text{step-}a_i,b_i\}$ for some finite I.

Let us prove first that every function $f = \cup_{i \in I}\{\text{step-}a_i,b_i\}$ is compact, when I is finite and $\cup_{i \in I}\{\text{step-}a_i,b_i\}$ exists, i.e., when for all subset J of I, $a_J = \cup_{i \in J}\{a_i\}$ exists $\Rightarrow$ $b_J = \cup_{i \in J}\{b_i\}$ exists (We then say that I is a **compatible** set of indices.)

Then, let $\{g_h\}_{h \in D}$ be a directed family in Y^X such that $f \leq \cup_{h \in D}\{g_h\}$. In particular, for every J in I as above, $f(a_J) = b_J \leq (\cup_{h \in D}\{g_h\})(a_J) = \cup_{h \in D}\{g_h(a_J)\}$. Clearly, for each J, b_J is compact and $\{g_h(a_J)\}_{h \in D}$ is directed. Let then $b_J \leq g_{h(J)}(a_J)$ for some $h(J) \in D$. Since I is finite and $\{g_h\}_{h \in D}$ is directed, let g_k, for $k \in D$, be such that $g_k \geq g_{h(J)}$ for all J in I. Clearly $f \leq g_k$ and we are done.

Prove now for exercise that for every continuous function $f: X \to Y$, one has

i. the set $F = \{ \cup_{i \in I}\{\text{step-}a_i,b_i\} \mid$ I finite, and $b_i \leq f(a_i) \}$ is directed

ii. $f = \cup F$

Suppose then that f is compact. We need to prove that $f = \cup_{i \in I}\{\text{step-}a_i,b_i\}$ for some finite compatible I. By the exercise, $f = \cup F$, for F directed; thus, there exists I such that $\cup_{i \in I}\{\text{step-}a_i,b_i\} \in F$ and $\cup_{i \in I}\{\text{step-}a_i,b_i\} \geq f = \cup F \geq \cup_{i \in I}\{\text{step-}a_i,b_i\}$. In conclusion $f = \cup_{i \in I}\{\text{step-}a_i,b_i\}$; that is, every finite element in X^Y has the form (*) and, in particular, X^Y is a Scott domain.

The function $\text{eval}_{X,Y}: Y^X \times X \to Y$ is defined by $\text{eval}_{X,Y}(f,x) = f(x)$. The proof that eval is continuous is straightforward.

Interesting examples of Scott Domains may be found everywhere in the literature of denotational semantics. Indeed, the Cartesian closedness of the category allows you to construct plenty of them as products and exponents over commonly used ground types. That is, consider your preferred types of data (integers, booleans, strings,etc.). Organize them as flat p.o.sets, i.e., add a least element $\perp$ and set $x \leq x'$ iff $x = \perp$ or $x = x'$. These are clearly objects of **D** as well as their products and exponents.

Other relevant examples are given, for example, by the p.o.sets P of the partial maps from ω to ω, the natural numbers, and Pω, the powerset of ω. The partial order, in these cases, is given by set inclusion, which on P means graph inclusion of functions, i.e., $f \leq g$ iff $\forall n$ $(f(n)\downarrow \Rightarrow g(n) = f(n))$. As an exercise, the reader may check that both P and Pω live in **D**.

Interestingly enough, these two familar structures are also reflexive objects in **D** . We sketch the proof of this for Pω , see section 9.6-2 for more on P.

Let $\{e_n\}_{n\in\omega}$ be a canonical (bijective and effective) enumeration of the finite subsets of ω and let $<,>: \omega\times\omega\to\omega$ be a canonical coding of pairs. Define then graph: $\mathbf{D}[P\omega,P\omega]\to P\omega$ by graph(f) = $\{<n,m> \mid m\in f(e_n)\}$ and fun: $P\omega\to\mathbf{D}[P\omega,P\omega]$ by fun(a)(b) = $\{m \mid \exists e_n \subseteq b\ <n,m>\in a\}$. It is a simple exercise to check that graph and fun are morphisms in **D**. Moreover, fun $\circ$ graph = id and, thus, $P\omega^{P\omega}<P\omega$.

This example, which played a relevant role in denotational semantics, has been directly inspired by Recursion Theory (see the references). Indeed, the work carried on so far can be naturally "effectivized."

2.4.1.5 Definition *A Scott domain* $\underline{X} = (X,\leq)$ *is* ***effectively given*** *if* $\exists e_0: \omega\to X_0$ *bijective and*

1. $\exists z\in X\ e_0(n), e_0(m) \leq z$ *is decidable in* n, m
2. $\exists g\in R\ (\exists z\in X\ e_0(n), e_0(m) \leq z \Rightarrow e_0(g(n,m)) = sup\{e_0(n),e_0(m)\})$.

Call **ED** the category of effectively given Scott domains and continuous functions. **ED** is a CCC. As a matter of fact, the effectiveness properties are easily inherited at higher types.

Observe that, instead of taking the least upper bounds (l.u.b.'s) of all directed sets, as required in the definition of **ED**, one may take only the **computable** l.u.b.'s, i.e. the l.u.b.'s of directed sets or ideals in (X_0,e_0) that are indexed over recursively enumerable (r.e.) sets. (One may independently choose directed sets or ideals and obtain the same collection of computable elements.)

These limits are computable in a very sound sense. For example, $(P\omega, \{e_n\}_{n\in\omega}, \subseteq)$ is in **ED** and its computable elements are exactly the r.e. sets.

Exercise Prove a similar fact for the set P of partial maps from ω to ω.

Call **constructive domain** a domain whose elements are the computable elements in an effectively given domain. Since **ED** is Cartesian closed, this may be done in any (higher) type. In particular, given the constructive domains $\underline{X}_c, \underline{Y}_c$ obtained from $\underline{X}$ and $\underline{Y}$, one may consider the constructive domain $\underline{Y}^{\underline{X}}_c$ of the computable elements of $\underline{Y}^{\underline{X}}$. Define then the following

2.4.1.6 ***CD*** *is the category of constructive domains and continuous and computable morphisms.*

Exercise One clearly has to check that, for $f\in\underline{Y}^{\underline{X}}_c$, $\forall x\in X_c$ $f(x)\in Y_c$.

By the Cartesian closedness of **ED**, **CD** also is a CCC. Observe that each $\underline{X}_c$ is countable and that it can be effectively enumerated by using an acceptable enumeration of the r.e. sets. Typical objects in **CD** are RE, the recursively enumerable sets, and PR (= P_c), the partial recursive functions (see the exercise above).

Thus, in a rather indirect way, that is by topological and order properties, we obtained a CCC of countable (and numbered) sets. The proof that **CD** is a full sub-CCC of **EN** requires an important generalization, in higher types, of the classical Myhill-Shepherdson theorem for enumeration operators. The main application of **CD** is the characterization of the partial (continuous) and computable functionals as the sub-CCC of **CD** generated by PR, i.e., taking PR and constructing all higher types within **CD** (see also section 8.4-I). Moreover, one can give a countable and effective interpretation to the recursive definitions of programs and data types within **CD** (by a constructive version of the "limit constructions" in chapter 10).

Exercise Prove that RE is reflexive in **CD** (use the full and faithful embedding of **CD** in **EN**).

2.4.2 Coherent Domains

2.4.2.1 Definition *A **coherent structure** is a pair $(|X|, \uparrow)$, where $|X|$ is a set and $\uparrow$ is a binary, reflexive, symmetric relation on $|X|$. The elements of $|X|$ are called **points**, and the relation $\uparrow$ is called **coherence**.*
*The **coherent domain** associated with $(|X|, \uparrow)$ is the collection X of subsets of $P(|X|)$ whose points are pairwise coherent. The elements of X are ordered by set-inclusion.*
Coherence is extended to X in the obvious way, that is: $A \uparrow B$ iff $A \cup B \in X$.

Exercise Prove, when X is a coherent domain, that
1. $\emptyset \in X$
2. X is closed under directed union
3. ($A \in X$ and $B \subseteq A$) $\Rightarrow$ $B \in X$

2.4.2.2 Definition *Let X, Y be two coherent domains. A function $F: X \to Y$ is **stable** iff*
i). F is continuous
ii. $\forall A, B \in X \quad A \uparrow B \Rightarrow F(A \cap B) = F(A) \cap F(B)$

2.4.2.3 Definition *The category **Stab** has coherent domains as objects and stable functions as morphisms.*

Given two coherent domains X and Y, their **product** $X \times Y$ is defined by:
i $|X \times Y| = \{(0,z) / z \in |X|\} \cup \{(1,z) / z \in |Y|\}$
ii. $(a,z) \uparrow (a',z')$ [mod $X \times Y$] iff $a = a' \Rightarrow z \uparrow z'$ [mod $D(a)$], where $D(0) = X$ and $D(1) = Y$.

Exercise Define the projections and check that they are stable, i.e., prove that **Stab** is Cartesian.

There is simple way to obtain stable functions over coherent domains.

2.4.2.4 Definition *Let X,Y be coherent domains. Let also f be an injective function from $|X|$ to $|Y|$ such that, for all $x,x' \in |X|$, one has $\{x, x'\} \in X \Leftrightarrow \{f(x), f(x')\} \in Y$. Define then $f^+ : X \to Y$ and $f^- : Y \to X$ by*

i. $f^+(a) = \{f(z) / z \in a\}$

ii. $f^-(b) = \{z / f(z) \in b\}$

It is a matter of a simple exercise to prove that both f^+ and f^- are stable functions.

We need to construct next an exponent object out of the set of stable maps over coherent domains.

2.4.2.5 Definition *Let $F: X \to Y$ be a stable function. The **Trace** of F is $Tr(F) = \{(a,z) / a \in X,$ a is finite, $z \in |Y|$, $z \in F(a)$, $(\forall a' \subseteq a , z \in F(a') \Rightarrow a = a')\}$.*

F is completely determined by its trace by means of the following equation: $F(A) = \{z \in |Y| / \exists a \subseteq A$ $(a,z) \in Tr(F)\}$.

Exercise Prove that the correspondence between stable functions and their traces is bijective.

Notation The symbol $\uparrow\uparrow$ is used to represent **strict coherence**, i.e., $A \uparrow\uparrow B$ iff $A \uparrow B$ and $A \neq B$.

2.4.2.6 Definition *Let $|Y^X| = \{(a,z) / a \in X, a$ is finite, $z \in |Y| \}$. Moreover, let $(a,z) \uparrow (a',z')$ iff*

i. $a \uparrow\uparrow a' \ [mod\ X] \Rightarrow z \uparrow\uparrow z' \ [mod\ Y]$ *and*

ii. $a \uparrow a' \ [mod\ X] \Rightarrow z \uparrow z' \ [mod\ Y]$.

*Then Y^X is the **arrow domain** (exponent object).*

Exercises

1. Prove that conditions (i) and (ii) may be stated equivalently as

   $(a,z) = (a',z')$ or $z \uparrow\uparrow z'$ or **not** $a \uparrow a'$.

2. Prove that every element of Y^X is a trace of some stable function from X to Y, and conversely that if $F: X \to Y$ is stable then $tr(F) \in Y^X$.

3. Let $f,g : X \to Y$ be two stable functions.

Define $f \leq_B g$ (**Berry's order**) iff $\forall x,y \in X$ $x \subseteq y \Rightarrow f(x) = f(y) \cap g(x)$

Prove that $f \leq_B g$ if and only if $Tr(f) \subseteq Tr(g)$. Let moreover $\leq_p$ be the pointwise order. Prove that:

i. $f \leq_B g \Rightarrow f \leq_p g$

ii. $f \uparrow g \Rightarrow (f \leq_B g \Leftrightarrow f \leq_p g)$

4. Let X,Y be coherent domains. A stable function f: X→Y is **linear** iff :

i. $a \cup b \in X \Rightarrow f(a \cup b) = f(a) \cup f(b)$

ii. $f(\emptyset) = \emptyset$

Prove that f: X→Y is linear iff its trace is formed of pairs (a,z), where the component a is a singleton. Observe that the maps f^+ and f^- in 2.4.2.4 are actually linear. Call **Lin** the category of coherent domains and linear maps.

5. Let f: X→Y, g: X→Y be two linear functions. Prove that Tr(f) ⊆ Tr(g) if and only if for all x in X f(x) ≤ g(x). Deduce as a corollary that on linear functions between coherent domains the order of Berry coincides with the pointwise order.

2.4.2.7 Definition *The function* $\boldsymbol{eval_{X,Y}}$*:* $Y^X \times X \to Y$ *is defined by the following equation:* $\forall A \in Y^X, \forall B \in X \ eval_{X,Y}(A,B) = \{y \mid \exists (b,y) \in A, b \subseteq B\}$.

We prove that $eval_{X,Y}$ is stable. Continuity is trivial. We must only check that if $(A,B) \uparrow (A',B')$ [mod Y^X] then $eval_{X,Y}((A,B) \cap (A',B')) = eval_{X,Y}((A,B)) \cap eval_{X,Y}((A',B'))$. The inclusion ⊆ is immediate by continuity.

Take then z in $eval_{X,Y}((A,B)) \cap eval_{X,Y}((A',B'))$. This implies $\exists (b,z) \in A$, $b \subseteq B$ and $\exists (b',z) \in A'$, $b' \subseteq B'$. Note that $B \uparrow B'$ by hypothesis and, thus, $b \uparrow b'$. Moreover, also by hypothesis, $A \uparrow A'$ and then, by definition of consistency mod Y^X, one has $b = b'$. This implies $(b,z) \in A \cap A', b \subseteq B \cap B' \Leftrightarrow z \in eval_{X,Y}((A \cap A'),(B \cap B')) \Leftrightarrow z \in eval_{X,Y}((A,B) \cap (A',B'))$.

In conclusion, the category **Stab** of coherent domains and stable functions is a CCC.

Exercise Let f^+ and f^- be defined as in 2.4.2.4, over coherent domains X and Y. Prove that $f^- \circ f^+ = id_X$, i.e., that X<Y via (f^+,f^-) in **Stab**.

By this technique and the following construction, one can easily construct, in each cardinal, a coherent domain of which all other coherent domains of the same cardinality are retracts. In particular, it will be so also its own function space.

2.4.2.8 Definition *If X is a coherent domain, then !X (read* ***of course*** *X) is the coherent domain defined by*

i. $|!X| = \{a \mid a \in X, a \text{ finite}\}$

ii. $a \uparrow b \ [mod\ !X]$ *iff* $a \cup b \in X$.

Let T be the three-element truth value poset {⊥,true, false}, i.e., the "lifting" of {true, false}. For simplicity, we look at the cardinal ω. Consider then the ω-power T^ω of T. That is, take all the functions from ω to T. When T is partially ordered in the usual way, with ⊥ least and true and false incomparable, then T^ω is clearly a coherent domain. One may also understand T^ω as the set of disjoint subsets of ω. Of course, $!T^\omega$ is aslo in **Stab** by the definition just given.

2.4.2.9 Theorem *Let D be a coherent domain with a countable |D|, and let e: ω→|D| be a bijective map. Then there exist an injective function f from |X| to $|!T^\omega|$ such that, for all x,x'∈|X|, one has {x, x'}∈X ⇔ {f(x), f(x')}∈$!T^\omega$.*

Proof Let f: $|D| \to |!T^\omega|$ be defined in the following way:

$$f(e(i)) = < \{i\}, \{j \,/\, j \leq i \text{ and } \textbf{not } e(i)\uparrow e(j)\} >$$

Obviously f is injective. It is also trivial that {e(i),e(j)}∈D implies {f(e(i)),f(e(j))}∈$!T^\omega$. Conversely suppose **not** e(i)↑e(j) [mod D] , and let i ≤ j (the other case is analogous). This implies that i∈$f(e(j))_1$ and then **not** f(e(i))↑f(e(j)) [mod $!T^\omega$].

2.4.2.10 Corollary *Let f^+ and f^- be as in 2.4.2.4, for f: |D|→$|!T^\omega|$. Then, for any coherent domain D with a countable |D|, one has D<$!T^\omega$ via (f^+,f^-).*

The proof easily follows from the exercise above. Note now that, if D is countably based, so is D^D, in **Stab** (check this for exercise). Since, in particular, $|!T^\omega|$ is countable, then $!T^\omega$ turns ou to be a reflexive object, as $!T^\omega !T^\omega <: !T^\omega$.

2.5 Equalizers and Pullbacks

Let f,g: A→B a pair of "parallel" functions in **Set**, i.e. f, g have the same source and target. The subset E of A on which f and g agree, i.e., E = {x / x∈A, and f(x) = g(x)} is called an **equalizer** of f and g. We try now to give a categorical characterization of previous set-theoretic notion. The starting point is that E being a subset of A it must be represented as a subobject, that is, as a mono i: E→A; moreover, i must enjoy the property f ∘ i = g ∘ i. But E is the maximal subset of A on which f and g agree, and in order to guarantee this condition we require that if h: C→A is any other function such that f ∘ h = g ∘ h, then h "factors" uniquely through i , that is, there exist a unique k: C→E such that h = i ∘ k. We now prove that the previous condition is enough to ensure, in **Set**, that i(E) contains *all* the x∈A, such that f(x) = g(x).

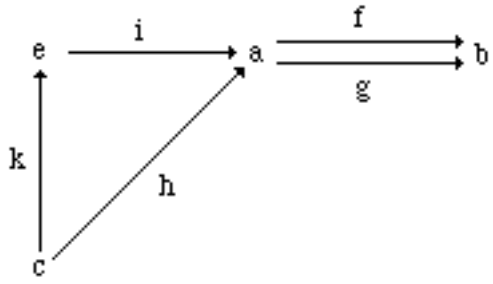
Suppose not, then there exist a∈A, a∉i(E) such that f(a) = g(a). Consider the function l: E∪{a}→A defined by l(e) = i(e) if e∈E, l(a) = a. Of course f ∘ l = g ∘ l, and therefore a morphism k: E∪{a}→E must exists such that l = i ∘ k. But then a = l(a) = i(k(a))∈i(E); this is a contradiction.

Since this condition of unique factorization also implies that i is mono (see proposition 2.5.2 below) we are led to the following definition:

2.5.1 Definition *Given a pair of morphisms* $f,g \in \mathbf{C}[a,b]$, *an* ***equalizer*** *of* f *and* g *is a pair* $(e, i \in \mathbf{C}[e,a])$ *such that :*

i. $f \circ i = g \circ i$

ii. for all $h \in \mathbf{C}[c,a]$, $f \circ h' = g \circ h'$ *implies* $\exists!\, k \in \mathbf{C}[c,e]$ $i \circ k = h$.



Coequalizers *are defined dually, that is a coequalizer of* f,g *is a pair* $(e, i \in \mathbf{C}[b,e])$ *such that:*

i. $i \circ f = i \circ g$

ii. for all $h \in \mathbf{C}[b,c]$, $h \circ f = h \circ g$ *implies* $\exists!\, k \in \mathbf{C}[e,c]$ $k \circ i = h$.

2.5.2 Proposition *Every equalizer is monic.*

Proof Let i: e→a be the equalizer of f, g: a→b. Let j, l : c→e, such that $i \circ j = i \circ l$.
Since $f \circ (i \circ j) = (f \circ i) \circ j = (g \circ i) \circ j = g \circ (i \circ j)$ there exists a unique h: c→e such that $(i \circ j) = i \circ h$, hence $j = h = l$. ♦

2.5.3 Proposition *Every epic equalizer is iso.*

Proof Let i: e→a be the equalizer of f, g: a→b. Since i is epic, and $f \circ i = g \circ i$, it follows that f = g. The identity id_a equalizes f and g, and there is a unique morphism h: a→e such that $id_a = i \circ h$. Moreover $i \circ h \circ i = id_a \circ i = i \circ id_a$, and since i is monic (by proposition 2.5.2), then $h \circ i = id_a$. ♦

We now introduce one of the most powerful notions of Category Theory: the pullback. In a sense, pullbacks generalize equalizers to pairs of morphisms with different sources.

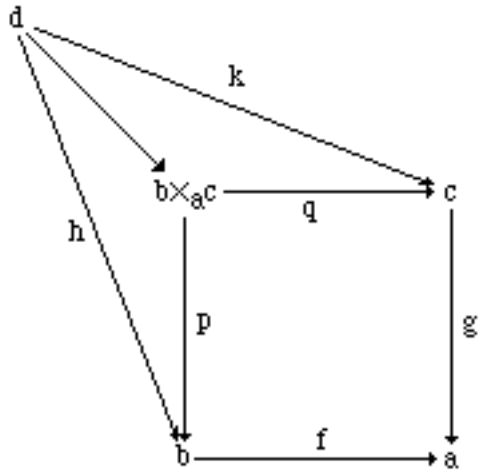
2.5.4 Definition *Given two arrows* $f: b \to a$ *and* $g: c \to a$ *with common target* a, *the* ***pullback*** *of* (f,g) *is an object* $b \times_a c$ *and two arrows* $p: b \times_a c \to b$, $q: b \times_a c \to c$, *such that*

1. $f \circ p = g \circ q: b \times_a c \to a$

2. for every other triple $(d, h: d \to b, k: d \to c)$ *such that* $g \circ k = f \circ h$, *there exists a unique arrow* $\langle h,k \rangle_a: d \to b \times_a c$ *such that* $p \circ \langle h,k \rangle_a = h$, *and* $q \circ \langle h,k \rangle_a = k$.

The dual notion is called ***pushout.***

A typical pullback diagram is as follows:



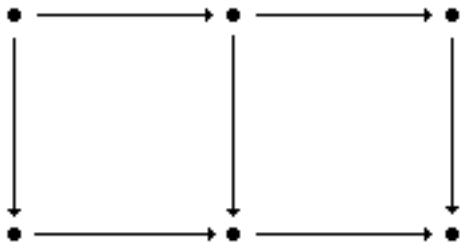
The lower "square" is also called "pullback square." Note that the notation used for pullbacks is quite similar to the one used for products; indeed, they behave similarly (products are just a particular case of pullbacks, see proposition. 2.5.5 below). Note also that the subscript a is meant to express the dependency of $b\times_a c$ and $\langle h,k\rangle_a$ on f and g, but $b\times_{f,g} c$ and $\langle h,k\rangle_{f,g}$, is too heavy a notation: the subscript must be considered essentially as a warning that we are dealing with a pullback, not just a product. Usually this omission of information is harmless, because the particular pullback we are considering is clear from the context.

Example In **Set** the pullback of (f: B→A, g: C→A) is as follows:
$(\{\langle x,y\rangle / x\in B, y\in C, f(x) = g(y)\}, p_1, p_2)$ where $p_1(\langle x,y\rangle) = x$ and $p_2(\langle x,y\rangle) = y$.

2.6.5 Proposition *Let **C** be a category with a terminal object t. For any object a of **C**, let !a be the unique morphism in **C**[a,t]. If **C** has pullbacks for every pair of arrows, then it also has products for every pair of objects.*
Proof *Hint*. Given a,b in C, let $(a\times b, p_1: a\times b\to a, p_2: a\times b\to b)$ be the pullback of (!a: a→t, !b: b→t). It is easy to verify that this is a product. ♦

2.6.6 Proposition *If a category **C** has pullbacks for every pair of arrows and it has terminal object, then it has an equalizer for every pair of arrows.*
Proof Let f,g: a→b. Let (c, fst:c→a, snd:c→a) be the pullback of $(\langle f,id_a\rangle: a\to b\times a, \langle g,id_a\rangle: a\to b\times a)$. Then the equalizer of f,g is (c, fst = snd). Indeed, $f\circ fst = p_1\circ\langle f\circ fst,fst\rangle = p_1\circ\langle f, id_a\rangle\circ fst = p_2\circ\langle g,id_a\rangle\circ snd = p_2\circ\langle g\circ snd,snd\rangle = g\circ snd$. Moreover, for any (c', h:c'→a) such that $f\circ h = g\circ h$, also $\langle f,id_a\rangle\circ h = \langle g,id_a\rangle\circ h$; by definition of pullback, there exists a unque k: c'→c such that $fst\circ k = h$. ♦

2.6.7 Pullback Lemma (PBL) *If a diagram of the form*



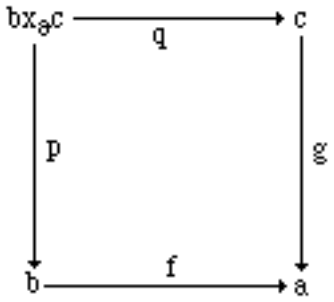
commutes, then

i. if the two small squares are pullbacks, then the outer rectangle is a pullback;

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

Proof Exercise. ♦

2.6.8 Proposition *If the square*

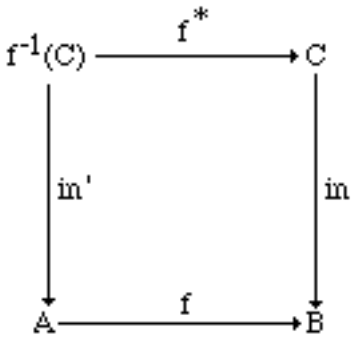


is a pullback and g is monic, then p is monic as well.

Proof: Exercise. ♦

The previous property suggests an interesting generalization of a common set-theoretic construction. If f is a function from a set A to a set B, and C is a subset of B, then the inverse image of C under f, denoted $f^{-1}(C)$ is that subset of A defined by $f^{-1}(C) = \{x/\ x \in A, f(x) \in C\}$.

It is easy to show that the diagram



is a pullback square in **Set**.

In general, given a monic g: c↣b and a morphism f: a→b, the **inverse image** of g under f is the subobject of a (if it exists) obtained by pulling back g along f.

2.6 Partial Morphisms and Complete Objects

As mentioned in the introduction, the common perspective of Programming Language Theory and Category Theory is due to the priority given to "functions" with respect to "sets." The latter notion, when required, is a derived notion of the former, in a sense.

There is an other aspect, though, that should be considered. In Computation Theory, as well as in actual programming, diverging computations cannot be avoided unless a restriction is made to a subclass of the computable functions.

The notion of partiality has a natural interpretation over sets. Let f: A→B be a partial function; the domain of convergence of f, call it $f{\downarrow}$, is just a subset of its "domain" A in the broader sense; moreover the restriction of f to $f{\downarrow}$ is a total map $f_{|(f{\downarrow})}: f{\downarrow} \to B$. Thus a partial map f may be represented by a pair of total functions (i: D→A, h: D→B), where D⊆A is the domain of convergence of f, h is the restriction of f to D, and i: D→A is the canonical injection .

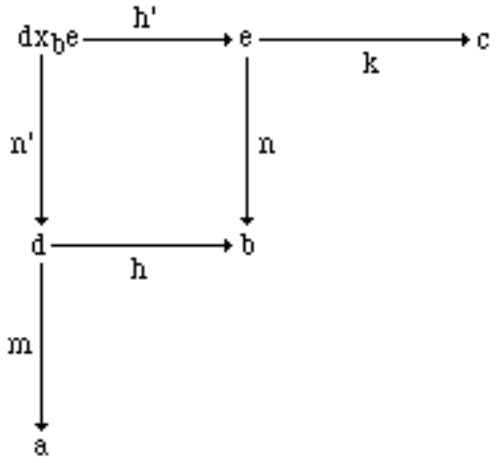
If we try to simulate the previous definition in categorical language, it is natural to define a partial map f between two objects a and b in a category **C** as a pair (m: d→a , h: d→b), where m is monic. However, as in the case of subobjects, we have no way to choose d in a canonical way, so we are forced to identify partial morphisms up to isomorphic variations of d (see section 1.5.)

2.6.1 Definition *Given a category **C** and two objects a and b, a partial map [m, h]: a→b is an equivalence class of pairs (m: d→a, h: d→b), where m is monic, with respect to the following relation R: (m: d→a, h: d→b) R (m': d'→a, h': d'→b) iff* $\exists k: d \to d'$*, k iso,* $m' = m \circ k$ *,* $h' = h \circ k$*.*

With a little abuse of language, we will often speak about a particular pair (m: d→a, h: d→b) when we actually mean $[(m: d \to a, h: d \to b)]_R$.

Our next aim is to define a category **pC** of partial maps on **C**. The main problem is in defining composition. If **C** has pullbacks for every pair of arrows, then the problem is resolved in the following way.

Given (n: e→b, k: e→c) and (m: d→a, h: d→b), define [n, k] ∘ [m, h]: a→b as the equivalence class determined by the outermost sides in the following diagram, i.e., ($m \circ n'$: $d \times_b e \to a$, $k \circ h'$: $d \times_b e \to c$) :



where the square is a pullback. Note that by proposition 2.5.8, since n is monic, n' is monic, too.

Let us see how the previous definition works in **Set**. For the equivalence relation defined on partial morphisms, we suppose $d \subseteq a$, $e \subseteq b$ and take $m: d \to a$, $n: e \to b$ as canonical injections. Then $d \times_b e = \{(x,y) / x \in d, y \in e, h(x) = n(y)\} = \{(x,y) / x \in d, y \in e, h(x) = y\} \cong \{x / x \in d, h(x) \in e\}$, that is, the expected domain of convergence of the composed function. The projections h' and n' associated with the pullback $\{x / x \in d, h(x) \in e\}$ are respectively $h_{|\{x / x \in d, h(x) \in e\}}$ and the canonical injection $i: \{x / x \in d, h(x) \in e\} \to d$. In conclusion, for all $x \in \{x / x \in d, h(x) \in e\}$, one has $k(h'(x)) = k(h(x))$, as we wanted.

Every arrow $f \in \mathbf{C}[a,b]$ has a natural associated arrow in **pC**, that is $(id: a \to a, f: a \to b)$. We say that a map in **pC** is **total** iff it has the above form (up to equivalences). $(\mathbf{pC})_\mathbf{t}$ is the subcategory of **pC** of total maps. As will become clearer in the next chapter, **C** is "isomorphic" to $(\mathbf{pC})_\mathbf{t}$.

Exercise Prove the following assertions:

1. id is total;
2. f,g total $\Rightarrow$ f∘g total;
3. f∘g total $\Rightarrow$ g total;

Example **pSet** is the category of sets with partial maps as morphisms. **pR** = **PR** is the monoid of the partial recursive functions. **pEN** is defined by using partial recursive functions in the diagram that defines the morphisms, instead of the total ones (see section 2.2). Observe that the diagrams commutes iff $e_b \circ f' = f \circ e_a$ for $f' \in \mathbf{PR}$ and, hence, $e_a(n) = e_a(m)$ and $f'(n)\downarrow \Rightarrow f'(m)\downarrow$. Clearly, $(\mathbf{pSet})_\mathbf{t} = \mathbf{Set}$, $\mathbf{PR}_\mathbf{t} = \mathbf{R}$ and $(\mathbf{pEN})_\mathbf{t} = \mathbf{EN}$.

Remark Sometimes, in the construction of the category **pC** from **C**, it may be intersting to restrict our attention to only a subset of the class of monics of **C**. Consider, for example, the category **PO** of p.o.sets: an interesting definition of a partial map f between two p.o.sets would also require f to be defined in an upward closed subset. Let **C** be a category with pullbacks for every pair of arrows. An

admissible family M of monics of **C**, is a family of monics closed under identities, compositions, and pullbacks. Every admissible M on a category **C** gives rise to a different category (M)(**pC**) of partial maps.

As already recalled, diverging computations play an essential role in the theory of computation. They also have an obvious interpretation in Set Theory. However, the set-theoretic understanding of computations may not suffice. Type-free languages and many typed ones, as we shall see later, escape a "naive" interpretation as sets and functions.

As a matter of fact, since the early days of denotational semantics of programming languages, the need to give meaning to (possibly) diverging computations over nontrivial mathematical structures suggested the introduction of various categories of p.o.sets with least elements. The naive understanding of this is immediate: add to all required data types, as sets, an extra element and give them a p.o.set structure as flat p.o.sets (i.e., $\perp$ is the least element and all others are incomparable). We already dealt with this in practice, when presenting various categories of p.o.sets as examples of CCC's. In a sense, the bottom element $\perp$ provides the first hint of the introduction of approximation and continuity notions to the mathematical semantics of programs.

As we have demonstrated, this concept is very clear matematically, at least in several specific categories, such as continuous or algebraic lattices, c.p.o's, and Scott domains. It is not so simple in interesting categories for computations such as **EN**, though. In order to understand it in a general setting and avoid any abuse of this simple concept, a category-theoretic perspective may provide a sound mathematical frame.

Notation In the rest of this section, **C** is a category of partial maps, i.e., **C = pD** for some category **D** with pullbacks for every pair of arrows. $\mathbf{C_t}$ is the associated category of total maps. Since $\mathbf{C_t}$ is a subcategory of **C**, we compose total and partial morphisms by using the same operation of composition. For typographical reasons, we write $a°$ instead of $a^{\perp}$.

The set-theoretic idea we try to formalize categorically is that, when an object a is "lifted" to $a°$ by adding an extra least element, then any hom-set of total maps with target $a°$ is isomorphic to the corresponding hom-set with target a.

2.6.2 Definition. *The **lifting** of $a \in Ob_{\mathbf{C}}$ is an object $a° \in Ob_{\mathbf{C_t}}$ together with a morphism $ex_a \in \mathbf{C}[a°,a]$ and for every $c \in Ob_{\mathbf{C}}$ an operation $\tau_c : \mathbf{C}[c,a] \to \mathbf{C_t}[c,a°]$ such that, for every $f \in \mathbf{C}[c,a]$ and for every $g \in \mathbf{C_t}[c,a°]$ one has*

1. $ex_a \circ \tau_c(f) = f$
2. $\tau_c(ex_a \circ g) = g$.

The operation τ_C: $\mathbf{C}[c,a] \to \mathbf{C_t}[c,a^\circ]$ is bijective, as τ_C^{-1} is defined by $\tau_C^{-1}(g) = ex_a \circ (g)$. Conversely, if τ_C is bijective and (1) holds, then also (2) does. Indeed, let $g \in \mathbf{C_t}[c,a^\circ]$; since τ_C is surjective, then there is $f \in \mathbf{C}[c,a]$ such that $\tau_C(f) = g$. That is, $\tau_C(ex_a \circ g) = \tau_C(ex_a \circ \tau_C(f)) = \tau_C(f) = h$ Thus, the following is an equivalent definition of the lifting object:

2.6.3 Definition. *The **lifting** of $a \in Ob_{\mathbf{C}}$ is an object $a^\circ \in Ob_{\mathbf{C}}$ together with a morphism $ex_a \in \mathbf{C}[a^\circ,a]$ such that, for every $f \in \mathbf{C}[c,a]$, there exists one and only one $g \in \mathbf{C_t}[c,a^\circ]$ satisfying the equation: $ex_a \circ g = f$.*

Thus, as we wanted, for all b, $\mathbf{C}[b,a]$ and $\mathbf{C_t}[b,a^\circ]$ are isomorphic, since any partial morphism $f \in \mathbf{C}[b,a]$ may be uniquely extended to a total one when the target object is lifted (and the lifting exists).

Exercise Prove that the lifting a° of an object a is unique, if it exists.

2.6.4 Proposition. *Let a° be the lifting of a and set $in_a = \tau_a(id_a)$. Then a is a retract of a° in $\mathbf{C}$ (notation: $a <_p a^\circ$) via (in_a, ex_a).*

Proof. $ex_a \circ in_a = (\tau_{a^\circ})^{-1}(id) \circ in_a$ by definition of $(\tau_{a^\circ})^{-1}$
$= (\tau_a)^{-1}(id \circ in_a)$
$= (\tau_a)^{-1}(\tau_a(id))$
$= id.$ ♦

2.6.5 Definition. *An object $a \in Ob_{\mathbf{C}}$ is a **complete** object iff $a < a^\circ$ in $\mathbf{C_t}$.*

The intuition should be clear. An object is complete when it "already contains," in a sense, the extra $\perp$. Think of an object d of **pCPO** and take its lifting d°, i.e., add a least element $\perp$ to d. Then d is complete, that is, $d < d^\circ$ via (i,j), iff d already contained a least element, $j(\perp)$ to be precise. It is actually easy to show that the complete objects in **pPO**, and likewise in **pCPO**, are exactly the partial ordered sets with a least element.

Note that in contrast to the partial retraction $a <_p a^\circ$ via (in,ex) in definition 2.6.4, which may be always given, only complete objects yield total retractions $a < a^\circ$.

2.6.6 Lemma. *If $a < a^\circ$ via (i,j) (in $\mathbf{C_t}$), then $\exists out \in \mathbf{C_t}[a^\circ,a]$ $a < a^\circ$ via $(\tau_a(id_a), out)$.*

Proof Set $out = j \circ \tau(ex_a \circ i \circ ex_a)$. Then

$out \circ \tau_a(id_a) = j \circ \tau(ex_a \circ i \circ ex_a) \circ \tau_a(id_a)$
$= j \circ \tau(ex_a \circ \tau(ex_a \circ i \circ ex_a) \circ \tau_a(id_a))$ by (2) in def. 2.6.2
$= j \circ \tau(ex_a \circ i \circ ex_a \circ \tau_a(id_a))$ by (1)

$= j \circ \tau(ex_a \circ i)$ by (1)
$= j \circ i$ by (2)
$= id_a$. ♦

Thus we may then assume that, if a is complete, $a < a^\circ$ via $(in=\tau_a(id_a),out)$.

The following fact gives the main motivation for the invention of complete objects: exactly on complete objects as targets all partial morphisms may be extended to total ones, *with the same target.*

2.6.7 Definition *$\underline{f} \in \mathbf{C}_t[b,a]$ **extends** $f \in \mathbf{C}[b,a]$ iff $\forall c \in Ob_{\mathbf{C}}$, $\forall h \in \mathbf{C}_t[c,b]$, $(f \circ h) \in \mathbf{C}_t[c,a] \Rightarrow \underline{f} \circ h = f \circ h$.*

2.6.8 Theorem. *Let a° be the lifting of $a \in Ob_{\mathbf{C}}$. Then $a < a^\circ \Leftrightarrow \forall b$, $\forall f \in \mathbf{C}[b,a]$, $\exists \underline{f} \in \mathbf{C}_t[b,a]$ such that $\underline{f}$ extends f.*

Proof.($\Rightarrow$) Set $\underline{f} = out \circ \tau_b(f)$. Then, for every $h \in \mathbf{C}_t[c,b]$, such that $(f \circ h) \in \mathbf{C}_t[c,a]$,

$\underline{f} \circ h = out \circ \tau_b(f) \circ h$
$= out \circ \tau_c(ex_a \circ \tau_b(f) \circ h)$ by (2) in def. 2.6.2
$= out \circ \tau_c(f \circ h)$ by (1)
$= out \circ \tau_c(ex_a \circ \tau_a(id_a) \circ f \circ h)$ by (1)
$= out \circ \tau_a(id_a) \circ f \circ h$ by (2)
$= f \circ h$

($\Leftarrow$) just take $\underline{ex} \in \mathbf{C}_t[a^\circ,a]$. ♦

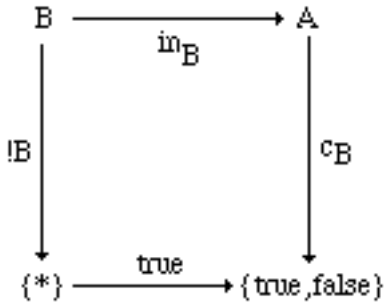
Exercises Prove the following facts:

1. $a^\circ < a^{\circ\circ}$.
2. $b < a < a^\circ \Rightarrow b < b^\circ$. (*Hint:* let $b < a$ via (i,j). For any $f \in \mathbf{C}[c,b]$ consider the extension $\underline{i \circ f} \in \mathbf{C}_t[c,a]$ of $i \circ f \in \mathbf{C}[c,a]$. Then $j \circ \underline{i \circ f} \in \mathbf{C}_t[c,b]$ and $\forall h$ into dom(f) $j \circ \underline{i \circ f} \circ h = j \circ (i \circ f \circ h) = f \circ h$.)

2.7 Subobject Classifiers and Topoi

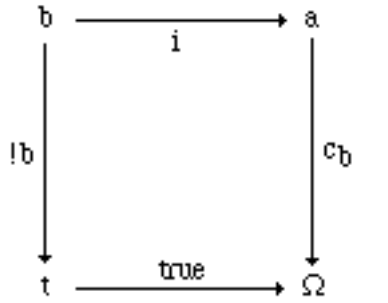
Subobjects and partial morphisms already forced some typical set-theoretic notions into the realm of categories. Let us take now a further step and categorically reconstruct power-sets and related constructions. By this, it will be possible to relativize these notions to structured sets. As for subobjects, the categorical version carries the structural information of the intended category.

In **Set** there is a one-to-one correspondence between subset of a set A and functions from A to 2={true, false}. The isomorphism takes every $B \subseteq A$ to its characteristic function $c_B: A \to 2$, defined by the following: $c_B(a)$ = true if $a \in B$, $c_B(a)$ = false if $a \notin B$. This isomorphism may be expressed in the categorical language by means of a pullback diagram



where in_B: B→A is the subset inclusion, and true: {*}→{true, false} takes * to "true."
The above diagram suggests the following definition in Category Theory:

2.7.1 Definition *Let* ***C*** *be a category with a terminal object t. A* ***subobject classifier*** *is an object* Ω *together with a morphism* ***true****: t→*Ω *such that for every monic i: b→a there is a unique arrow* c_b*: a→*Ω *such that the following diagram is a pullback diagram:*



Exercise A subobject classifier, when it exists in **C**, is unique up to isomorphisms.

The subobject classifier Ω plays a central role in the translation of Set Theory into Category Theory. It allows the simulation into the categorical language of concepts like intersection, union, and complement, by the definition of a Heyting algebra of truth-morphisms over Ω.

2.7.2 Definition *A* ***topos*** *is a category* ***C*** *with a terminal object, a subobject classifier, pullbacks for every pairs of arrows, and exponents for all pairs of objects.*

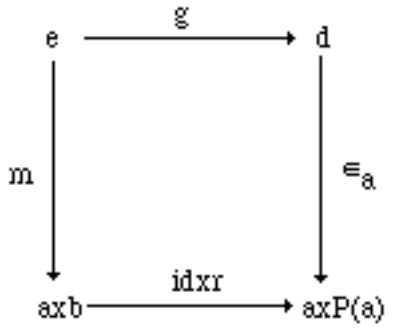
It will turn out that a topos is a "universe" where we can carry out constructions with almost the same confidence as we do in **Set**. Of course, **Set** itself is a topos.

Exercises

1. Prove that if **C** is a topos, then **C** is a CCC.
2. In a topos **C**, what is the arrow eval: $\Omega^a \times a \rightarrow \Omega$?

In the spirit of a categorical description of set-theoretic concepts, one may also talk in topoi of "relations", "powersets", and so forth. Quite generally, given objects a and b, a **relation** on a and b, is simply a monic r: d→a×b. In particular, then, one may consider power-objects P(a) as given by a relation $\in_a$: d→a×P(a) with the following universal property:

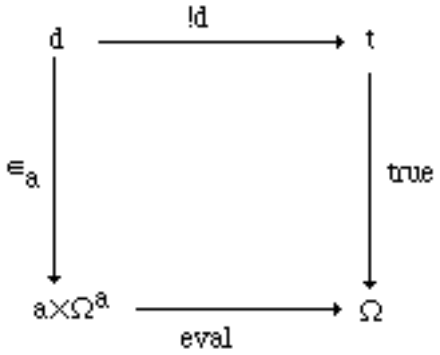
2.7.3 Definition *Let* ***C*** *be a cartesian category and* $a \in Ob_{\mathbf{C}}$. *The* ***power-object*** *P(a) is an object of* ***C*** *together with an object d and a monic* $\in_a$: *d→a×P(a) (a* ***membership-relation****) which is universal in the sense that, for any object b and monic m : e→a×b, there is a unique map r: b→P(a) and a (forcedly unique) map g: e→d to make the following square a pullback:*



As Ω is the truth value object, the set-theoretic intuition suggests that, in a topos, the object Ω^a should represent exactly the power-object of a.

2.7.4 Proposition *In a topos* ***C*** *every object a has a power-object* Ω^a.

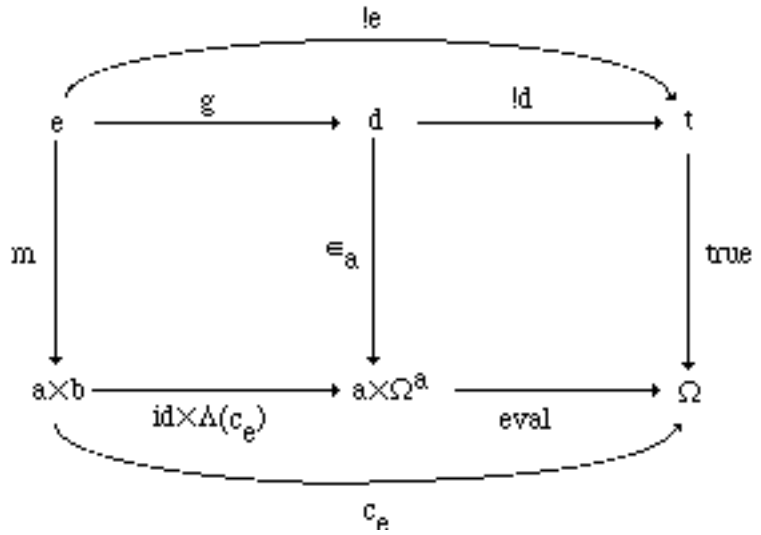
Proof. Let $\in_a$: d→a×P(a) be defined by the following pullback square:



Then the required universality of $\in_a$ is given by the properties

1. of the subobject classifier, i.e. the existence of the characteristic map c_e of m;
2. of the map eval , i.e. the existence and unicity of $\Lambda(c_e)$;
3. of the above pullback ;

an by an application of the pullback lemma 2.5.7. All this is described by the following commuting diagram, where the squares are pullbacks:



◆

The reader who has completed exercise (2) above may easily understand the intuitive meaning of the construction in 2.7.3 and compare it to his set-theoretic understanding. In particular, eval says whether "an element of a is in a given subset of a ."

Exercises

1. Prove that any topos has lifting.
2. Prove that a category **C** is a topos if and only if it has a terminal objects, and all pullbacks and powerobjects.

References The general notions can be found in the texts mentioned at the end of chapter 1, though their presentation and notation may be different. For example, in the case of CCC's, this is so also because these categories play a greater role in the categorical approach to Type Theory, or to denotational semantics, than in other applications of Category Theory. Notice that **Grp** and **Top** are not CCC's and consider that the origin of Category Theory is largerly indebted to algebraic geometry.

Applications of universal concepts from Category Theory to Programming Language Theory have been developed by several authors, in particular for program specification. For instance, the work by the ADJ group is explicitly based on universal conditions for specification (initial algebras of abstract data types; see Goguen et al. (1973/78)). An early insight in the connections between programs and categories may be found in Burstall and Thatcher (1974). All these aspects go beyond the limited aims of this book, which privileges "theories of functions" over "algebraic theories." Some more notions with an algebraic flavor may be found in chapter 4.

As for the examples we have provided, they originated entire areas of research. In particular, the category **EN** was first introduced by Malcev, as a general setting of Recursion Theory, and widely used in Ershov (1973/75). Scott domains and related categories are broadly treated in several places,

e.g. Scott (1981/82), Scott and Gunter (1989) or many books in denotational semantics. Scott (1976) presents Pω as a reflexive object and discusses categories of retractions. The (generalized) Myhill-Shepherdson theorem and related matters may be found in Scott (1976), Giannini and Longo (1984), Berger (1986), Rosolini (1986), and Longo (1988a), among others.

Coherent domains are given in Berry (1978) and used in Girard (1986), where linear maps are introduced (with some relevant consequences, see section 4.4).

There is a broad literature on categories with partial maps. We partly followed the approach in DiPaola and Heller (1984), and in Longo and Moggi (1984), where the notions of "complete object" and *partial* Cartesian Closed Category were introduced (and applied in Asperti and Longo (1987)). The properties of complete objects carry on also when using a more topos-theoretic perspective, as shown in Moggi (1988), which is devoted to a deeper insight into the concepts just sketched here. New results and surveys on categories with partial morphisms may be also found in Rosolini (1986), Moggi (1988a), Robinson and Rosolini (1988) and Curien and Obtulowicz (1988). Independently of the category-theoretic approach, Plotkin (1984) used the category **pCPO** for the purposes of denotational semantics.

Toposes originated by works of Lawvere and Tierney. A (difficult) introduction to Topos Theory may be found in Johnstone (1977). The reader may also consult Goldblatt (1979) or Barr and Wells (1985) on this subject.

Chapter 3

FUNCTORS AND NATURAL TRANSFORMATIONS

The starting point of Category Theory is the premise that every kind of mathematically structured object comes equipped with a notion of "acceptable" transformation or construction, that is, a morphism that preserves the structure of the object. This premise holds for categories themselves: a **functor** is the "natural" notion of morphisms between categories. By a further step, the question about what a morphism between functors should look like suggests the notion of **natural transformation**.

Of course, this is not a matter of arbitrary generalizations: it is a habit of mathematics to build constructions on top of constructions, since one may understand in this way, by the uniformity of methods and language, the reasonableness of specific constructions. And this theoretical unification, suggesting power and intellectual unity, is one of the major merits of Category Theory and its applications.

Moreover, as we shall see in several examples, there are notions which are clarified in an essential way or even suggeted by the categorical language of functors and natural transformations. Indeed, there are even too many examples for our purposes, and we will be able to mention only a few that are easily met in computer science.

3.1 Functors

If a transformation F between two categories **C** and **D** must map the categorical structure of **C** to that of **D**, it must take objects and morphisms of **C** to objects and morphisms of **D**; moreover, it must preserve source, target, identities and composition. Such a transformation F: $\mathbf{C}\to\mathbf{D}$ is called a **functor**.

3.1.1 Definition *Let* $\boldsymbol{C}$ *and* $\boldsymbol{D}$ *be categories. A (covariant)* ***functor*** $F : \boldsymbol{C} \to \boldsymbol{D}$ *is a pair of operations* $F_{ob}: Ob_{\boldsymbol{C}} \to Ob_{\boldsymbol{D}}$, $F_{mor}: Mor_{\boldsymbol{C}} \to Mor_{\boldsymbol{D}}$ *such that, for each* $f: a\to b$, $g: b\to c$ *in* $\boldsymbol{C}$,

- $F_{mor}(f) : F_{ob}(a)\to F_{ob}(b)$
- $F_{mor}(g \circ f) = F_{mor}(g) \circ F_{mor}(f)$
- $F_{mor}(id_a) = id_{F_{ob}(a)}$.

It is usual practice to omit the subscripts "ob" and "mor" as it is always clear from the context whether the functor is meant to operate on objects or on morphisms.

1.2 Examples

1. If **C** and **D** are preorders, then a functor F: **C**→**D** is just a monotone function from the objects of **C** to the objects of **D**, indeed $a <_{\mathbf{C}} b \Leftrightarrow \exists f \in \mathbf{C}[a,b] \Rightarrow F(f) \in \mathbf{D}[F(a),F(b)] \Leftrightarrow F(a) <_{\mathbf{D}} F(b)$. Conversely, given a monotone function f between two partial orders C and D, there is of course only one way to extend f to a functor among the categories associated with C and D.

2. The identity functor **I**: **C**→**C** is defined by a pair of identity operations both on objects and on morphisms of **C**. Similarly, we define the **inclusion functor Inc**: **C**→**D** when **C** is a subcategory of **D**.

3. If **C** is a Cartesian category define ×: **C**×**C**→**C** as the pair of operations that take $(a,b) \in Ob_{\mathbf{C}\times\mathbf{C}}$ to $\times(a,b) = a \times b \in Ob_{\mathbf{C}}$, and $(f,g) \in \mathbf{C}\times\mathbf{C}[(a,b),(c,d)]$ to $\times(f,g) = f \times g = \langle f \circ p_1, g \circ p_2 \rangle \in \mathbf{C}[a \times b, c \times d]$. × is a functor. Indeed × preserves sources and targets and, moreover,

$$\times(id_a,id_b) = id_a \times id_b = \langle p_a,p_b \rangle = id_{a\times b} = id_{\times(a,b)}$$

$$\times((f,g) \circ (h,k)) = \times(f \circ h, g \circ k) = (f \circ h) \times (g \circ k) = (f \times g) \circ (h \times k) = \times(f,g) \circ \times(h,k)$$

× is usually called **product functor** (since products are only determined up to isomorphisms, we must assume here a canonical choiche, for each object a, b, of their product a×b).

4. The **power-set functor P**: **Set**→**Set** takes every set A to its power-set **P**(A) and every function f: A→B to the function **P**(f): **P**(A)→**P**(B) defined by

$$\forall A' \subseteq A \quad \mathbf{P}(f)(A') = \{y \in B \mid \exists x \in A', y = f(x)\}.$$

Note that the set **P**(f)(A') is what is usually called f(A').

A functor F: **C**→**D** **preserves** a property P that an arrow f may have in **C**, if the arrow F(f) has the same property in **D**. For example, every functor preserves isomorphisms, or the property of being a component of a retraction pair, as it is stated in the following proposition:

3.1.3 Proposition *Let F: **C**→**D** be a functor. If a<b (a ≅ b) in **C**, then F(a) <F(b) (F(a) ≅ F(b)) in **D**.*

Proof. If f∘g = id, then F(f)∘F(g) = F(f∘g) = F(id) = id. ♦

Exercise Does every functor F preserve the property of being monic or epic?

A functor F: **C**→**D** is **faithful** if for all $a,b \in Ob_{\mathbf{C}}$ and for all $f,g \in \mathbf{C}[a,b]$, F(f) = F(g) implies f = g; it is **full** if for all $a,b \in Ob_{\mathbf{C}}$ and every $h \in \mathbf{D}[F(a),F(b)]$ there is $g \in \mathbf{C}[a,b]$ such that h = F(g). A functor F: **C**→**D** is a **full embedding** iff it is full and faithful, and it is also injective for objects.

A functor that "forgets" (part of) the intended structure of the objects is called **forgetful**: typical examples are the functors from **Grp**, **Top**, or **PO** to **Set** which assign to each object its set of elements and to each arrow the function associated with it. This notion is not a precise one, but it is

usually clear when a functor can be considered "forgetful"; note however that a forgetful functor should always be faithful (and very rarely full).

The definition of functor we have given preserves also the *direction* of the arrows in **C**. In some cases, it is interesting to study transformations among categories that reverse such directions, that is, by mapping sources to targets and vice versa. A transformation of this kind between two categories **C** to **D** may be considered as a functor from the dual category $\mathbf{C}^{op}$ to **D**, and it is known as **contravariant functor** (from **C** to **D**). Explicitly, note the following definition

3.1.4 Definition *Let **C** and **D** be categories. A **contravariant functor** $F: \mathbf{C}\to\mathbf{D}$ is a pair of operations $F_{ob}: Ob_{\mathbf{C}}\to Ob_{\mathbf{D}}$, $F_{mor}: Mor_{\mathbf{C}}\to Mor_{\mathbf{D}}$ such that for each $f: a\to b$, $g: b\to c$ in **C**,*

- $F_{mor}(f) : F_{ob}(b)\to F_{ob}(a)$
- $F_{mor}(g\circ f) = F_{mor}(f)\circ F_{mor}(g)$
- $F_{mor}(id_a) = id_{F_{ob}(a)}$

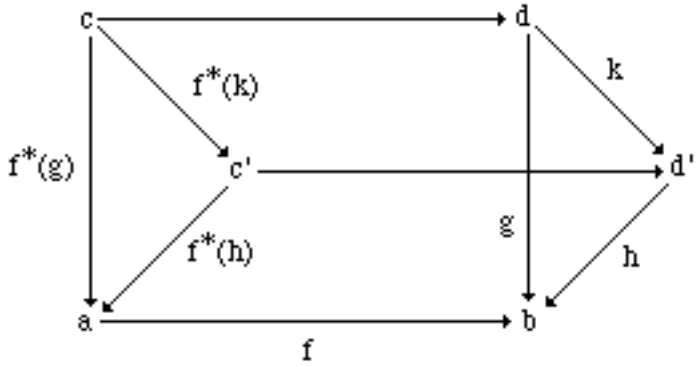
Examples

1. Each functor $F: \mathbf{C}\to\mathbf{D}$ defines a contravariant functor $F^{op}: \mathbf{C}^{op}\to\mathbf{D}$ that coincides with F on objects and such that $F^{op}(f) = F(f^{op})$, $F^{op}(g\circ f) = F^{op}(f)\circ F^{op}(g)$. Of course, $F = (F^{op})^{op}$.

2. The **duality** functor $(\ )^{op}: \mathbf{C}\to\mathbf{C}^{op}$ is a contravariant functor such that $(a)^{op} = a$ for $a\in Ob_{\mathbf{C}}$ and $(f)^{op} = f$ for $f\in Mor_{\mathbf{C}}$.

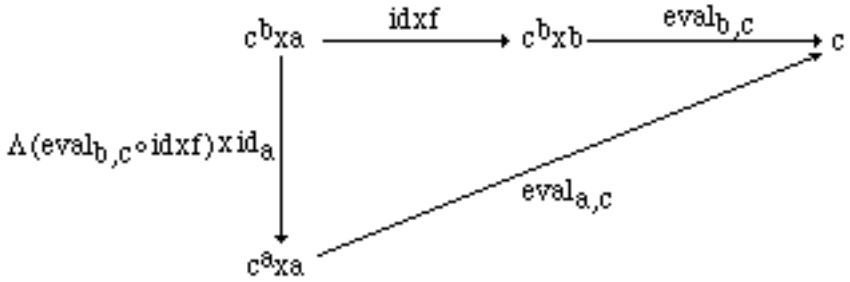
3. The function that takes every set A to its power-set **P**(A) may be also extended in a natural way to a **contravariant power-set functor P**: **Set**→**Set**. Just define, for $f: A\to B$, $\mathbf{P}(f): \mathbf{P}(B)\to\mathbf{P}(A)$ as the function that assigns to each $B'\subseteq B$ the set $\mathbf{P}(f)(B') = f^{-1}(B') = \{x\in A / \exists y\in B', f(x)=y\}$.

4. Let **C** be a category with pullbacks, and let $f\in\mathbf{C}[a,b]$. Then f induces a **pullback functor** f^*: $\mathbf{C}\downarrow b\to\mathbf{C}\downarrow a$, whose action is described in the following diagram:



In this diagram, g and h are objects of $\mathbf{C}\downarrow b$, and $k\in\mathbf{C}\downarrow b[g,h]$; $f^*(g)$ and $f^*(h)$ are the pullbacks of g and h along f, yielding a unique arrow $f^*(k)$ making the above diagram commute. The proof that f^* is a functor is a consequence of elementary properties of pullback, and it is left as an exercise for the reader.

5. Let **C** be a CCC. For every $c \in Ob_{\mathbf{C}}$ the **exponent functor** $\mathbf{exp_c}: \mathbf{C} \to \mathbf{C}$ is defined as follows. For every $a \in Ob_{\mathbf{C}}$, set $\mathbf{exp_c}(a) = c^a$, and for every $f \in \mathbf{C}[a,b]$, $\mathbf{exp_c}(f) = \Lambda(eval_{b,c} \circ id \times f) : c^b \to c^a$. As a diagram,



In order to check that $\mathbf{exp_c}$ is actually a contravariant functor, we must take these steps:

$\mathbf{exp_c}(id) = \Lambda(eval \circ id \times id) = \Lambda(eval) = id$

$\mathbf{exp_c}(f \circ g) = \Lambda(eval \circ id \times(f \circ g))$

$= \Lambda(eval \circ id \times f \circ id \times g)$

$= \Lambda(eval \circ \Lambda(eval \circ id \times f) \times id \circ id \times g)$ by the diagram

$= \Lambda(eval \circ id \times g \circ \Lambda(eval \circ id \times f) \times id)$

$= \Lambda(eval \circ id \times g) \circ \Lambda(eval \circ id \times f)$

$= \mathbf{exp_c}(g) \circ \mathbf{exp_c}(f)$. ♦

Note that the composition of two functors is still a functor and that there exists the identity functor which is right and left invariant w.r.t. composition. Thus, it is sound to define the category **Cat** whose objects are categories and whose morphisms are functors. It is worth pointing out that this is a convenient and unusual algebraic property. The collection of groups is not (naturally) a group; similarly for topological spaces and so on. The definition of **Cat**, so similar to the paradoxical "set of all sets" of Set Theory, must be used with some caution, however. It is not our intention to engage in foundational questions in this book, but it is time to say a few more words about our "axiomatic" defintion of Category. So far we have made no assumption at all about the dimension of the classes $Ob_{\mathbf{C}}$ and $Mor_{\mathbf{C}}$; the word "class" itself has been used in an intuitive sense, without any reference to an underlying Set Theory. From now on, though, the reader may refer to the notions of set and class as used, say, in Gödel-Bernays Set Theory, in order to make this distinction clear. From this perspective, a Category **C** such that $Ob_{\mathbf{C}}$ and $Mor_{\mathbf{C}}$ are sets, is called **small**. In the sequel, when we refer to **Cat**, we mean the category of all small categories; of course, **Cat** itself is not a small category, and thus it is not among its own objects.

(*Question*: which of the categories mentioned so far are small, and which are not?)

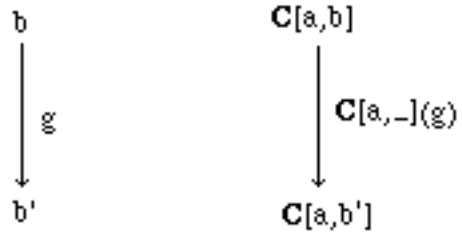
If **C** is a small category, then, for all $a,b \in Ob_{\mathbf{C}}$, $\mathbf{C}[a,b]$ is a set, usually called **hom-set** of a and b (some authors use the word "hom-set" in an arbitrary category). We say that a category **C** is **locally small** when for all $a,b \in Ob_{\mathbf{C}}$, $\mathbf{C}[a,b]$ is a set and not a class. If **C** is a locally small

category, it is possible to define some functors from **C** to **Set**, called **hom-functors**, that play a central role in the developments of Category Theory. We shall see later that most of this work may be done at a more abstract level, taking an arbitrary topos **D** instead of **Set**.

3.1.5 Definition *Let* $\mathbf{C}$ *be a locally small category. Given* $a \in Ob_{\mathbf{C}}$, *the* ***covariant hom-functor*** $\mathbf{C}[a,\_]: \mathbf{C} \to \mathbf{Set}$ *is given by:*

i. for every $b \in Ob_{\mathbf{C}}$ $\mathbf{C}[a,\_](b) = \mathbf{C}[a,b] \in Ob_{\mathbf{Set}}$;

ii. for every $g \in \mathbf{C}[b,b']$, $\mathbf{C}[a,\_](g): \mathbf{C}[a,b] \to \mathbf{C}[a,b']$ *is the function that takes* $f \in \mathbf{C}[a,b]$ *to* $(g \circ f) \in \mathbf{C}[a,b']$ *that is,*

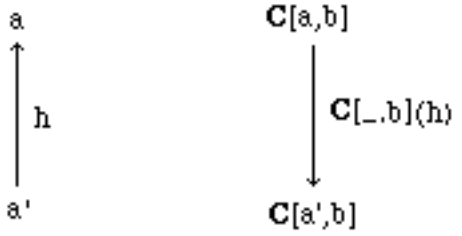


The function $\mathbf{C}[a,\_](g)$ will be denoted in the sequel by "$\mathbf{C}[a,g]$" or also by the suggestive "$g \circ \_$" (some authors use also $g*$). Note that the drawing above is clearly an implication between diagrams. Their plain juxtaposition will be used often in the sequel with this meaning, as a special case of the "conditional equational reasoning" mentioned earlier in section 1.2.

3.1.6 Definition *Let* $\mathbf{C}$ *be a locally small category. Given* $b \in Ob_{\mathbf{C}}$, *the* ***contravariant hom-functor*** $\mathbf{C}[\_,b]: \mathbf{C} \to \mathbf{Set}$ *is given by:*

i. for every $a \in Ob_{\mathbf{C}}$ $\mathbf{C}[\_,b](a) = \mathbf{C}[a,b] \in Ob_{\mathbf{Set}}$

ii. for every $h \in \mathbf{C}[a',a]$, $\mathbf{C}[\_,b](h): \mathbf{C}[a,b] \to \mathbf{C}[a',b]$ *is the function that takes* $f \in \mathbf{C}[a,b]$ *to* $f \circ h \in \mathbf{C}[a',b]$ *that is*



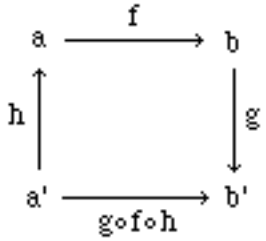
The function $\mathbf{C}[\_,b](h)$ will be denoted in the sequel by "$\mathbf{C}[h,b]$" or by the suggestive "$\_ \circ h$" (some authors use also h^*.)

3.1.7 Definition *Let* $\mathbf{C}$ *be a locally small category. Given* $b \in Ob_{\mathbf{C}}$, *the* ***hom-functor*** $\mathbf{C}[\_,\_]: \mathbf{C}^{op} \mathbf{x} \mathbf{C} \to \mathbf{Set}$ *is given by:*

i. for every $(a,b)\in Ob_{\mathbf{C}\times\mathbf{C}}$ $\mathbf{C}[\_,\_](a,b) = \mathbf{C}[a,b]\in Ob_{\mathbf{Set}}$

ii. for every $h\in\mathbf{C}[a',a]$, $g\in\mathbf{C}[b,b']$, $\mathbf{C}[\_,\_](h,g): \mathbf{C}[a,b]\to\mathbf{C}[a',b']$ *is the function that takes* $f\in\mathbf{C}[a,b]$ *to* $g\circ f\circ h\in\mathbf{C}[a',b']$

C[_,_] is contravariant in the first argument and covariant in the second. One may understand how **C**[_,_] works on morphisms by the following commutative diagram:



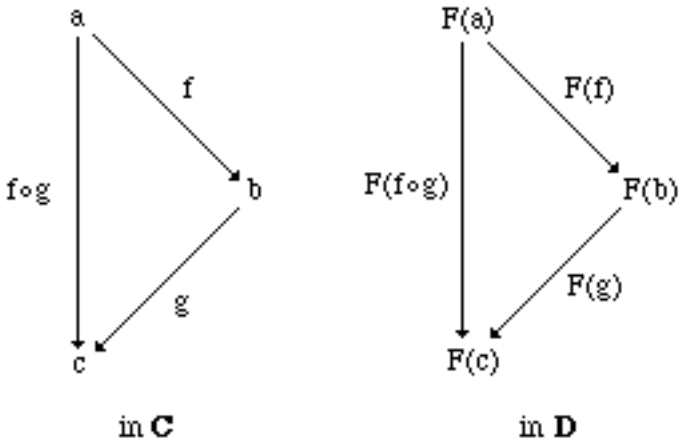
The function **C**[_,_](h,k) will be denoted by "**C**[h,k]" or by "g ∘_ ∘ h."

Exercise: prove in details that **C**[a,_] and **C**[_,b] are a covariant and a contravariant functor, respectively.

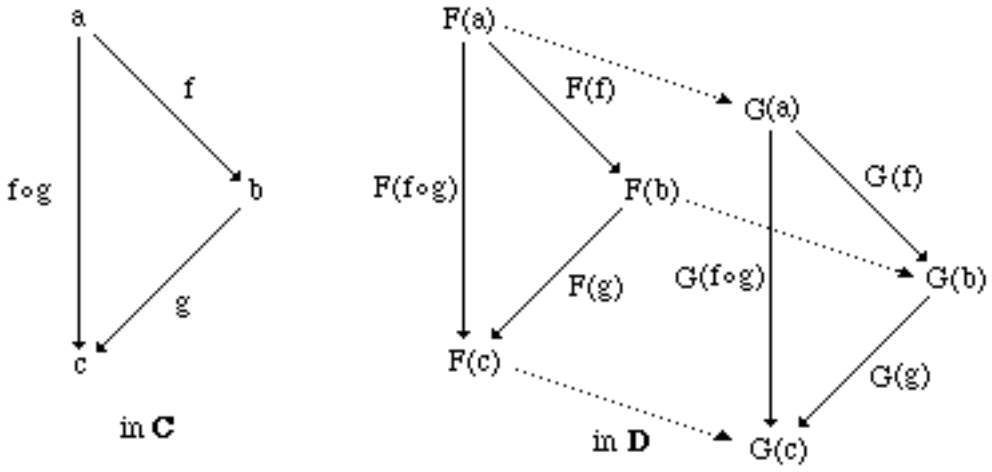
Exercise A category **C** has enough points iff the functor **C**[t, -] is faithful, when t is terminal.

3.2 Natural Transformations

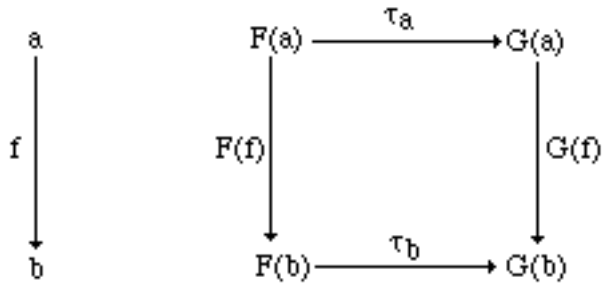
The fact that F is a functor from a category **C** to a category **D** may be equivalently expressed by F(id) = id and, for every f and g in $\mathrm{Mor}_{\mathbf{C}}$, by the following (implication between) diagrams:



Consider now two functors F, G: **C**→ **D**. A quite reasonable idea of transformation from F to G is a "translation" as described in the following picture, where the dotted lines should yield commutative squares



Thus, the "translation" can be defined by assigning to each object $a \in Ob_{\mathbf{C}}$ an arrow τ_a: $F(a) \to G(a)$, with the only condition that, for every $f \in \mathbf{C}[a,b]$, the following diagram commutes:



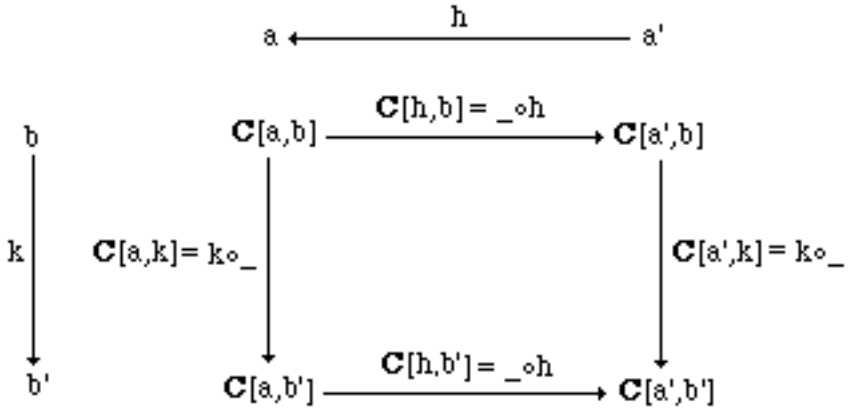
The properties described in this diagram are equivalently formalized by the following definition.

3.2.1 Definition. *Let* $F,G : \mathbf{C} \to \mathbf{D}$ *be functors. Then* $\tau : F \to G$ *is a* ***natural transformation*** *from* F *to* G *iff:*

i. $\forall a \in Ob_{\mathbf{C}} \quad \tau_a \in \mathbf{D}[F(a),G(a)]$

ii. $\forall f \in \mathbf{C}[a,b] \quad \tau_b \circ F(f) = G(f) \circ \tau_a$.

3.2.2 Example Let $\mathbf{C}$ be a small category, and $h \in \mathbf{C}[a',a]$. The collection (in **Set**) of morphisms $\{\mathbf{C}[h,b] / \mathbf{C}[a,b] \to \mathbf{C}[a',b] \}_{b \in \mathbf{C}}$, defines a natural transformation $\mathbf{C}[h,\_]$ from the (contravariant) hom-functor $\mathbf{C}[a,\_]$ to the (contravariant) hom-functor $\mathbf{C}[a',\_]$. Note the following diagram:



The same diagram proves that, given $k \in \mathbf{C}[b,b']$, the collection of morphisms $\{\mathbf{C}[a,k] / \mathbf{C}[a,b] \to \mathbf{C}[a,b']\}_{a \in \mathbf{C}}$ defines a natural transformation $\mathbf{C}[\_,k]$ from the hom-functor $\mathbf{C}[\_,b]$ to the hom-functor $\mathbf{C}[\_,b']$.

It is easy to close up natural transformations under composition by setting $(\tau \circ \beta)_a = (\tau_a) \circ (\beta_a)$. This composition of natural transformation is usually called **vertical**, as opposed to the **horizontal composition**, defined at the end of this section. Since the identity transformation from a functor F to itself is defined in the obvious way, we have actually constructed a new category, starting from any two given categories **C** and **D**.

The new category is called the **category of functors** from **C** to **D**, **(C→D) = Funct(C,D)**; its objects are functors and the morphisms are natural transformations. In particular, if F,G: **C→D**, **Funct(C,D)**[F,G] is the collection of all the natural transformations from F to G; in the following we shall use the abbreviation **Nat**(F,G) instead of **Funct(C,D)**[F,G].

Two functors from **C** to **D** are equivalent (or **naturally isomorphic**) iff they are isomorphic as objects of **Funct(C,D)**. For example, it is well understood that any set A is isomorphic to A×1, where 1 is a singleton. For arbitrary cartesian categories, this corresponds to saying that the functor _×1 and the identity functor Id are naturally isomorphic. If **F**: **C→D** is a full embedding, then **C** is isomorphic to a full subcategory of **D**. The next section will present further examples of natural isomorphisms.

The concept of natural isomorphism of functors also allows us to define a notion of "equivalence" between categories, which captures better than the notion of isomorphism the sense that two categories can be said to be "essentially the same." Two categories **C** and **D** are **equivalent** if and only if there are two functors F: **C→D** and G : **D→C** such that $G \circ F \cong id_{\mathbf{C}}$ and $F \circ G \cong id_{\mathbf{D}}$ (note that **C** is isomorphic to **D** iff $G \circ F = id_{\mathbf{C}}$ and $F \circ G = id_{\mathbf{D}}$).

3.2.3 Proposition *Let* $F,G : \mathbf{C} \to \mathbf{D}$ *be functors and* $\tau : F \to G$ *be a natural transformation from F to G. Assume that, for each* $a \in Ob_{\mathbf{C}}$, $\tau_a \in \mathbf{D}[F(a),G(a)]$ *is an isomorphism. Then* τ *is a natural isomorphism.*

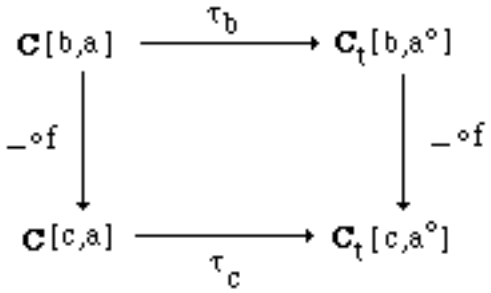
Proof Define $\tau^{-1} : G \to F$ by $\tau^{-1}{}_a = (\tau_a)^{-1}$. τ^{-1} is natural, since $\forall f \in \mathbf{C}[a,b]$

$$\begin{aligned} \tau^{-1}{}_b \circ G(f) &= \tau_b{}^{-1} \circ G(f) \circ \tau_a \circ \tau_a{}^{-1} \\ &= \tau_b{}^{-1} \circ \tau_b \circ F(f) \circ \tau_a{}^{-1} \\ &= G(f) \circ \tau^{-1}{}_a. \ \blacklozenge \end{aligned}$$

Examples A simple example of natural transformation may be given by studying "liftings" (see section 2.6), in various categories. One may actually understand the general notion better, by completing a little exercise on natural transformations. Indeed, what is hidden behind definition 2.6.2, is the "naturality" of τ_c. When writing this down explicitly, the definition is tidier and more expressive. Let $\mathbf{C}$ be a Category of partial maps, $\mathbf{C_t}$ be the associated category of total maps, and **Inc**: $\mathbf{C_t} \to \mathbf{C}$ be the obvious inclusion. Thus, 2.6.2 may be simply restated as

*The **lifting** of $a \in Ob_{\mathbf{C}}$ is the object a° such that the functors $\mathbf{C}[\_,a] \circ \mathbf{Inc}, \mathbf{C_t}[\_,a^\circ] : \mathbf{C_t} \to \mathbf{Set}$ are naturally isomorphic.*

Then, by definition of natural transformation and hom-functor, this requires the existence of a function τ such that the following diagram commutes, for any object b and c in $\mathbf{C}$ and $f \in \mathbf{C_t}[c,b]$



That is, $\tau_c(g \circ f) = \tau_b(g) \circ f$ and $(\tau_c)^{-1}(h \circ f) = (\tau_b)^{-1}(h) \circ f$, for any total f, since τ is an isomorphism.

With this definition, to prove unicity of liftings is even smoother than in section 2.5. Indeed, let τ be the given natural isomorphism, and let a' and β be an alternative lifting and natural isomorphism. Set $\phi = \tau \circ \beta^{-1}$. Then $\mathbf{C_t}[\_,a']$ and $\mathbf{C_t}[\_,a^\circ]$ are naturally isomorphic via ϕ and, for $f = \phi_{a'}(id): a' \to a^\circ$ and $g = (\phi_{a^\circ})^{-1}(id): a^\circ \to a'$, one has:

$$\begin{aligned} g \circ f &= (\phi_{a^\circ})^{-1}(id) \circ f \\ &= (\phi_{a'})^{-1}(id \circ f) \qquad \text{by naturality} \\ &= (\phi_{a'})^{-1}(\phi_{a'}(id)) \\ &= id. \end{aligned}$$

Similarly for $f \circ g = id$ (on a°).

Since we are now familiar with functors, we may look also at lifting as a functor.

Let **C** be a pC and assume that for each $a \in Ob_{\mathbf{C}}$ there exists the lifting a°. Then there is a (unique) extension of the map $a \vdash a^\circ$ to a functor $\_^\circ : \mathbf{C} \to \mathbf{C_t}$ (the **lifting functor**).

The reader may check this as an exercise (*hint:* observe that $ex_a = (\tau_{a^\circ})^{-1}(id): a^\circ \to a$ and use the naturality of τ) .

As for specific examples, the lifting functor for **pSet** is obvious. It can be easily guessed also for the category **pPo** of p.o.sets and partial monotone functions with upward closed domains: just add a fresh least element and the rest is easy for the reader who has completed the last exercise. Note, and it is crucial, that by monotonicity the lifting functor does not exist if one doesn't assume that the domains are upward closed.

The category **pCPO** is given by defining complete partial orders under the assumption that directed sets are not empty. Thus, the objects of **pCPO** do not need to have a bottom element. As for morphisms, take the partial continuous functions with open subsets as domains. Clearly, the lifting functor is defined as it is for **pPo.**

A more complex example is given by **EN**, the category of numbered sets in section 2.2. Let **pEN** be the partial category of numbered sets in the example before 2.5.2. Given $\underline{a} = (a,e) \in Ob_{\mathbf{pEN}}$, define $\underline{a^\circ} = (a^\circ, e^\circ)$ by adding a new element $\perp$ to the set a and by defining $e^\circ(n) = \underline{\text{if}}\ \phi_n(0)$ converges $\underline{\text{then}}\ e(\phi_n(0))\ \underline{\text{else}}\ \perp$. Clearly, $e^\circ : \omega \to a^\circ$ is surjective. Let now $\underline{b} = (b,e')$ and $f \in \mathbf{pEN}[\underline{b},\underline{a}]$. By definition, there exists $f' \in PR$ such that $f \circ e' = e \circ f'$. We define the extension $\underline{f} \in \mathbf{EN}[\underline{b},\underline{a^\circ}]$ of f by giving $\underline{f}' \in R$ which represents $\underline{f}$. That is, set $\phi_{\underline{f}'(n)}(0) = f(n)$. Such an $\underline{f}' \in R$ exists by the s-m-n (iteration) theorem. Thus,

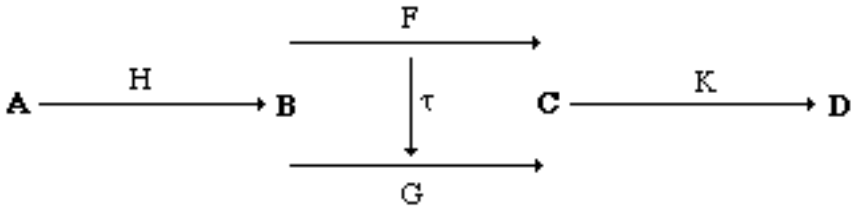
$$\begin{aligned} \underline{f}(e'(n)) &= e \circ (\underline{f}'(n)) \\ &= \underline{\text{if}}\ \phi_{\underline{f}'(n)}(0) \text{ converges } \underline{\text{then}}\ e(\phi_{\underline{f}'(n)}(0))\ \underline{\text{else}}\ \perp . \end{aligned}$$

Therefore, if $f(e'(n)) = e(f'(n))$ is defined, then $\phi_{\underline{f}'(n)}(0)$ converges and, hence, $\underline{f}(e'(n)) = e(\phi_{\underline{f}'(n)}(0)) = f(e'(n))$. Finally, set $\tau_{\underline{a}\underline{b}}(f) = \underline{f}$. For each $\underline{a}$, $\tau_{\underline{a}}$ gives the required natural isomorphism, as $\forall \underline{g} \in \mathbf{EN}[\underline{b},\underline{a^\circ}]\ \exists! f \in \mathbf{pEN}[\underline{b},\underline{a}]\ f'(n) = \phi_{\underline{g}'(n)}(0)$. (**Exercise**: check the due diagram). By the fact above, this defines the lifting functor in **pEN**.

Exercise Define the category **ER** of equivalence relations on ω and effective maps (*hint:* the objects are quotient sets on ω, and the morphisms are induced by total recursive functions similarly as for **EN**). Observe that **ER** and **EN** are equivalent, but not isomorphic. Indeed, one is small, while the other is not.

We next discuss ways to derive natural transformations from a given one and, finally, the notion of horizontal composition.

Let $H: A \to B$, $F: B \to C$, $G: B \to C$, $K: C \to D$ be functors, and let $\tau: F \to G$ be a natural transformation as shown in the following diagram:



τ induces two natural transformations $K\tau: KF \to KG$, and $\tau H: FH \to GH$, respectively defined by

$$(K\tau)_b = K(\tau_b) : KF(b) \to KG(b);$$

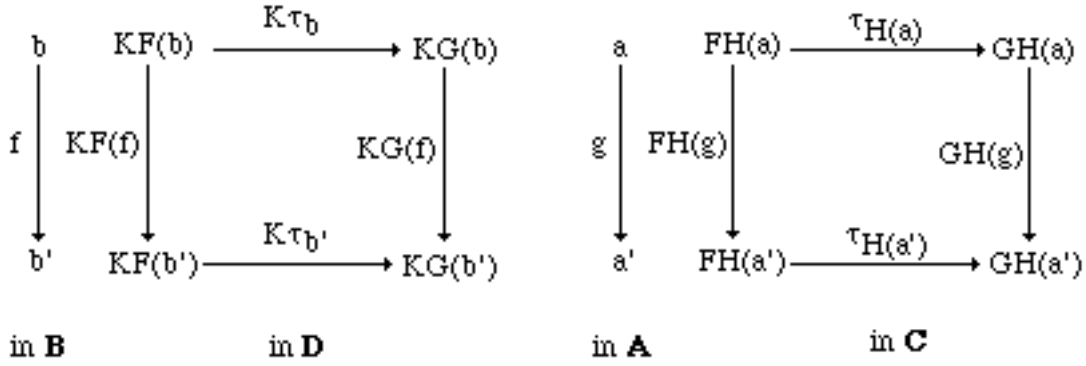
$$(\tau H)_a = \tau_{H(a)} : FH(a) \to GH(a)$$

We have, for every $f \in \mathbf{B}[b,b']$ and every $g \in \mathbf{A}[a,a']$,

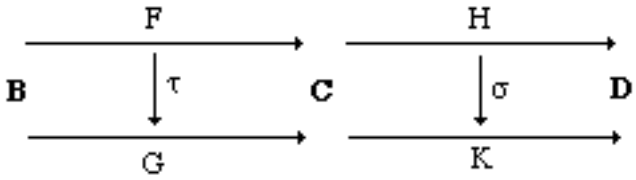
$$(K\tau)_{b'} \circ KF(f) = K(\tau_b) \circ K(F(f)) = K(\tau_b \circ F(f)) = K(G(f) \circ \tau_b) = KG(f) \circ K\tau_b = KG(f) \circ (K\tau)_b$$

$$(\tau H)_{a'} \circ FH(g) = \tau_{H(a')} \circ F(H(g)) = G(H(g)) \circ \tau_{H(a)} = GH(g) \circ (\tau H)_a$$

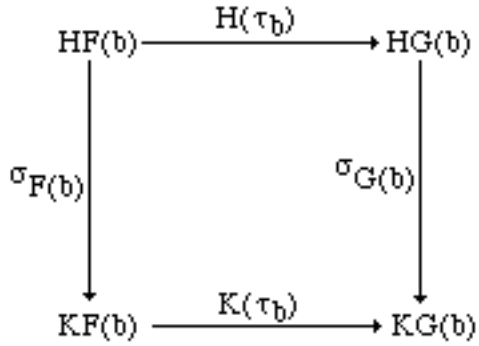
that proves the naturality of $K\tau$ and τH.



Consider now categories, functors, and natural transformations as described in the following diagram:



Then, for the naturality of σ with respect to the arrow τ_b, the following diagram (in **D**) commutes for every $b \in Ob\mathbf{B}$:



The **horizontal composition** of σ and τ is the natural trasformation $\sigma\tau$: $HF \to KG$ defined by, for every $b \in Ob_{\mathbf{B}}$, $\sigma\tau_b = \sigma_{G(b)} \circ H(\tau_b) = K(\tau_b) \circ \sigma_{F(b)}$.
We check the naturality of $\sigma\tau$. Let $f \in \mathbf{B}[b,b']$, then

$$\begin{aligned}
\sigma\tau_{b'} \circ HF(f) &= \sigma_{G(b')} \circ H(\tau_{b'}) \circ HF(f) \\
&= \sigma_{G(b')} \circ H(\tau_{b'} \circ F(f)) \\
&= \sigma_{G(b)} \circ H(G(f) \circ \tau_b) && \text{by the naturality of } \tau \\
&= \sigma_{G(b)} \circ H(G(f) \circ \tau_b) && \text{by the diagram} \\
&= K(G(f) \circ \tau_b) \circ \sigma_{F(b)} \\
&= KG(f) \circ K(\tau_b) \circ \sigma_{F(b)} \\
&= KG(f) \circ \sigma\tau_b
\end{aligned}$$

Note that if we identify the functors K and H with the identity natural transformation id_K and id_H, $K\tau$ and τH may be understood as particular cases of the horizontal application between natural transformations (why?).

Exercise Prove the following equality among natural transformations (**interchange law**):

$$(\nu \circ \mu)(\tau \circ \sigma) = (\nu\tau) \circ (\mu\sigma) .$$

3.3 Cartesian and Cartesian Closed Categories Revisited

By definition, a category $\mathbf{C}$ is Cartesian iff it contains a terminal object t, and for every $a,b \in Ob_{\mathbf{C}}$ there is an object $a \times b$ together with two morphisms $p_1: a \times b \to a$, $p_2: a \times b \to b$, and for every object c an isomorphism $< , >_c : \mathbf{C}[c,a] \times \mathbf{C}[c,b] \to \mathbf{C}[c,a \times b]$ such that for all morphisms $f: c \to a$, $g: c \to b$, $h: c \to a \times b$, the following equations hold:

i. $p_1 \circ <f,g>_c = f$
ii. $p_2 \circ <f,g>_c = g$

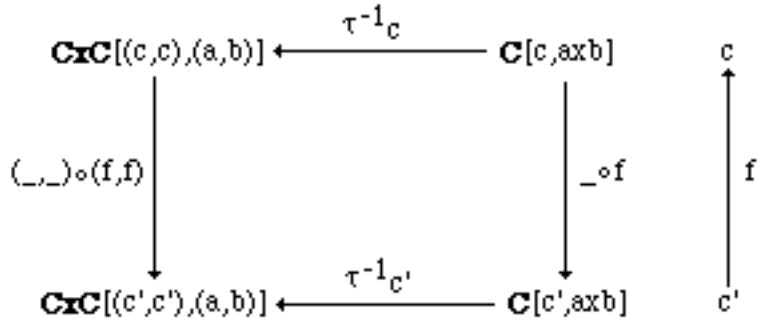
We want now to show that $< , >_c$ is also "natural in c", i.e. it satisfies the property:

(nat) for every $k: c \to c'$ $<f,g>_{c'} \circ k = <f \circ k, g \circ k>_c$

Indeed, we have

$$\begin{aligned}
<f,g>_{c'} \circ k &= <p_1 \circ <f,g>_{c'} \circ k, p_2 \circ <f,g>_{c'} \circ k>_c && \text{by ii)} \\
&= <f \circ k, g \circ k>_c && \text{by i)}
\end{aligned}$$

The previous "naturality" property suggests that $< , >$ is actually a natural isomorphism. Indeed, let $\Delta: \mathbf{C} \to \mathbf{C} \times \mathbf{C}$ the functor defined by $\Delta(c) = (c,c)$, $\Delta(f) = (f,f)$ (Δ is called the **diagonal functor**); then $< , >$ is a natural isomorphism from $\mathbf{C} \times \mathbf{C}[\_,(a,b)] \circ \Delta$ to $\mathbf{C}[\_,a \times b]$. Note that $\mathbf{C} \times \mathbf{C}[\_,(a,b)] \circ \Delta$ and $\mathbf{C}[\_,a \times b]: \mathbf{C} \to \mathbf{Set}$ are contravariant functors. Conversely suppose to have for all objects a and b an object $a \times b$ and a natural isomorphism $\tau : \mathbf{C} \times \mathbf{C}[\_,(a,b)] \circ \Delta \cong \mathbf{C}[\_,a \times b]$. The naturality of τ^{-1} is expressed by the following commutative diagram:



Let $(q_1,q_2) = \tau^{-1}_{a\times b}(id_{a\times b})$. Note that q_1: $a\times b\to a$, q_2: $a\times b\to b$. We want to prove that $a\times b$ is a product with projections q_1 and q_2. Indeed, let $f = \tau_c(h,g)$: $c\to a\times b$ in the above diagram, then we have $(q_1,q_2)\circ(\tau_c(h,g),\tau_c(h,g)) = \tau^{-1}_c(id_{a\times b}\circ\tau_c(h,g)) = (h,g)$ and, in particular, $q_1\circ\tau_c(h,g) = h$; $q_2\circ\tau_c(h,g) = g$.

The previous considerations suggest another characterization of Cartesian Category:

3.3.1 Proposition *A category* ***C*** *is Cartesian iff it contains a terminal object t, and for every* $a,b\in Ob_{\mathbf{C}}$ *there is an object* $a\times b$ *and a natural isomorphism* $< , > : \mathbf{C}\times\mathbf{C}[\_,(a,b)]\circ\Delta \to \mathbf{C}[\_,a\times b]$.

The situation is quite similar, and perhaps even simpler, in the case of exponents. Remember that a category **C** is a CCC iff it is Cartesian and has exponents for every pair of objects, i.e. for every a and b in **C** there is an object b^a together with a morphism $eval_{a,b}$: $b^a\times a\to b$ (evaluation map) and, for every object c , a function Λ_c : $\mathbf{C}[c\times a,b]\to\mathbf{C}[c,b^a]$ such that, for all morphisms f: $c\times a\to b$, h: $c\to b^a$, one has

$$\beta.\ \ eval_{a,b}\circ(\Lambda(f)\times id_a) = f$$

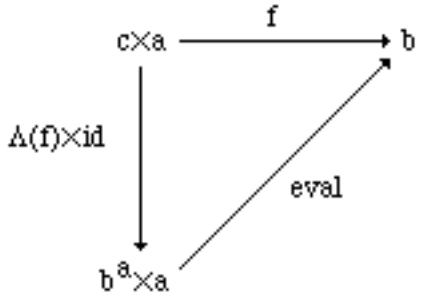
$$\eta.\ \ \Lambda(eval\circ(h\times id)) = h$$

It turns out that Λ_c is "natural in c", in the sense that for any $f\in\mathbf{C}[c\times a,b]$, $g\in\mathbf{C}[d,c]$

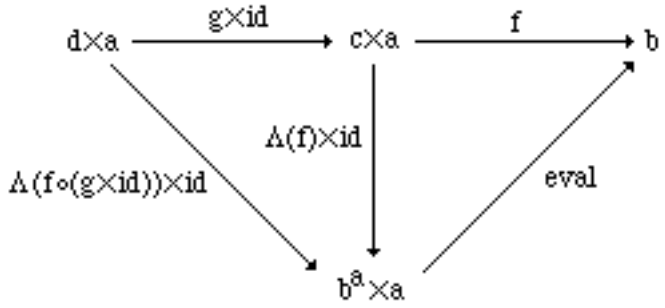
$$\Lambda_{c'}(f)\circ g = \Lambda_c(f\circ g\times id)\ ,$$

In order to check this, recall that (β) corresponds to:

(Diag.Eval)



Thus, by twice (Diag.Eval), the following diagram commutes



Moreover, set $\Lambda^{-1} = \mathrm{eval}\circ(\_\times id)$.

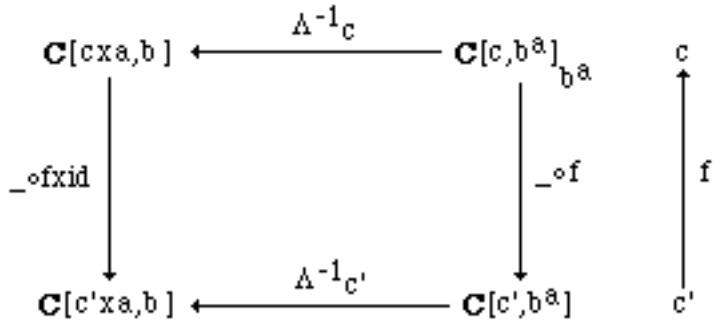
Then $\Lambda^{-1}\circ\Lambda = id$ by (β)

and $\Lambda\circ\Lambda^{-1} = id$ by (η).

That is, for each $c\in Ob\mathbf{C}$, Λ_c: $\mathbf{C}[c\times a,b]\to\mathbf{C}[c,b^a]$ is an isomorphism. Thus Λ is a natural isomorphism, by proposition 3.2.3.

Note that, formally, Λ is a natural transformation from the contravariant functor $\mathbf{C}[\_,b]\circ\_\times a$ to the contravariant functor $\mathbf{C}[\_,b^a]$ (where $\_\times a$ is the product functor defined in the obvious way). We abbreviate $\mathbf{C}[\_,b]\circ\_\times a$ as $\mathbf{C}[\_\times a,b]$. Note also that $\mathbf{C}[\_\times a,b]$, $\mathbf{C}[\_,b^a]$: $\mathbf{C}\to\mathbf{Set}$.

Conversely, suppose that, for all objects a and b, there is an object b^a and a natural isomorphism Λ: $\mathbf{C}[\_\times a,b]\cong\mathbf{C}[\_,b^a]$. Then the naturality of Λ^{-1} is expressed by the following commutative diagram:



Set now $\mathrm{eval}_{a,b} = \Lambda^{-1}{}_{ba}(id_{ba})$ and note that $\mathrm{eval}_{a,b}$: $b^a\times a\to b$. We want to prove that b^a is an exponent with $\mathrm{eval}_{a,b}$ as evaluation map. Indeed, let $f = \Lambda_c(g)$: $c\to b^a$ in the above diagram. Then we have:

$$\mathrm{eval}_{a,b}\circ(\Lambda_c(g)\times id) = \Lambda^{-1}{}_c(id_{ba}\circ\Lambda_c(g)) = g$$

This argument gives the following equivalent characterization of CCC:

3.3.2 Proposition *$\mathbf{C}$ is a **Cartesian closed category** iff it is Cartesian and for every $a,b\in Ob\mathbf{C}$ there is an object b^a and a natural isomorphism $\Lambda : \mathbf{C}[\_\times a,b]\to\mathbf{C}[\_,b^a]$.*

Observe that definitions 3.3.1 and 3.3.2 require that the category **C** be locally small, since they are based on hom-functors. Thus, in a sense, the equational definition is more general.

3.3.3 Remark It is easy to prove that the following (natural) isomorphisms hold in all CCC's, for any object A, B, and C:

1. $A \cong A$;
2. $t \times A \cong A$;
3. $A \times B \cong B \times A$;
4. $(A \times B) \times C \cong A \times (B \times C)$;
5. $(A \times B) \to C \cong A \to (B \to C)$;
6. $A \to (B \times C) \cong (A \to B) \times (A \to C)$;
7. $t \to A \cong A$;
8. $A \to t \cong t$.

What is worth mentioning, though, is that these are exactly the isomorphisms that hold in all CCC's, i.e., no other isomorphism is valid in all CCC's. The proof of this fact is nontrivial and is a nice application of lambda calculus to categories (an application in the other direction from what is meant by the title of this book!). Its key idea will be mentioned in chapter 9.

3.4 More Examples of CCC's

Both examples here derive from bordering areas of (generalized) computability and Proof Theory. The first, in particular, like many aspects of these theories, is widely used in the type theoretical understanding of programming languages constructs.

3.4.1 Partial Equivalence Relations

A very relevant example of CCC is suggested by a long story. It began with Kleene's realizability interpretation of intuitionistic logic and continued with the work of Kreisel, Girard and Troelstra in Proof Theory. The idea is to look at functions as computable ones, as in the case of the category **EN**, but by a simple and insightful way cartesian closedness is obtained. The approach is also used in the "quotient-set" semantics of types, in functional programming. It will give us a paradigmatic structure in the categorical semantics of polymorphism in PART II.

As usual, let $\varphi: \omega \to PR$ an acceptable goedel numbering of the partial recursive functions. Let $K = (\omega, \cdot)$ be **Kleene's applicative structure**, where $\cdot : \omega \times \omega \to \omega$ is the partial application defined by: $m \cdot n = \varphi_m(n)$. A **partial equivalence relation** R on a set V is a symmetric and transitive relation, not necessarely reflexive, on V.

(**Notation**: n R m iff n relates to m in R; $\{p\}_R = \{q \mid q\ R\ p\}$; $Q(R) = \{\{p\}_R \mid p\ R\ p\}$) .

3.4.1.1 Definition *The category* ***PER*** *of partial equivalence relations on* ω *has as objects the symmetric and transitive relations on* ω. *Morphisms are defined by*

$$f \in \mathbf{PER}[A,B] \quad \text{iff} \quad f : Q(A) \to Q(B) \ \text{ and } \ \exists n \ \forall p \ (pAp \Rightarrow f(\{p\}_A) = \{n\cdot p\}_B) .$$

Thus, the morphisms in **PER** are "computable" because they are fully described by partial recursive functions, which are total on the domain of the source relation.

Let now $<\,,>: \omega\times\omega\to\omega$ be an effective and bijective pairing. For $A,B\in Ob_{\mathbf{PER}}$, the **product** $A\times B$ is defined by

$$\forall m,n,m',n' \quad <m,n> A\times B <m',n'> \ \Leftrightarrow \ (\, m\ A\ m' \ \text{ and } \ n\ B\ n').$$

It is easy to check that this actually defines a product functor in **PER**.

The **exponent object** B^A is defined by

$$\forall m,n, \quad m\ (B^A)\ n \ \Leftrightarrow \ \forall p,q \ (p\ A\ q \ \Rightarrow \ m\cdot p\ B\ n\cdot q) .$$

Cartesian closedness follows by giving a natural isomorphism $\Lambda : \mathbf{PER}[A\times C,B] \cong \mathbf{PER}[A,B^C]$. Let s be the recursive function of the s-m-n (or iteration) theorem, i.e. $\varphi_{s(n,p)}(q) = \varphi_n(p,q)$. Then set $\Lambda(f)(\{p\}_A) = \{s(n,p)\}_{C\to B}$, where n is an index for f. In other words $\{n\cdot<p,q>\}_B = \{s(n,p)\cdot q\}_B$. Observe that Λ is a well-defined function from $\mathbf{PER}[A\times C,B]$ to $\mathbf{PER}[A,B^C]$, since s is computable and, then, any index for the recursive function $p \vdash s(n,p)$, computes $\Lambda(f)\in\mathbf{PER}[A,B^C]$. As for the naturality of Λ, one may prove it by the argument above, which also gives some information on the evaluation map.

Let $\mathbf{eval_{A,B}}$: $B^A\times A\to B$ be defined by $\mathrm{eval}(\{<m,n>\}_{B^A{}_{\times A}}) = \{m\cdot n\}_B$.

In order to prove that $\mathrm{eval}_{A,B}$ is a morphism in the category we must find $e_{A,B}\in\omega$ such that

$$\mathrm{eval}(\{<m,n>\}_{B^A{}_{\times A}}) = \{m\cdot n\}_B = \{e_{A,B}\cdot(<m,n>)\}_B$$

Let u be the "universal" function, i.e., the partial recursive function such that $u(<m,n>) = m\cdot n$, and let e be an index for it, i.e., $u = \varphi_e$. It is easy to observe that one can set $e_{A,B} = e$ for all A, B in $Ob_{\mathbf{PER}}$, since

$$\{e\cdot(<m,n>)\}_B = \{u(<m,n>)\}_B = \{m\cdot n\}_B .$$

Then, (β) is simply

$$\mathrm{eval}(\{<s(n,p),q>\}_{B^A{}_{\times A}}) = \{n\cdot<p,q>\}_B, \ \text{ by definition of } s .$$

Similarly for (η).

3.4.2 Limit and Filter Spaces

There is an elegant, unifying way to understand the various approaches to generalized computability proposed in the 60's and 70's. The connecting point is the construction of categories of sets where a suitable notion of limit gives an abstract notion of computability. The idea is to generalize the technique used when defining the computable elements in Scott domains **D** (see 2.4.1), in the way explained below (in short, the computable elements are the limits of recursively enumerable indexed

sequences.) This suggests several CCC's, such as limit spaces (**L-spaces**) and filter spaces (**FIL**) with their relevant subcategories. They will be introduced here and discussed also in the examples in section 5.3 and section 8.4, toghether with other ideas for higher type computations.

3.4.2.1 Definition *A **limit space** (L-space) $(X,\downarrow)$ is a set X and a relation (convergence) between countable sequences $\{x_i\}_{i\in\omega}\subseteq X$ and elements $x\in X$ (notation: $\{x_i\}\downarrow x$) such that*

1. *if all but finitely many x_i are x, i.e., $\{x_i\}$ is eventually x, then $\{x_i\}\downarrow x$;*
2. *if $\{x_i\}\downarrow x$ and $k(0) < k(1) < \ldots < k(n) < \ldots$, then $\{x_{k(i)}\}\downarrow x$;*
3. *if not$(\{x_i\}\downarrow x)$, then there is $k(0) < k(1) < \ldots < k(n) < \ldots$ such that for no subsequence $h(0) < h(1) < \ldots < h(n) < \ldots$ one has $\{x_{(i)}\}\downarrow x$.*

An L-space $(X,\downarrow)$ has a **countable basis** (is **separable**) iff for some given countable $X_o \subseteq X$, $\forall x\in X\ \exists\{x_i\}\subseteq X_o\ \{x_i\}\downarrow x$.

From now on we assume that each countably based L-spaces $(X,\downarrow)$ comes with a given surjective enumeration $e: \omega\to X_o$ of the base. An immediate example of separable L-spaces is the set of real numbers endowed with the usual notion of sequence converge (Cauchy).

3.4.2.2 Definition *The hom-set $\mathbf{L}[X,Y]$ between L-spaces $(X,\downarrow)$ and $(Y,\downarrow)$ consists of all continuous functions, i.e.,*

$$f\in\mathbf{L}[X,Y] \quad \textit{iff} \quad \forall x\in X\ \forall\{x_i\}\downarrow x\ \{f(x_i)\}\downarrow f(x),$$

where convergence is given in the intended spaces.

This category has exponents and products, as $(\mathbf{L}[X,Y],\downarrow)$ also is an L-space by

$$\{f_i\}\downarrow f \quad \text{iff} \quad \forall x\in X,\ \forall\{x_i\}\downarrow x\ \{f_i(x_i)\}\downarrow f(x)\ ,$$

while products are given by componentwise convergence.

Finally, eval: $\mathbf{L}[X,Y]\times X\to Y$, with $\text{eval}(f,x) = f(x)$, is continuous. As a matter of fact, $(\mathbf{L}[X,Y],\downarrow)$ is the coarsest limit structure (i.e., with more converging sequences) such that eval is continuous.

One can also show that if $(X,\downarrow)$ and $(Y,\downarrow)$ are separable, then also $(\mathbf{L}[X,Y],\downarrow)$ is separable. Indeed, L-spaces and separable L-spaces, with continuous maps as morphisms, form Cartesian closed categories.

Yet another CCC of limit spaces may be given by the Moore-Smith net-convergence or, equivalently, by filter-convergence. Just recall that a **filter** Φ (on a given set X) is a set of (sub)sets closed by intersection and such that $A\in\Phi$ and $A\subseteq B$ imply $B\in\Phi$ (e.g., the collection of subsets of A which contain a given $x\in A$ is the (ultra)filter generated by x). A **filter base** is a non empty collection of non empty subsets of X such that $A\in\Phi$ and $B\in\Phi$ imply $\exists C\in\Phi\ C\subseteq A\cap B$. A filter base Φ generates a unique filter $[\Phi] = \{B\subseteq X \mid \exists A\in\Phi\ A\subseteq B\}$. Define then

3.4.2.3 Definition *(X, F) is a* ***filter space*** *iff* $\forall x \in X$ $F(x)$ *is a filter of filters such that the ultrafilter generated by* x *is in* $F(x)$*. Given a filter base* Φ*, we write* $\Phi \downarrow x$ *iff* $[\Phi] \in F(x)$ *.*

Exercise Prove that the category **FIL** of filter spaces with continuous maps (where f is continuous iff $\Phi \downarrow x$ implies $f(\Phi) \downarrow x$) is a CCC. Give a full and faithful functor $F : \mathbf{FIL} \to \mathbf{L\text{-}spaces}$. (*Hint*: a filter structure on $\mathbf{FIL}[X,Y]$ is given by $\Xi \downarrow f$ iff $\Phi \downarrow x$ implies $\Xi(\Phi) \downarrow f(x)$, where $\Xi(\Phi)$ is the set of all $W(U)$ with $W \in \Xi$ and $U \in \Phi$ and $W(U) = \cup\{f(U) \mid f \in W\}$; moreover, given a filter Φ and a sequence $\{x_i\}$, define $Con(\Phi,\{x_i\})$ iff $\forall U \in \Phi\ \exists k\ \forall n \geq k\ x_n \in U$ and set $\{x_i\} \downarrow x$ iff $\exists \Phi \downarrow x\ Con(\Phi,\{x_i\})$).

A notion of separable filter space is easily given, by taking a countable base of filters (i.e., a countable collection of sets such that each converging filter is generated by elements of the base).

Clearly, each (separable) topological space (X, top) may be turned into a (separable) filter space: just take, for each point x, the collection $F(x)$ of all filters containing the filter of neighborhoods of that point. The reader will have a better insight into the "injections"

$$\mathbf{Top} \to \mathbf{FIL} \to \mathbf{L\text{-}spaces}$$

when looking at examples of adjunctions, in section 5.3. As for now, it may suffice to say that there exist filter spaces whose limit structure is not topological. Some of these filter spaces are among those needed for the study of the total computable functionals.

In the very general setting of L-spaces we can now hint a notion of computability which specializes to the one given over Scott domains, when the intended limit structure is derived from the Scott topology.

Let R be the total recursive functions and $(X, X_0, e, \downarrow)$ a separable L-space, where e is a given enumeration of the base X_0. Define then, in a slightly incomplete way (see the comment below), the collection $X_c \subseteq X$ of **computable elements** by $x \in X_c$ iff $\exists f \in R\ \{e_{f(i)}\} \downarrow x$.
In other words, given a countably based limit structure, an element is computable (or recursive) when it is the limit of a countable sequence indexed over an r.e. set.

Comment L-spaces actually carry too little structure to yield a good definition of "recursive" just by taking arbitrary limits of recursively enumerable converging sequences. Their simplicity and generality, though, should give an immediate intuition of what is going on. Indeed, the technique in the definition of computability tidily borrows from similar methods in mathematics. Consider, say, "smooth manifolds." They are defined on the base of the familiar notion of differentiability in $\mathbb{R}^n$, which is extended by a system of local coordinates to abstract spaces. Similarly here, one takes for granted the recursive functions and extends computability to an abstract setting (and higher types, in

particular) by "local" properties of convergence as in the equation above. As we shall see later, too, the categorical language relates and unifies the various classes of structures where this is done.

L-spaces suggest how to express computability by limits very simply; however, the weakness of these structures is that limits are far away from being uniquely determined and that there is no obvious way to characterize "interesting" sequences and limit points.

The point then is to take only "some" limits. This is done by directed sets in Scott domains (see 2.4.1), which are particular converging sequences with a privileged limit, the least upper bound. It will be described for **FIL** in the examples at the end of section 5.3.

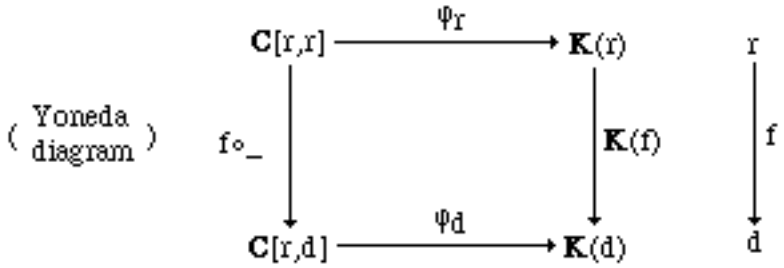
3.5 Yoneda's Lemma

Let **C** be a Cartesian closed category. Let $\mathbf{F} = \mathbf{C}\times\mathbf{C}[\_,(a,b)] \circ \Delta : \mathbf{C}\to\mathbf{Set}$, $\mathbf{G} = \mathbf{C}[\_\times a,b]: \mathbf{C}\to\mathbf{Set}$. In §3.3 we proved that for both **F** and **G** there exists an object, respectively called $a\times b$ and b^a, such that $\mathbf{F} \cong \mathbf{C}[\_,a\times b]$ and $\mathbf{G} \cong \mathbf{C}[\_,b^a]$. In general, functors which enjoy this property are called (co-)**representable**. The formal definition, in case of covariant functors, is the following (note that **F** and **G** are contravariant; we leave as an exercise for the reader to derive the dual definition) :

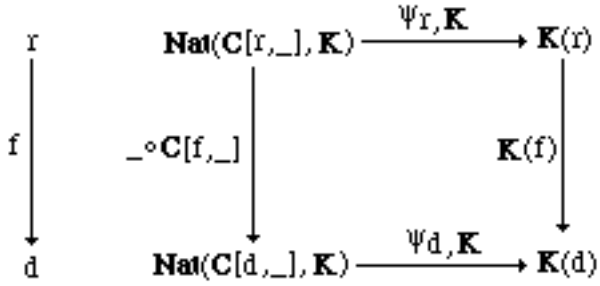
3.5.1 Definition *A functor $K: \mathbf{C}\to\mathbf{Set}$ is representable iff there exist an object r in $\mathbf{C}$ and a natural isomorphism $\phi: K \cong \mathbf{C}[r,\_]$.*

Exercise Give another definition of CC and CCC by using the previous notion.

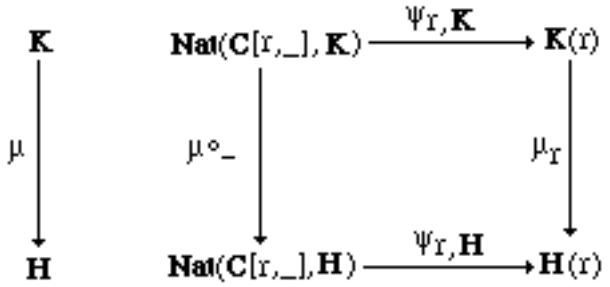
Note that, if $\mathbf{K}: \mathbf{C}\to\mathbf{Set}$, every natural transformation $\varphi: \mathbf{C}[r,\_]\to\mathbf{K}$ is fully determined by the image of the identity id_r. Indeed, for every $f\in\mathbf{C}[r,d]$, $\varphi_d(f) = \varphi_d(f \circ id_r) = \mathbf{K}(f) \circ \varphi_r(id_r)$, by the following diagram:



3.5.2 Lemma (Yoneda) *Let $K: \mathbf{C}\to\mathbf{Set}$ be a functor. The map $\psi_{r,\mathbf{K}}: Nat(\mathbf{C}[r,\_], \mathbf{K})\to\mathbf{K}(r)$ that takes every natural transfomation $\varphi: \mathbf{C}[r,\_]\to\mathbf{K}$ to $\varphi_r(id_r)\in\mathbf{K}(r)$ is an isomorphism. Moreover, it is natural in r and $\mathbf{K}$, that is,*



(where $\mathbf{C}[f,\_] : \mathbf{C}[d,\_] \to \mathbf{C}[r,\_]$ *is the natural transformation defined by* $\mathbf{C}[f,\_]_c = \mathbf{C}[f,c] = \_ \circ f : \mathbf{C}[d,c] \to \mathbf{C}[r,c]$*; see example 3.3.2), and*



Proof By the Yoneda diagram above and a routine verification of the commutativity of the diagrams in the definition. ♦

If we take $\mathbf{K}$ in the previous lemma to be $\mathbf{C}$[d,_], where d is a generic object in $\mathbf{C}$, this results in the statement that there is a natural bijection between arrows $g \in \mathbf{C}[d,r]$ (that is, elements in $\mathbf{C}$[d,_](r)) and natural transformations from $\mathbf{C}$[r,_] to $\mathbf{C}$[d,_].

3.5.3 Proposition (Yoneda embedding)

i. Let Y be the map which takes every $r \in Ob_{\mathbf{C}}$ *to the hom-functor* $\mathbf{C}[r,\_]$*, and every* $g \in \mathbf{C}[d,r]$ *to the natural transformation* $\mathbf{C}[g,\_]: \mathbf{C}[r,\_] \to \mathbf{C}[d,\_]$ *defined by* $\mathbf{C}[g,\_]_c = \mathbf{C}[g,c] = \_ \circ g : \mathbf{C}[r,c] \to \mathbf{C}[d,c]$*. Then Y is a full embedding from* $\mathbf{C}^{op}$ *to* $\mathbf{Funct}(\mathbf{C},\mathbf{Set})$ *.*

ii. Let Y be the map which takes every $r \in Ob_{\mathbf{C}}$ *to the hom-functor* $\mathbf{C}[\_,r]$ *, and every* $g \in \mathbf{C}[d,r]$ *to the natural transformation* $\mathbf{C}[\_,g]: \mathbf{C}[\_,d] \to \mathbf{C}[\_,r]$ *defined by* $\mathbf{C}[\_,g]_c = \mathbf{C}[c,g] = g \circ \_ : \mathbf{C}[c,d] \to \mathbf{C}[c,r]$ *. Then Y is a full and faithful covariant functor from* $\mathbf{C}$ *to* $\mathbf{Funct}(\mathbf{C}^{op},\mathbf{Set})$*.*

Proof We prove only (i), since the other proof is dual.

$Y(id_r) = id_{\mathbf{C}[r,\_]}$ is immediate. Given $g \in \mathbf{C}[d,r]$, $f \in \mathbf{C}[s,r]$, $Y(f \circ g): \mathbf{C}[s,\_] \to \mathbf{C}[d,\_]$ is defined as follows: for every $c \in Ob_{\mathbf{C}}$ $Y(g \circ f)_c = \_ \circ f \circ g : \mathbf{C}[s,c] \to \mathbf{C}[d,c]$. Y is a functor, as, for every $h \in \mathbf{C}[s,c]$,

$$\begin{aligned} Y(g \circ f)_c(h) &= h \circ f \circ g \\ &= Y(g)_c(f \circ h) \\ &= Y(g)_c(\, Y(f)_c(h)\,) \end{aligned}$$

$$= (Y(g)_C \circ Y(f)_C)(h).$$

The fact that Y is full and faithful follows, as already observed, from the Yoneda lemma with **C**[d,_] instead of K. ♦

3.6 Presheaves

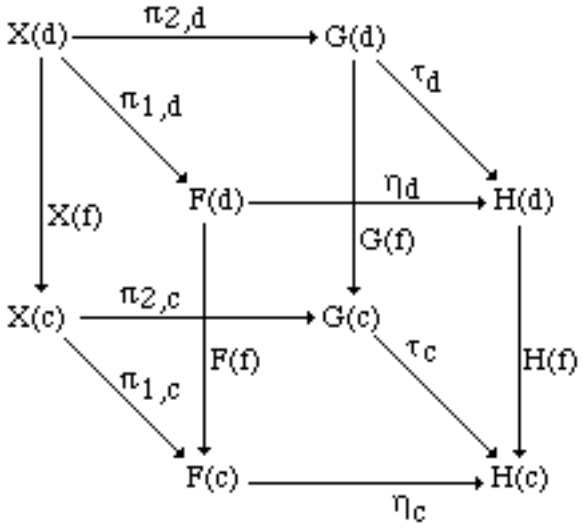
In the last section we proved that every small category **C** is isomorphic to a full subcategory of **Funct**($\mathbf{C}^{op}$,**Set**) through the Yoneda embedding Y. For its relevance, the category **Funct**($\mathbf{C}^{op}$,**Set**) has its own name: a functor F: $\mathbf{C}^{op} \to$ **Set** is called **presheaf** on **C**, and **Funct**($\mathbf{C}^{op}$,**Set**) is the category of **presheaves** on **C**.

The category of presheaves inherits many interesting properties from **Set**, and in particular it is itself a topos. For the moment, we only prove the following properties:

3.5.1 Theorem *Given a category **C**, the category of presheaves on **C** has pullbacks for every pair of morphisms and is Cartesian closed.*

Proof. The terminal object is the functor T : $\mathbf{C}^{op} \to$ **Set**, which takes every object c in $Ob_{\mathbf{C}}$ to the single set {*} and every arrow to the identity on this set.

Given two natural transformations η: F→H and τ: G→H , their pullback is defined objectwise: for every c in $Ob_{\mathbf{C}}$ let ($\pi_{1,c}$: X_c→F(c), $\pi_{2,c}$: X_c→G(c)) be the pullback in **Set** of η_c: F(c)→H(c) and τ_c: G(c)→H(c) . Then define a functor X: $\mathbf{C}^{op} \to$ **Set** where X(c) = X_c , and for every f:c→d, X(f) is the only arrow that makes the following diagram commute:



In the same way we define two natural transformations π_1: X→F, π_2: X→F, where for every c in $Ob_{\mathbf{C}}$, $\pi_1(c) = \pi_{1,c}$ and $\pi_2(c) = \pi_{2,c}$. Then it is easily verified that (π_1: X→F, π_2: X→F) is the pullback of η: F→H and τ: G→H.

As always, the pullback of two functors F and G on the terminal T gives the product $F\times G$. It is easy to verify that $F\times G(c) = F(c)\times G(c)$ and $F\times G(f) = F(f)\times G(f)$.

Exponents are defined by using the Yoneda lemma.

If G^F is the exponent of F and G, we must have an isomorphism $\mathbf{Nat}[H\times F,G] \cong \mathbf{Nat}[H,G^F]$ for every H. In particular, if H is $\mathbf{C}[\_,c]$, and since by the Yoneda lemma, $\mathbf{Nat}[\mathbf{C}[\_,c],K] \cong K(c)$ for every K, we have: $\mathbf{Nat}[\mathbf{C}[\_,c]\times F,G] \cong \mathbf{Nat}[\mathbf{C}[\_,c],G^F] \cong G^F(c)$. Thus, we *define* $G^F(c) = \mathbf{Nat}[\mathbf{C}[\_,c]\times F,G]$.

Given an arrow $f: c\to d$, we must now define $G^F(f): \mathbf{Nat}[\mathbf{C}[\_,d]\times F,G]\to\mathbf{Nat}[\mathbf{C}[\_,c]\times F,G]$. Let σ be a natural transformation from $\mathbf{C}[\_,d]\times F$ to G; then $G^F(f)(\sigma)$ must be a natural transformation from $\mathbf{C}[\_,c]\times F$ to G. We define $G^F(f)(\sigma) = \sigma \circ \mathbf{C}[\_,f]\times id_F$ (see example 3.2.2 for the definition of the natural transformation $\mathbf{C}[\_,f] : \mathbf{C}[\_,c]\to\mathbf{C}[\_,d]$).

G^F is a functor, indeed $G^F(id)(\sigma) = \sigma$. Moreover :

$$\begin{aligned} G^F(f \circ g)(\sigma) &= \sigma \circ \mathbf{C}[\_, f \circ g]\times id_F \\ &= \sigma \circ \mathbf{C}[\_,f]\times id_F \circ \mathbf{C}[\_,g]\times id_F \\ &= G^F(g)(\sigma \circ \mathbf{C}[\_,f]\times id_F) \\ &= G^F(g)(G^F(f)(\sigma)). \end{aligned}$$

Let us define the natural transformation of evaluation $\varepsilon_{F,G}: G^F\times F\to G$. For every d in $Ob_{\mathbf{C}}$, $\varepsilon_{F,G}(d)$: $\mathbf{Nat}[\mathbf{C}[\_,d]\times F,G]\times F(d)\to G(d)$ is defined by $\varepsilon_{F,G}(d)(\sigma,n) = \sigma(d)(id_d,n)$.

We prove now that for any $H: \mathbf{C}^{op}\to\mathbf{Set}$ we have an isomorphism Θ: $\mathbf{Nat}[H\times F,G] \cong \mathbf{Nat}[H,G^F]$. Let $\tau: H\times F\to G$ be a natural transformation. For any c in $Ob_{\mathbf{C}}$, $\Theta(\tau)(c)$ ought to be a function from $H(c)$ to $G^F(c) = \mathbf{Nat}[\mathbf{C}[\_,c]\times F,G]$. By the Yoneda lemma, we have for every c in $Ob_{\mathbf{C}}$ an isomorphism γ_c: $\mathbf{Nat}[\mathbf{C}[\_,c],H] \cong H(c)$; thus, if $m\in H(c)$,

$$\gamma_c^{-1}(m)\in\mathbf{Nat}[\mathbf{C}[\_,c],H]$$
$$\tau \circ (\gamma^{-1}(m)\times id_F) \in\mathbf{Nat}[\mathbf{C}[\_,c]\times F,G].$$

Define then $\Theta(\tau)(c) = \lambda m.\ \tau \circ (\gamma_c^{-1}(m)\times id_F)$: $H(c)\to G^F(c) = \mathbf{Nat}[\mathbf{C}[\_,c]\times F,G]$.

We must prove that, for every $\tau: H\times F\to G$, $\mu: H\to G^F$,

1. $\varepsilon_{F,G} \circ (\Theta(\tau)\times id_F) = \tau$
2. $\Theta(\varepsilon_{F,G} \circ (\sigma\times id_F)) = \sigma$

For (1) we have, for every $d\in Ob_{\mathbf{C}}$, $m\in H(d)$, $n\in F(d)$, the following:

$$\begin{aligned} &(\varepsilon_{F,G}(d) \circ (\Theta(\tau)\times id_F)(d))(m,n) = \\ &\quad = \varepsilon_{F,G}(d)\ (\Theta(\tau)(d)(m),n) \\ &\quad = \varepsilon_{F,G}(d)\ (\ \tau \circ (\gamma_d^{-1}(m)\times id_F),n) && \text{by def. of } \Theta \\ &\quad = (\tau \circ (\gamma_d^{-1}(m)\times id_F))(d)(id_d,n) && \text{by def. of } \varepsilon_{F,G} \\ &\quad = \tau\ (d)(m,n) && \text{as } (\gamma_d^{-1}(m))(d)(id_d) = m \end{aligned}$$

For (2) we have, for every $d\in Ob_{\mathbf{C}}$, $m\in H(d)$, $c\in Ob_{\mathbf{C}}$, $h\in\mathbf{C}[c,d]$, $n\in F(d)$, the following:

$$\begin{aligned} &(\Theta(\varepsilon_{F,G} \circ (\mu\times id_F))(d)(m)(c)(h,n) = \\ &\quad = (\varepsilon_{F,G} \circ (\mu\times id_F) \circ (\gamma_d^{-1}(m)\times id_F))(c)(h,n) && \text{by def. of } \Theta \end{aligned}$$

$= \varepsilon_{F,G}(c)\ (\ (\mu \times id_F)(c)(\ \gamma_d^{-1}(m)(c)(h),\ n\ )\ )$

$= \varepsilon_{F,G}(c)\ (\ (\mu \times id_F)(c)(\ H(h)(m),\ n\ )$ by def of γ_d^{-1}

$= \varepsilon_{F,G}(c)\ (\mu(c)(H(h)(m)),\ n)$

$= \mu(c)(H(h)(m))(c)\ (id_c, n)$ by def. of $\varepsilon_{F,G}(c)$

$= G^F(h)(\mu(d)(m))(c)\ (id_c, n)$ by naturality of μ

$= (\mu(d)(m) \circ \mathbf{C}[\_,h] \times id_F)\ (c)(id_c, n)$ by def. of $G^F(h)$

$= (\mu(d)(m)(c)\ ((\mathbf{C}[c,h] \times id_{F(c)})(id_c, n))$

$= (\mu(d)(m)(c)(h,n).$ ♦

References We only give some references for the examples we mentioned, as they are not usually presented in other books and are mostly endebted to the theory of computing. Partial equivalence relations as a model for higher type computations were given in Kreisel (1959) and later applied to the semantics of higher order intuitionistic logic in Girard (1972) and Troelstra (1973c). They recently came again to the limelight as relevant structures for the semantics of polymorphism in functional languages (see chapter 12). Limit spaces and their closure properties can be found in Kuratowski (1952). With our perspective, filter spaces are used in Hyland (1979).

For (pre)sheaves and related notions, the reader should consult the previous references for Topos Theory (as well as Fourman (1977) and Lambek and Scott (1986), among others). See also Scott (1980) for an application of Yoneda's embedding of arbitrary CCC's (and their reflexive objects, if any) into topoi of presheaves.

Chapter 4

CATEGORIES DERIVED FROM FUNCTORS AND NATURAL TRANSFORMATIONS

This chapter has two main motivations. One derives from the use of algebraic methods in computer science, the other from recent developments in (applied) Proof Theory. The two research directions are brought together nicely by the underlying categorical structures, which tidily generalize two constructions crucial form the perspective of this book namely, products and exponents. Here, they will be discussed in the context of monoidal and monoidal closed categories.

As already mentioned, geometry and algebra provided the background and motivations for the early developments of Category Theory. In these areas, Category Theory often suggested both a unified language and effective tools for an abstract description or specification of mathematical structures. This method, which is typical of the categorical approach, has been widely explored in computer science, in connection with ideas from universal algebra. The point is that, before performing a complicated task, a programmer needs a clear specification of it. This may be given, for example, by a set of equations, or by a logical system, or also by declarations of types (or sorts) and operations on them. Inference rules may specify a theory.

This abstract approach, in computer science as well as in mathematics, is meant to simplify the work for a concrete implementation, since only the essential or desired aspects of a task, a problem, or a mathematical structure are focused on and dealt with. Unstructured lists of goals or properties are hard to understand, are prone to errors, and may hide the core of the issue.

As we shall recall in section 4.1, algebras are usually described as sets with operations. Now (binary) operations need some kind of "product" on the carrier sets to be specified or typed, e.g., a group operation "$\cdot$" is $\cdot : G \times G \to G$. However, "$\times$" does not need to be the familar Cartesian product, as several relevant examples may be given by using (binary) functors that do not need to possess projections. One may then ask whether these functors may relate to suitable exponents, in a way that is similar for CCC's. This further step will take us to the natural (and fruitful) generalization of CCC's as monoidal closed categories and will relate the algebraic perspective to the "functional" one, which permeates this book.

4.1 Algebras Derived from Functors

An algebra is a set, or **carrier**, together with a family of functions, or **operations**, on the carrier. One may define categories of algebras by taking, as morphisms, well-behaved functions, the homomorphisms, between algebras of the same kind.

More precisely, let Ω be a set of operator symbols indexed by their arities. Ω_n, say, is the set operators of arity n.

4.1.1 Definition *The category* $\mathbf{Alg}_{\Omega}$, *of* ***Ω-algebras****, has as objects pairs* (A,α), *where* A *is a carrier set and, for each* n *and each operator symbol* $\rho_n \in \Omega_n$, α *yields a function* $\alpha_\rho : A^n \to A$. *An morphism* f *from* (A,α) *to* (A',α') *is a function* $f: A \to A$ *such that*

$$f(\alpha_\rho(a_1, \ldots, a_n)) = \alpha'_\rho(f(a_1, \ldots, f(a_n))$$

for all n, $\rho_n \in \Omega_n$ *and* $a_1, \ldots, a_n \in A$.

For example, any monoid $(A,\cdot)$ is an Ω-algebra, with a binary operation on a carrier set A. As an instance of this, take Kleene's $(PR,\circ)$, i.e., the monoid of the partial recursive functions, is an Ω-algebra over PR, as carrier, with the binary operation of composition, as operation.

Remark (Commutative) monoids provide the basic instances of linear and multilinear algebra. In particular, there is a tensor product $A \otimes B$ of commutative monoids that can be characterized by a universal bilinear map $\mu: A \times B \to A \otimes B$ such that for each bilinear $f: A \times B \to C$, there is a unique monoid homomorphism $g: A \otimes B \to C$ such that $f = g \circ \mu$. This idea, as well as the generality of monoids, provide the basic mathematical intuition for the categorical notions developed in this chapter.

One may specify more, though. For example, over $(PR,\circ)$ one may require the existence of distinguished functions, such as constants for the identity, the successor and everywhere constant functions. The behavior of these elements is specified by a set of well-known equations.

In general, this technique defines a class of Ω-algebras, called a **variety**. Varieties form full subcategories of the intended category $\mathbf{Alg}_{\Omega}$.

A well-known generalization of Ω-algebras is given by the many sorted, or heterogeneous, case. That is, the operators are specified over more than one carrier. The applications of these notions are nowadays a broadly construed area in computer science. However, until now, they have been more indebted to universal algebra and Abstract Model Theory than to Category Theory (see the References).

Example A **stack** is made out of finite words over a set **el** of elements $A \cup \{\text{error}\}$. Let $\{tt, ff\}$ be the boolean values and A^* the finite words on A. The following signature specifies a stack as a many sorted algebra with sorts el, stack, bool, and operators:

push : stack el → stack;
pop : stack → stack;
top : stack → el;

iserrel : el → bool;
iserrstack : stack → bool;

For variables x: stack and e: el, one has to set the following:

empty = λ
push(x,α) = "if x∈A and α∈A^* then xα else error"
pop(xα) = "if x∈A and α∈A^* then α else error"
top(xα) = "if x∈A and α∈A^* then x else error"
iserrel(error) = tt
iserrstack(error) = tt

A few more equations specify the behavior of "iserr . . . " on elements and the stack, in the obvious way.

Observe now that a set Ω of operators determines a functor T: **Set→Set** defined by

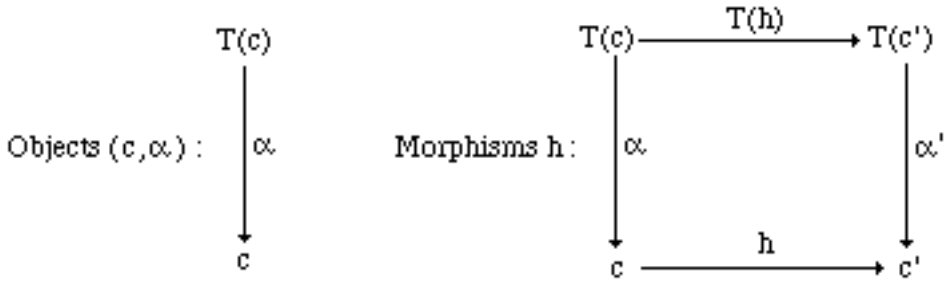
$$T(A) = \{\rho(a_1, \ldots, a_n) \mid n\in\omega,\ \rho\in\Omega_n,\ a_1, \ldots, a_n\in A\}.$$

Then an ω-algebra (A,α) determines a set-theoretic function f : F(A)→A by

$$f(\rho(a_1, \ldots, a_n)) = \alpha_\rho(a_1, \ldots, a_n).$$

This informal remark suggests another generalization of the notion of Ω-algebra. Ω-algebras are based on carriers as sets; we may obtain a more category-theoretic description of the general concept of algebra by taking endofunctors over arbitrary categories, instead of T: **Set→Set** .

4.1.2 Definition *Let* ***C*** *be a category and T:* ***C→C*** *be an endofunctor. The category* ***T-alg*** *of T-algebras is defined as follows: the objects of* ***T-alg*** *are the pairs (c,α) with c∈Ob$_{\mathbf{C}}$ and α∈**C**[T(c),c]; the arrows between two objects (c,α) and (c',α') are all the arrows h∈**C**[c,c'] such that α'∘T(h) = h ∘α . Graphically,*



Identities and composition are both inherited from ***C****.*

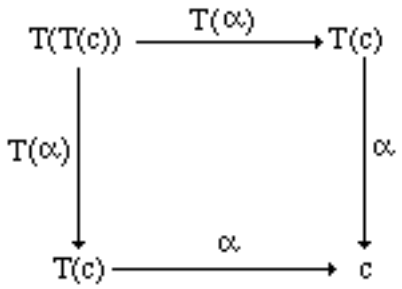
Thus Ω-algebras are T-algebras for T: **Set→Set**, but T-algebras, in general, are defined over categories of structured data, not just sets. One may take, for example, a collection of data types which form a category **C** and include the type **p** of programs, over those data types, as object. Then

T: **C**→**C** and α:T(c)→c, given by T(c) = **p**×c and α(i,x) = "apply program i to input x" give a T-algebra (c,α) , for each object c (see example 1, after definition 4.2.3).

Consider now a preorder (P,≤) as a category. A functor T: P→P is a monotone function. In this setting, a T-algebra is a prefixed point for the functor T as the mere existence of α:T(e)→e corresponds to T(e) ≤ e, which means exactly that e is a prefixed point for T. Recall also that the *least* prefixed point of a monotonic function, if it exists, is always the least fixed point as well. This property has the following correspondent in Category Theory:

4.1.3 Proposition *Let* ***C*** *be a category and T:* ***C***→***C****. If (c,*α*) is an initial T-algebra, then* α *is an isomorphism from T(c) to c in* ***C****.*

Proof Consider the following commutative diagram

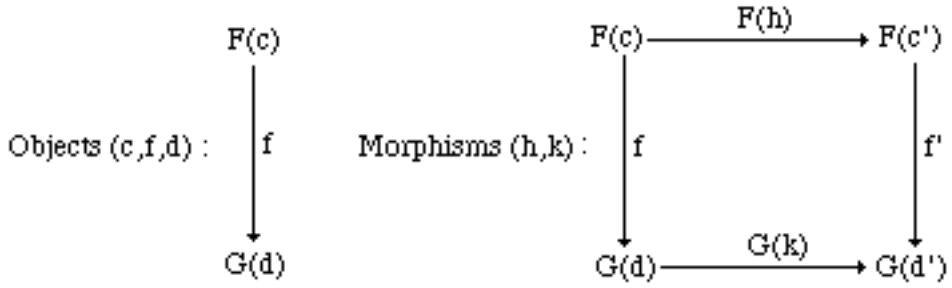


(T(c),T(α)) is a T-algebra, and α∈**T-alg**[(T(c),T(α)),(c,α)]. Let η∈**T-alg**[(c,α),(T(c),T(α))] be the unique morphism given by initiality. Then $\alpha\circ\eta$ and id_c are both morphisms in **T-alg**[(c,α),(c,α)] and by initiality they must be equal. Moreover,

$\eta \circ \alpha$ = T(α) ∘ T(η) as η∈**T-alg**[(c,α), (T(c),T(α))]
= T($\alpha \circ \eta$) since T is a functor
= T(id_c)
= id_c since T is a functor. ♦

We can now go a step further and extend T-algebras to a more general notion, based on functors that share only the target category, instead of endofunctors. This further step may be understood in a very abstract way. Look at the diagrams for definition 4.1.2 and generalize them by using, in the lower line, new objects d, d', say, instead of c, c', and a different functor instead of the (implicit) identity functor. This gives comma categories, which we denote by (F□↓G) instead of the frequently used (F□,G).

4.1.4 Definition *Let F:* ***C***→***A****, G:* ***D***→***A*** *be functors. The* ***comma category*** *(F□↓G) is defined as follows: the objects of (F□↓G) are the triples (c,f,d) with c*∈*Ob*$_{\mathbf{C}}$*, d*∈*Ob*$_{\mathbf{D}}$*, f*∈***A****[F(c),G(d)]; the arrows between two objects (c,f,d) and (c',f',d') are all the pairs (h: c*→*c',k: d*→*d') such that f' ∘F(h) = G(k) ∘f . Graphically,*



The identity of (c,f,d) is (id_C, id_d); composition is defined componentwise, that is, (h',k') ∘ (h,k) = (h' ∘ h, k' ∘ k).

Observe that by this further abstraction, we hit upun a notion examined in another context. Indeed, if a is an object of **C**, and K_a: **C**→**C** is the constant functor taking every object to a, and every morphism to id_a, then (K_a↓$id_{\mathbf{C}}$) is just the category of objects over a, or slice category, mentioned in definition 1.3.4.

4.1.5 Example A (possibly finite) **graph** G is a triple (T,∂,V), where T is a set of arcs (or edges), V a set of nodes, and ∂: T→V×V a function that gives the source and target of each arc. A morphism h from G to G' is a pair of functions <f,g>, f: T→T' and g: V→V' such that

$$g \circ p_0 \circ \partial = p_0 \circ \partial' \circ f \quad \text{and}$$
$$g \circ p_1 \circ \partial = p_1 \circ \partial' \circ f ,$$

where p_0 and p_1 are the projections.

This, with the obvious componentwise composition of morphisms, defines the category **Graph**. **Graph** is a comma category (Id↓Δ) defined by the identity and the diagonal functors Id, Δ: (**Fin**)**Set**→(**Fin**)**Set**.

Exercises

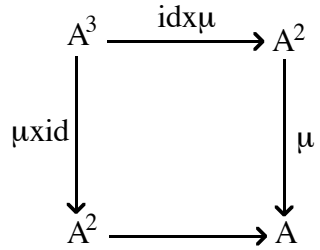
1. Prove that **Graph** is cartesian.
2. Observe that a category is a graph with some extra structure: identities for each object (node) and composition for morphisms (arcs) of the due types. Give a (forgetful) functor from **Cat**, the category of small categories, to **Graph**. Is this functor full?

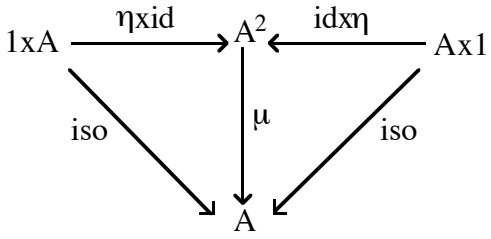
4.2 From monoids to monads

Consider again a monoid (A,·,e), i.e., a carrier A, a binary associative operation and a (specified) left and right identity for "·".

The monoid may be described as the set A together with two functions μ: A×A→A and η:1→A, where 1 is the singleton set (or terminal object in **Set**). The idea is that μ describes the

internal operation "·" and η picks up the identity in A. Associativity of the application and the properties of the identity are given by the following commutative diagrams:

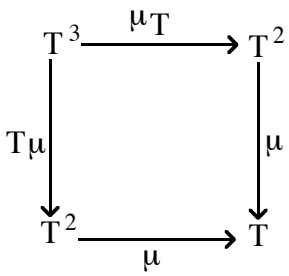


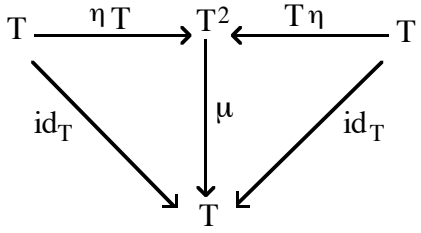


The abstract specification of monoids above can be generalized in several ways. We consider here the case where functors are used as carriers, instead of sets. To this purpose, recall that each endofunctor $T: \mathbf{C} \to \mathbf{C}$ has composition $T^{n+1} = T^n \circ T: \mathbf{C} \to \mathbf{C}$. Moreover, if $\mu: T^2 \to T$ is a natural transformation, whose components are $\mu_c: T^2c \to Tc$ for every $c \in Ob_{\mathbf{C}}$, then $T\mu: T^3 \to T^2$ and $\mu T: T^3 \to T^2$ are the natural transformations obtained by horizontal compositions, as in section 3.2. Their components are $(T\mu)_c = T(\mu_c): T^3c \to T^2c$ and $(\mu T)_c = \mu_{Tc}: T^3c \to T^2c$, respectively.

The idea is that an endofunctor of a category forms a monoid when the product of sets above is interpreted as a composition of functors.

4.2.1 Definition *A **monad** over a category $\mathbf{C}$ is a triple (T, μ, η), where $T: C \to C$ is a functor, $\mu: T^2 \to T$ and $\eta: Id_C \to T$ are natural transformations, and the following diagrams commute:*





As in the case of monoids over a set, η and μ give the identity and the internal operation. The diagrams describe the behaviour of the (left and right) identity and associativity.

Remark Many other names have also been used in literature in place of "monad": the most common is "triple," but sometimes, you will find "triad" or "standard construction." See Barr and Wells (1985) for an interesting account of the history of this name.

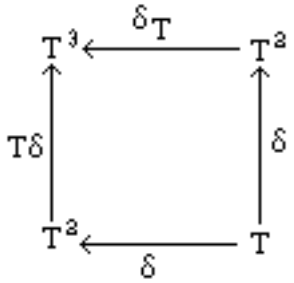
Examples

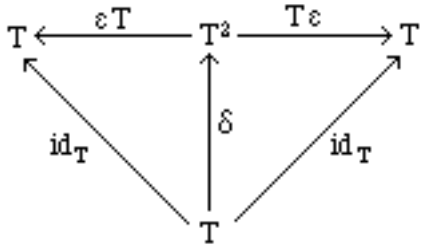
1. Let M be a monoid and define $T: \mathbf{Set} \to \mathbf{Set}$ by $T(A) = M \times A$, $T(f) = id_M \times f$. Let $\eta_A: A \to M \times A$ and $\mu_A: M \times M \times A \to M \times A$ be, respectively, the functions that take a to $(1_M, a)$ and (m,n,a) to (mn,a). Then the associative and unarity identities follows from those of M.

2. Let T be the covariant power-set functor, i.e., T: **Set**→**Set** as given by $T(A) = \mathsf{P}(A)$, the powerset of A, and $T(f)(B) = \{f(a) \mid a \in B \subseteq A\}$ as the image of B along f. Then η_A is the singleton map $\eta_A(a) = \{a\}$ and μ_A is the union map $\mu_A(B) = \cup B$. As for the next example, it is an easy exercise to show that (T, μ, η) is a monad.

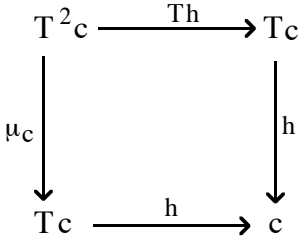
3. As mentioned below definition 4.1.2, a preorder $(P,\leq)$ yields a category where endofunctors are monotone functions. If one has also a monad (T, μ, η), the natural transformations η and μ give that, for any $a \in P$, $a \leq Ta$ and $T(Ta) \leq Ta$, since $\eta_a: a \to Ta$, and $\mu_a: T(Ta) \to Ta$. By putting together these inequalities with the monotonicity of T, one has $Ta \leq T(Ta) \leq Ta$. Therefore, a monad in a partial order is a closure operator, since, in this case, $T(Ta) = Ta$.

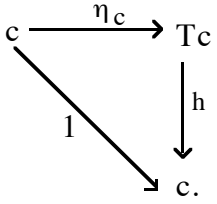
4.2.2 Definition *A **comonad** over a category **C** is a triple (T, δ, ε), where $T: \mathbf{C} \to \mathbf{C}$ is a functor, and $\delta: T \to T^2$ and $\varepsilon: T \to id_{\mathbf{C}}$ are natural transformations, such that the following diagrams commute:*





In definition 4.1.2, we defined the notion of T-algebra over a generic endofunctor $T: \mathbf{C} \to \mathbf{C}$. Thus, one may have T-algebras, if any, over a monad (T, η, μ). In this case, though, it is sound to impose some extra conditions and ask that the monad's operation and identity, given by μ and η, commute with the operation of T-algebra. That is, a T-algebra (c, α) is **given by a monad** (T, η, μ) over a category **C** if it also satisfies the following commutative diagrams:





Or, equivalently, $h \circ T(h) = h \circ \mu_c$;

$h \circ \eta_c = id_c$.

When T is the functor of a monad (T, η, μ), we shall simply say T-algebras with the intended meaning to be derived by the monad. This collection of T-algebras can be organized in a category by adopting the same notion of morphism as in definition 4.1.2.

2.3 Definition *Let (T, η, μ) be a monad over a category $\mathbf{C}$. The* ***Eilenberg-Moore Category*** $\mathbf{C}^T$ *associated with the monad has T-algebras for objects, and for morphisms from (c, α) to (c',α') all the arrows $h \in \mathbf{C}[c,c']$ such that $\alpha' \circ T(h) = h \circ \alpha$.*

Examples

1. Consider the monad (T, η, μ) associated with a monoid M as in example 1 above. Then a T-algebra is a pair (A, α) where $\alpha: T(A) = M \times A \to A$. Let us write $m*a$ in place of $\alpha(m,a)$; then, for every $a \in A$ and $m,n \in M$, the equations of a T-algebra read:

$$1*a = a ;$$
$$(m\ n)*a = m*(n*a) .$$

In other words, an algebra for the monad associated with a monoid M is just what is usually called an **M-set**. Moreover the equation for a morphism h of T-algebras gives, for every $a \in A$ and $m \in M$, $h(ma) = m\ h(a)$, that is the usual notion of homomorphism of M-sets.

2. In connection with example 3 above, observe that if the category **C** is a preorder, then $c \leq T(c)$, by η_c, and $T(c) \leq c$, by α of the T-algebra. Thus, if **C** also happens to be a partial order, any T-algebra given by a monad is a fixed point in the p.o.set.

Let (T, η, μ) be a monad over a category **C**. Then, for every $c \in Ob_{\mathbf{C}}$, $(T(c), \mu_c)$ is a T-algebra. Indeed,

$$\mu_c \circ T(\mu_c) = \mu_c \circ \mu_{T(c)} \quad \text{by the associative law of } (T, \eta, \mu)$$
$$\mu_c \circ \eta_{T(c)} = id_c \quad \text{by the unity law of } (T, \eta, \mu)$$

The algebra $(T(c), \mu_c)$ is called the **free algebra** generated by c (with respect to T).

The computational significance of monads has been stressed recently in suggestions that they may help in understanding programs "as functions from values to computations." The approach we sketch here, and which seems very promising, still belongs to a denotational view in program semantics; however, it suggests an alternative to the conceptual gap between the intensional (operational) and the extensional (denotational) approach to the semantics of programming languages. The idea, roughly, is to give a denotational semantics to computations. The intuition we have been mostly referring to, in the examples and in the presentation so far, is that objects are data types and morphisms are functions or programs in extenso. However, one may need a better display of the intensional aspects of computing and distinguish between values and computations. From this point of view, we can try to look at programs not as transformations from values to values, but as transformations from values (or

programs) *to programs*. This should be done by stressing the effectiveness and the intensional nature of actual computations, without losing the insight and the elegance of the intended extensional approach. That is one should preserve the conceptual unity of the mathematical view (e.g., by categories and related structures), and without getting lost in the taxonomy and details of the operational descriptions.

The idea of a monad (T, η, μ) as a model for computations is that, for each set of values of type A, $T(A)$ is the object of computations of "type A." Then one may understand that $\eta_A: A \to TA$ maps values to computations, and $\mu_A: T^2A \to TA$ flattens a computation of a computation into a computation. If one also requires that η_A is a mono for every $A \in C$, then values form a "subset" of computations (see section 1.5, for subobjects).

Examples

1. The above example of the power-set functor as a monad may give a description of nondeterministic computations by looking at nondeterministic computations as sets of values. η_A picks up a specific computation, as $\eta_A(a)=\{a\}$, and μ_A says that a nondeterministic computation over non-deterministic computations, yields a nondeterministic computation, as $\mu_A(B) = \cup B \in \mathcal{P}(A)$.
2. Even more expressively, computations with side effects may be described by the functor $T(A) = (A \times S)^S$, where S is a set of stores. Intuitively a computation takes a store and returns a value together with the modified store. Then a monad is obtained by setting, for each type (or object) A,

$$\eta_A(a) = \lambda s{:}S.(a, s) \text{ and } \mu_A(f) = \lambda s{:}S.\mathrm{eval}(fs)) .$$

The meaning of η_A should be clear; while for each $f \in (A \times S)^S$, $\mu_A(f)$ is the computation that, given a store s, first computes the pair (computation,store) given by $(f',s') = fs$; then evaluates f' applied to s' and returns a new pair (value,store).

In the application of monads the idea is to go from objects, or types, to the image by a functor of an object. More specifically, in the examples, programs take values in A, say, to computations in TB . Kleisli categories provide the right setting to describe this approach.

4.2.4 Definition *Given a monad (T, μ, η) over C, the* ***Kleisli category*** $\boldsymbol{C_T}$*, is the category whose objects are those of* $\boldsymbol{C}$ *; the set* $\boldsymbol{C_T}[A,B]$ *of morphisms from A to B in* $\boldsymbol{C_T}$ *is* $\boldsymbol{C}[A,TB]$*; the identity in* $\boldsymbol{C_T}[A,A]$ *is* $\eta_A : A \to TA$ *. Moreover, the composition of* $f \in \boldsymbol{C_T}[A,B]$ *and* $g \in \boldsymbol{C_T}[B,C]$ *in* $\boldsymbol{C_T}$ *is* $g \circ f = \mu_C \circ Tg \circ f : A \to TB \to T^2C \to TC$.

Exercises

1. Use the lifting functor $(\_)^\circ$ after proposition 3.2.3 and observe that the Kleisli category is the category of "total maps" over a given category of partial morphisms.
2. Give the notion of co-Kleisli category (see section 5.5).

Eilenberg-Moore and Kleisli categories will be used in the next chapter when discussing adjunctions and monads, and they will be applied in the semantics of linear logic.

4.3 Monoidal and monoidal closed categories

In section 4.2 we began by describing, with a diagram, the familiar notion of monoid. This motivated the definition of monads, when Cartesian products in **Set** are substituted with composition of functors, as in definition 4.2.1.

In both cases, we worked essentially "up to isomorphisms". As for monoids, we identified $(A\times A)\times A$ with $A\times(A\times A)$, in the upper left vertex of the first diagram in section 4.2, and called both A^3. Similarly for T^3, which is short for $T\circ(T\circ T)$ and $(T\circ T)\circ T$, we followed a similar procedure in definition 4.2.1. This is perfectly sound as both Cartesian product and composition are associative. The next step now is the explicit formalization of these (implicit) properties, including the behavior of the terminal object 1, as a left and right unit, since they are needed in all categories where it may be possible to define monoids and derived notions.

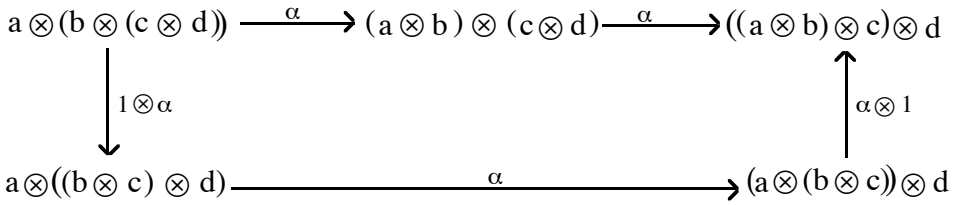
4.3.1 Definition *A **monoidal category** is a category $\mathbf{C}$ with a bifunctor $\otimes : \mathbf{C}x\mathbf{C} \to \mathbf{C}$, a (left and right) identity $e \in Ob_{\mathbf{C}}$ and natural isomorphisms, for each $a, b, c \in Ob_{\mathbf{C}}$,*

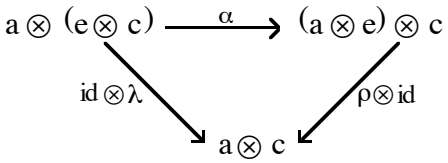
$\alpha_{a,b,c} : a\otimes(b\otimes c)\to(a\otimes b)\otimes c$;

$\lambda_a : e\otimes a\to a$;

$\rho_a : a\otimes e\to a$;

such that the following diagrams commute:





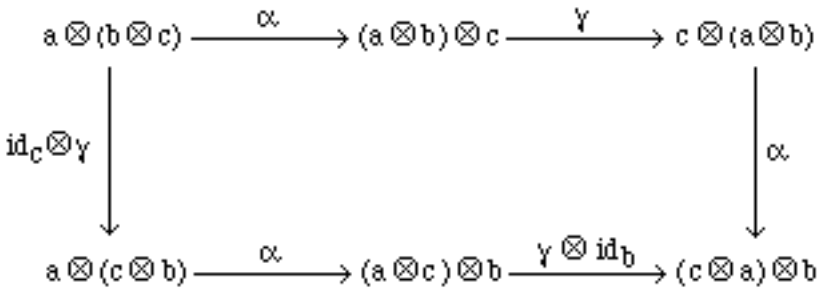
Indexes for the natural transformations α, λ, and ρ will usually be skipped, when they are clear from the context.

Remarks 1 - The "tensor product" ⊗ in a monoidal category does not need to be unique, in contrast with the Cartesian product (see the example below in the category **Stab** and **Lin**). Observe also that the dual of a tensor product is, formally, just (another) tensor product, since isomorphisms define the monoidal structure and the reverse of an isomorphism is still an isomorphism.
2 - The motivation for the two diagrams in definition 4.3.1 originate from a relevant fact, whose treatment goes beyond our limited aims. In short, it may be shown that in momoidal categories, as defined above, any two (natural) isomorphisms built out of α, λ, ρ and id, by using ⊗ and composition, actually coincide ("coherence theorem"). For example, a⊗(b⊗c)⊗(a'⊗b') and (a⊗b)⊗(c⊗a'⊗b') are isomorphic in just one way.

4.3.2 Definition A **symmetric monoidal** *category* $(\mathbf{C}, \otimes, e, \alpha, \lambda, \rho)$ *is a monoidal category such that for all objects* a,b *there is a natural isomorphism* $\gamma_{a,b}: a\otimes b \to b\otimes a$ *and*

$$\gamma_{a,b} \circ \gamma_{b,a} = id_{b,a}$$
$$\rho_b = \lambda_b \circ \gamma_{b,e} : b\otimes e \to b$$

and moreover the following diagram commutes:



The diagram in 4.3.2 is motivated by an extension of the coherence theorem mentioned in the remark above.

Example Every Cartesian category is a symmetric monoidal category, with respect to the categorical product, and the obvious choice for the isomorphisms. Roughly, the tensor product, in the sense of monoidal categories, differs from the Cartesian product in that it has no projections and pairing functions.

Exercise Let **C** be a category with a coproduct for every pair of objects (i.e., **C** is co-Cartesian). Prove that **C** is a symmetric monoidal category.

Example In section 2.4.2, we introduced the category **Stab** of coherent domains and stable functions. Any coherent domain X is obtained from a set |X| of points and a binary relation ↑ ,

coherence, on $|X|$. Stable maps are continuous functions which also preserve intersection of coherent arguments. **Stab** is a CCC.

In the same example, we noticed that one could also consider the linear maps between coherent domains. They are stable ones which also commute with respect to arbitrary unions (see exercise 4 below definition 2.4.2.6). **Lin** is the category of coherent domains and linear functions. Products in **Lin** are defined as for **Stab**, as the projections happen to be linear maps; thus, **Lin** is also Cartesian.

However, there is another relevant functor from **Lin**×**Lin** to **Lin**. Given coherent domains X and Y, let $(|X|\times|Y|,\uparrow^\times)$ be given by $(x,y)\uparrow^\times(x',y')$ iff $x\uparrow x'$ and $y\uparrow y'$. Let $X\otimes Y$ be the coherent domain obtained from $(|X|\times|Y|,\uparrow^\times)$. This is not a product in **Lin**, but the reader may easily check that it turns **Lin** into a symmetric monoidal category.

We will mention again the categories **Lin** and **Stab**, and their interesting interplay, in section 4.4, since they inspired Linear Logic.

Exercise Let $S^\oplus$ be the free commutative monoid generated by a set S. Prove then that, for $S^\oplus\otimes S'^\oplus = (S\times S')^\oplus$, the freely generated commutative monoids form a symmetric monoidal category.

Example Our next example is borrowed from recent investigations of parallelism and concurrency, based on a categorical description of well-known structures for those aspects of computations, namely, Petri nets.

The category of graphs was defined in the example 4.1.5. For readability we write $\partial_0 = p_0 \circ \partial$ and $\partial_1 = p_1 \circ \partial$ and set $(\partial_0,\partial_1: T\to V)$ for the graph $\mathbf{G} = (T, \partial, V)$. Now, every graph $\mathbf{G}$ may be turned into a category. Call $C(\mathbf{G})$ the category whose objects are the nodes V of $\mathbf{G}$, whose arrows are generated from the arcs T of $\mathbf{G}$ by adding the identity arc for each node and closing freely w.r.t. composition "$\circ$". Clearly, C is a functor from the category **Graph** in the example 4.1.5 to **Cat** (this method will be discussed when presenting adjunctions, see example 2 in section 5.3).

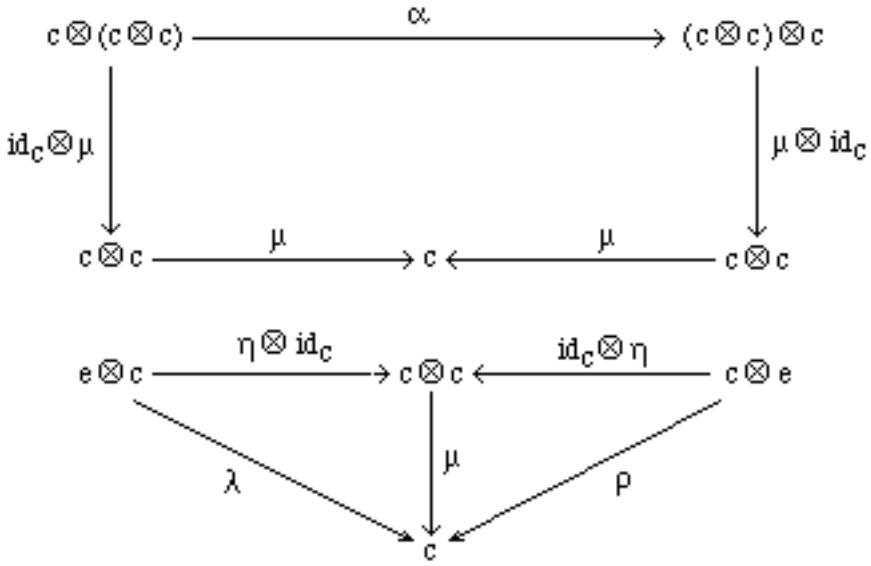
A (transition/place) **Petri net** is a graph $\mathbf{N} = (\partial_0,\partial_1: T\to S^\oplus)$, where the arcs are called transitions and whose nodes are the elements of the free commutative monoid $S^\oplus$, generated by a (possibly finite) set S of labels, the places. Similarly for graphs, we may perform the free construction of a category $C(\mathbf{N})$, on top of a Petri net $\mathbf{N}$. However, $\mathbf{N}$ has an extra structure: namely the monoid structure of $S^\oplus$. Thus we may obtain a monoidal category as follows. Let $C^\otimes(\mathbf{N})$ be the category whose objects are the nodes of $\mathbf{N}$, with $u\otimes v = u\cup v$ in $S^\oplus$, and whose arrows are freely generated from the transitions T of $\mathbf{N}$ with respect to composition "$\circ$" and the monoidal operation $\otimes$ as well. In short, for each pair $f: u\to u'$ and $g: v\to v'$ one also has $f\otimes g: u\otimes v\to u'\otimes v'$ in $C^\otimes(\mathbf{N})$. The functoriality of $\otimes: C^\otimes(\mathbf{N})\to C^\otimes(\mathbf{N})$ is expressed by

$$(1) \qquad (f\otimes g)\circ(f'\otimes g') = (f\circ f')\otimes(g\circ g') \ .$$

The intended meaning of the monoidal operation $\_\otimes\_$ is the parallel composition of arrows, while "$\circ$" is the sequential composition. Thus, net computations are understood as the closure of the transitions with respect to the parallel and the sequential composition. (*Exercise*: discuss the meaning of (1) above).

Monoidal categories provide the right setting for a generalization of the notion of "monoid".

4.3.3 Definition *Let* $(\mathbf{C}, \otimes, e, \alpha, \lambda, \rho)$ *a monoidal category. A* ***monoid*** *in* $\mathbf{C}$ *is an object* c *together with two arrows* $\mu: c\otimes c \to c$ *and* $\eta: e \to c$ *such that the following diagrams commute:*



A ***morphism*** *between two monoids* (c, μ, η) *and* (c', μ', η') *is an arrow* $f: c \to c'$ *in* $\mathbf{C}$ *such that*

$$f \circ \mu = \mu' \circ (f\otimes f) \; : c\otimes c \to c'$$
$$f \circ \eta = \eta' : e \to c'.$$

It is easy to prove that with the previous definition of morphisms, monoids over a monoidal category $\mathbf{C}$ form a category $\mathbf{Mon}_{\mathbf{C}}$.

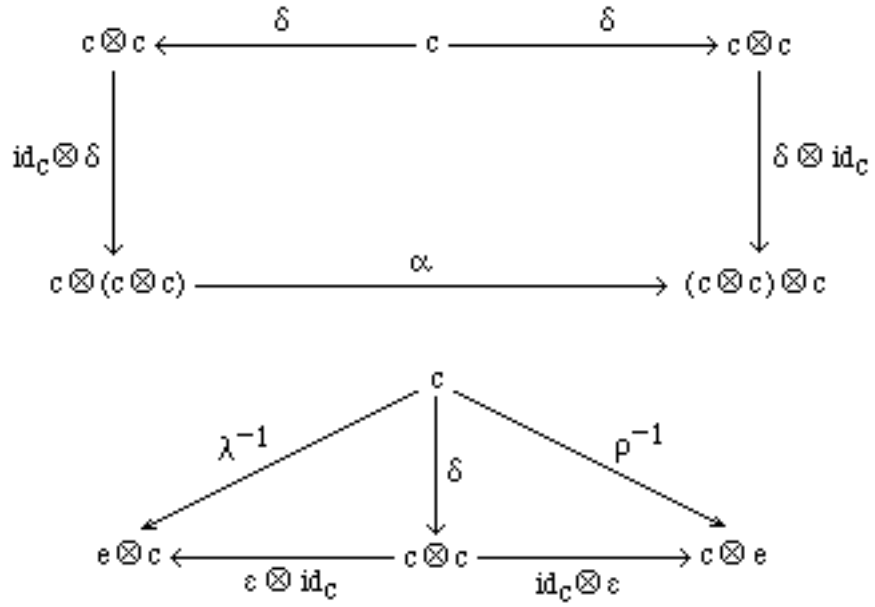
Examples

1. **Set**, with a Cartesian product, is monoidal and the monoids in it are exactly the ordinary monoids.

2. Given an arbitrary category $\mathbf{C}$, consider the category of endofunctors $\mathbf{C}^{\mathbf{C}}$, with natural transformations as morphisms. Then composition "$\circ$" of functors is a bifunctor from $\mathbf{C}^{\mathbf{C}}\times\mathbf{C}^{\mathbf{C}}$ to $\mathbf{C}^{\mathbf{C}}$ which turns $\mathbf{C}^{\mathbf{C}}$ into a monoidal category. The reader may easily check for exercise that the monoids in this category are exactly the monads in definition 4.2.1 and thus understand the elegant conceptual frame provided by the categorical developments of the notion of monoid.

Dually, we have the concept of "comonoid."

4.3.4 Definition *Let* $(\mathbf{C}, \otimes, e, \alpha, \lambda, \rho)$ *be a monoidal category. A* ***comonoid*** *in* $\mathbf{C}$ *is an object* c *together with two arrows* $\delta: c \to c \otimes c$, $\varepsilon: c \to e$ *such that the following diagrams commute:*

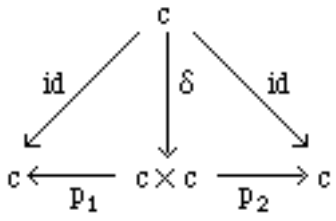


A ***morphism*** *between two comonoids* (c, δ, ε) *and* $(c', \delta', \varepsilon')$ *is an arrow* $f: c \to c'$ *in* $\mathbf{C}$ *such that*

$$f \circ \delta' = (f \otimes f) \circ \delta \; : c \to c' \otimes c'$$
$$\varepsilon' \circ f = \varepsilon : c \to e' .$$

As we have pointed out, monoidal categories are given by an abstract notion of "product," described only in terms of (natural) isomorphisms or, equivalently, in terms of collections of isomorphisms between objects with no further properties (cartesian products also have projections).

Observe that, in the special case that $\mathbf{C}$ is Cartesian, each object is a comonoid: just set $\delta = \langle id,id \rangle$, i.e.,



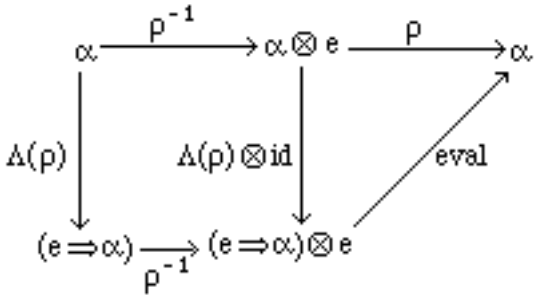
Similarly to the construction of cartesian closed categories from cartesian ones, one may extend the abstract or "equational" approach for monoidal categories and give a notion of monoidal *closed* categories.

4.3.5 Definition *Let* $\mathbf{C}$ *be a symmetric monoidal category, with respect to the bifunctor* $\otimes$*. Then* $\mathbf{C}$ *is* ***monoidal closed*** *if there is also a bifunctor* $\Rightarrow: \mathbf{C}\times\mathbf{C}\to\mathbf{C}$ *such that, for every object b, there exists an isomorphism* $\Lambda : \mathbf{C}[a\otimes b,c] \cong \mathbf{C}[a,b\Rightarrow c]$ *that is natural in a and c.*

Clearly, any CCC is monoidal closed.

3.6 Exercise

1. (Important for the purposes of sections 4.4 and 5.5) **Lin**, with $\otimes$ as in the example below definition 4.3.2, is symmetric monoidal closed (*hint*: see section 2.4.2 and use the coherent domain of traces of linear maps as "$\Rightarrow$"; note that this is not an exponent for the Cartesian product in **Lin**).
2. Consider the tensor product $\otimes$ of monoids given in the remark below definition 4.1.1. There is a natural isomorphism $A\otimes B \cong B\otimes A$, and the monoid homomorphisms between two commutative monoids A and B form a commutative monoid $[A\to B]$. Prove then that there is a natural isomorphism $[(A\otimes B)\to C] \cong [A\to[B\to C]]$, making the category **cMon**, of commutative monoids, a symmetric monoidal closed category.
3. Let e be the identity of a monoidal closed category. Prove then that $e\Rightarrow\alpha \cong \alpha$ for any object α. (*Hint:* use the diagram



in your proof.)

Example We already defined Petri nets and turned each individual net into a (monoidal) category. We could thus view the relation between parallelism and sequentiality as a functorial notion (see the example below definition 4.3.2). We consider now the collection of Petri nets as a category. The category **Petri**, of (transition/place) Petri nets, has Petri nets $(\partial_0,\partial_1: T\to S^{\oplus})$ as objects, where $S^{\oplus}$ is the free commutative monoid generated by S. Morphisms are pairs $\langle f,g\rangle$, $f: T\to T'$ and $g: S^{\oplus}\to S'^{\oplus}$ as for graphs (see the example 4.1.5), with the extra condition that g is monoid homomorphism. (**Exercise**: describe **Petri** as a comma category). As the careful reader should have checked, given two graphs $G = (\partial_0,\partial_1: T\to V)$ and $G' = (\partial'_0, \partial'_1: T'\to V')$, their cartesian product is the graph $G\times G' = (\partial_0\times\partial'_0,\partial_1\times\partial'_1: T\times T'\to V\times V')$. When considering Petri nets $N = (\partial_0,\partial_1: T\to S^{\oplus})$ and $N' = (\partial'_0,\partial'_1: T'\to S'^{\oplus})$, their product as graphs, that is

$$N\times N' = (\partial_0\times\partial'_0,\partial_1\times\partial'_1 : T\times T'\to S^{\oplus}\times S'^{\oplus}),$$

is also a Petri net, since $S^{\oplus}\times S'^{\oplus} \cong (S+S')^{\oplus}$, i.e., the product of two free commutative monoids $S^{\oplus}$ and $S'^{\oplus}$ coincides with the monoid freely generated by $(S+S')$. The Petri net $N\times N'$ is called the *synchronous product* of the nets N and N'. Intuitively, the sychronous product of two Petri nets is the results of a composition operation with synchronization: the places of the result are the union of the places of the factors, while the transition in the synchronous product are pairs (i.e., synchronization) of the given transitions. Since $S^{\oplus}\oplus S'^{\oplus} \cong (S+S')^{\oplus}$ is valid as well, the category **Petri** has also coproducts, namely,

$$N\oplus N' = ([\partial_0,\partial'_0],[\partial_1,\partial'_1] : T+T' \to (S+S')^{\oplus}),$$

where $[\partial_i,\partial'_i]$ denotes the function induced on the coproduct $T+T'$ by the functions ∂_i and ∂'_i. Intuitively, the coproduct of two Petri nets is the result of a composition operation without synchronization: the two nets laid aside without interaction. The choice is nondeterministic because of the freedom of choosing an arbitrary initial state.

The initial net has no transition and no places, while the final net has one transition and no places. As the reader has proved for exercise, the tensor product of free commutative monoids satisfies $S^{\oplus}\otimes S'^{\oplus} = (S\times S')^{\oplus}$. Thus, one may define $\_ \otimes \_ : \mathbf{Petri}^2 \to \mathbf{Petri}$, the tensor product $N\otimes N'$ of two Petri nets N and N', by taking as transitions the Cartesian product of their transitions, as places the cartesian product of their places, and using the equation $S^{\oplus}\otimes S'^{\oplus} = (S\times S')^{\oplus}$ to define nodes from places. The unit object I is the Petri net $(\partial_0,\partial_1: [1]\to[1]^{\oplus})$, with $\partial_0 = \partial_1$ the inclusion of $[1]$ in $[1]^{\oplus}$. In conclusion **Petri** is cartesian, cocartesian and is a monoidal category with a tensor product derived from that operation on monoids.

The next step is to discuss the monoidal closed structure. This is easy when considering finite Petri nets since, whenever $S = \{a_1, ..., a_n\}$ is finite, we have

$$[S^{\oplus} \to S'^{\oplus}] \cong \overbrace{S'^{\oplus} \times \ldots \times S'^{\oplus}}^{n} \cong \overbrace{S'^{\oplus} \oplus \ldots \oplus S'^{\oplus}}^{n} \cong \overbrace{(S' + \ldots + S')}^{n}{}^{\oplus}.$$

Define then the "$\Rightarrow$" functor, on finite Petri nets, as follows. Given Petri nets N and N', $N\Rightarrow N'$ has as arrows the set of triples

$\{(h: T\to T', g: S^{\oplus}\to S'^{\oplus}, g': S^{\oplus}\to S'^{\oplus})$ / g,g' are monoid homomorphisms, $\partial'_0 \circ h = g \circ \partial'_0$ and $\partial'_1 \circ h = g' \circ \partial_1\}$.

For $S = \{a_1, ..., a_n\}$ the monoid of nodes is the free commutative monoid $(S'+ ... +S')^{\oplus}$, n times, and ∂_0 and ∂_1 are the second and third projection. It is then clear that (finite) **Petri** is a symmetric monoidal closed category.

Monoidal closed categories generalize Cartesian closed ones in that they also possess exponent objects $a\Rightarrow b$, or b^a, which "internalize" the hom-sets. One may then ask if there is a way to describe "internally" the behavior of funtors on morphisms. That is, given a monoidal closed category **C** and a functor $F: \mathbf{C}\to\mathbf{C}$, consider, say, $f\in\mathbf{C}[a,b]$. Then $F(f)\in\mathbf{C}[F(a),F(b)]$. Since b^a

and $F(b)^{F(a)}$ represent $\mathbf{C}[a,b]$ and $\mathbf{C}[F(a),F(b)]$ in $\mathbf{C}$, one may study the conditions under which F is "represented" by a morphism in $\mathbf{C}[b^a, F(b)^{F(a)}]$, for each a and b.

4.3.7 Definition *Let $\mathbf{C}$ be a monoidal closed category. A functor $F: \mathbf{C}\to\mathbf{C}$ is **closed** if, for each a and b in $\mathbf{C}$, there exists $f_{ab}\in\mathbf{C}[b^a, F(b)^{F(a)}]$ such that, for all $g\in\mathbf{C}[a,b]$,*

$$f_{ab}\circ\Lambda(g\circ\lambda_a) = \Lambda(F(g)\circ\lambda_{F(a)})$$

where $\Lambda : \mathbf{C}[e\otimes a,b] \cong \mathbf{C}[e,b^a]$ and $\lambda_a: e\otimes a\to a$ are as in 4.3.1.

The notion of a closed functor soundly formalizes our aim above, in view of proposition.4.3.8 below. This essentially says that, internally, f_{ab} takes the identity to the identity and behaves correctly w.r.t. to composition. We will call f_{ab} the **action** of F on b^a.

4.3.8 Proposition *Let $\mathbf{C}$ be monoidal closed category and $F: \mathbf{C}\to\mathbf{C}$ a closed functor. Then the collection $\{f_{ab} \mid a,b\in Ob_{\mathbf{C}}\}$ of its actions is natural in a and b. Moreover, let $comp_{abc}: a^b\otimes b^c\to a^c$ be the natural transformation which internalizes composition, and $ID_a: e\to a^a$ be the natural transformation which "picks up" the identity in a^a, i.e., $ID_a = \Lambda(\lambda_a)$. Then the following equations hold:*

i. $f_{aa}\circ ID_a = ID_{F(a)}$

ii. $\forall g: b\to a,\ \forall h: c\to b,\ f_{ac}\circ comp\circ\Lambda(g\circ\lambda_b)\otimes\Lambda(h\circ\lambda_c) = comp\circ f_{ab}\otimes f_{bc}\circ\Lambda(g\circ\lambda_b)\otimes\Lambda(h\circ\lambda_c)$

Proof Exercise. ♦

4.3.9 Remark By similar techniques for monoidal categories, one may define other classes of categories. In particular, this can be done in order to study very weak frames dealing with a notion of exponent object. Consider for example a category $\mathbf{C}$, not necessarily monoidal, but with an "exponent" functor which realizes the following natural isomorphism, $\mathbf{C}[a,c^b] \cong \mathbf{C}[b,c^a]$ satisfying suitable identities. $\mathbf{C}$ is called **symmetric closed**. Clearly any symmetric monoidal closed category is symmetric closed. This description of categories by (natural) isomorphisms may remind the reader of remark 3.3.3, where we described the set of isomorphisms which hold in all CCC's. Those equations do not characterize CCC's, as cartesian closure requires further structural properties, i.e. projections. They may be used instead to define the larger class of **symmetric monoidal closed categories with a terminal object**. (The reader may try to complete, as an exercise, the definition of the classes of categories in this remark.)

4.4. Monoidal Categories and Linear Logic

The perspective of this book is to introduce and use Category Theory mainly in order to understand "types as objects" of (Cartesian Closed) Categories. Indeed, this will be the focus of the chapters in

Part II. In the literature, so far, the categorical semantics of types has been mostly applied to the functional notion of type, in connection with proof theoretic investigations. The link to functional languages is described by the motto "propositions as types," which proved successful both in the mathematical investigation and in concrete applications as the design of new functional languages with powerful type systems. It seems promising to explore similar analogies for other approaches to computing, where nonfunctional constructs, e.g., parallel features, are considered. These motivations and the properties of relevant categories, namely **Stab** and **Lin**, the categories of stable and linear maps widely discussed in section 2.3 and in the previous section, have suggested a new formal system, **linear logic**.

The crucial difference between the old and new paradigms may be informally summarized as follows. When looking at "propositions as objects" one understands "proofs as functions", while one of the possible perspectives suggested by linear logic is the description of "proofs as actions". An action is roughly an operation which modifies its premises, by a sort of "physical reaction." In short, a step of logical inference modifies the state of the premises: for example, when deducing B from A, some resource in A is consumed. This is in contrast with classical and intuitionistic logic, which deal with static situations: if A implies B and A holds, then we deduce B, but A holds as before, also after the deductive process. Observe that, when ignoring the reaction, one obtains "proofs as functions" again, and thus the new approach may be considered a refinement of the "propositions as objects" analogy.

The proof theoretic description of this difference is based on a rewriting of the structural rules of deduction. We present next the core of linear logic and refer the reader to references for further readings or comparisons with other systems in Proof Theory.

Linear Logic is formally a Gentzen-like logic. The calculus of sequences of Gentzen seems a very suitable formalism for the study of proofs as dynamic actions; indeed the relevance of the *structural rules*, which instead play a quite obscure role in natural deduction, for example, allows a better formalization of the handling of formulas during the inference process, if one wants to focus on its dynamic aspects. The main difference between linear logic and Gentzen calculus of sequences is just in these structural rules. In particular, linear logic drops **weakening** and **contraction** rules, i.e.,

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad \text{and} \qquad \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}$$

Intuitively, formulas can be thought of as resources, and the interpretation of a sequent of the form $A_1, \ldots, A_n \vdash B$ is that of a process (action) which *consumes* the resourses $A_1, \ldots, A_n$ and *produces* B. Thus, resources cannot be freely duplicated or erased; at most, they can be reordered. Moreover, this lack of structural rules suggests a duplication of the binary connectives of conjunction and disjunction. Namely, $\wedge$ and $\vee$ will be described as $\otimes$ and $\cup$ in the "multiplicative" case and

as $\cap$ and $\oplus$ in the "additive" case below. The reason for this is clearly understood, say, in the introduction and elimination rules for $\wedge$: in the presence of contraction and weakening, Gentzen calculus does not distinguish between $(\otimes, r)$ and $(\cap, r)$, nor between $(\otimes, l)$ and $(\cap, l, 1)$ plus $(\cap, l, 2)$. Similarly for $\vee$, which is also described by two connectives, $\cup$ and $\oplus$.

In this section, we define and give categorical meaning to the core of "classical" linear logic by suggesting the basic structural properties of suitable categories where the reader may carry on the details of the interpretation. Other connectives or modalities, essentially the exponential connective "!" (of course), will be discussed in section 5.5.

4.4.1 Alphabet

1. atomic propositions, ranged over by A, B, C, . . .
2. logical symbol:
   2.1. multiplicative symbols:
   constants: 1 (mult. true), $\perp$ (mult. false)
   unary connective: $( )^{\perp}$ (linear negation)
   bynary connectives: $\otimes$ (mult. and), $\cup$ (mult. or), —o (mult. or linear implication).
   2.2. additive symbols:
   constants: T (add. true), 0 (add. false)
   bynary connectives: $\cap$ (add. and), $\oplus$ (add. or).

The well-formed formulae are defined in the obvious way.
Greek capital letters $\Gamma, \Delta, \ldots$ denote finite (possibly empty) sequences of formulas separated by commas. The expression $\Gamma \vdash \Delta$ is called sequent.

4.4.2 Axioms and rules

Each set of (nonstructural) axioms and rules below begins with axioms and rules which deal with the identities relative to the specified connective: 1 for $\otimes$, $\perp$ for $\cup$, T for $\cap$, 0 for $\oplus$. The other rules are introduction rules.

1. structural rules

(id) $A \vdash A$

(exc, r) $\dfrac{\Gamma \vdash \Delta}{\Gamma' \vdash \Delta'}$ where Γ', Δ' are permutations of Γ, Δ.

$$\text{(cut)}\quad \frac{\Gamma_1 \vdash A, \Delta \qquad A, \Gamma_2 \vdash \Delta'}{\Gamma_1, \Gamma_2 \vdash \Delta, \Delta'}$$

2. multiplicative rules

$$\text{(1, r)}\quad \vdash 1 \qquad\qquad \text{(1, l)}\quad \frac{\Gamma \vdash \Delta}{\Gamma, 1 \vdash \Delta}$$

$$\text{(}\bot\text{, l)}\quad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta} \qquad\qquad \text{(}\bot\text{, l)}\quad \bot \vdash$$

$$\text{(}\otimes\text{, r)}\quad \frac{\Gamma_1 \vdash A, \Delta \qquad \Gamma_2 \vdash B, \Delta'}{\Gamma_1, \Gamma_2 \vdash A \otimes B, \Delta, \Delta'} \qquad\qquad \text{(}\otimes\text{, l)}\quad \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}$$

$$\text{(}\cup\text{, r)}\quad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \cup B, \Delta} \qquad\qquad \text{(}\cup\text{, l)}\quad \frac{\Gamma_1, A \vdash \Delta \qquad \Gamma_2, B \vdash \Delta'}{\Gamma_1, \Gamma_2, \ A \cup B \vdash \Delta, \Delta'}$$

$$\text{(—o, r)}\quad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \text{—o} B, \Delta} \qquad\qquad \text{(—o, l)}\quad \frac{\Gamma_1 \vdash A, \Delta \qquad \Gamma_2, B \vdash \Delta'}{\Gamma_1, \Gamma_2, A \text{—o} B \vdash \Delta, \Delta'}$$

3. additive rules

$$\text{(T, r)}\quad \Gamma \vdash T, \Delta \qquad\qquad \text{(0, l)}\quad \Gamma, 0 \vdash \Delta$$

$$\text{(}\cap\text{, r)}\quad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \cap B, \Delta} \qquad\qquad \text{(}\cap\text{, l, 1)}\quad \frac{\Gamma, A \vdash \Delta}{\Gamma, A \cap B \vdash \Delta}$$

$$\text{(}\cap\text{, l, 2)}\quad \frac{\Gamma, B \vdash \Delta}{\Gamma, A \cap B \vdash \Delta}$$

$$(\oplus,r,1)\quad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A\oplus B, \Delta}$$

$$(\oplus,r,2)\quad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A\oplus B, \Delta} \qquad (\oplus, l)\quad \frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A\oplus B \vdash \Delta}$$

3. linear negation

$$(\perp,r)\quad \frac{\Gamma,A \vdash \Delta}{\Gamma \vdash A^{\perp},\Delta} \qquad (\perp,l)\quad \frac{\Gamma \vdash A, \Delta}{\Gamma, A^{\perp} \vdash \Delta}$$

As suggested by the rule $(\otimes, l)$, the comma in the left hand side of a sequent has the same logical meaning as $\otimes$, while the commas in the right hand side correspond to the tensor sum $\cup$, the disjunction which is dual to $\otimes$.

Observe that the rules $(\perp, r)$ and $(\perp, l)$ are equivalent to

$$(\text{contrap})\quad \frac{\Gamma,A \vdash B,\Delta}{\Gamma, B^{\perp} \vdash A^{\perp},\Delta}$$

$(\perp, 1)$ $1 \vdash \perp^{\perp}$ and $(\perp, 2)$ $1^{\perp} \vdash \perp$

(*Hint*: $1,\perp \vdash \perp$ gives both $\perp \vdash 1^{\perp}$ and $1 \vdash \perp^{\perp}$, while $1 \vdash 1,\perp$ gives both $\perp^{\perp} \vdash 1$ and $1^{\perp} \vdash \perp$; the rest is obvious).

From the rules one may easily derive the following sequents:

(μ_{AB}) $A,B \vdash A\otimes B$ by $(\otimes, r)$

(eval_{AB}) $A, A\multimap B \vdash B$ by $(\multimap, l)$

In a few steps, one also obtains

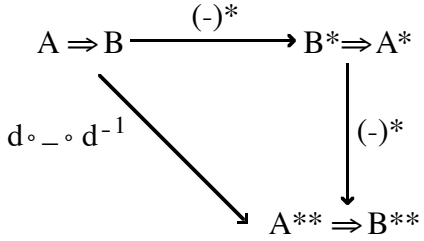
(α_{ABC}) $(A\otimes B)\otimes C \vdash A\otimes(B\otimes C)$ $(\alpha^{-1}{}_{ABC})$ $A\otimes(B\otimes C) \vdash (A\otimes B)\otimes C$.

Exercise Show the logical equivalence of $(A \otimes B) \multimap C$ and $A \multimap (B \multimap C)$. Derive also that $A \vdash (A^{\perp})^{\perp}$ and $(A^{\perp})^{\perp} \vdash A$, i.e. $(\ )^{\perp}$ is "dualizing".

This may be enough to suggest that the categorical meaning of linear logic may be found in particular symmetric monoidal closed categories, where $\otimes$ and $\multimap$ are interpreted by the tensorial product and the bifunctor $\Rightarrow$ given in definition 4.3.5, respectively. Observe that the cartesian product is needed too, so to provide an interpretation for $\cap$ and its identity $\top$ (see later). Negation and all dual constructions are taken care of by a "dualizing" endofunctor $(-)^*$, which is closed in the sense of 4.3.7. Indeed, theorem 4.4.6 below shows that this functor, given $\otimes$ and $\cap$, immediately yields their duals.

4.4.4 Definition *A* ***∗-autonomous*** *category* $\mathbf{K}$ *is a symmetric monoidal closed category and a contravariant closed functor* $(-)^*: \mathbf{K} \to \mathbf{K}$ *such that:*

- *there exists a natural isomorphism* $d : Id \cong (-)^{**}$
- *the following diagram commutes (where* $(-)^*$ *is the action of the functor on exponents)*



A ∗-autonomous category is ***linear*** *if it is also cartesian.*

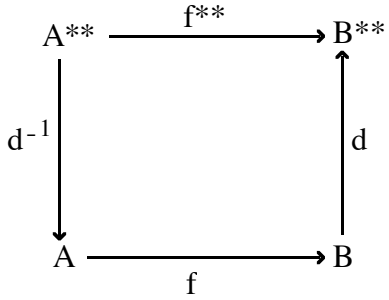
In other words, a linear category has both a monoidal and a Cartesian structure, $\otimes$ and $\cap$, plus a dualizing functor $(-)^*$.

4.4.5 Proposition *Both the vertical and the horizontal arrows in the diagram in definition 4.4.4 are isomorphisms. Moreover,*

i. $A^* \cong B^*$ *iff* $A \cong B$;

ii. $A^* \cong A \Rightarrow 1^*$.

Proof By definition, the following diagram commutes and $d \circ \_ \circ d^{-1}$ is an isomorphism,



Then the horizontal (-)*, in the diagram in definition 4.4.4, is a split mono (see section 1.4) and the vertical one is a split epi. Since the latter is an instance of the former, they are both split monos and split epis as well and, thus, isomorphisms. In conclusion, $A \Rightarrow B \cong B^* \Rightarrow A^*$. Finally, $A^* \cong B^*$ implies $A \cong A^{**} \cong B^{**} \cong B$ and, by exercise 4.3.6.3, $A^* \cong 1 \Rightarrow A^*$. Thus, $A^* \cong B^*$ iff $A \cong B$ and, moreover, $A^* \cong A \Rightarrow 1^*$. ♦

4.4.6 Theorem *Let* ***K*** *be a linear category. Define*

$$A \cup B = (A^* \otimes B^*)^*$$
$$A \oplus B = (A^* \cap B^*)^*.$$

Then $\cup$ *is a dual to the tensor product and* $\oplus$ *is the coproduct in* ***K****. Their identies are* $\perp = 1^*$ *and* $0 = T^*$*, respectively.*

Proof $\cup$ is a well-defined dual to $\otimes$: just compare with definition 4.3.1 (and the remark afterwards) and note that

- $A \cup \perp = (A^* \otimes 1)^* \cong A^{**} \cong A$ (similarly for the identity to the left),
- $(A^* \otimes B^*) \otimes C^* \cong A^* \otimes (B^* \otimes C^*)$ implies $(A \cup B)^* \otimes C^* \cong A^* \otimes (B \cup C)^*$ implies $((A \cup B)^* \otimes C^*)^* \cong (A^* \otimes (B \cup C)^*)^*$ implies $(A \cup B) \cup C \cong A \cup (B \cup C)$.

As for the Cartesian coproduct, note that, if p_{A^*} and p_{B^*} are the projections for $A^* \cap B^*$, then their images $(p_{A^*})^*$ and $(p_{B^*})^*$ via the (-)* functor are the injections for $A \oplus B$. Finally, $A \cup \perp = (A^* \otimes 1)^* \cong A$ and $A \oplus 0 = (A^* \cap T)^* \cong A$. ♦

The results in proposition 4.4.5 and theorem 4.4.6 prove the "dualizing" role of the (-)* functor and of $\perp = 1^*$. In particular, theorem 4.4.6 is a version of De Morgan rules in the semantic structures for linear logic, when described in categorical terms. Note also that

$$\begin{aligned} A^* \cup B &\cong (A \otimes B^*)^* \\ &\cong (A \otimes B^*) \Rightarrow \perp \\ &\cong A \Rightarrow (B^* \Rightarrow \perp) \\ &\cong A \Rightarrow (1 \Rightarrow B) \\ &\cong A \Rightarrow B \end{aligned}$$

which gives the "classic" flavor of this fragment of Linear Logic.

The meaning of linear logic as a deductive system is then given by interpreting each entailment $A \vdash B$ as a morphism between the interpretation of the formulas. Linear categories yield such an interpretation, which we sketch here, by induction on rules. Here are some of the basic cases. Some crucial ones are simply the properties corresponding to (μ_{AB}), (eval_{AB}), (α_{ABC}) and (α^{-1}_{ABC}), given before definition 4.4.4, in monoidal closed categories. As for the others,

the rules	*are interpreted by*	*the fact that*

for 1 and ⊥	1 and ⊥ are (the unique) identies for ⊗ and ∪, which are expressed by commas, on the left or right, respectively, of ⊢.
(⊗,r) and (∪,l)	⊗ and ∪ are bifunctors
(∩,r) and (⊕,l)	∩ and ⊕ have pairing and sum of morphisms
(∩,l,1) and (∩,l,2)	there exist projections for ∩
(⊕,r,1) and (⊕,r,2)	there exist injections for ⊕
(contrap)	(-)* is so defined in linear categories
(⊥,⊥)	⊥ = 1* and, thus, ⊥* ≅ 1 .

(By the argument before 4.4.4, (contrap) and (⊥,⊥) are equivalent to (⊥,r) and (⊥,l) .)

Example The category **Lin** of coherent domains and linear maps, in 2.4.2, not only provides an example of linear category, but it has even been the source of inspiration for linear logic, since linear functions depend on inputs "additively" as deductions from assumptions in this logic. (There is more than that analogy, though, as this example and section 5.5 should clarify.)

We noticed that **Lin** is Cartesian in 2.3.7(II). **Lin** is also co-Cartesian. Indeed, let $(|X|,\uparrow)$ and $(|Y|,\uparrow)$ be coherent structures. Define then the **coproduct** $X \oplus Y$ as the coherent domain associated with the coherent structure $(\{(0,z) \mid z \in |X|\} \cup \{(1,z) \mid z \in |Y|\}, \uparrow^{\oplus})$, where $(a,x) \uparrow^{\oplus} (a',y)$ iff $a = a'$ and $x \uparrow y$. (Note the difference with the product: the support is the same, but the coherence relation changes. Give the embedding linear maps for exercise.)

The singleton coherent domain $T = 0 = \{\varnothing\}$, associated with the empty coherent structure, is both terminal and initial in the category. Indeed, it is the identity for both the product and the coproduct.

In the previous section we turned **Lin** into a symmetric monoidal closed category (see exercise 4.3.6). Recall, in particular, that the **tensor product** $A \otimes B$ is the coherent domain associated with the coherent structure $(|A| \times |B|, \uparrow^{\otimes})$, where $(x,y) \uparrow^{\otimes} (x',y')$ iff $x \uparrow x'$ and $y \uparrow y'$.

The dual of the tensor product, is ∪ given by the **tensor sum** $A \cup B$, that is, the coherent domain associated with the coherent structure $(|A| \times |B|, \uparrow^{\cup})$, where $(x,y) \uparrow^{\cup} (x',y')$ iff ($(x,y) = (x',y')$ or $x \uparrow \uparrow x'$ or $y \uparrow \uparrow y'$). Also in this case, the tensor product and sum have the same support, but the coherence relation changes.

Notice that the identity for both the tensor product and its dual is the coherent domain $\underline{1} = \bot = \{\varnothing,\{1\}\}$, associated with $(\{1\}, =)$.

As for the contravariant functor (-)* , given a coherent structure $(|A|, \uparrow)$, define A^* as the coherent domain associated with $(|A|, \uparrow^*)$, where $x \uparrow^* y$ iff $\sim(x \uparrow y)$ in A . On a linear function $f: A \to B$, $f^*: B^* \to A^*$ is defined in the following way: $(y,x) \in Tr(f^*)$ iff $(x,y) \in Tr(f)$.

As one could expect (and easily compute), in coherent domains one has $T^* = 0 = T$ and $\underline{1}^* = \bot = \underline{1}$. Moreover, the equations in theorem 4.4.6 are realized in **Lin**.

Memo For the reader's convenience, we summarize the identities in linear logic (and linear categories):

$$1 \text{ for } \otimes\,;\ \bot \text{ for } \cup\,;\ T \text{ for } \cap\,;\ 0 \text{ for } \oplus$$

which are interpreted in **Lin** as $\underline{1} = \bot = \{\varnothing, \{1\}\}$ and $T = 0 = \{\varnothing\}$, with the obvious coherent structure.

References Textbooks, which stress the applications of Category Theory to computer science with an algebraic perspective, are Arbib and Manes (1975) and Rydeheard and Burstall (1988). Goguen and Burstall (1984) contains a survey and references to part of this area, broadly construed, which ranges from the work in Elgot (1971), to the contributions of the ADJ group (see, at the end of this volume, the many references which include the names of Goguen, Thatcher, Wagner, or Wright).

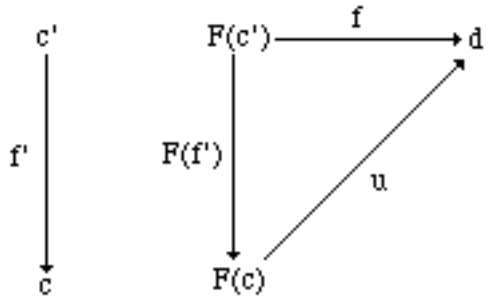
Besides the references to general Category Theory, the reader may consult Barr (1979) and Barr and Wells (1985) for monoidal categories, monads and their mathematical applications. Further references, which also discuss Linear Logic, are Barr (1990) and Marti-Oliet and Meseguer (1990). The recent applications of "monoidal notions" we described are borrowed from Moggi (1989), as for the understanding of programs as "functions from values to computations", and from Meseguer and Montanari (1988) or Degano and al. (1989), as for the examples on Petri nets (see also Marti-Oliet and Meseguer (1989)).

The main reference for linear logic is Girard (1987). Further work may be found, among others, in Girard and Lafont (1987), Lafont (1988), DePavia (1987), and Seely (1987), where $*$-autonomous categories are proposed for its semantics. Lambek (1968) contains early work on the categorical significance of logical systems and discusses weakenings of the structural rules.

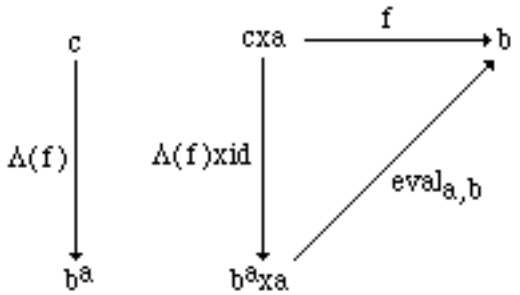
Chapter 5

UNIVERSAL ARROWS AND ADJUNCTIONS

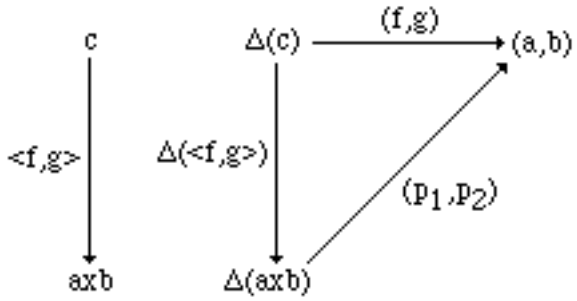
In chapter 3, we introduced the notion of functor as a uniform way to describe "acceptable" constructions in Category Theory. The reader may have noticed that in some cases, for a given construction $F: \mathbf{C} \to \mathbf{D}$, there exists a particular object c in $\mathbf{C}$ and an arrow $u: F(c) \to d$, which has a "universal behaviour" with respect to d, in the sense that every other arrow $f: F(c') \to d$ uniquely factorizes through u, i.e., there exists a unique $f': c \to c'$ such that $f = u \circ F(f')$ or the following diagram commutes:



Consider for example the product functor $\_ \times a: \mathbf{C} \to \mathbf{C}$ where $\mathbf{C}$ is a CCC and $a \in Ob_{\mathbf{C}}$. The arrow $eval: b^a \times a \to b$ is universal with respect to b, in the previous sense, since for every arrow $f: c \times a \to b$ there exists a unique arrow $\Lambda(f): c \to b^a$ such that $f = eval \circ \Lambda(f)$, i.e.,



Also the product of two objects may be described in terms of universal arrows. Just take the diagonal functor $\Delta: \mathbf{C} \to \mathbf{C}\mathbf{x}\mathbf{C}$, with $\Delta(c) = (c,c)$ and $\Delta(f) = (f,f)$ and observe that the following diagram commutes:



The aim of this chapter is to study the concept of universal arrow outlined above, and some of its implications. In particular we prove that given a functor $F: \mathbf{C}\to\mathbf{D}$, if for every $d\in\mathbf{D}$ there exists an universal arrow $u_d: F(c_d)\to d$, then the function $g: Ob_{\mathbf{D}}\to Ob_{\mathbf{C}}$ that takes d to c_d may be extended to a functor $G: \mathbf{D}\to\mathbf{C}$. Such a functor G is called (right) **adjoint** to F. Remarkably, if G is a (right) adjoint to F, then F is a (left) adjoint to G, in a dual sense, and each one describes the other up to isomorphism. From another point of view, F and G can be seen as a pair of "mutually representable" functors.

The notion of adjointness is probably the most profound and original contribution of Category Theory to mathematics. The reader must not expect to understand its relevance upon first reading the definition, or the few examples and applications in this chapter: only by a systematic use of adjunctions will she or he become competent on the subject.

5.1 Universal arrows

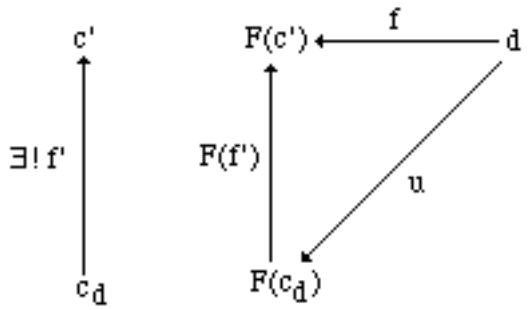
In addition to the previous remarks, the careful reader has surely noticed that most of the definitions of the constructions defined in chapter 2 (products, coproducts, terminal object, exponents . . .) have the following common pattern:

for all . . . there exist a unique . . . such that . . . .

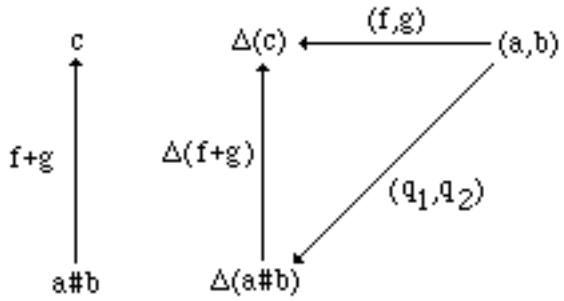
As a matter of fact, all those definitions can be seen as particular cases of a same notion (or its dual): the **universal arrow**. For historical reasons, universal arrows are defined dually with respect to the examples just mentioned (exponents, products . . .). Couniversal maps, to be defined next, will fit the examples.

5.1.1 Definition. *Let* $F: \mathbf{C}\to\mathbf{D}$ *be a functor and* $d\in Ob_{\mathbf{D}}$. *Then* $\langle u,c_d\rangle$ *is* ***universal*** *from* d *to* F *iff* $u\in\mathbf{D}[d,F(c_d)]$, $c_d\in Ob_{\mathbf{C}}$, *and* $\forall c'\in Ob_{\mathbf{C}}$, $\forall f\in\mathbf{D}[d,F(c')]$, $\exists!\, f'\in\mathbf{C}[c_d,c']$ $f = F(f')\circ u$.

As usual, this may be equivalently visualized by



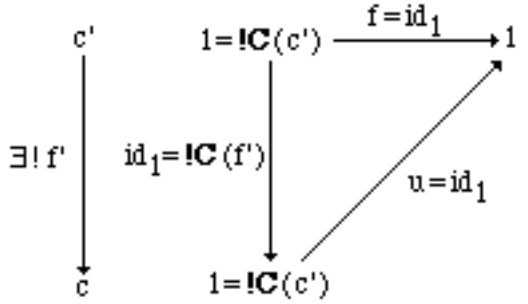
Example $<(q_1,q_2),(a\#b,a\#b)>$ is universal from (a,b) to Δ, where Δ: $\mathbf{C}\to\mathbf{C}\times\mathbf{C}$ is the diagonal functor.



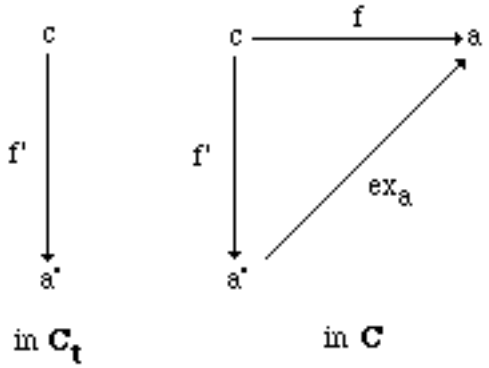
5.1.2 Definition. *Let* $F: \mathbf{C}\to\mathbf{D}$ *be a functor and* $d\in Ob_{\mathbf{D}}$. $<u,c_d>$ *is* ***(co)universal*** *from* F *to* d *iff* $u\in\mathbf{D}[F(c_d),d]$, $c_d\in Ob_{\mathbf{C}}$, *and* $\forall c'\in Ob_{\mathbf{C}}$, $\forall f\in\mathbf{D}[F(c'),d]$ $\exists!\, f'\in\mathbf{C}[c',c_d]$ $f = u\circ F(f')$.

The diagrams and examples in the introduction to this chapter refer this "dual" notion. Here are other interesting examples:

5.1.3 Example Let **!C** be the unique functor from the category **C** to the category **1**; if $id_1 = u \in \mathbf{1}[\mathbf{!C}(c),1]$ is universal, then c is terminal in **C**:



5.1.4 Example Let **C** be a category of partial morphisms, $\mathbf{C_t}$ be the associated category of total morphisms, and **Inc**: $\mathbf{C_t}\to\mathbf{C}$ be the embedding functor (see section 2.6). Recall that, by definition, the lifting of $a\in Ob_{\mathbf{C}}$ is an object $a^\circ\in Ob_{\mathbf{C}}$ together with a morphism $ex_a\in\mathbf{C_t}[a^\circ,a]$ such that for every $f\in\mathbf{C}[c,a]$, there exists one and only one $f'\in\mathbf{C_t}[c,a^\circ]$ satisfying the equation $ex_a\circ f' = f$. This says exactly that $ex_a: a^\circ\to a$ is an universal arrow from **Inc** to a :



Exercise Define an initial object as a universal arrow from 1 to $!\mathbf{C}$.

5.1.5 Proposition. *Let* $F: \mathbf{C} \to \mathbf{D}$ *be a functor and* $d \in Ob_{\mathbf{D}}$ *. Assume that* $\langle u,c \rangle$ *is universal from* d *to* F *. Then*

1. $\langle u',c' \rangle$ *is universal from* d *to* $F \Rightarrow c \cong c'$;
2. $c \cong c'$ *via* $(h,h^{-1}) \Rightarrow \langle F(h) \circ u,c' \rangle$ *is universal from* d *to* F.

Proof

1. $\exists!h \in \mathbf{C}[c,c']$ $\exists!h' \in \mathbf{C}[c',c]$ $u' = F(h) \circ u = F(h) \circ F(h') \circ u' = F(h \circ h') \circ u'$. But $u' = F(id) \circ u'$.

By unicity, $h \circ h' = id$. Similarly, $h' \circ h = id$.

2. Exercise. ♦

By the proof of proposition 5.1.5, one even has that the isomorphism is unique. This is a strong property of universal constructions as a very common tool in mathematics (see the examples below). It is hard to think of a more suitable language than the categorical one for expressing properties like this.

We next give two alternative characterizations of the notion of universal arrow: the first one, in theorem 5.1.6, is of an equational nature; the second one, in theorem 5.1.7, makes use only of the notion of a natural isomorphism. In a sense, the definition of universal arrow given in definition 5.1.1 is in the middle way: it is based on one equation and on the existence of an isomorphism (does the reader see any analogy with the various characterizations of products and exponents we have given?).

5.1.6 Theorem *Let* $G: \mathbf{C} \to \mathbf{D}$ *be a functor,* $d \in Ob_{\mathbf{D}}$ *and* $c_d \in Ob_{\mathbf{C}}$. *Then there exists* $u \in \mathbf{D}[d,G(c_d)]$ *such that* $\langle u,c_d \rangle$ *is universal from* d *to* G *iff, for every* $c \in Ob_{\mathbf{C}}$ *there is an operation* $\tau_c: \mathbf{D}[d,G(c)] \to \mathbf{C}[c_d,c]$ *such that, for every* $f \in \mathbf{D}[d,G(c)]$ *and every* $h \in \mathbf{C}[c_d,c]$ *,*

1. $G(\tau_c(f)) \circ u = f$
2. $\tau_c(G(h) \circ u) = h$

Proof

($\Leftarrow$) let $\langle u, c_d \rangle$ be universal from d to G. For every $f \in \mathbf{D}[d, G(c)]$ define $\tau_c(f)$ as the unique arrow f' such that $G(f') \circ u = f$. (1) is then immediate by definition, and (2) follows by unicity.

($\Rightarrow$) let $\tau_c : \mathbf{D}[d, G(c)] \to \mathbf{C}[c_d, c]$ be an operation which satisfies (1) and (2) above. We must only prove that it is an isomorphism. Define then, for every $h \in \mathbf{C}[c_d, c]$ $\tau_c^{-1}(h) = G(h) \circ u$; thus, we have:

$\tau_c^{-1}(\tau_c(f)) = G(\tau_c(f)) \circ u = f$ by (1)

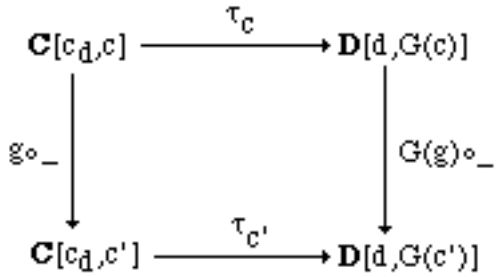
$\tau_c(\tau_c^{-1}(h)) = \tau_c(G(h) \circ u) = h$ by (2). ♦

5.1.7 Theorem. *Let $\mathbf{C}$, $\mathbf{D}$ be locally small categories, $G: \mathbf{C} \to \mathbf{D}$, $d \in Ob_{\mathbf{D}}$ and $c_d \in Ob_{\mathbf{C}}$. Then there exists $u \in \mathbf{D}[d, G(c_d)]$ such that $\langle u, c_d \rangle$ is universal from d to G iff $\mathbf{C}[c_d, \_] \cong \mathbf{D}[d, G\_]$.*

Proof

($\Leftarrow$) For $c' \in Ob_{\mathbf{C}}$ and $f' \in \mathbf{C}[c_d, c']$, set $\tau_{c'}(f') = G(f') \circ u \in \mathbf{D}[d, Fc']$.

Then $\tau : \mathbf{C}[c_d, \_] \to \mathbf{D}[d, G\_]$ is a natural transformation, since $\tau_{c'}(g \circ h) = G(g \circ h) \circ u = G(g) \circ \tau_c(h)$. That is, for $g \in \mathbf{C}[c, c']$, the following diagram commutes:



Moreover, $\forall f \in \mathbf{D}[d, Fc'] \ \exists! f' \in \mathbf{C}[c_d, c'] \ f = \tau_{c'}(f')$, by definition. Thus, by proposition 3.2.3, τ is a natural isomorphism.

($\Rightarrow$) Let $\tau : \mathbf{C}[c_d, \_] \cong \mathbf{D}[d, G\_]$ and set $u = \tau_{c_d}(id)$. Then $u \in \mathbf{D}[d, G(c_d)]$. By the naturality of τ, for all $c, c' \in Ob_{\mathbf{C}}$, $G(g) \circ \tau_c(\_) = \tau_{c'}(g \circ \_)$ and, hence, $\forall f \in \mathbf{D}[d, G(c')]$

$$\begin{aligned} G(\tau_{c'}^{-1}(f)) \circ u &= G(\tau_{c'}^{-1}(f)) \circ \tau_{c_d}(id) \\ &= \tau_{c'}(\tau_{c'}^{-1}(f)) \\ &= f \ . \end{aligned}$$

That is, $\forall f \in \mathbf{D}[d, G(c')] \ \exists! f' (= \tau_{c'}^{-1}(f)) \in \mathbf{C}[c_d, c']$ such that $f = G(f') \circ u$. ♦

Exercise: Give the dual version of theorems 5.1.6 and 5.1.7.

Recall now that a functor $F : \mathbf{C} \to \mathbf{Set}$ is representable if there exists a $c \in \mathbf{C}$ such that $F \cong \mathbf{C}[c, \_]$ naturally.

5.1.8 Corollary *Let* $\mathbf{C}, \mathbf{D}$ *be small categories,* $G: \mathbf{C}\to\mathbf{D}$, $d\in Ob_{\mathbf{D}}$ *and* $c_d\in Ob_{\mathbf{C}}$. *Then there exists* $u\in\mathbf{D}[d,G(c_d)]$ *such that* $\langle u,c_d\rangle$ *is universal from* d *to* G *iff the functor* $\mathbf{D}[d,G\_]: \mathbf{C}\to\mathbf{Set}$ *is representable.*

5.2 From Universal Arrows toward Adjunctions

The construction of a universal arrow $u_d: G(c_d)\to d$ from $G: \mathbf{C}\to\mathbf{D}$ to d usually depends on d. If this construction can always be performed, the function $d \vdash c_d$ can be extended to a functor $F: \mathbf{D}\to\mathbf{C}$. We shall see in the next section that such G and F relate in an important way called adjunction; for the moment we concentrate on the construction of the functor F.

In this and in the following section, we assume that we are dealing with locally small categories.

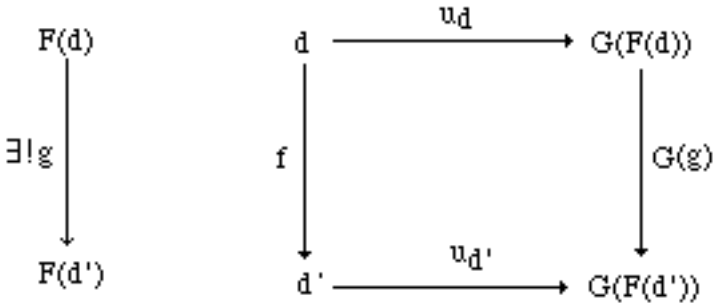
5.2.1 Theorem. *Let* $G: \mathbf{C}\to\mathbf{D}$ *be a functor such that* $\forall d\in Ob_{\mathbf{D}}\ \exists\langle u_d,c_d\rangle$ *universal from* d *to* G *Then there exists a functor* $F: \mathbf{D}\to\mathbf{C}$ *such that*

i. $F(d) = c_d$

ii. $\mathbf{C}[F\_,\_] \cong \mathbf{D}[\_,G\_]$.

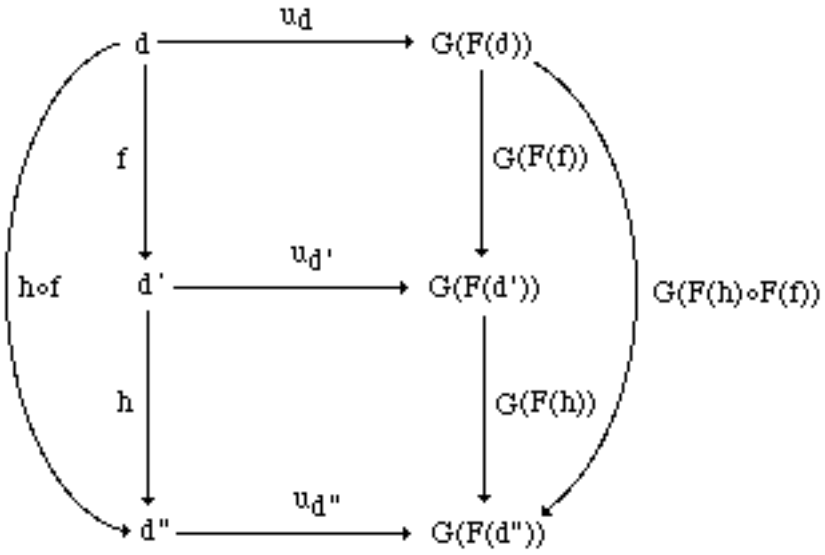
(Note that $\mathbf{C}[F\_,\_]$, $\mathbf{D}[\_,G\_] : \mathbf{D}^{op}\times\mathbf{C}\to\mathbf{Set}$*). Moreover, the functor* F *is unique, up to isomorphism.*

Proof: By assumption we know that, for all $f\in\mathbf{D}[d,d']$,

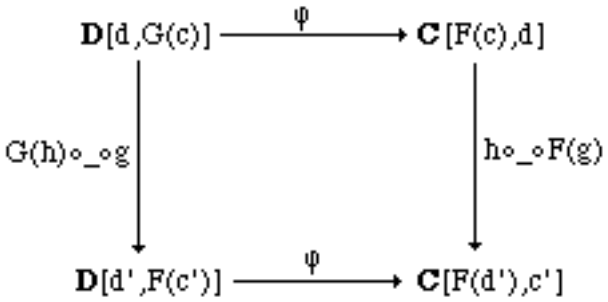


Set then $F(f) = g$, that is, $u_{d'}\circ f = G(Ff)\circ u_d$. By the uniqueness property, $G(id) = id$.

Moreover, by twice the definition of F,



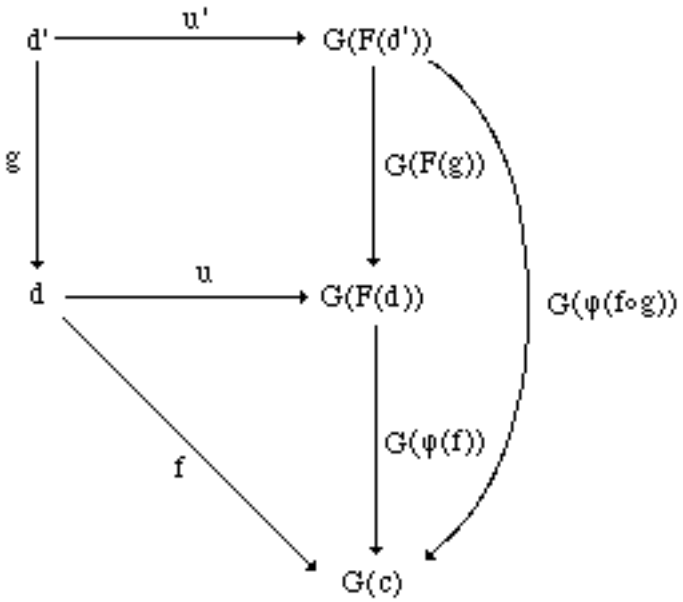
And again by unicity of $F(h \circ f)$, one then has $F(f \circ h) = F(f) \circ F(h)$.
We need now to define a natural isomorphism φ: $\mathbf{D}[\_,G\_] \cong \mathbf{C}[F\_,\_]$. Thus we first need to check, for a suitable φ, that for all $g \in \mathbf{D}[d',d]$ and $h \in \mathbf{C}[c,c']$ the following diagram commutes:



Equivalently,

1. $\forall f \in \mathbf{D}[d,Fc]\ \ \varphi(G(h) \circ f \circ g) = h \circ \varphi(f) \circ Fg$.

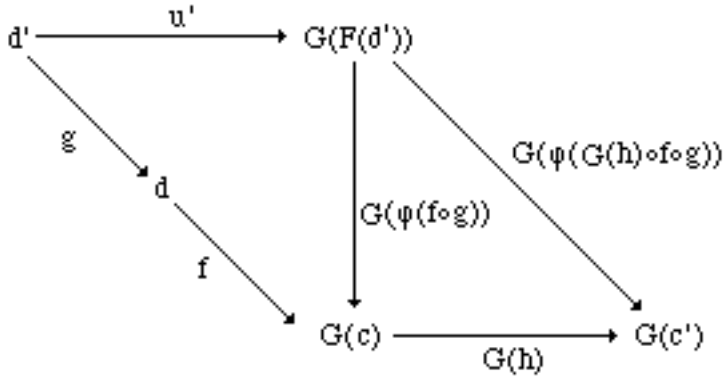
Now write u (u') for u_d ($u_{d'}$, respectively). We know then that $\forall f \in \mathbf{D}[d,G(c)]\ \ \exists! f' \in \mathbf{C}[F(d),c]$ $f = G(f') \circ u$. Define $\varphi(f) = f'$, that is, $f = G(\varphi(f)) \circ u$ (compare with the definition of F). φ is clearly a set-theoretic isomorphism; thus, we have only to prove the naturality (1).
By the definition of the functors G and F, the following diagram commutes:



That is, $G(\varphi(f\circ g))\circ u' = G(\varphi(f)\circ F(g))\circ u'$, since G is a functor. By unicity,

2. $\varphi(f \circ g) = \varphi(f) \circ F(g)$.

Moreover, for all $f \in \mathbf{D}[d,G(c)]$,



by the definition of G .

Therefore,

$(G(h) \circ f \circ g) = h \circ \varphi(f \circ g)$ by the diagram and unicity

$= h \circ \varphi(f) \circ F(g)$ by (2).

This proves (1), i.e. the naturality of φ , and by proposition 3.2.3 the proof is completed. ♦

Dually, we have the following:

5.2.2 Theorem. Let F: $\mathbf{D}\rightarrow\mathbf{C}$ be a functor such that $\forall c \in Ob_{\mathbf{C}}$ $\exists \langle u_c, d_c \rangle$ universal from F to c. Then there exists a (unique) functor G: $\mathbf{C}\rightarrow\mathbf{D}$ such that

i. $G(c) = d_c$

ii. $\mathbf{C}[F\_,\_] \cong \mathbf{D}[\_,G\_]$.

Proof The result follows by duality; anyway we explicitly reprove it, but by using a different technique from the one used above. As the reader will see, the difference is essentially notational, but she or he is invited to study both proofs since they are good examples of two common proof styles in Category Theory.

Let $G_{Ob}: Ob_{\mathbf{C}} \to Ob_{\mathbf{D}}$ be the function defined by $G_{Ob}(c) = d_c$, where $u_c: F(d_c) \to c$ is the universal arrow. We have

$$\forall f \in \mathbf{C}[F(d),c] \;\; \exists! g \in \mathbf{D}[d,G_{Ob}(c)] \;\; f = u_c \circ F(g)$$

Now define, $\forall g \in \mathbf{D}[d,G_{Ob}(c)] \;\; \tau_{d,c}(g) = u_c \circ F(g)$.

For every $d \in Ob_{\mathbf{D}}$ and $c \in Ob_{\mathbf{C}}$, $\tau_{d,c}: \mathbf{D}[d,G_{Ob}(c)] \to \mathbf{C}[F(d),c]$ is clearly a set-theoretic isomorphism. Note that, $\forall h \in \mathbf{D}[d',d]$,

1. $\tau_{d',c}(g \circ h) = u_c \circ F(g \circ h) = u_c \circ F(g) \circ F(h) = \tau_{d,c}(g) \circ F(h)$

By taking $\tau_{d,c}^{-1}(f)$ for g in (1), we have $\tau_{d',c}(\tau_{d,c}^{-1}(f) \circ h) = f \circ F(h)$, or equivalently,

2. $\tau_{d,c}^{-1}(f) \circ h = \tau_{d,c}^{-1}(f \circ F(h))$

For simplicity, we now omit the indexes of τ and τ^{-1} .

Let $G_{Mor}: Mor_{\mathbf{C}} \to Mor_{\mathbf{D}}$ be the function defined by

$$\forall k \in \mathbf{C}[c,c'] \;\; G_{Mor}(k) = \tau^{-1}(k \circ u_c) \in \mathbf{D}[G_{Ob}(c), G_{Ob}(c')]$$

We want to prove that $G = (G_{Ob}, G_{Mor})$ is a functor:

$$\begin{aligned} G(id_c) &= \tau^{-1}(u_c) \\ &= \tau^{-1}(u_c \circ id_{F(G(c))}) \\ &= \tau^{-1}(u_c \circ F(id_{G(c)})) \\ &= \tau^{-1}(\tau(id_{G(c)})) \\ &= id_{G(c)} \end{aligned}$$

and, for every $f: c' \to c''$, $k: c \to c'$,

$$\begin{aligned} G(f \circ k) &= \tau^{-1}(f \circ k \circ u_c) \\ &= \tau^{-1}(f \circ u_{c'} \circ F(\tau^{-1}(k \circ u_c))) \\ &= \tau^{-1}(f \circ u_{c'}) \circ \tau^{-1}(k \circ u_c) \qquad \text{by (2)} \\ &= G(f) \circ G(k) \end{aligned}$$

(1) proves the naturality of $\tau_{d,c}$ in the component d . We have still to prove the naturality in c, that is, $\forall g \in \mathbf{D}[d,G_{Ob}(c)]$, $\forall k \in \mathbf{C}[c,c']$

3. $\tau_{d,c'}(G(k) \circ g) = k \circ \tau_{d,c}(g)$

We have:

$$\begin{aligned} \tau(G(k) \circ g) &= \tau(\tau^{-1}(k \circ u_c) \circ g) \\ &= \tau(\tau^{-1}(k \circ u_c \circ F(g))) \qquad \text{by (2)} \\ &= k \circ u_c \circ F(g) \\ &= k \circ \tau(g) \end{aligned}$$

By taking $\tau_{d,c}^{-1}(f)$ for g in (3) we obtain $\tau_{d,c'}(G(k) \circ \tau_{d,c}^{-1}(f)) = k \circ f$, or equivalently:

4. $G(k) \circ \tau_{d,c}^{-1}(f) = \tau_{d,c'}^{-1}(k \circ f)$.

(2) and (4) state the naturality of τ^{-1}.
Note that since G must satisfies (4) , then

$$G(k) = G(k) \circ id = G(k) \circ \tau^{-1}(u_c) = \tau^{-1}(k \circ u_c)$$

which shows that the adopted definition for G was actually forced. This proves the unicity of the functor G. ♦

5.2.3 Example An interesting example of application of theorem 5.2.2 refers to Cartesian closed categories. By the previous section, we know that if **C** is a CCC, then for all a,b in $Ob_{\mathbf{C}}$, (p: $a\times b\to a$, p_2: $a\times b\to b$) is universal from Δ to (a,b), and $eval_{a,b}$: $b^a\times a\to b$ is universal from $\_\times a$ to b. Then the functions $\_\times\_$: $Ob_{\mathbf{C}\times\mathbf{C}}\to Ob_{\mathbf{C}}$ and $\_^a$: $Ob_{\mathbf{C}}\to Ob_{\mathbf{C}}$ which respectively take (a,b) to $a\times b$ and b to b^a, can be extended to two functors $\_\times\_$: $\mathbf{C}\times\mathbf{C}\to\mathbf{C}$ and $\_^a$: $\mathbf{C}\to\mathbf{C}$. The explicit definition is the following: for every f: $a\to c$, g: $b\to d$

$$(\_\times\_)(f,g) = f\times g = \langle f \circ p_1, g \circ p_2 \rangle : a\times b\to c\times d$$
$$(\_^a)(g) = \Lambda(eval_{a,b} \circ g) : b^a\to d^a$$

For every object c in **C**, even the unique arrow !c: $c\to t$ may be seen as universal arrow from the unique functor $\mathbf{!C}$: $\mathbf{C}\to\mathbf{1}$ to t. In this case, the extension of the function that takes $1\in Ob_{\mathbf{1}}$ to t to a functor **T**: $\mathbf{1}\to\mathbf{C}$ is trivial, but it is interesting that the existence of the terminal object t in **C** may be expressed by the natural isomorphism $\mathbf{1}[\mathbf{!C}(c)=1\ ,1] \cong \mathbf{C}[c,\mathbf{T}(1)=t]$.

5.2.4 Example Consider **C**, $\mathbf{C_t}$ and **Inc** as in example 5.1.4, and assume that for each object $a\in Ob_{\mathbf{C}}$ there exists the lifting a°. By example 5.1.4 we know that ex_a: $a^\circ\to a$ is an universal arrow from the embedding functor $\mathbf{Inc}$: $\mathbf{C_t}\to\mathbf{C_p}$ to a. By theorem 5.2.2 the function $\_^\circ$: $Ob_{\mathbf{C}}\to Ob_{\mathbf{C}}$ which takes every object a to its lifting a°, may be extended to a functor $\_^\circ$: $\mathbf{C_p}\to\mathbf{C_t}$. The explicit definition of the functor $\_^\circ$ on a partial arrow f: $b\to c$ is the following

$$(\_^\circ)(f) = f^\circ = \tau(f \circ ex_b)\in\mathbf{C_t}[b^\circ,c^\circ]$$

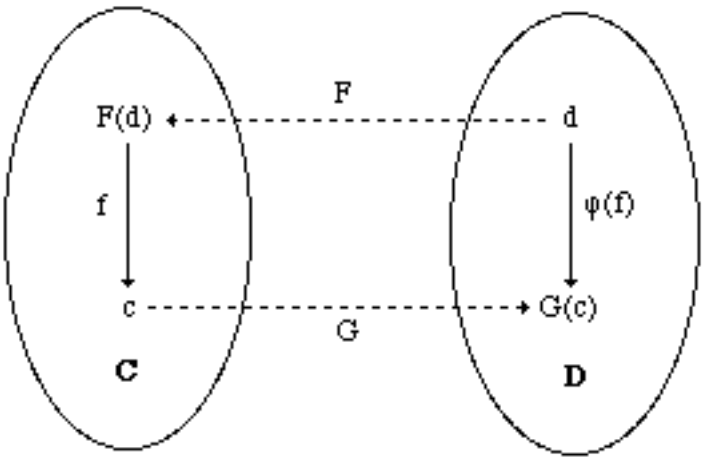
where $\tau(f\circ ex_a)$ is the only arrow such that $ex_c \circ \tau(f\circ ex_b) = f\circ ex_b$.

Note that nearly all the facts about partiality and extendability we proved depend directly on properties of natural transformations and adjunctions. That is, it was not possible to derive the properties of the lifting of b by assuming just a set-theoretic isomorphism between $\mathbf{C_p}[a,b]$ and $\mathbf{C_t}[a,b^\circ]$ for all a, as one may be tempted at first thought. The expressive categorical notion of natural transformation turns out to be essential for these purposes.

5.3 Adjunctions

In this section we derive a general notion from the previous constructions and say that the functors F and G in theorems 5.2.1 and 5.2.2 are "adjoint" to one another. The idea is that an **adjunction** establishes a relation between two categories **C** and **D** through two functors F: $\mathbf{D}\to\mathbf{C}$ and G:

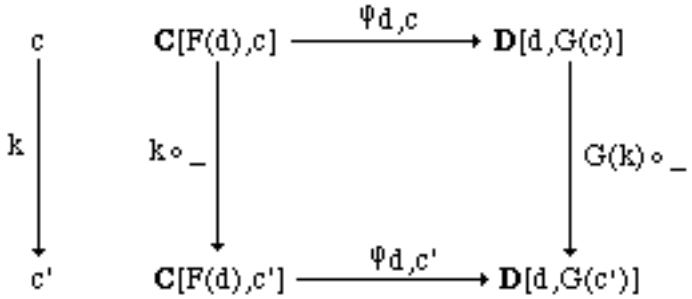
C→D; this relation creates a bijective correspondence φ of arrows in the two categories of the kind described by the following picture:

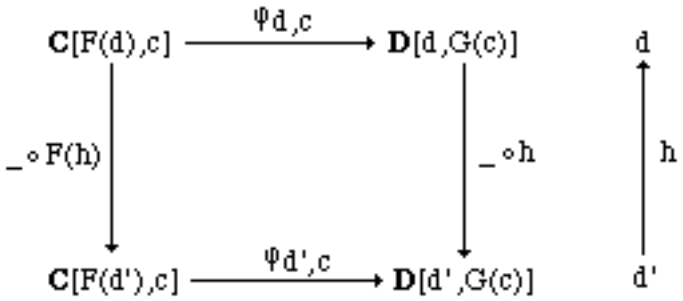


5.3.1 Definition *Let* $F: \mathbf{D}\to\mathbf{C}$ *and* $G: \mathbf{C}\to\mathbf{D}$ *be functors. Then an* ***adjunction*** *from* **D** *to* **C** *is a triple* $\langle F,G,\varphi\rangle$ *such that* $\varphi: \mathbf{C}[F\_,\_] \cong \mathbf{D}[\_,G\_]$ *is a natural isomorphism. F is called* ***left adjoint*** *of G, and G is called* ***right adjoint*** *of F.*

The naturality of the isomorphism φ deserves to be spelled out. For any $f\in\mathbf{C}[F(d),c]$, $k\in\mathbf{C}[c,c']$ and $h\in\mathbf{D}[d',d]$, we have

1. $\varphi_{d,c'}(k \circ f) = G(k) \circ \varphi_{d,c}(f)$
2. $\varphi_{d',c}(f \circ F(h)) = \varphi_{d,c}(f) \circ h$





It is equivalent to require that φ^{-1} is natural, that is, for any $g \in \mathbf{C}[d,G(c)]$, $k \in \mathbf{C}[c,c']$ and $h \in \mathbf{D}[d',d]$,

3. $\varphi^{-1}_{d,c'}(G(k) \circ g) = k \circ \varphi^{-1}_{d,c}(g)$
4. $\varphi^{-1}_{d',c}(g \circ h) = \varphi^{-1}_{d,c}(g) \circ F(h)$.

Examples

1. Let **D**, **C** be partial order categories, and $(Ob_{\mathbf{D}},\leq_D)$, $(Ob_{\mathbf{C}},\leq_C)$ the associated p.o.sets. An adjunction from **D** to **C** is a pair of monotone functions $f: Ob_{\mathbf{D}} \to Ob_{\mathbf{C}}$, $g: Ob_{\mathbf{C}} \to Ob_{\mathbf{D}}$ such that, for every $d \in Ob_{\mathbf{D}}$, $c \in Ob_{\mathbf{C}}$,

$$f(d) \leq_C c \quad \Leftrightarrow \quad d \leq_D g(d) .$$

Consider for example the partial order **Z** of relative numbers, and the partial order **R** of real numbers. Let $I: \mathbf{Z} \to \mathbf{R}$ be the obvious inclusion, and $\lfloor\_\rfloor: \mathbf{R} \to \mathbf{Z}$ be the function that takes a real number r to its lower integer part $\lfloor r \rfloor$. Then I and $\lfloor\_\rfloor$ define an adjunction from **Z** to **R**, since

1. $I(z) \leq_{\mathbf{R}} r \quad \Leftrightarrow \quad z \leq_{\mathbf{Z}} \lfloor r \rfloor$

Conversely let $\lceil\_\rceil: \mathbf{R} \to \mathbf{Z}$ be the function that takes a real number r to its upper integer part $\lceil r \rceil$. Then $\lceil\_\rceil$ and I define an adjunction from **R** to **Z**, since

2. $\lceil r \rceil \leq_{\mathbf{Z}} z \quad \Leftrightarrow \quad r \leq_{\mathbf{R}} I(z)$

Note that $\lfloor\_\rfloor$ and $\lceil\_\rceil$ are respectively the right and left adjoint to the same functor I. Note, moreover, that $\lfloor\_\rfloor$ and $\lceil\_\rceil$ are the unique functions that respectively satisfy conditions (1) and (2) for all r and z.

Another interesting example of adjunctions between partial orders as categories is the following: consider the p.o.set of positive integers **N**. For every natural number n, let $\_ \cdot n : \mathbf{N} \to \mathbf{N}$ be the function that takes a natural numbers m to the product $m \cdot n$. The right adjoint to $\_ \cdot n$ is the the function $div(\_,n): \mathbf{N} \to \mathbf{N}$ that takes q to (the lower integer part of) q divides n .

Indeed, for every m, q, $m \cdot n \leq q \quad \Leftrightarrow \quad m \leq div(q,n)$

Analogously the "minus" operation is right adjoint to "plus."

2. This further example uses familiar notions and applies the categorical understanding of a fundamental technique in (universal) algebra. Given a category **C** of structures and a category **D** of slightly more general ones, the right adjoint of the forgetful functor from **C** to **D** defines the "free structures" over the objects in the category **D**. This technique is widely explained in several places (see references), so that we just hint at it here.

The category **Graph** was defined in the example 4.1.5. Recall now that a graph **G** is given by:

- a set V of objects (nodes)
- a set T of arrows (edges)
- a function $\partial_1: T \to O$ which assigns to each arrow f its range $\partial_1(f)$
- a function $\partial_2: T \to O$ which assigns to each arrow f its target $\partial_2(f)$.

Morphisms of graphs **G**, **G'** are pairs <f,g>, where f: T→T' and g: V→V' have the properties in the example 4.1.5. We already mentioned (see the exercise following that example) that each small category **C** may be regarded as a graph **G** = U(**C**), just forgetting identities and composition. Of course, U takes objects to nodes and arrows to edges. Moreover, every functor F: **C**→**D** gives a morphisms H = U(F): U(**C**)→U(**D**) between the associated graphs; the reader should have checked that U: **Cat** →**Graph** is actually a (forgetful) functor. Conversely every graph **G** generates a category **C** = C(**G**) with the same objects of **G**, and, for arrows, the finite strings $(f_1,...,f_n)$ of composable arrows of **G**, i.e., of arrows in the due types (the empty strings are the identities in C(**G**)). Composition in C(**G**) is just string concatenation, that is,

$$(f_1, \ldots ,f_n) \circ (g_1, \ldots ,g_m) = (f_1, \ldots ,f_n,g_1, \ldots ,g_m) .$$

Note that $(f_1,...,f_n) = f_1 \circ \ldots \circ f_n$. The category C(**G**) is called the **free category** generated by **G**.

This construction may be extended to morphisms of graphs: if H: **G**→**G'** then C(H): C(**G**)→C(**G'**) is the functor that coincides with H on objects, and that is defined on morphisms by:

$$C(H)(f_1,...,f_n) = (H(f_1),... H(f_n)).$$

It is easy to prove that C is a functor from **Grph** to **Cat**. Actually, we have an adjoint situation, since there is an isomorphism Θ : **Cat**[C(**G**), **C**] ≅ **Grph**[**G**, U(**C**)] which is natural in **G** and **C**. The isomorphism Θ takes every functor F: C(**G**)→**C** to the morphism Θ(F): **G**→**C**, which is the "restriction" of F on **G**. For the nature of C(**G**), every functor F: C(**G**)→**C** is uniquely determined by its behavior on the arrows of **G**, indeed if $(f_1,\ldots,f_n)$ is an arrow in C(**G**), by definition of a functor, $F((f_1,\ldots,f_n)) = F(f_1 \circ \ldots \circ f_n) = F(f_1) \circ \ldots \circ F(f_n)$. This proves that Θ is injective. But Θ is also surjective, since if H: **G**→U(**C**) , we can define a functor F: C(**G**)→**C** by $F((f_1,\ldots f_n)) = H(f_1) \circ ... \circ H(f_n)$, and clearly Θ(F) = H. We leave it to the reader to prove the naturality of the isomorphism.

Exercise In section 4.3 we turned each Petri net N into a monoidal category $C^{\otimes}(N)$. Describe $C^{\otimes}(N)$ as a freely generated category.

Exercise Let **C** and **D** be discrete categories (i.e., the only morphisms of the categories are identities). Prove that <G,F,τ>: **C**→**D** is an adjunction if and only if G and F define an isomorphism between **C** and **D**.

In the previous section, we have actually shown how to construct an adjunction when one can uniformly obtain a universal arrow $<u_d,c_d>$ from each object d . Now we show how to obtain universal arrows out of an adjunction, and put together the two results.

5.3.2 Theorem. *If* $< F: \mathbf{D}\to\mathbf{C} , G: \mathbf{C}\to\mathbf{D} , \varphi >$ *is an adjunction from* **D** *to* **C**, *then*
1. $< u = \varphi(id_{F(d)}) : d\to G(F(d)) , F(d)>$ *is universal from d to G*

u is called ***unit*** *of the adjunction*

2. $\langle u'=\varphi^{-1}(id_{G(c)}): F(G(c))\to c, G(c)\rangle$ *is universal from F to c*

u' is called ***counit*** *of the adjunction.*

Conversely, if $G: \mathbf{C}\to\mathbf{D}$ *is a functor and (1) holds (or* $F: \mathbf{D}\to\mathbf{C}$ *is a functor and (2) holds), then* $\langle F,G,\varphi\rangle$ *is an adjunction from* $\mathbf{D}$ *to* $\mathbf{C}$.

Proof. (1) is given by theorem 5.1.6 ($\Leftarrow$) and the definition of G. Note that (2) follows dually. The converse is stated in theorems 5.2.1 and 5.2.2. ♦

Thus, if $\langle$ F, G, $\varphi\rangle$ is an adjunction, then the functor F of theorem 5.2.1 is the left adjoint of G and, conversely, G in theorem 5.2.2 is the right adjoint of F. In view of the expressive power of the notion of adjunction, we can now state in one line some of the concepts we introduced in the previous chapters.

5.3.3 Corollary *Let* $\mathbf{C}$ *be a category. Then*

i. $\mathbf{C}$ *has a terminal object iff the unique functor* $!\mathbf{C}: \mathbf{C}\to\mathbf{1}$ *has a right adjoint;*

ii. $\mathbf{C}$ *has finite products iff the diagonal functor has a right adjoint;*

iii. $\mathbf{C}$ *is a CCC iff it is cartesian (i.e.,* $!\mathbf{C}: \mathbf{C}\to\mathbf{1}$ *and* $\Delta : \mathbf{C}\to\mathbf{C}\times\mathbf{C}$ *have right adjoints) and, for each* $a\in Ob_{\mathbf{C}}$, *the functor* $\_\times a : \mathbf{C}\to\mathbf{C}$ *has a right adjoint.*

Proof. Immediate by theorem 5.2.2 and the considerations in example 5.2.3. ♦

5.3.4 Corollary *Let* $\mathbf{C}$ *be a category of partial morphisms. The lifting functor* $\_^{\circ} : \mathbf{C}\to\mathbf{C}_t$ *is the right adjoint of the embedding functor* $\mathbf{Inc}: \mathbf{C}_t\to\mathbf{C}$.

Proof Immediate by 5.2.2 and the considerations in example 5.2.4. ♦

As the reader probably expects, it is also possible to give a fully equational characterization of adjunctions.

5.3.5 Theorem *An adjunction* $\langle F, G,\tau\rangle : \mathbf{C}\to\mathbf{D}$ *is fully determined by the following data:*

- *the functor* $G: \mathbf{D}\to\mathbf{C}$
- *a function* $f: Ob_{\mathbf{C}}\to Ob_{\mathbf{D}}$ *such that, for every object c of* $\mathbf{C}$, $f(c) = F(c)$
- *for every object c of* $\mathbf{C}$, *an arrow* $unit_c\in\mathbf{C}[c, G(f(c))]$
- *for every object c of* $\mathbf{C}$ *and d of* $\mathbf{D}$, *a function* $\tau_{c,d}^{-1}: \mathbf{C}[c,G(d)]\to\mathbf{D}[f(c),d]$

such that, for every $h\in\mathbf{C}[c,G(d)]$ *and* $k\in\mathbf{D}[f(c),d]$,

1. $G(\tau_{c,d}^{-1}(h)) \circ unit_c = h$;
2. $\tau_{c,d}^{-1}(G(k) \circ unit_c) = k$.

Proof The theorem is an immediate consequence of theorems 5.3.2 and 5.1.6. A direct proof is not difficult, and its study is a good exercise for the reader since it summarizes many of the previous results. Here it is:

The function f may be extended to a functor F by setting, for $k \in \mathbf{C}[c,c']$, $F(k) = \tau_{c',d}^{-1}(unit_{c'} \circ k)$. Note that

$$\begin{aligned} F(id_c) &= \tau_{c,f(c)}^{-1}(unit_c) \\ &= \tau_{c,f(c)}^{-1}(id_{G(f(c))} \circ unit_c) \\ &= \tau_{c,f(c)}^{-1}(G(id_{f(c)}) \circ unit_c) \\ &= id_{f(c)} . && \text{by (2)} \end{aligned}$$

and moreover, omitting the indexes for notational convenience,

$$\begin{aligned} F(h \circ k) &= \tau^{-1}(unit \circ h \circ k) \\ &= \tau^{-1}(G(\tau^{-1}(unit \circ h)) \circ unit \circ k) && \text{by (1)} \\ &= \tau^{-1}(G(\tau^{-1}(unit \circ h)) \circ G(\tau^{-1}(unit \circ k)) \circ unit) && \text{by (1)} \\ &= \tau^{-1}(G(\tau^{-1}(unit \circ h) \circ \tau^{-1}(unit \circ k)) \circ unit) \\ &= \tau^{-1}(unit \circ h) \circ \tau^{-1}(unit \circ k) && \text{by (2)} \\ &= F(h) \circ F(k) . \end{aligned}$$

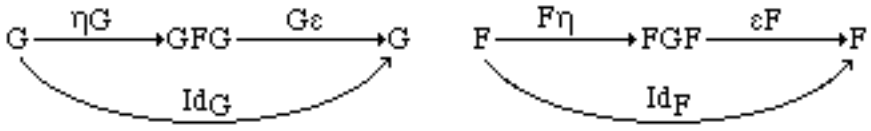
Let now, for every object c of **C** and d of **D**, $\tau_{c,d}$: $\mathbf{D}[f(c),d] \rightarrow \mathbf{C}[c,G(d)]$ be the function defined by $\tau_{c,d}(k) = G(k) \circ unit_c$. Equations (1) and (2) express exactly the fact that $\tau_{c,d}$ and $\tau_{c,d}^{-1}$ define an isomorphism. We have still to prove their naturality. Let $k \in \mathbf{D}[d,d']$, $h \in \mathbf{C}[c,G(d)]$, $h' \in \mathbf{C}[c',c]$, and $k' \in \mathbf{D}[f(c),d]$; then

$$\begin{aligned} \text{nat-1.} \quad \tau^{-1}(G(k) \circ h) &= \tau^{-1}(G(k) \circ G(\tau^{-1}(h)) \circ unit_c) \\ &= \tau^{-1}(G(k \circ \tau^{-1}(h)) \circ unit_c) \\ &= k \circ \tau^{-1}(h) ; \\ \text{nat-2.} \quad \tau^{-1}(h \circ k) &= \tau^{-1}(G(\tau^{-1}(h)) \circ unit \circ k) && \text{by (1)} \\ &= \tau^{-1}(h) \circ \tau^{-1}(unit \circ k) && \text{by (nat-1)} \\ &= \tau^{-1}(h) \circ F(k) ; \\ \text{nat-3.} \quad \tau(k' \circ F(h')) &= \tau (\tau^{-1}(\tau(k')) \circ F(k)) \\ &= \tau (\tau^{-1}(\tau(k') \circ h') && \text{by (nat-2)} \\ &= \tau(k') \circ h' ; \\ \text{nat-4.} \quad \tau(k \circ k') &= G(k \circ k') \circ unit \\ &= G(k) \circ G(k') \circ unit \\ &= G(k) \circ \tau(k'). \ \blacklozenge \end{aligned}$$

5.3.6 Proposition *Let* $\langle F,G,\tau \rangle: \mathbf{C} \rightarrow \mathbf{D}$ *be an adjunction. Then there exist two natural transformations* $\eta : Id_{\mathbf{C}} \rightarrow GF$ *and* $\varepsilon: FG \rightarrow Id_{\mathbf{D}}$ *such that, for every c in* **C** *and d in* **D**, $\eta(c)$ *and* $\varepsilon(d)$ *are respectively the unit and counit of the adjunction.*

Proof: exercise. ♦

In other words, one may construct the unit and counit "uniformely" and naturally. Observe also that, if $\eta : Id_C \to GF$ and $\varepsilon: FG \to Id_D$ are the natural transformations in proposition 5.3.6, then the following diagram commutes:



The previous diagrams fully characterize an adjunction: as a matter of fact many authors prefer to define an adjunction between two categories **C** and **D** as a quadruple $(F, G, \eta, \varepsilon)$ where $F: \mathbf{C} \to \mathbf{D}$ and $G: \mathbf{D} \to \mathbf{C}$ are functors, and $\eta : Id_C \to GF$, $\varepsilon: FG \to Id_D$ are natural transformations such that

$$(G\varepsilon) \circ (\eta G) = id_G ;$$
$$(\varepsilon F) \circ (F\eta) = id_F .$$

We leave it as an exercise for the reader to prove the equivalence of this notion with the one we have adopted. We shall use the definition of adjunction as a quadruple in the next section, since it simplifies the investigation of the relation between adjunctions and monads.

Exercises

1. An adjointness $(F, G, \eta, \varepsilon)$ from **C** to **D** is an **adjoint equivalence** if and only if η and ε are natural isomorphisms. Prove that given two equivalent categories **C** and **D** (see section 3.2) it is always possible to define an adjoint equivalence between them.
2. Given an adjointness $(F, G, \eta, \varepsilon)$ from **C** to **D**, prove the equivalence of the following statements:
   i. $\eta GF = GF\eta$;
   ii. ηG is an isomorphism;
   iii. $\varepsilon FG = FG\varepsilon$;
   iv. εF is an isomorphism.

5.3.7 Example In section 3.4 we defined the CCCs of limit and filter spaces, **L-spaces** and **FIL** respectively, which generalize topological spaces, **Top**. The functorial "embeddings" mentioned in that example are actually adjunctions. Recall that $H : \mathbf{Top} \to \mathbf{FIL}$ is given by

$$H((X,top)) = (X,F) \text{ where } F(x) = \{\Phi \mid \Phi \text{ is a filter and } \forall 0 \in top\ (x \in 0 \Rightarrow 0 \in \Phi)\}.$$

$H(f) = f$ by the definition of continuity. H has a left adjoint $T : \mathbf{FIL} \to \mathbf{Top}$ defined by

$$T((X, F)) = (X,top) \text{ where } 0 \in top \text{ iff } \forall x \in 0\ \forall \Phi \in F(x)\ \ 0 \in \Phi .$$

Also in this case, filter continuity corresponds to topological continuity, i.e., $T(f) = f$. The reader may easily define the natural isomorphism τ.

In general, limits are not unique in filter spaces. A stronger notion of convergence, to be used for computability (see the final remark in section 8.4), may be given as follows. For (X,F) in **FIL** consider (X,top) = T((X, F)) and define

(s-conv.) $\Phi \downarrow^{s} x$ iff $\Phi \in F(x)$ and $\forall 0 \in \Phi \cap top$ $x \in 0$.

Then top is T_0 iff s-convergent filters have a unique limit.

Let $N = (\omega, F)$ be the natural numbers with the filter structure induced by the discrete topology and $M = N^N$. With some work (see references) one can show that M^M is not topological, i.e. for no (X,top) one has $M^M = H((X,top))$. The idea is that each topological filter space has a least filter for each F(x), the neighborhood filter at x; deduce from this that the associated adjointness (T,H,η,ε) to (T,H,τ) is not an adjoint equivalence (which one of η and ε is not a natural isomorphism?).

Exercises *(based on the previous example and exercises)*

1. Consider the full subcategory of **FIL** given by the filter spaces (X,F) such that, for each x, there is a least $\Phi \in F(x)$. Give an adjoint equivalence between **Top** and this category.
2. Check that the functors between **L-spaces** and **FIL** defined in the exercise in section 3.4.2 yield an adjunction, which is not an adjoint equivalence.
3. Give directly an adjunction between **Top** and **FIL**, and compare the definition with the adjunction obtained by composition of functors. (*Hint for the direct construction:* given an L-space $(X,\downarrow)$, define (X,top) by $0 \in top$ iff $\forall x \in 0$, $\forall \{x_i\} \downarrow x$, $\{x_i\} \subseteq 0$ eventually. Conversely, for (X, top) topological space, define $(X,\downarrow)$ by $\{x_i\} \downarrow x$ iff $\forall 0 \in top$ $(x \in 0 \Rightarrow \{x_i\} \subseteq 0$ eventually).

5.4 Adjunctions and Monads

In this section we study the relation between two seemingly distant concepts as adjunction and monad. As a matter of fact, every adjunction immediately defines a monad, and conversely every monad can be thought of as generated by an adjunction, called a **resolution** for the monad (see 5.4.2 below). Resolutions for a given monad can be build up in a category by introducing a natural notion of morphism between them; it then happens that the Eilenberg-Moore and the Kleisli Categories associated with the monad (see definitions 4.2.3 and 4.2.4) are respectively the terminal and initial object of the category.

The presentation is rather technical; at first reading, the reader may just look at the first theorem, which will be applied in the next section.

5.4.1 Proposition *Let $(F, G, \eta, \varepsilon)$ be an adjunction from **C** to **D**; then $(T = GF, \eta, \mu = G\varepsilon F)$ is a monad on **C** and $(T = FG, \delta = G\eta F, \varepsilon)$ is a comonad.*

Proof Note first that GF, η, and $G\varepsilon F$ have the correct types, i.e., $T = GF: \mathbf{C}\to\mathbf{C}$, $\eta: Id_{\mathbf{C}}\to GF$, and $\mu = G\varepsilon F: GFGF\to GF$. We must prove the unity and associative laws for the monad.
As for the unity laws we have

$$\mu \circ T\eta = G\varepsilon F \circ GF\eta = G(\varepsilon F \circ F\eta) = G(id_F) = id_{GF}$$
$$\mu \circ \eta T = G\varepsilon F \circ \eta GF = (G\varepsilon \circ \eta G)F = id_G(F) = id_{GF}$$

For the associative law, note first that

$$\varepsilon \circ \varepsilon FG = \varepsilon \circ FG\varepsilon .$$

Indeed, for any $d\in Ob\mathbf{D}$, and letting $f = \varepsilon_d: FG(d)\to d$,

$$\begin{aligned} \varepsilon_d \circ \varepsilon_{FG(d)} &= f \circ \varepsilon_{FG(d)} \\ &= \varepsilon_d \circ FG(f) && \text{by naturality of } \varepsilon \\ &= \varepsilon_d \circ FG(\varepsilon_d) \end{aligned}$$

Then one has the following:

$$\begin{aligned} \mu \circ \mu T &= G\varepsilon F \circ G\varepsilon FGF \\ &= G(\varepsilon \circ \varepsilon FG)(F) \\ &= G(\varepsilon \circ FG\varepsilon)(F) \\ &= G\varepsilon F \circ GFG\varepsilon F \\ &= \mu \circ T\mu . \end{aligned}$$

The rest is an exercise in duality. ♦

5.4.2 Definition *Let (T, η, μ) be a monad over a category $\mathbf{C}$. A **resolution** for (T, η, μ) is a category $\mathbf{D}$ and an adjunction $(F, G, \eta, \varepsilon)$ from $\mathbf{C}$ to $\mathbf{D}$ such that $T = GF$ and $\mu = G\varepsilon F$. A **morphism** between two resolutions $(F, G, \eta, \varepsilon)$: $\mathbf{C}\to\mathbf{D}$ and $(F', G', \eta, \varepsilon')$: $\mathbf{C}\to\mathbf{D}'$ (for the same monad) is a functor $H: \mathbf{D}\to\mathbf{D}'$ such that $F' = H \circ F$, $G = G' \circ H$, and $H\varepsilon = \varepsilon' H$.*

It is easily proved that resolutions with the associated morphisms form a category. Now we are going to prove that the Eilenberg-Moore and Kleisli categories associated with a monad (T, η, μ) both give rise to resolutions. In particular, they are respectively the terminal and initial objects in the category of all resolutions for that monad.

5.4.3 Proposition *Let (T, η, μ) be a monad over a category $\mathbf{C}$, and let $\mathbf{C}^T$ be the Eilenberg-Moore category associated with the monad. Then there exists a resolution for (T, η, μ) which is an adjunction from $\mathbf{C}^T$ to $\mathbf{C}$.*

Proof Let $U^T: \mathbf{C}^T\to\mathbf{C}$ be the forgetful functor that takes every algebra (c,α) to c, and every morphism of algebras h to the same h regarded as a morphism in $\mathbf{C}$. Let $F^T: \mathbf{C}\to\mathbf{C}^T$ be the functor which takes every object c to its free algebra $(T(c),\mu_c)$, and every morhism $f: c\to c'$ to $F^T(f) = T(f)$. Let $\varepsilon^T: F^TU^T\to id_{\mathbf{C}}T$ be the natural transformation defined by $\varepsilon^T(c,\alpha) = \alpha$ (note that $\alpha: T(c)=F^TU^T(c,\alpha)\to c$). We want to prove that $(F^T, U^T, \eta, \varepsilon^T)$ is a resolution for (T, η, μ).

Obviously $U^T \circ F^T = T$. $U^T \varepsilon^T F^T = \mu$, since for any object c one has

$$\begin{aligned}(U^T \varepsilon^T F^T)(c) &= U^T(\varepsilon^T {}_{F^T(c)}) \\ &= U^T(\varepsilon^T(T(c), \mu_c)) && \text{by def. of } F^T \\ &= U^T(\mu_c) && \text{by def. of } \varepsilon^T \\ &= \mu_c && \text{by def. of } U^T\end{aligned}$$

We still haveto prove that $(F^T, U^T, \eta , \varepsilon^T)$ is an adjunction from $\mathbf{C}^T$ to $\mathbf{C}$, that is,

$$(U^T\varepsilon^T) \circ (\eta U^T) = id_{U^T}$$
$$(\varepsilon^T F^T) \circ (F^T \eta) = id_{F^T}$$

One has, for every T-algebra (c,α),

$$\begin{aligned}(U^T\varepsilon^T \circ \eta U^T)\,(c,\alpha) &= U^T(\varepsilon^T{}_{(c,\alpha)}) \circ \eta_{U^T(c,\alpha)} \\ &= \alpha \circ \eta_c && \text{by def. of } \varepsilon^T \text{ and } U^T \\ &= id_c && \text{by def. of T-algebra}\end{aligned}$$

And for every $c \in Ob_{\mathbf{C}}$:

$$\begin{aligned}(\varepsilon^T F^T \circ F^T \eta)\,(c) &= \varepsilon^T{}_{F^T(c)} \circ F^T(\eta_c) \\ &= \mu_c \circ T(\eta_c) && \text{by def. of } \varepsilon^T \text{ and } F^T \\ &= id_{T(c)} && \text{by the unity law of the monad.}\blacklozenge\end{aligned}$$

We say that the resolution $(F^T, U^T, \eta , \varepsilon^T)$ from $\mathbf{C}$ to the Eilenberg-Moore category $\mathbf{C}^T$, and given by proposition 5.4.3, is **associated with** $\mathbf{C}^T$.

5.4.4 Proposition *Let (T, η , μ) be a monad over a category $\boldsymbol{C}$ .Then the resolution $(F^T, U^T, \eta, \varepsilon^T)$: $\boldsymbol{C} \to \boldsymbol{C}^T$, associated with the Eilenberg-Moore Category $\boldsymbol{C}^T$, is a terminal object in the category of all the resolution for the monad (T, η , μ).*

Proof Let $(F, G, \eta , \varepsilon)$: $\mathbf{C} \to \mathbf{D}$ be another resolution for (T, η , μ). We must prove that there exists a unique arrow from $(F, G, \eta , \varepsilon)$ to $(F^T, U^T, \eta , \varepsilon^T)$. Remember (cf. definition 5.4.2) that such an arrow is a functor $H: \mathbf{D} \to \mathbf{C}^T$, such that $F^T = H \circ F$, $G = U^T \circ H$, and $H\varepsilon = \varepsilon^T H$.

Define, for any object d, and any morphism f of $\mathbf{D}$,

$$H(d) = (G(d), G(\varepsilon(d)))$$
$$H(f) = G(f).$$

Then one has, for any $c \in Ob_{\mathbf{C}}$, any $h \in Mor_{\mathbf{C}}$,

$$H(F(c)) = (G(F(c)), G(\varepsilon(F(c)))) = (T(c), \mu_c) = F^T(c)$$
$$H(F(h)) = G(F(h)) = T(h) = F^T(h)$$

that proves the equality $H \circ F = F^T$.

Moreover, for any $d \in Ob_{\mathbf{D}}$, and any $f \in Mor_{\mathbf{D}}$,

$$U^T(H(d)) = U^T(G(d), G(\varepsilon(d))) = G(d)$$
$$U^T(H(f)) = U^T(G(f)) = G(f), \text{ as } U^T \text{ is the identity on morphisms.}$$

That proves the equality $G = U^T \circ H$.

Finally, for any $d \in Ob_{\mathbf{D}}$,

$$\varepsilon^T_{H(d)} = \varepsilon^T_{(G(d),\, G(\varepsilon(d)))} = G(\varepsilon(d)) = H(\varepsilon(d))$$

that proves the equality $H\varepsilon = \varepsilon^T H$.

We have still to prove that H is the unique morphism from $(F, G, \eta, \varepsilon)$ to $(F^T, U^T, \eta, \varepsilon^T)$. Let H' be another morphism; then, for any $f \in Mor_{\mathbf{D}}$,

$$H'(f) = U^T(H'(f) = G(f) = U^T(H(f) = H(f)$$

and, for any $d \in Mor_{\mathbf{D}}$,

$$\begin{aligned} H'(d) &= (\, U^T(H'(d),\, \varepsilon^T_{H'(d)}) && \text{by def. of } U^T \text{ and } \varepsilon^T \\ &= (\, G(d),\, H'(\varepsilon^T(d))\,) && \text{as } G = U^T \circ H' \\ &= (\, U^T(H(d),\, H(\varepsilon^T(d))\,) && \text{as } H'(f) = H(f) \\ &= (\, U^T(H'(d),\, \varepsilon^T_{H(d)}) \\ &= H(d). \end{aligned}$$

This completes the proof. ♦

The unique functor to $(F^T, U^T, \eta, \varepsilon^T)$ in the category of all resolutions for a given monad (T, η, μ) is called **comparison functor** and it is usually denoted by K^T.

The category of resolutions of a monad has also an initial object, which is based on the Kleisli category associated with the monad.

5.4.5 Proposition *Let (T, η, μ) be a monad over a category $\mathbf{C}$, and let $\mathbf{C}_T$ be the Kleisli category associated with the monad. Then there exists a resolution for (T, η, μ) that is an adjunction from $\mathbf{C}_T$ to $\mathbf{C}$.*

Proof: Let $U_T: \mathbf{C}_T \to \mathbf{C}$ be the functor defined by the following:

for any object c of $\mathbf{C}_T$ (i.e., of $\mathbf{C}$), and any morphism $h \in \mathbf{C}_T[c,c']$ (and thus $h \in \mathbf{C}[c,T(c')]$)

$$U_T(c) = T(c);$$

$$U_T(h) = \mu_{c'} \circ T(h).$$

Let $F_T: \mathbf{C} \to \mathbf{C}_T$ be functor defined by the following:

for any object c of $\mathbf{C}$, and any morphism $f \in \mathbf{C}[c,c']$

$$F_T(c) = c;$$

$$F_T(f) = \eta_{c'} \circ f \;\; (= T(f) \circ \eta_c).$$

Let $\varepsilon_T: F_T U_T \to id$ be the natural transformation defined by the following:

for any object c of $\mathbf{C}_T$

$$\varepsilon_T(c) = id_{T(c)} \quad (\text{in } \mathbf{C}).$$

We want to prove that $(F_T, U_T, \eta, \varepsilon_T): \mathbf{C} \to \mathbf{C}_T$ is a resolution for (T, η, μ).

Obviously, $U_T \circ F_T = T$.

Moreover, $U_T\, \varepsilon_T\, F_T = \mu$, since for any object c one has

$$(U_T\, \varepsilon_T\, F_T)(c) = U_T(\varepsilon_T(c)) \qquad \text{by def. of } F_T$$

$= U_T(id_{T(c)})$ by def. of ε_T

$= \mu_c$. by def. of U_T

We have still to prove that $(F_T, U_T, \eta , \varepsilon_T)$ is an adjunction from $\mathbf{C}$ to $\mathbf{C}_T$, that is,

$(U_T\varepsilon_T) \circ (\eta U_T) = id: U_T \to U_T$

$(\varepsilon_T F_T) \circ (F_T\eta) = id: F_T \to F_T$

One has, for every object c of $\mathbf{C}_T$:

$(U_T\varepsilon_T \circ \eta U_T)(c) = U_T(id_{T(c)}) \circ \eta_{T(c)}$ by def. of ε_T and U_T

$= \mu_c \circ T(id_{T(c)}) \circ \eta_{T(c)}$ by def. U_T on morphisms

$= \mu_c \circ \eta_{T(c)}$ as T is a functor

$= id_c$. by the unity law of the monad

And, for every $c \in Ob\mathbf{C}$,

$(\varepsilon_T F_T \circ F_T\eta)(c) = \varepsilon_T(c) \circ (\eta_{T(c)} \circ \eta_c)$ by def. of F_T

$= id_{T(c)} \circ (\eta_{T(c)} \circ \eta_c)$ by def. of ε_T

$= \mu_C \circ T(id_{T(c)}) \circ \eta_{T(c)} \circ \eta_c$ by def. of composition $\circ$ in $\mathbf{C}_T$

$= \mu_C \circ \eta_{T(c)} \circ \eta_c$

$= \eta_c$ by the unity law of the monad

$= id_c$ by def. of the identity in $\mathbf{C}_T$. ♦

5.4.6 Proposition *Let (T, η, μ) be a monad over a category* $\boldsymbol{C}$*. The resolution* $(F_T, U_T, \eta , \varepsilon_T)$*:* $\boldsymbol{C} \to \boldsymbol{C}_T$ *associated with the Kleisli Category* $\boldsymbol{C}_T$ *is an initial object in the category of resolutions for the monad (T, η , μ).*

Proof Let $(F, G, \eta , \varepsilon)$: $\mathbf{C} \to \mathbf{D}$ be another resolution for (T, η , μ). We must prove that there exists a unique arrow from $(F_T, U_T, \eta , \varepsilon_T)$ to $(F, G, \eta , \varepsilon)$, that is a unique functor $K: \mathbf{C}_T \to \mathbf{D}$, such that $F = K \circ F_T$, $U_T = G \circ K$, and $K\varepsilon_T = \varepsilon K$.

Define, for any object c of $\mathbf{C}_T$, and any morphism $f \in \mathbf{C}_T[c,c']$,

$K(c) = F(c)$;

$K(f) = \varepsilon_{F(c')} \circ F(f)$.

where c and f are regarded as object and morphism of $\mathbf{C}$.

Then one has, for any $c \in Ob\mathbf{C}$, any $h \in \mathbf{C}[c,c']$,

$K(F_T(c)) = K(c) = F(c)$

$K(F_T(h)) = K(\eta_{c'} \circ h)$ by def. of F_T

$= \varepsilon_{F(c')} \circ F(\eta_{c'} \circ h)$ by def. of K

$= \varepsilon_{F(c')} \circ F(\eta_{c'}) \circ F(h)$ as F is a functor

$= F(h)$ as $(F, G, \eta , \varepsilon)$ is an adjunction

This proves the equality $K \circ F_T = F$.

Moreover, for any object c of $\mathbf{C}_T$, and any morphism $f \in \mathbf{C}_T[c,c']$,

$G(K(c)) = G(F(c)) = T(c) = U_T(c)$

$$\begin{aligned} G(K(f)) &= G(\varepsilon_{F(c')} \circ F(f)) && \text{by def. of } K \\ &= G(\varepsilon_{F(c')}) \circ G(F(f)) && \text{as G is a functor} \\ &= \mu_{c'} \circ T(f) && \text{as } (F, G, \eta, \varepsilon) \text{ is a resolution} \\ &= U_T(f) && \text{by def. of } U_T \end{aligned}$$

that proves the equality $U_T = G \circ K$.

Finally, for any $d \in Ob\mathbf{D}$,

$$\begin{aligned} K(\varepsilon_T(c)) &= K(id_{T(c)}) && \text{by def. of } \varepsilon_T \\ &= \varepsilon_{F(c')} \circ F(id_{T(c)}) && \text{by def. of } K \\ &= \varepsilon_{F(c')} && \text{as } F \text{ is a functor} \\ &= \varepsilon_{K(c')} && \text{by def. of } K \end{aligned}$$

that proves the equality $K\varepsilon_T = \varepsilon K$.

We have still to prove that K is the unique morphism from $(F_T, U_T, \eta, \varepsilon_T)$ to $(F, G, \eta, \varepsilon)$.

Let $K' : \mathbf{C}_T \to \mathbf{D}$ be another morphism; then, for every object c of $\mathbf{C}_T$,

$$\begin{aligned} K'(c) &= K'(F_T(c)) && \text{by def. of } F_T \\ &= F(c) && \text{as } K' \circ F_T = F \\ &= K(F_T(c)) && \text{as } K \circ F_T = F \\ &= K(c) && \text{by def. of } F_T \end{aligned}$$

and, for any $f \in \mathbf{C}_T[c,c']$,

$$\begin{aligned} K'(f) &= K'(id_{c'} \circ f) \\ &= K'(\mu_{c'} \circ \eta_{T(c')} \circ f) && \text{by the unitary law of the monad} \\ &= K'(\mu_{c'} \circ T(id_{T(c')}) \circ \eta_{T(c')} \circ f) \\ &= K'(id_{T(c')} \circ (\eta_{T(c')} \circ f)) && \text{by def. of composition } \circ \text{ in } \mathbf{C}_T \\ &= K'(id_{T(c')}) \circ K'(\eta_{T(c')} \circ f)) && \text{as } K' \text{ is a functor} \\ &= K'(\varepsilon_T(c')) \circ K'(F_T(f))) && \text{by def. of } \varepsilon_T \text{ and } F_T \\ &= \varepsilon_{K'(c')} \circ F(f) && \text{as } K'\varepsilon_T = \varepsilon K' \text{ and } F = (K' \circ F_T) \\ &= \varepsilon_{F(c')} \circ F(f) && \text{as } K'(c) = K(c) = F(c) \\ &= K(f). && \text{by def. of } K \end{aligned}$$

This completes the proof. ♦

For consistency with the terminology adopted for the comparison functor, we shall denote by K_T the unique arrow from the initial object $(F_T, U_T, \eta, \varepsilon_T)$ in the category of all resolutions for a monad (T, η, μ).

Consider now the comparison functor from the initial to the terminal object. Of course, it must be $K^T = K_T$; let us check this explicitly.

For any object c in $\mathbf{C}_T$,

$$\begin{aligned} K^T(c) &= (U_T(c), U_T(\varepsilon_T(c))) && \text{by def. of } K^T \\ &= (T(c), \mu_c \circ T(id_{T(c)})) && \text{by def. of } U_T \text{ and } \varepsilon_T \end{aligned}$$

$$= (T(c), \mu_c)$$
$$= F^T(c) \qquad \text{by def. of } F^T$$
$$= K_T(c). \qquad \text{by def. of } K_T$$

And for any morphism $f \in \mathbf{C}_T[c,c']$,

$$K^T(f) = U_T(f) \qquad \text{by def. of } K^T$$
$$= \mu_{c'} \circ T(f) \qquad \text{by def. of } U_T$$
$$= \varepsilon^T_{(T(c'),\, \mu(c'))} \circ T(f) \qquad \text{by def. of } \varepsilon^T$$
$$= \varepsilon^T_{FT(c')} \circ F^T(f) \qquad \text{by def. of } F^T$$
$$= K_T(f)\ . \qquad \text{by def. of } K_T$$

Exercises

1. Prove that the comparison functor $K_T = K^T : \mathbf{C}_T \to \mathbf{C}^T$ is full and faithful.
2. Prove that the Kleisli Category is isomorphic to the full subcategory of $\mathbf{C}^T$ consisting of all free algebras.

5.5 More on Linear Logic

In this section, we complete an introductory presentation of linear logic and its categorical meaning, initiated in section 4.4. As already mentioned, the leading idea of this system refers to the "linear" use of "resources" or logical assumptions. From assuming A, one derives less than from assuming A, A, i.e., twice A . The logic-oriented reader, mostly used to classical or intuitionistic reasoning, may find this a little strange. Probably, though, this habit hides a nonconstructive view which may result in a limitation of our understanding of effective processes. Indeed, the alternative approach proposed by linear logic, which enriches and complements the traditional ones, seems to suggest formalizations and understanding of processes, such as parallel ones, which have so far escaped to a description by usual tools (see the examples on monoids and Petri nets in section 4.3 and the references, for recent developments of this idea).

As the reader may recall, the changes in the structural rules motivate a duplication of the connectives (see section 4.4). However, there is a way to recover the usual possibility, in classical as well as in intuitionistic logic, of an iterated use of assumptions. The idea is to introduce a connective "!" (read "of course"), which allows to assume as (finitely) many times one wishes a given assumption. This connective has a categorical meaning, which may be given in the terms of adjunctions and monads, following the previous section. The interesting categorical significance of this relation to classical and intuitionistic logic, as well as the categorical understanding we described in section 4.4, via structures such as the categories **Stab** and **Lin**, is probably what makes the difference between linear logic and previous formal experiments with the structural rules in other areas of logic.

The rules below are meant to extend the system in section 4.4. Observe that the structural rules of weakening and contractions apply with respect to the connective !. This is exactly what it is introduced for: it is meant to allow copies of assumptions and observe that they lead to the same consequences. This is expressed also by the rules (!,r) and (!,l) . Following Girard's work, we explicitly introduce the dual "?" (read "why not"), of the connective ! . Its operational behaviour is described by the rules, which mimic those for ! on the other side of the entailment (cf. the duality of $(-)^{\perp}$ in 4.4.3), and by the equivalence of !A and $(?(A^{\perp}))^{\perp}$, proved below.
The exponential fragment of Linear Logic is as follows:

5.5.1 exponential unary connectives: ! (of course), ? (why not)

5.5.2 exponential rules

$$\text{(weak-l)}\ \frac{\Gamma \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad \text{(weak-r)}\ \frac{\Gamma \vdash \Delta}{\Gamma \vdash ?A, \Delta}$$

$$\text{(contr-l)}\ \frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad \text{(contr-r)}\ \frac{\Gamma \vdash ?A, ?A, \Delta}{\Gamma \vdash ?A, \Delta}$$

$$\text{(!,l)}\ \frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad \text{(!,r)}\ \frac{!\Gamma \vdash A, ?\Delta}{!\Gamma \vdash !A, ?\Delta}$$

$$\text{(?,l)}\ \frac{!\Gamma, A \vdash ?\Delta}{!\Gamma, ?A \vdash ?\Delta} \qquad \text{(?,r)}\ \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash ?A, \Delta}$$

The duality between ! and ? is easily obtained by the following deductions:

$$\text{(!,l)}\ \frac{A \vdash A}{!A \vdash A} \qquad \text{(}\perp\text{,r)}\ \frac{A \vdash A}{\vdash A^{\perp}, A}$$

$$\text{(}\perp\text{,l)}\ \frac{!A \vdash A}{!A, A^{\perp} \vdash} \qquad \text{(?,r)}\ \frac{\vdash A^{\perp}, A}{\vdash ?A^{\perp}, A}$$

$$\text{(?,l)}\ \frac{!A, A^{\perp} \vdash}{!A, ?A^{\perp} \vdash} \qquad \text{(!,r)}\ \frac{\vdash ?A^{\perp}, A}{\vdash !A, ?A^{\perp}}$$

$$\text{(}\perp\text{,r)}\ \frac{!A, ?A^{\perp} \vdash}{!A \vdash (?A^{\perp})^{\perp}} \qquad \text{(}\perp\text{,l)}\ \frac{\vdash !A, ?A^{\perp}}{(?A^{\perp})^{\perp} \vdash !A}$$

Exercise Prove that $!(A\cap B) \vdash !A \otimes !B$ and $!A \otimes !B \vdash !(A\cap B)$.

5.5.3 Remark The connective "!" is exactly what is needed to recover the intuitionistic calculus: it is possible to prove that there is an embedding of intuitionistic (and classical) logic into linear logic. The embedding maps every intuitionistic formula A to a linear formula $\underline{A}$ in the following way:

$\underline{A} = A$ if A is an atomic formula

$\underline{A \wedge B} = \underline{A} \cap \underline{B}$

$\underline{A \vee B} = !\underline{A} \oplus !\underline{B}$

$\underline{A \Rightarrow B} = !\,\underline{A} \multimap \underline{B}$

(! is supposed to bind tighter than $\multimap$ and $\cup$).

The absurdum *F* of intuitionistic is translated into 0, the identity for $\oplus$. Thus $\underline{\sim A} = !\underline{A} \multimap 0$. In other words, the iterated use of the premises, in a linear implication, gives exactly the intuitionistic implication. Then, if, $\Gamma \vdash_i A$, in intuitionistic Logic, then $!\,\Gamma \vdash \underline{A}$ in Linear Logic.

Our aim now is to give categorical meaning to the connective !, of course. By duality, in linear categories (see definition 4.4.4), we also obtain an interpretation for ?, why not.

We have already remarked that a resource such as !A can be duplicated and erased, and in a sense these properties characterize the meaning of the connective ! . Thus, at the sematic level, we expect to have two morphisms $\delta: !A \to !A \otimes !A$, and $\varepsilon: !A \to 1$ where 1 is the identity of the monoidal category. (Commutative) comonoids in 4.3.4 seem the right structure for this, as they are characterized by a sort of diagonal map, such as δ, and a map ε which dualizes the map η in definitions 4.2.1 and 4.3.3.

We start then with a monoidal category **C**. By definition, **C** must satisfy certain natural isomorphisms, given in 4.3.1, which we rebaptize in this section, for convenience, with more suggestive names

1. assoc: $X \otimes (Y \otimes Z) \cong (X \otimes Y) \otimes Z$
2. ins-l: $X \cong 1 \otimes X$
3. exch: $X \otimes Y \cong Y \otimes X$

Let also:

4. ins-r = exch $\circ$ ins-l : $X \cong X \otimes 1$

As mentioned in section 4.4, the connective of linear implication $\multimap$ is interpreted by the right adjoint to the tensor product $\otimes$, when **C** is a monoidal closed category, as defined in section 4.3.

Recall that the category $\mathbf{CoMon}_{\mathbf{C}}$ of commutative comonoids over a monoidal category **C**, has as objects, for c in **C**, $(c, \delta: c \to c\otimes c, \varepsilon: c \to 1)$, and morphisms $f: (c, \delta, \varepsilon) \to (c', \delta', \varepsilon')$, for each arrow $f: c \to c'$ in **C** , such that

$\delta' \circ f = (f \otimes f) \circ \delta : c \to c' \otimes c'$,

$\varepsilon' \circ f = \varepsilon: c \to 1$.

Given a commutative comonoid $(c, \delta: c\to c\otimes c, \varepsilon: c\to 1)$ observe that the following equations hold:

$$(\varepsilon\otimes id_C)\circ\delta = ins\text{-}l : c\to 1\otimes c$$
$$exch\circ\delta = \delta : c\to c\otimes c$$
$$assoc\circ(id_C\otimes\delta)\circ\delta = (\delta\otimes id_C)\circ\delta : c\to(c\otimes c)\otimes c\ .$$

Exercise As pointed out after definition 4.3.6, if **C** is Cartesian, in the sense that $\otimes$ is actually a cartesian product $\cap$ and the isomorphisms are the canonical ones, then all the maps and isomorphisms above can be constructed for each object in **C** (in particular, recall that δ and ε are canonically given; namely, $\delta = \langle id,id\rangle : c\to c\cap c$ is the diagonal and $\varepsilon: c\to t$ is the unique map to the terminal object). Prove that, if **C** is Cartesian, then **C** is actually isomorphic to $\mathbf{CoMon_C}$. Does the converse hold?

5.5.4 Definition *A* ***! - model*** *is a linear category* ***C*** *and a comonad* $(!,D,E)$ *such that there exist natural isomorphisms*

$$I : !(A\cap B)\cong !(A)\otimes !(B)$$
$$J : !t\cong 1$$

where t *and* 1 *are the identities for* $\cap$ *and* $\otimes$*, the Cartesian and tensor products in* ***C****.*

Indeed, by definition, in a linear category one has both a monoidal and a Cartesian structure. The relation established by the natural isomorphisms gives the monoids we need.

5.5.5 Lemma *Let* $\langle \mathbf{C},(!,D,E)\rangle$ *be an !-model. Then, for each object* c *in* ***C****, there exist maps* $\delta': !c\to !c\otimes !c$ *and* $\varepsilon': !c\to 1$*, such that* $(!c, \delta', \varepsilon')$ *is a comonoid.*

Proof. Just set $\delta' = I\circ !\delta : !c\to !(c\cap c)\to !c\otimes !c$

$$\varepsilon' = I\circ !\varepsilon : !c\to !t\to 1$$

where $\delta = \langle id,id\rangle : c\to c\cap c$ and $\varepsilon: c\to t$ are the monoidal maps in the remark above, w.r.t. the Cartesian product $\cap$. The rest is easy.♦

Thus, the comonad $(!: \mathbf{C}\to\mathbf{C}, D: !\to !\circ !, E: !\to Id_{\mathbf{C}})$ associated with a !-model gives all the ingredients for the interpretation of the connective !, of course. In view of the above lemma, we can define the functor $\mathbf{!}: \mathbf{C}\to \mathbf{CoMon_C}$ by

$$\mathbf{!}(c) = (!c, \delta': !c\to !c\otimes !c, \varepsilon': !c\to 1)\ .$$

This gives the required monoids, while the natural transformations D and E uniformly yield maps $D_c : !c\to !\circ !c$ and $E_c : !c\to c$, which are needed to interpret the rules in (!,r) and (!,l).

We already mentioned in section 4.4 that the idea of the interpretation relies on viewing entailments as morphisms. In short, observe that, with an informal blend of syntax and semantics,

the rules	*are interpreted by*	*the fact that*
(weakenings)		each morphism $f: 1\to\Delta$ gives a morphism $f\circ\varepsilon_a: !a\to\Delta$
(contractions)		each morphism $f: !a\otimes!a\to\Delta$ gives a morphism $f\circ\delta_a: !a\to\Delta$
(!,l)		each morphism $f: a\to\Delta$ gives a morphism $f\circ E_a: !a\to\Delta$
(!,r)		each morphism $f: !c\to a$ gives a morphism $!f\circ D_c: !c\to!a$.

As for the rules which contain ?, their meaning is easily derivable by duality. The idea is to define a functor $? : \mathbf{C}\to\mathbf{C}$ by

$$? = * \circ ! \circ *$$

that is $?A = (!A^*)^*$. Then the following theorem gives the categorical meaning of the modality ?, why not.

5.5.6 Theorem *Let* $<\mathbf{C},(!,D,E)>$ *be an !-model. Then there exist a monad* $(?, D': ?\circ ?\to ?, E': Id\to ?)$ *and natural isomorphisms*

$$I': ?(A\oplus B) \cong ?(A)\cup ?(B)$$
$$J': ?0 \cong \bot ,$$

where 0 *and* $\bot$ *are the identities for* $\oplus$ *and* $\cup$, *the duals of the Cartesian and tensor products in* $\boldsymbol{C}$.

Proof. Set $? = * \circ ! \circ * : \mathbf{C} \to \mathbf{C}$ and, for each object A, $D'_A = (D_{A^*})^*$ and $E'_A = (E_{A^*})^*$. As $D: !\to!\circ!$, one has

$D_{A^*}: !A^*\to!\circ!A^*$
$(D_{A^*})^*: (!\circ!A^*)^*\to(!A^*)^*$ by def.of (-)*
$(D_{A^*})^*: (!\circ^*\circ^*\circ!A^*)^*\to(!A^*)^*$ by $Id \cong (-)^{**}$
$D'_A: ?\circ?A\to?A$

Each of these steps is an isomorphism, uniform in A, and gives a natural transformation $D': ?\circ?\to?$. Similarly, from $E: !\to Id_{\mathbf{C}}$ one has $E_{A^*}: !A^*\to A^*$ and, thus, $E'_A = (E_{A^*})^*: A\to?A$. The properties required for a monad follow by duality.

As for the natural isomorphisms, compute

$?(A\oplus B) = * \circ ! \circ *(A\oplus B)$
$= * \circ !(A^*\cap B^*)$ by theorem 4.4.6

$\cong (!(A^*)\otimes!(B^*))^*$ by def. of !-model

$= ?A \cup ?B$ by theorem 4.4.6.

Finally, $?0 = * \circ ! \circ *0 \cong * \circ ! \, t \cong 1^* \cong \bot$, by definition and theorem 4.4.6. ♦

Exercise Endow a structure of monoid over each object in a monoidal category whose tensor product is actually a Cartesian coproduct. Then give the details of the interpretation of the rules for ?.

Next we find, within any categorical model of linear logic, an interpretation for the intuitionistic connectives $\cap$ and $\Rightarrow$, by using the comonad construction in the !-model. Namely, given an !-model **C**, one may interpret intuitionistic "and" and "implication" by Cartesian product and exponential in a suitable category derived from **C**. As the purpose of the iterator ! was to take us back to intuitionistic logic, we use its categorical meaning to construct this new category.

As a matter of fact, in the remark 5.5.3, we hinted how to derive intuitionistic connectives from linear ones, once the connective ! is available. The following result gives the categorical counterpart of that construction.

Observe that in general, given a comonad (T, δ, ε) over **C**, the co-Kleisli category **K** is the category whose objects are those of **C**, and the set **K**[A,B] of morphisms from A to B in **K** is **C**[T(A),B]. The identity in **K**[A,A] is ε_A: T(A)→A. The composition of f∈**K**[A,B] and g∈**K**[B,C] in **K** is

$$g \circledcirc f = g \circ T(f) \circ \delta_A : T(A) \to T^2(A) \to T(B) \to C$$

(see definition 4.2.4 where Kleisli categories over monads were defined).

5.5.7 Theorem *If **C** be an !-model. Then the co-Kleisli category **K** associated with the comonad (!,D,E) is Cartesian closed.*

Proof (*hint*) The exponent of two objects B and C is (!B—oC). We then have the following chain of isomorphisms:

$\mathbf{K}[A \cap B, C] \cong \mathbf{C}[!(A \cap B), C]$ by definition of **K**

$\cong \mathbf{C}[!(A) \otimes !(B), C]$ as $!(A \cap B) \cong !(A) \otimes !(B)$

$\cong \mathbf{C}[!(A), !B \multimap C]$ as **C** is monoidal closed

$\cong \mathbf{K}[A, !B \multimap C]$ by definition of **K**. ♦

Example In section 2.4.2 we defined the category **Stab** of coherent domains and stable functions. In that section (see exercise 4) the subcategory **Lin**, with linear maps, was also introduced and, later (see section 4.4), it was given as an example of linear category. We also defined a function ! on coherent domains as follows: if X is a coherent domain, then !X is the coherent domain defined by:

i. $|!X| = \{a \,/\, a \in X, a \text{ finite}\}$;

ii. $a \uparrow b \,[\text{mod } !X]$ iff $a \cup b \in X$.

We need now to extend it to a functor ! : **Stab**→**Lin**. Recall that a linear map g: Z→Z' is uniquely determined by its behavior on the points of the coherent domain Z, i.e., on the elements of |Z|. Moreover, any stable map may be equivalently described in terms of its trace. Set then, for each stable map f : X→Y,

Tr(!f) = {({a}, b) | b∈Y, b finite, a∈X, a finite and least such b ⊆ f(a) }.

Next, we define an adjunction between ! and the obvious inclusion functor **Inc** from **Lin** into **Stab**. This is given by a natural isomorphism

(iso) φ : **Lin**[!A,B] ≅ **Stab**[A,B]

where the inclusion functor is omitted.

Once more we use traces, that is, for each g∈**Lin**[!A,B] set

Tr(φ(g)) = {(a, y) | ({a}, y)∈Tr(g) } .

The reader may prove for exercise the naturality of φ. In particular, the unit and counit of the adjunction are given, as usual, by

$\eta_A = \varphi(id_{!A})$: A→!A where Tr(η_A) = {(a, a) | a∈A finite }

$\varepsilon_A = \varphi^{-1}(id_A)$: !A→A where Tr(ε_A) = {({x}, x) | x∈|A| }.

Exercise Check, by actual computations in the structure, that !f = $\varphi^{-1}(\eta_A \circ f)$ and f = $\varphi(\varepsilon_A \circ !f)$.

Following theorem 5.4.1, (!, Inc, η, ε) yields a comonad

(! = !∘Inc: **Lin**→**Lin**, D = !η**Inc**: !→!∘!, E = ε: !→$Id_{\mathbf{C}}$)

as required to turn **Lin** into an !-model. Moreover, it is a matter of a simple observation on the "hardware" of coherent domains to show that the isomorphisms needed to complete the definition hold in **Lin**, namely, that !(A∩B) ≅ !(A)⊗!(B) and !t ≅ 1 are uniformly valid in this model (see the example in section 4.4).

Interestingly enough, by (iso) above, **Stab** is the co-Kleisli category associated with the comonad (!, D, E) on **Lin**.

We conclude this section by identifying a class of categories which yield an interesting interpretation of the modality ! . The idea is to interpret !A as the commutative comonoid freely cogenerated by A, not just as a comonoid in the intended linear category.

5.5.8 Definition *Let* ***C*** *be a linear category and* ***U***: $\mathbf{CoMon}_{\mathbf{C}}$→***C*** *be the forgetful functor which takes (c, δ, ε) to c. Then* ***C*** *is a* ***free !-model*** *if there exists a right adjoint to* ***U****, that is a functor* ***!*** : ***C***→$\mathbf{CoMon}_{\mathbf{C}}$ *and a natural isomorphism* Ω: ***C***[c,a] ≅ $\mathbf{CoMon}_{\mathbf{C}}$[(c,δ,ε), ***!***(a)] .

We need to show that free !-models are indeed !-models. This follows from the simple, but powerful, adjointness property stated in 5.5.8. As already recalled, by proposition 5.4.1, each

adjunction yields a comonad. We explicitly reconstruct the units and counits as they bear some information.

5.5.9 Lemma *Let* $\mathbf{C}$ *be a free !-model and* $\langle \mathbf{U},!,\Omega \rangle$ *be the given adjunction. Then, for* $! = \mathbf{U} \circ \mathbf{!}$*, there exist natural transformations* $D: ! \to ! \circ !$ *and* $E: ! \to Id_{\mathbf{C}}$ *such that*

$$(!: \mathbf{C} \to \mathbf{C},\ D: ! \to ! \circ !,\ E: ! \to Id_{\mathbf{C}})$$

is the comonad associated with $\mathbf{C}$*, in the sense of proposition 5.4.1.*

Proof By the definition of morphism in $\mathbf{CoMon}_{\mathbf{C}}$, for every $h \in \mathbf{C}[c,a]$, the morphism $\Omega(h) \in \mathbf{CoMon}_{\mathbf{C}}[(c,\delta,\varepsilon),\mathbf{!}(a)]$ satisfies the following equations:

hom-1. $\delta_a \circ \Omega(h) = (\Omega(h) \otimes \Omega(h)) \circ \delta : c \to !a \otimes !a$;

hom-2. $\varepsilon_a \circ \Omega(h) = \varepsilon : c \to 1$.

Moreover, the naturality of Ω is expressed by the following equations:

for every $h \in \mathbf{C}[c,a]$, $f \in \mathbf{C}[a,b]$, $g \in \mathbf{CoMon}_{\mathbf{C}}[(c',\delta',\varepsilon'), (c,\delta,\varepsilon)]$:

nat-1. $\Omega(f \circ h) = \mathbf{!}(f) \circ \Omega(h)$;

nat-2. $\Omega(h \circ \mathbf{U}(g)) = \Omega(h) \circ g$.

The counits of the adjunction $(\Omega, \mathbf{U}, \mathbf{!}): \mathbf{CoMon}_{\mathbf{C}} \to \mathbf{C}$, are arrows $E_c = \Omega^{-1}(id_{\mathbf{!}(c)}): !c \to c$. By equation (nat-1) above, for $h = E_c$, we obtain $\mathbf{!}(f) = \Omega(f \circ E_c)$, and by equation (nat-2), $E_c \circ \mathbf{U}(\Omega(h)) = h$. The family of arrows $\{E_c\}_{c \in \mathbf{C}}$ defines a natural transformation $E: (\mathbf{U} \circ \mathbf{!}) \to \mathbf{I}$. Dually the units of the adjunction define a natural transformation $H : \mathbf{I} \to (\mathbf{!} \circ \mathbf{U})$, where: $H_{(c,\delta,\varepsilon)} = \Omega(id_c) : (c,\delta,\varepsilon) \to !c$. The adjunction between $\mathbf{CoMon}_{\mathbf{C}}$ and $\mathbf{C}$ is thus equivalently expressed by the parameters $(\mathbf{U}, \mathbf{!}, H: \mathbf{I} \to (\mathbf{!} \circ \mathbf{U}), E: (\mathbf{U} \circ \mathbf{!}) \to \mathbf{I})$.

Remember now that a comonad over a category $\mathbf{C}$ is a comonoid in the category of endofunctors from $\mathbf{C}$ to $\mathbf{C}$ (with composition as product, see 4.2.2).

By proposition 5.4.1, every adjunction $(F, G, \eta: Id_{\mathbf{C}} \to G \circ F, \varepsilon: F \circ G \to Id_{\mathbf{C}'})$ from $\mathbf{C}$ to $\mathbf{C}'$ determines a comonad $(T = F \circ G, \delta = F\eta G: T \to T \circ T, \varepsilon : T \to Id_{\mathbf{C}'})$ over $\mathbf{C}'$.

In particular the adjunction $(\mathbf{U}, \mathbf{!}, H: \mathbf{I} \to (\mathbf{!} \circ \mathbf{U}), E: (\mathbf{U} \circ \mathbf{!}) \to \mathbf{I})$: $\mathbf{CoMon}_{\mathbf{C}} \to \mathbf{C}$, defines a comonad $(! = \mathbf{U} \circ \mathbf{!} , D = \mathbf{U}H\mathbf{!} : ! \to ! \circ ! , E : ! \to Id_{\mathbf{C}})$ over the !-model $\mathbf{C}$. ♦

Finally we derive the natural isomorphisms in definition 5.5.4.

5.5.10 Theorem *Let* $\mathbf{C}$ *be a free !-model and* $(!: \mathbf{C} \to \mathbf{C},\ D: ! \to ! \circ !,\ E: ! \to Id_{\mathbf{C}})$ *be the comonad associated with it by the lemma. Then there exist natural isomorphisms*

$$I: !(A \cap B) \cong !(A) \otimes !(B)$$

$$J: !t \cong 1$$

where t *and* 1 *are the identities for* $\cap$ *and* $\otimes$*, the Cartesian and tensor products in* $\mathbf{C}$*.*

Proof Consider the comonoid $(!(A) \otimes !(B), \delta, \varepsilon)$ where

$$\delta = \mathrm{mix} \circ (\delta_A \otimes \delta_B) : !(A) \otimes !(B) \to (!(A) \otimes !(B)) \otimes (!(A) \otimes !(B))$$

$$\varepsilon = \text{ins-r}^{-1} \circ (\varepsilon_A \otimes \varepsilon_B) : !(A) \otimes !(B) \to 1$$

and

$$\text{mix} : (!(A) \otimes !(A)) \otimes (!(B) \otimes !(B)) \to (!(A) \otimes !(B)) \otimes (!(A) \otimes !(B))$$

is the obvious isomorphism.

Then, by hypothesis, we have an isomorphism

$$\Omega: \mathbf{C}[!(A) \otimes !(B), A \cap B] \cong \mathbf{CoMon}_{\mathbf{C}}[\ (!(A) \otimes !(B), \delta, \varepsilon),\ !(A \cap B)\ ]$$

The isomorphism $I_{A,B}$ from $!(A \cap B)$ to $!(A) \otimes !(B)$, which we write I for short, is given by

$$I = (!\text{fst} \otimes !\text{snd}) \circ \delta_{A \cap B} : !(A \cap B) \to !(A) \otimes !(B)$$

Note that I is a morphism of comonoids, that is, as it is easily verified,

$$\delta \circ I = (I \otimes I) \circ \delta_{A \cap B}$$

$$\varepsilon \circ I = \varepsilon_{A \cap B}$$

The inverse image of I is defined in the following way.

Let

$$k_1 = E_A \circ \text{ins-r}^{-1} \circ (\text{id}_{!A} \otimes \varepsilon_B) : !(A) \otimes !(B) \to A$$

$$k_2 = E_B \circ \text{ins-l}^{-1} \circ (\varepsilon_A \otimes \text{id}_B) : !(A) \otimes !(B) \to B$$

and

$$k = \langle k_1, k_2 \rangle : !(A) \otimes !(B) \to A \cap B$$

Then the inverse image of I is $\mathbf{U}(\Omega(k)) = \Omega(k) : !(A) \otimes !(B) \to !(A \cap B)$, indeed:

$\Omega(k) \circ I =$

$= \Omega(k \circ I)$ by (nat-2)

$= \Omega(\langle k_1, k_2 \rangle \circ I)$ by def. of k

$= \Omega(\langle k_1 \circ I, k_2 \circ I \rangle)$

$= \Omega(\langle E_A \circ \text{ins-r}^{-1} \circ (\text{id}_{!A} \otimes \varepsilon_B) \circ (!\text{fst} \otimes !\text{snd}) \circ \delta_{A \cap B},\ E_B \circ \text{ins-l}^{-1} \circ (\varepsilon_A \otimes \text{id}_B) \circ (!\text{fst} \otimes !\text{snd}) \circ \delta_{A \cap B} \rangle)$

by def. of k_1, k_2 and I

$= \Omega(\langle E_A \circ \text{ins-r}^{-1} \circ (\text{id}_{!A} \otimes \varepsilon_A) \circ (!\text{fst} \otimes !\text{fst}) \circ \delta_{A \cap B},\ E_B \circ \text{ins-l}^{-1} \circ (\varepsilon_B \otimes \text{id}_B) \circ (!\text{snd} \otimes !\text{snd}) \circ \delta_{A \cap B} \rangle)$

as $\varepsilon_B \circ !\text{snd} = \varepsilon_{A \cap B} = \varepsilon_A \circ !\text{fst}$

$= \Omega(\langle E_A \circ \text{ins-r}^{-1} \circ (\text{id}_{!A} \otimes \varepsilon_A) \circ \delta_A \circ !\text{fst},\ E_B \circ \text{ins-l}^{-1} \circ (\varepsilon_B \otimes \text{id}_B) \circ \delta_B \circ !\text{snd} \rangle)$

as !fst and !snd are comonoid morphisms

$= \Omega(\langle E_A \circ !\text{fst},\ E_B \circ !\text{snd} \rangle)$ by properties of comonoids

$= \Omega(\langle \text{fst} \circ E_{A \cap B},\ \text{snd} \circ E_{A \cap B} \rangle)$ by naturality of E

$= \Omega(E_{A \cap B})$

$= \text{id}_{!(A \cap B)}$. by def. of E

$I \circ \Omega(k) =$

$= (!\text{fst} \otimes !\text{snd}) \circ \delta_{A \cap B} \circ \Omega(k)$

$= (!\text{fst} \otimes !\text{snd}) \circ (\Omega(k) \otimes \Omega(k)) \circ \delta$ by (hom-1)

$= (!\text{fst} \circ \Omega(k)) \otimes (!\text{snd} \circ \Omega(k)) \circ \delta$

$= (\Omega(\mathrm{fst} \circ k)) \otimes (\Omega(\mathrm{fst} \circ k)) \circ \delta$ by (nat-1)

$= (\Omega(k_1)) \otimes (\Omega(k_2)) \circ \delta$ by def. of k

$= (\Omega(E_A \circ \mathrm{ins\text{-}r}^{-1} \circ (\mathrm{id}_{!A} \otimes \varepsilon_B)) \otimes (\Omega(E_B \circ \mathrm{ins\text{-}l}^{-1} \circ (\varepsilon_A \otimes \mathrm{id}_B)) \circ \delta$ by def. of k_1 and k_2

$= (\mathrm{ins\text{-}r}^{-1} \circ (\mathrm{id}_{!A} \otimes \varepsilon_B)) \otimes (\mathrm{ins\text{-}l}^{-1} \circ (\varepsilon_A \otimes \mathrm{id}_B)) \circ \delta$ by (nat-2) and def. of E

$= (\mathrm{ins\text{-}r}^{-1} \circ (\mathrm{id}_{!A} \otimes \varepsilon_B)) \otimes (\mathrm{ins\text{-}l}^{-1} \circ (\varepsilon_A \otimes \mathrm{id}_B)) \circ \mathrm{mix} \circ (\delta_A \otimes \delta_B)$ by def. of δ

$= (\mathrm{ins\text{-}r}^{-1} \circ (\mathrm{id}_{!A} \otimes \varepsilon_A)) \otimes (\mathrm{ins\text{-}l}^{-1} \circ (\varepsilon_B \otimes \mathrm{id}_B)) \circ (\delta_A \otimes \delta_B)$ by application of mix

$= (\mathrm{ins\text{-}r}^{-1} \circ (\mathrm{id}_{!A} \otimes \varepsilon_A) \circ \delta_A) \otimes (\mathrm{ins\text{-}l}^{-1} \circ (\varepsilon_B \otimes \mathrm{id}_B) \circ \delta_B)$ by properties of comonoids

$= \mathrm{id}_{!(A) \otimes !(B)}$

$= \mathrm{id}_{!(A) \otimes !(B)}$.

The construction is clearly uniform in A and B.

As for the natural isomorphism J, note that $!A \cong !(A \cap t) \cong !A \otimes !t$, $!A \cong !(t \cap A) \cong !t \otimes !A$ and that the right and left identity, in a monoidal category, are unique. ♦

References Universal arrows and adjunctions are fundamental notions in Category Theory. Their treatment, in various forms, and references to their origin may be found in all textbooks we mentioned in the previous chapters. References for Linear Logic have been given in chapter 4.

Chapter 6

CONES AND LIMITS

In chapter 2, we learned how common constructions can be defined in the language of Category Theory by means of equations between arrows of given objects. In chapter 4, we saw that those definitions were based on the existence of an universal arrow to a given functor. The category-theoretic notion of limit is merely a generalization of those particular constructions, as it stresses their common universal character. From another point of view, the limit is a particular and important case of universal arrow, where the involved functor is a "diagonal," or "constant" functor, as we shall see. To help the reader become confident with this new notion, we begin this chapter by looking back at the constructions of chapter 2 and we regard them as particular instances of limits. Then we study some relevant properties concerning existence, creation, and preservation of limits. As for computer science, limits have been brought to the limelight mainly by the semantic investigation of recursive definition of data types: this particular application of the material in this chapter will be discussed in chapter 10.

6.1 Limits and Colimits

The concept of limit embodies the general idea of universal construction, that is, of an entity which has a privileged behavior amongt a class of objects that satisfy a certain property. The only way to define a property in the categorical language is by specifying the existence and equality of certain arrows, that is, essentially by imposing the existence of a particular commutative diagram amongt objects inside the category.

6.1.1 Definition *A **diagram** D in a category $\mathbf{C}$ is a directed graph whose vertices $i \in I$ are labeled by objects d_i and whose edges $e \in E$ are labeled by morphisms f_e.*

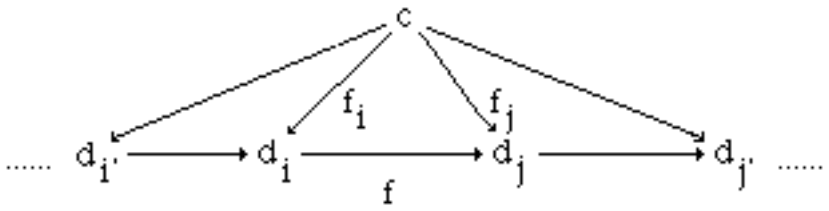
A diagram D in $\mathbf{C}$ is similar to a subcategory of $\mathbf{C}$; however, it does not need to contain identities, nor must it be closed under composition of morphisms.

More formally, a diagram in a category $\mathbf{C}$ should be defined as a graph homomorphism D from an index graph I to the (graph underlyng the) category $\mathbf{C}$. Such a diagram is called "of type I". For the adjunction between graphs and categories, this is exactly the same as a functor from the category $\mathbf{I}$ freely generated by the graph I (the index category) to $\mathbf{C}$. A graph is called small when the index category is small.

6.1.2 Definition. *Let* $\mathbf{C}$ *be a category and* D *a diagram with objects* d_i, $i \in I$. *Then a* ***cone to*** D *is an object* c *and a family of morphisms* $\{f_i \in \mathbf{C}[c,d_i] \mid i \in I\}$ *such that*

$$\forall i,j \in I \;\; \forall e \in E \;\; f_e \in \mathbf{C}[d_i,d_j] \;\; \Rightarrow \;\; f_e \circ f_i = f_j \, .$$

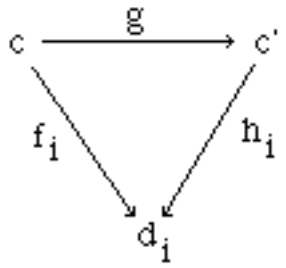
A cone may be visualized by



Cocones are defined dually.

Example In a partial order **P**, cones correspond to lower bounds, cocones to upper bounds.

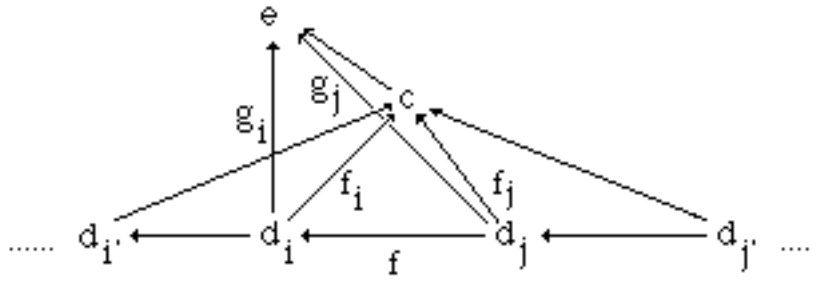
Note now that, given a diagram D, the cones to D form a category, call it $\mathbf{Cones}_{\mathbf{C},\mathbf{D}}$. Just take as morphisms from $(c,\{f_i \in \mathbf{C}[c,d_i] \mid i \in I\})$ to $(c',\{h_i \in \mathbf{C}[c',d_i] \mid i \in I\})$ any $g \in \mathbf{C}[c,c']$ such that $\forall i \in I \;\; h_i \circ g = f_i$. That is,



Clearly, $\mathbf{Cones}_{\mathbf{C},\mathbf{D}}$ is a category. Dually one defines the category $\mathbf{Cocones}_{\mathbf{C},\mathbf{D}}$.

6.1.3 Definition. *Let* $\mathbf{C}$ *be a category and* D *a diagram. Then a* ***limit*** *for* D *is a terminal object in* $\boldsymbol{Cones_{C,D}}$. ***Colimits*** *are defined dually.*

$(c,\{f_i \in \mathbf{C}[d_i,c] \mid i \in I\})$ is the initial object in $\mathbf{Cocones}_{\mathbf{C},\mathbf{D}}$, it may be visualized by the following commutative diagram:



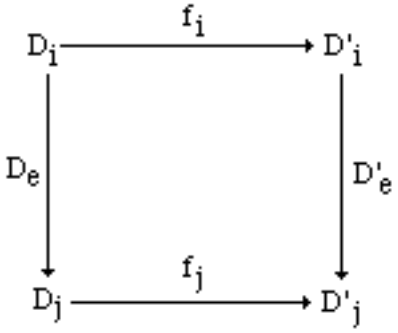
Limits are also called universal cones, as any other cone uniquely factorizes via them. Dually, colimits are called universal cocones.

Examples

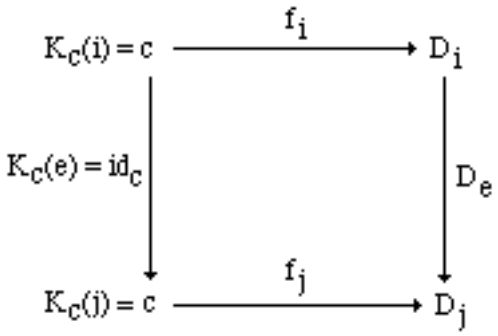
1. Let **P** be a partial order. Then limits correspond to greater lower bounds, while colimits correspond to least upper bounds.
2. Let $D = (\{d_i\}_{i\in\omega}, \{f_i \in \mathbf{Set}[d_i, d_{i+1}]\}_{i\in\omega})$ be a diagram in **Set** such that $d_i \subseteq d_{i+1}$, and f_i = incl (the set-theoretic inclusion). Then the colimit of $d_0 \rightarrow \ldots\ldots d_i \rightarrow d_{i+1} \rightarrow \ldots\ldots$ is $\cup\{d_i\}$ (**exercise**: what is the limit of the same diagram?) .

Exercise Prove that the colimits in **C** are the limits in $\mathbf{C}^{\mathbf{op}}$ of the dual diagram.

Consider now a diagram as a functor from an index category **I** to **C**. Note first that any object c of the category **C** is the image of a constant functor K_c: **I**→**C**, and so K_c can be regarded as a degenerate diagram of type I in **C**. Once diagrams are defined as functors, it makes sense to consider natural transformations between diagrams. If D and D' are two diagrams of type I, a natural transformation from D to D' is a family of arrows f_i indexed on objects in **I** such that for each arrow e in **I** (each edge of the graph of type I)



A cone for a diagram D of type I from an object c is then a natural transformation from the constant diagram K_c to D

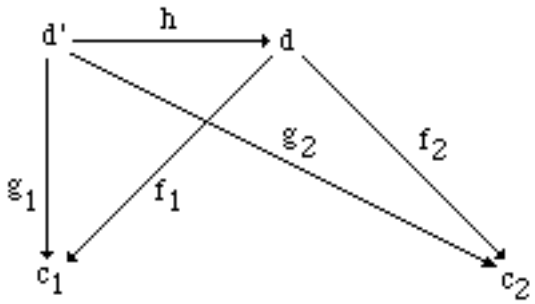


Dually, a cocone for a diagram D of type I to an object c is a natural transformation from D to the constant diagram K_c.

6.2 Some Constructions Revisited

Let D be an empty diagram, that is a diagram with no objects and no arrows. By definition, a cone in **C** to D is then just an object c of **C**, with no other structure (and every object of **C** can be seen as a cone). A limit for the empty diagram is then an object t such that for any other object c there is exactly one arrow from c to t, i.e., it is a **terminal** object. Dually, the **initial** object is the colimit of the empty diagram.

A graph is called **discrete** if it has no arrows. For example the set {1,2} can be regarded as a discrete graph. A diagram of type {1,2} in a category **C** is an ordered pair of objects, (c_1,c_2). A limit for such a diagram is an object d, together with two arrows $f_1: d \to c_1$ and $f_2: d \to c_2$, such that for any other cone $(d',\{g_i \in \mathbf{C}[d',c_i] \mid i \in \{1,2\}\})$ there exists exactly one arrow $h: d' \to d$, with $f_i \circ h = g_i$ for $i \in \{1,2\}$.

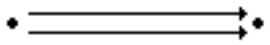


But this is just the definition of **product** d of c_1 and c_2 with $f_1: d \to c_1$ and $f_2: d \to c_2$ as projections.

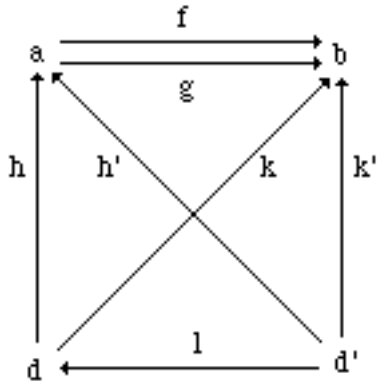
Dually, the **coproduct** $c_i \# c_j$, if it exists, is just the the colimit of the diagram $\{c_i,c_j\}$.

The product of any indexed collection of objects in a category is defined analogously as the limit of the diagram $D: I \to \mathbf{C}$ where I is the index set considered as a discrete graph. This product is usually denoted by $\Pi_{i \in I} D_i$, although explicit mention of the index set is often omitted.

Consider the graph I with two vertices and two edges



A diagram of type I in a category **C** is a pair of objects, a and b, and a pair of parallel arrows $f,g \in \mathbf{C}[a,b]$. A cone for this diagram consists of an object d, and two arrows $h \in \mathbf{C}[d,a]$ and $k \in \mathbf{C}[d,b]$ such that $g \circ h = k$ and $f \circ h = k$. A limit is a cone $(d,\{h,k\})$ that is universal, that is, for any other cone $(d',\{h',k'\})$ there exists exactly one arrow $l: d' \to d$ such that $h \circ l = h'$, and $k \circ l = k'$.

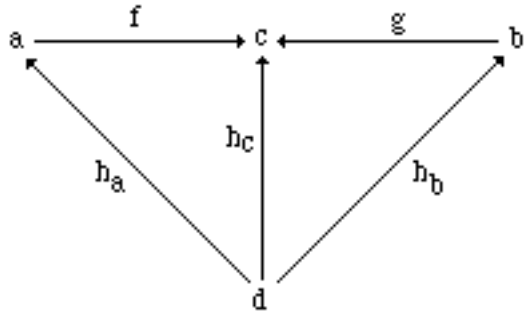


Note now that the existence of two arrows, h and k, such that $g \circ h = k$ and $f \circ h = k$, is equivalent to the existence of an arrow h such that $g \circ h = f \circ h$. Moreover, $h \circ l = h'$ implies $k \circ l = k'$, since $k \circ l = f \circ h \circ l = f \circ h' = k'$, thus the above limit is just the **equalizer** of f and g.

Dually, **coequalizers** are the colimits for the same diagram.

Consider now the following graph:

A diagram of this type in a category **C** is a given by three objects, a, c, and b, and two morphisms, $f \in \mathbf{C}[a,c]$ and $g \in \mathbf{C}[b,c]$. A cone to this diagram is an object d, together with three morphisms $h_a \in \mathbf{C}[d,a]$, $h_c \in \mathbf{C}[d,c]$ and $h_b \in \mathbf{C}[d,b]$, such that the following diagram commutes:



A cone $(d, \{h_a, h_b, h_c\})$ is a limit, if for any other cone $(d', \{h'_a, h'_b, h'_c\})$ there exists a unique arrow $k: d' \to d$ such that $h'_i = h_i \circ k$, for $i \in \{a,b,c\}$.

The commutativity of the previous diagram implies that $f \circ h_a = g \circ h_b$; conversely, given two arrows h_a and h_b such that $f \circ h_a = g \circ h_b$, one obtains a cone by defining $h_c = f \circ h_a = g \circ h_b$. Thus, the diagram for the cone $(d, \{h_a, h_b, h_c\})$ is equivalently expressed by giving only the outer commutative "square", i.e., by giving $(d, \{h_a, h_b\})$. In conclusion, a universal cone for a diagram of this type turns out to be just a **pullback**.

As usual, by taking the colimit of the same diagram we obtain the dual notion of **pushout**.

6.3 Existence of limits

In this secton, we study the important question about the existence of limits in a given category. Starting with the familiar category of sets, we generalise a common construction that allows the existence of complex limits to be states, provided that simpler ones exist.

Note first that every diagram D has limit in **Set**. It is obtained as follows.

Let $\{D_i\}_{i\in I}$ be a family of objects in D and consider the object $\Pi_{i\in I}D_i$, i.e., the product indexed by I. The elements of $\Pi_{i\in I}D_i$ are tuples $\{x_0, x_1, x_2, ...\}$ such that $x_i\in D_i$, for all $i\in I$, or equivalently functions $f : I \to \cup_{i\in I}D_i$, such that $f(i)\in D_i$.

$\Pi_{i\in I}D_i$ has projections $p_i : \Pi_{i\in I}D_i \to D_i$ for all $i\in I$, defined by $p_i(\{x_0,x_1,x_2, ...\}) = x_i$. In general these projections do not form a cone on D, that is, if $f_e: D_i\to D_j$ is an edge of D, one may have $p_j \neq f_e\circ p_i$. The idea is to take the subset L of $\Pi_{i\in I}D_i$ of all the tuples that satisfy the condition $p_j = f_e\circ p_i$. That is, $\{x_0, x_1, x_2, ...\}\in L$ if and only if, for all edges $f_e: D_i\to D_j$, one has $x_j = f_e(x_i)$. Let then γ_i be the projection p_i restricted to L. Then $(L,\{ \gamma_i \in \mathbf{C}[L,D_i] \mid i \in I\})$ is the limit (prove it as an exercise).

This set-theoretic construction is better formalized in Category Theory in the following way.

Let $\Pi(D_j / \exists i\in I\ \exists e\in E\ f_e: D_i\to D_j)$ be the product of all codomains of edges in D, with projections $\pi_j: \Pi(D_j / \exists i\in I\ \exists e\in E\ f_e: D_i\to D_j) \to D_j$. By definition of product, there is a unique function

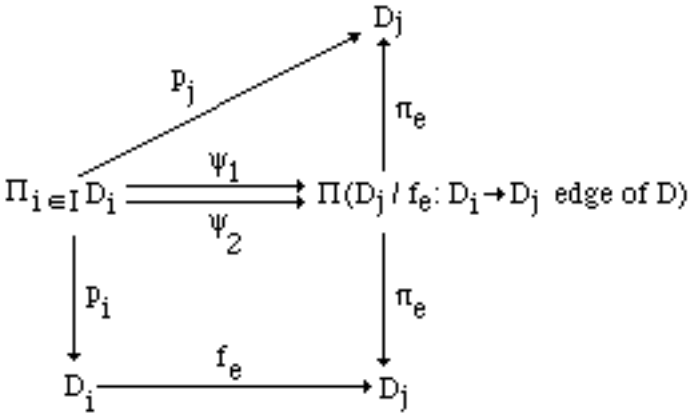
$$\psi_1: \Pi_{i\in I}D_i \to \Pi(D_j / \exists i\in I\ \exists e\in E\ f_e: D_i\to D_j)$$

such that $p_j = \pi_j\circ\psi_1$ for any edge $f_e: D_i\to D_j$ of D. Analogously there is a unique function

$$\psi_2: \Pi_{i\in I}D_i \to \Pi(D_j / \exists i\in I\ \exists e\in E\ f_e: D_i\to D_j)$$

such that $f_e\circ p_i = \pi_j\circ\psi_2$ for any edge $f_e: D_i\to D_j$ of D.

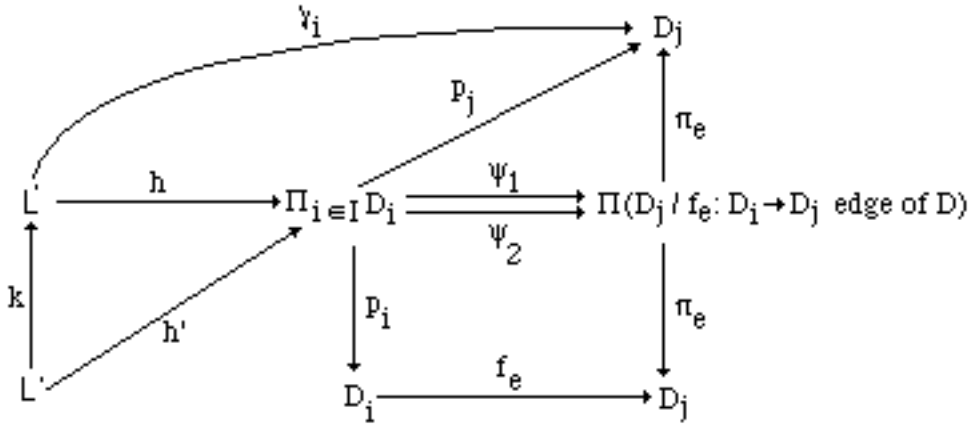
This is visualized in the following diagram:



Note now that, in set-theoretic terms, for all the tuples $\{x_0, x_1, x_2, ...\}$ in $\Pi_{i\in I}D_i$ the following properties are equivalent:

1. for all edges $f_e: D_i\to D_j$, $x_j = f_e(x_i)$
2. $\psi_1(\{x_0, x_1, x_2, ...\}) = \psi_2(\{x_0, x_1, x_2, ...\})$

Then, what we are looking for is the maximal subset L of $\Pi_{i\in I}D_i$ whose elements satisfy (2), but we aleady know that this is none other than the equalizer of ψ_1 and ψ_2. By a diagram,



We are now ready to generalize to every category **C** the previous construction of limits in **Set.**

6.3.1 Theorem *Let D be a diagram in **C** with sets I of vertices and E of edges. If every I-indexed family and every E-indexed family of objects has a product, and every pair of morphisms has an equalizer, then D has a limit.*
Proof Exercise (use the previous diagrams). ♦

6.3.2 Corollary *If a category **C** has arbitrary products, and equalizers for every pair of morphisms, then every diagram has a limit.*

6.3.3 Corollary *If a category **C** has all finite products, and coequalizers for every pair of morphisms, then every finite diagram has a limit.*

The relevance of theorem 6.3.1 is that, in general, it is simpler to check the existence of products and equalizers than to prove directly the existence of limits.

Example Corollary 6.3.2 may be used to prove that every diagram has a limit in **CPO**. If $\{C_i\}_{i\in I}$ is a family of c.p.o.'s, let $\Pi_{i\in I}C_i$ be the product indexed by I. $\Pi_{i\in I}C_i$ may be given a c.p.o. structure by the componentwise order, that is, $(c_i)_{i\in I} \le (d_i)_{i\in I}$ iff $\forall\, i\in I\ \ c_i \le d_i$. The projections p_i: $\Pi_{i\in I}(C_i)\to C_i$ are defined by $p_i((c_i)_{i\in I}) = c_i$. It is easy to prove that $\Pi_{i\in I}(C_i)$ is indeed a cpo, that the projections are continuous, and that $\Pi_{i\in I}(C_i)$ satisfies the universal property of the product.

Given $f,g : A\to B$, their equalizer is $h: A'\to A$, where $A' = \{a\in A \,/\, f(a) = g(a)\}$ with the ordering inherited by A, and h is the injection. A' is a c.p.o. Indeed, let D be a direct subset of

A'; then D is also a direct subset of A, and thus $f(\cup D)= \cup_{a\in D} f(a) = \cup_{a\in D} g(a) = g(\cup D)$. By this, $\cup D \in A'$. The continuity of h and the universal property for equalizers are easy to prove.

In propositions 2.5.5 and 2.5.6 we showed how to define products and equalizers from terminal objects and pullbacks. This suggests an even simpler sufficient (and necessary) condition for the existence of all finite limits.

6.3.4 Corollary *If **C** has a terminal objects and pullbacks for every pair of morphisms, then it has all finite limits.*

Exercise State the dual versions of theorem 6.3.1 and corollaries 6.3.1 to 6.3.4.

6.4 Preservation and Creation of Limits

In this section we study some cases of functors which "preserve" the property of objects to be limits of a diagram.

6.4.1 Definition *Let $G: \mathbf{A}\to\mathbf{X}$ be a functor, and let $(a,\{\tau_i \in \mathbf{A}[a,d_i] \mid i\in I\})$ be an universal cone from a on the diagram D in **A**. We then say that G **preserves the limit** $(a,\{\tau_i \in A[a,d_i] \mid i\in I\})$ if and only if $(Ga,\{G\tau_i \in \mathbf{X}[Ga,Gd_i] \mid i\in I\})$ is an universal cone from Ga on the diagram G(D) in X . **Preservation of colimits** is defined dually.*

6.4.2 Theorem *If the functor $G: \mathbf{A}\to\mathbf{X}$ has a left adjoint $F: \mathbf{X}\to\mathbf{A}$, and the diagram $D = (\{d_i\}_{i\in I}, \{f_e\}_{e\in E})$ in **A** has limit $(a, \{\tau_i\in\mathbf{A}[a,d_i] \mid i\in I\})$, then $G(D) = (\{Gd_i\}_{i\in I}, \{Gf_e\}_{e\in E})$ has a limit in **X**, and the limit is $(Ga,\{G\tau_i\in\mathbf{X}[Ga,Gd_i] \mid i\in I\})$.*

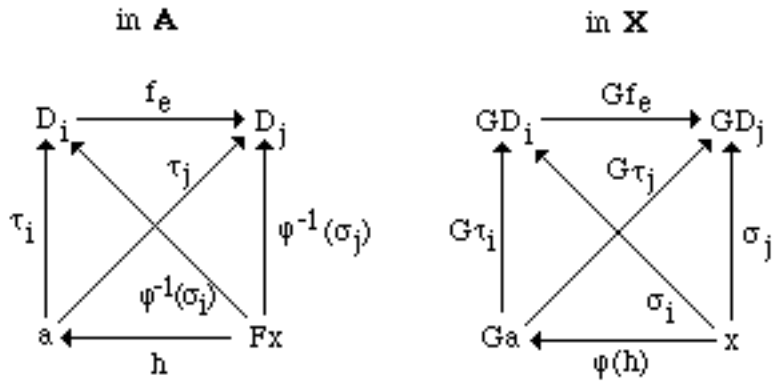
Proof By the properties of functors, $(\{Gd_i\}_{i\in I}, \{Gf_e\}_{e\in E})$ is a cone; we only need to prove that it is universal. Let $(x,\{\sigma_i\in\mathbf{X}[x,Gd_i] \mid i\in I\})$ be another cone, and let φ: $\mathbf{A}[Fx,d_i] \cong \mathbf{X}[x,Gd_i]$ be the isomorphism of the adjunction. Then $(Fx, \{\varphi^{-1}(\sigma_i)\in\mathbf{A}[Fx,d_i] \mid i\in I\})$ is a cone. Indeed, for all $f_e: d_i\to d_j$ one has

$$\begin{aligned} f_e\circ\varphi^{-1}(\sigma_i) &= \varphi^{-1}(G(f_e)\circ\sigma_i) && \text{by naturality} \\ &= \varphi^{-1}(\sigma_j) && \text{because } (\sigma_i) \text{ is a cone on } G(D) . \end{aligned}$$

By the universality of $(a, \{\tau_i\in\mathbf{A}[a,d_i] \mid i\in I\})$ there exists a unique arrow $h: Fx\to a$ such that $\forall i\in I\ \ \tau_i\circ h = \varphi^{-1}(\sigma_i)$. Take then $\varphi(h): x\to Ga$. Since $G\tau_i\circ\varphi(h) = \varphi(\tau_i\circ h) = \sigma_i$, one has that $\varphi(h)$ is a mediating morphism between the cones $(x, \{\sigma_i\in\mathbf{X}[x,Gd_i] \mid i\in I\})$ and $(Ga, \{G\tau_i\in\mathbf{X}[Ga,Gd_i] \mid i\in I\})$.

Moreover, $\varphi(h)$ is unique, for, if $\varphi(h')$ is another mediating morphism, then h' is a mediating morphism between $(Fx,\{\varphi^{-1}(\sigma_i)\in\mathbf{A}[Fx,d_i] \mid i\in I\})$ and $(a, \{\tau_i\in\mathbf{A}[a,d_i] \mid i\in I\})$. By universality, $h' = h$ (see the diagram below). ♦

The proof of theorem 6.4.2 may be visualized by the following commutative diagrams:



Exercise Give the dual statement of theorem 6.4.2 .

An example of application of (the dual of) theorem 6.4.2 is the following.

6.4.3 Theorem *In every Cartesian closed category **C**, products distribute over colimits.*
Proof Just note that by definition of CCC the functor $-\times a$: $\mathbf{C}\to\mathbf{C}$ has a right adjoint for each $a\in Ob_{\mathbf{C}}$, and apply the dual of theorem 6.4.2. ♦

6.4.4 Corollary *Let **C** be a CCC. Suppose, moreover, that it contains an initial object 0, and coproducts for each pair of objects. Then, for all $X,Y,Z\in Ob_{\mathbf{C}}$, one has*

i. $0 \times Z \cong Z$

ii. $(X + Y) \times Z \cong (X \times Z) + (Y \times Z)$

Exercises (Huwig-Poigné) A category **C** has **fixpoints** if for every morphism $f: X\times X'\to X'$ there exists a morphism $Y(f): X\to X'$ such that $f \circ \langle id_X, Y(f)\rangle = Y(f)$. Prove then the following facts:

1. **CPO** has fixpoints.

2. If **C** is a CCC and it has an initial object 0 and fixpoints, then it is inconsistent , i.e. all objects are isomorphic. (*Hint*: let t the terminal object, and consider the projection p_2: $tx0\to 0$. Then $Y(p_2)$: $t\to 0$. Deduce from this an isomorhism between 0 and t ...) .

3. (difficult) If **C** is a CCC and it has fixpoints and binary coproducts, then **C** is inconsistent. *Hint*: consider the object $2=t+t$ and interpret the injection **tt:** $t\to 2$ and **ff**: $t\to 2$ as denoting "truth" and

"falsehood." Then all finitary truth tables can be expressed by morphisms in $2\times2\times....\times2 \to 2$. The existence of a fixpoint for "**not**" induces the following identities:

$$\mathbf{tt} = Y(\mathbf{not})\ \mathbf{or}\ \mathbf{not}(Y(\mathbf{not})) = Y(\mathbf{not})\ \mathbf{or}\ Y(\mathbf{not}) = Y(\mathbf{not})$$

$$\mathbf{ff} = Y(\mathbf{not})\ \mathbf{and}\ \mathbf{not}(Y(\mathbf{not})) = Y(\mathbf{not})\ \mathbf{and}\ Y(\mathbf{not}) = Y(\mathbf{not})$$

Hence the injections **tt** , **ff** : $t\to2$ are identified. As, for all objects X in **C**, $X+X = (t\times X) + (t\times X) = (t+t)\times X$, one may deduce the equality of the coproduct injections $u,v: X\to X\times X$ for all X . By this it is easy to obtain the inconsistency.

Fixed points will be widely discussed in chapter 8. The reader may already understand, though, that from the point of view of denotational semantics, this is a negative result: coproducts (i.e. disjoint sums) are incompatible with fixed point operators. As is well known, both constructions are rather relevant in semantic domains.

Another important case of limit-preserving functor is the hom-functor.

6.4.5 Theorem *Let* $\boldsymbol{C}$ *be a small category. For any object* $c\in Ob_{\boldsymbol{C}}$*, the hom-functor* $hom[c,\_]: \boldsymbol{C}\to\boldsymbol{Set}$ *preserves limits.*

Proof: Consider the diagram $D = (\{d_i\}_{i\in I}, \{f_e\}_{e\in E})$ in **C**, and let $(a, \{\tau_i\in\mathbf{A}[a,d_i] \mid i\in I\})$ be a limit. We must prove that the diagram $S = (\{hom[c,d_i]\}_{i\in I}, \{f_e \circ \_\}_{e\in E})$ has a limit in **Set**.

Take $L = (hom[c,a],\{\tau_i \circ \_ : hom[c,a]\to hom[c,d_i] \mid i\in I\})$ as a limit.

Since $hom[c,\_]$ is a functor, L is a cone for S. We have only to prove that it is universal. Suppose then that $L' = (X, \{\gamma_i : X\to hom[c,d_i] \mid i\in I\})$ is another cone for S. This means that for any $f_e: d_i\to d_j$, and any $x\in X$, $f_e \circ \gamma_i(x) = \gamma_j(x)$. For any $x\in X$, $(c, \{\gamma_i(x): c\to d_i \mid i\in I\})$ is then a cone for D, and by universality of $(a, \{\tau_i\in\mathbf{A}[a,d_i] \mid i\in I\})$, there exists a unique morphism $h_x: c\to a$ in **C** such that $\gamma_i(x) = \tau_i \circ h_x$ for all i. Define then $h: X\to hom[c,a]$ by $h(x) = h_x$. We have $(\tau_i \circ \_) \circ h = \gamma_i$, since for every $x\in X$, $\tau_i \circ h(x) = \tau_i \circ h_x = \gamma_i(x)$. Unicity follows by unicity of h_x for any x. ♦

Exercise Use theorem 6.4.5 and prove theorem 6.4.2 in case the categories considered are small.

4.6 Definition *A functor* $F: \boldsymbol{A}\to\boldsymbol{X}$ ***creates*** *limits for a given diagram D if, whenever* $(x, \{\sigma_i\in\boldsymbol{X}[x,F(d_i)] \mid i\in I\})$ *is a limit for* $F(D)$ *in* $\boldsymbol{X}$*, then there exists a unique cone* $(a, \{\tau_i \in\boldsymbol{A}[a,d_i] \mid i\in I\})$ *over* D *in* $\boldsymbol{A}$*, such that* $F(a) = x$ *and* $F(\tau_i) = \sigma_i$ *for every* $i\in I$*, and* $(a,\{\tau_i \in\boldsymbol{A}[a,d_i] \mid i\in I\})$ *is a limit.*

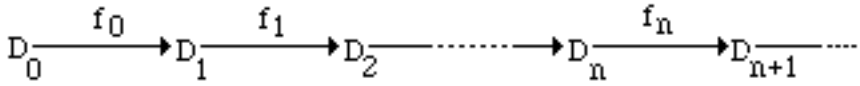
Example The forgetful functor U from **Grp** to **Set** creates all limits. For instance, the fact that it creates products is another way of stating that, given two groups G and G', there is a unique group structure on $U(G)\times U(G')$, which gives their product in Grp.

6.5 ω-limits

An important case of diagrams in a category **C** is that of infinite chains of objects. These diagrams, and the associated limits, are particularly relevant for the denotational semantics of programming languages, since they provide the base for the solution of recursive domain equations with the so-called *least fixed point technique* (see chapter 10)

6.5.1 Definition

*i) An **ω-diagram** in a category **C** is a diagram with the following structure:*



*(dually, one defines **ω^{OP}-diagrams** by just reversing the arrows).*

*ii. A category **C** is **ω-complete** (**ω-cocomplete**) iff it has limits (colimits) for all ω-diagrams.*

*iii. A functor F: **C**→**C** is **ω-continuous** iff it preserves all colimits of ω-diagrams.*

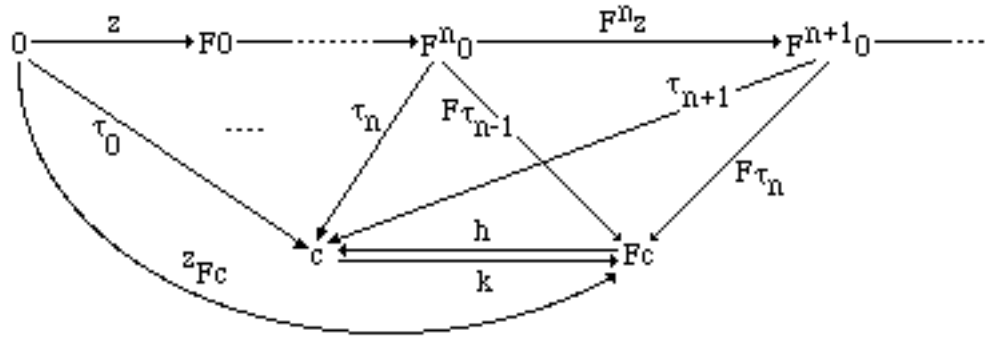
If **C** is a partial order then,

i. an ω-diagram in **C** is an ω-chain
ii. **C** is ω-cocomplete if and only if **C** is a cpo
iii. a functor F: **C**→**C** is ω-continuous iff the associated function on object of **C** is continuous.

6.5.2 Theorem *Let **C** be a category with initial object 0. Let F: **C**→**C** be an ω-continuos (covariant) functor and $z \in \mathbf{C}[0,F(0)]$ be the unique arrows defined by the initiality of 0. Assume also that $(c, \{\tau_i \in \mathbf{C}[F^i(0),c]_{i\in\omega}\})$ is a colimit for the ω-diagram $(\{F^i(0)\}_{i\in\omega}, F^i(z)\}_{i\in\omega})$, where $F^0(0) = 0$ and $F^0(z) = z$. Then $c \cong Fc$.*

Proof By the hypothesis, one has that $(Fc, \{F\tau_i \in \mathbf{C}[F^{i+1}(0),Fc]_{i\in\omega}\})$ is a limit for $(\{F^{i+1}(0)\}_{i\in\omega}, \{F^{i+1}(z)\}_{i\in\omega})$ and $(c, \{\tau_{i+1} \in \mathbf{C}[F^{i+1}(0),c]_{i\in\omega}\})$ is a cone for the same diagram. Thus, by universality, there exists a unique arrow $h: Fc \to c$ such that $\forall i\in\omega \; h \circ F\tau_i = \tau_{i+1}$. Now add to $(Fc, \{F\tau_i \in \mathbf{C}[F^{i+1}(0),Fc]_{i\in\omega}\})$ the unique arrow $z_{Fc} \in \mathbf{C}[0,Fc]$. This gives a cone for $(\{F^i(0)\}_{i\in\omega}, \{F^i(z)\}_{i\in\omega})$ and, by the universality of $(c,\{\tau_i \in \mathbf{C}[F^i(0),c]_{i\in\omega}\})$, there exists a unique arrow $k: c \to Fc$ such that $\forall i\in\omega \; k \circ \tau_{i+1} = F\tau_i$ (of course $k \circ \tau_0 = z_{Fc}$). But, then, $\forall i\in\omega$ $h \circ k \circ \tau_{i+1} = h \circ F\tau_i = \tau_{i+1}$ (and $h \circ k \circ \tau_0 = \tau_0$), thus $h \circ k$ is a mediating morphism between $(c, \{\tau_i \in \mathbf{C}[F^i(0),c]_{i\in\omega}\})$ and itself. Thus, by unicity, $h \circ k = id$.
In the same way, one proves that $k \circ h = id$. ♦

This is all summarized by the following diagram:



Theorem 6.5.2 tells us how to give meaning to recursive definitions of data types under certain circumstances. Very informally, assume that types are interpreted as objects of a category. Then in a recursive definition X = [...X...] of a data type of data X, the transformation [... _ ...] may be understood as an endofunctor F(_) for which we are seeking a fixed point. Indeed, if F satisfies the properties in theorem 6.5.2, then the theorem "solves" the equation (or recursive definition) X = [...X...]. In a sense, this construction gives meaning to X = [...X...], over a suitable categorical structure, in the same way that the equation $x = x^2+7$ is "given meaning" over the complex numbers by finding a solution for it.

However, the assumptions on F are too strong and leave out several significant cases (e.g., hom-functors or exponents). Chapter 10 is entirely devoted to a nontrivial extension of this technique in order to handle a more relevant class of recursive definitions of data types.

References Main textbooks.

Chapter 7

INDEXED AND INTERNAL CATEGORIES

7.1 Indexed Categories

In this section we introduce the basic notions of the Theory of Indexed Categories. In order to improve readability, the following exposition is an (over)simplification of the usual, and more general, approach. In particular, many of the concepts we define up to equality can be defined up to a fixed collection of *canonical* isomorphisms. In this case, the indexed notions introduced in the theory are required to satisfy a suitable set of coherence conditions, which play a quite marginal role, but conversely can easily puzzle the reader who is approaching the Theory of Indexed Categories for the first time. The reader who is interested in more notions in this branch of Category Theory should consult the References.

7.1.1 Definition *Let* ***CAT*** *be the (meta)category of all categories, and S be a category. An* ***S-indexed category*** *is a functor* $A: S^{op} \to \mathbf{CAT}$.

More explicitly, an S-indexed category A is defined by the following data:
i. for every object s of S, a category A(s), called the category of s-indexed families of objects of A;
ii. for every morphism f: s→s' of S, a functor A(f): F(s')→F(s), called the substitution functor determined by f, and frequently denoted as f^*.

Example A simple but important example is the S-indexing S/: $S^{op} \to \mathbf{CAT}$ of S itself. S/ takes every object r of S to the comma category S/r. Remember that the objects of S/s are arrows h: s→r with codomain r. These arrows should be intuitively thought of as families $\{h^{-1}(i)/\ i \in r\}$. If f: s→s' is an arrow of S, then f^*: S/s'→S/s is the pulling back functor. Note that pullbacks are usually defined only up to isomorphism, while we are here implicitly supposing a canonical choice. As a matter of fact, the pullback and the associated "functor" are the basic examples of notions profitably defined up to isomorphism, which we mentioned in the introduction.

7.1.2 Definition *Let* $A, B: S^{op} \to \mathbf{CAT}$ *be two S-indexed categories.*
1. The ***product category*** $A \times B: S^{op} \to \mathbf{CAT}$ *is defined by*

$$A \times B(s) = A(s) \times B(s)$$
$$A \times B(f) = A(f) \times B(f);$$

2. The ***dual category*** $A^{op}: S^{op} \to \mathbf{CAT}$ *is defined by*

$$A^{op}(s) = A(s)^{op}$$

$$A^{op}(f) = A(f)^{op}$$

where $A(f)^{op} : A(s')^{op} \to A(s)^{op}$ *is the defined in the obvious way;*

3. If r *is an object of* S*, the S-indexed category* $\mathbf{A^r}$ *is defined by*

$$A^r(s) = A(r \times s)$$
$$A(f) = A(id_r \times f).$$

7.1.3 Definition *Let A, B be two S-indexed categories. An* ***S-indexed functor*** $H: A \to B$ *is a natural transformation from* $A: S^{op} \to \mathbf{CAT}$ *to* $B: S^{op} \to \mathbf{CAT}$.

Thus, an S-indexed functor H: A→B is a collection of functors H(s): A(s)→B(s), for s object of S, such that for any f: s→s' in S, H(s) ∘ A(f) = B(f) ∘ H(s') (H(s) ∘ f^* = f^* ∘ H(s').

Given two indexed functors H: A→B and K: B→C, their composition K ∘ H : A→C is defined component-wise (being the composition of natural transformations), i.e., (K ∘ H)(s) = K(s) ∘ H(s). The identity id_A: A→A, is the identity natural transformation from A to A.

7.1.4 Definition *Let* $H: A \to B$, $K: A \to B$, *be two S-indexed functors. An* ***S-indexed natural transformation*** $\tau: H \to K$ *consists of a natural transformation* $\tau(s): H(s) \to K(s)$ *for any object* s *of* S *such that, for any* $f: s \to s'$ *in* S,

$$(\dagger) \qquad \tau(s) \circ A(f) = B(f) \circ \tau(s') \qquad (\tau(s) \circ f^* = f^* \circ \tau(s')\ ).$$

The previous definition is more complex than it seems at first sight. Note that τ(s): H(s)→K(s), τ(s'): H(s')→K(s') are natural transformations, while A(f): A(s')→A(s) and B(f): B(s')→B(s) are functors. We are thus composing natural transformations and functors in the way described at the end of section 3.2. τ(s) ∘ A(f) and B(f) ∘ τ(s') are natural transformations of the following type:

τ(s) ∘ A(f) : H(s) ∘ A(f) → K(s) ∘ A(f)

B(f) ∘ τ(s') : B(f) ∘ H(s') → B(f) ∘ K(s').

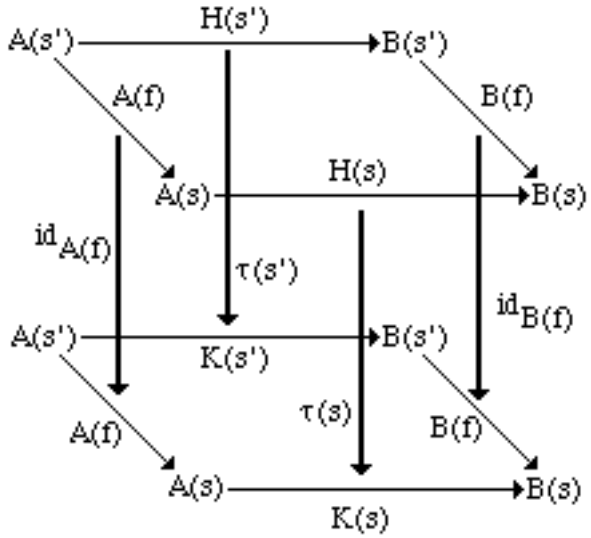
But, according to the definition of S-indexed functors, for any f: s→s', one has H(s) ∘ A(f) = B(f) ∘ H(s') and K(s) ∘ A(f) = B(f) ∘ K(s'), thus equation (†) is well typed.

Spelling out the composition of natural transformations and functors in (†), we have for any f: s→s' in S and any object a in A(s'),

$$\tau(s)_{A(f)(a)} = B(f)(\tau(s')_a)$$

where the previous equation holds in the category B(s).

The previous situation can be summarized in the following diagram:



(Vertical) composition of S-indexed natural transformations is defined componentwise, that is, given H, K, L : A→B,τ: H→K and ρ: K→L, ρ ∘ τ: H→L is given by $(\rho \circ \tau)(s) = \rho(s) \circ \tau(s)$. This is a good definition since, for any f: s→s' in S and any object a in A(s'),

$$\begin{aligned}(\rho \circ \tau)(s)_{A(f)(a)} &= (\rho(s) \circ \tau(s))_{A(f)(a)} \\ &= \rho(s)_{A(f)(a)} \circ \tau(s)_{A(f)(a)} \\ &= B(f)(\rho(s')_a) \circ B(f)(\tau(s')_a) \\ &= B(f)(\rho(s')_a \circ \tau(s')_a) \\ &= B(f)((\rho \circ \tau)(s')_a).\end{aligned}$$

7.1.5 Definition *Let A, B be S-indexed categories, H: A→B, K: B→A be S-indexed functors, and η: id_A→K∘H, ε: H∘K→id_B be S-indexed natural transformations. <H, K, η, ε> : A→B is an* ***S-indexed adjunction*** *if and only if*

$$(K\varepsilon) \circ (\eta K) = id_K$$
$$(\varepsilon H) \circ (H\eta) = id_H .$$

The notion of indexed adjunction is the obvious generalization of the usual notion of adjunction. In particular it is easy to check that for any object s of S, <H(s), K(s), η(s), ε(s)> : A(s)→B(s) is an adjunction in the usual sense.

The main problem with the definition of adjunction as a quadruple <H, K, η, ε> : A→B is in its generalization of the case with parameters (remember that the definition of exponents requires an adjunction of this kind). As a triple, an indexed adjunction can be defined in the following, somewhat informal, way:

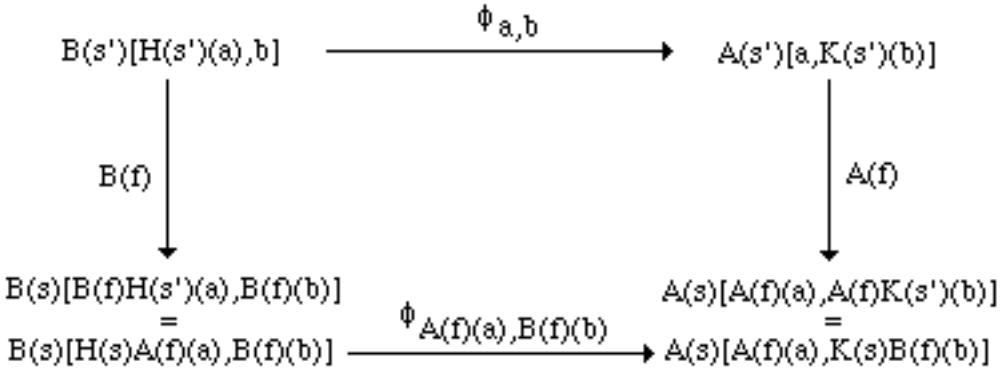
7.1.6 Definition *Let A, B be S-indexed categories, and H: A→B, K: B→A be S-indexed functors. <H, K, ϕ> : A→B is an* ***S-indexed adjunction*** *if and only if, for every f: s→s' in S,*

i. $\langle H(s), K(s), \phi(s)\rangle : A(s)\to B(s)$ *is an adjunction*

ii. $\phi(s) \circ B(f) = A(f) \circ \phi(s')$ $\quad (\phi(s) \circ f^* = f^* \circ \phi(s'))$

Equation ii expresses the naturality of the isomorphism ϕ with respect to the index s. Spelling out the composition in ii, we can say that for any f: s→s', a in A(s'), b in B(s'), and g: H(s')(a)→b in B(s'),

$$\phi(s)_{A(f)(a),B(f)(b)}\,(B(f)(g)) = A(f)\,(\phi(s')_{a,b}(g))$$



Suppose we have an adjunction $\langle H, K, \eta, \varepsilon\rangle : A\to B$. Then we obtain ϕ in definition 7.1.6 by letting, for any a in A(s), b in B(s), and g: H(s)(a)→b in B(s),

$$\phi(s)_{a,b}(g) = \varepsilon(s)_b \circ H(s)(g)$$

As we know from chapter 5, for any s in S, $\phi(s)_{a,b}$: B(s)[H(s)(a),b]→A(s)[a,K(s)(b)] is an isomorphism. We now prove that the previous definition of $\phi(s)$ satisfies equation ii in definition 7.1.6. For any f: s→s', a in A(s'), b in B(s'), and g: H(s')(a)→b in B(s'), we have

$$\begin{aligned} A(f)\,(\phi(s')_{a,b}(g)) &= A(f)\,(\varepsilon(s')_b \circ H(s)(g)) && \text{by def. of } \phi(s') \\ &= A(f)\,(\varepsilon(s')_b) \circ A(f)\,(H(s)(g)) && \text{since A(f) is a functor} \\ &= \varepsilon(s)_{B(f)(b)} \circ H(s)(B(f)(g)) && \text{by naturality of } \varepsilon \text{ and H} \\ &= \phi(s)_{A(f)(a),B(f)(b)}\,(B(f)(g)) && \text{by def. of } \phi(s) \end{aligned}$$

Conversely, given an adjunction $\langle H, K, \phi\rangle : A\to B$, we obviously obtain η, ε by the following:

$$\eta(s)_a = \phi(s)_{a,H(s)(a)}(id_{H(s)(a)}) : a\to K(s)H(s)a$$
$$\varepsilon(s)_b = \phi(s)^{-1}_{K(s)(b),b}(id_{K(s)(b)}) : H(s)K(s)b\to b.$$

Definition 7.1.6 has a straightforward generalization to the case with parameters.

7.1.7 Definition *Let A, B, D be S-indexed categories, and* $H: A\times D\to B$, $K: D^{op}\times B\to A$ *be S-indexed functors.* $\langle H, K, \phi\rangle : A\to B$ *is an* ***S-indexed adjunction with parameters in D*** *if and only if, for every* $f: s\to s'$ *in S,*

i. $\langle H(s), K(s), \phi(s)\rangle : A(s)\to B(s)$ *is an adjunction with parameters in D(s);*

ii. $\phi(s) \circ B(f) = A(f) \circ \phi(s')$ $\quad (\phi(s) \circ f^* = f^* \circ \phi(s'))$.

7.2 Internal Category Theory

A category C is **small** when the collection Mor_C of its morphisms is a set. Clearly, then, the collection Ob_C of objects of C is also a set. Moreover, there are set-theoretic functions DOM,COD: $\mathrm{Mor}_C \to \mathrm{Ob}_C$ that specify source and target of every morphism, a function ID: $\mathrm{Ob}_C \to \mathrm{Mor}_C$ that defines the identity morphism for every object, and a partial function COMP: $\mathrm{Mor}_C \times \mathrm{Mor}_C \to \mathrm{Mor}_C$ for the composition. Given two morphisms f and g, their composition is defined if and only if DOM(f) = COD(g); the domain of COMP is thus the set $\{(f,g) \mid \mathrm{DOM}(f) = \mathrm{COD}(g)\}$, that is, the pullback of the two functions DOM,COD: $\mathrm{Mor}_C \to \mathrm{Ob}_C$. All these functions must also satisfy the obvious equations stating the behavior of the identity morphism with respect to composition, the associativity law for compositition, and the rules which specify domain and target for the identity morphism and for the result of a composition. Thus every small category may be completely described *internally* to the category **Set**, which becomes a sort of "universe of discourse." The previous discussion, however, has made very little use of the specific structure of **Set**; we only needed the existence of pullbacks in order to define the correct domain of the function COMP. In this section, we will show that most of the basic definitions of Category Theory, such as category, functor, natural transformation and so on, can be recasted *inside* any category with *all finite limits*. This means that any such a category may be considered a fairly big universe inside which we can carry out constructions with almost the same confidence as we do in **Set**. This branch of Category Theory is known as "internal," since it describes notions of Category Theory by using the categorical language as a metalanguage.

For many fields of mathematics, from Set Theory to Algebra and Geometry, treatments in the language of Category Theory, even of well-known results, have never been worthless since most of the time they created a new, sometimes unexpected, sense of explanation. The same holds for Category Theory itself: in a sense, Internal Category Theory plays with respect to the general theory the same role that Category Theory plays with respect to Set Theory. If a notion of Category Theory cannot be described internally in a simple way, then there is surely something in that notion that is worth spelling out. As we shall see, this is, for example, the case of the hom-functor and, more generally, of every presheaf.

Internal Category Theory allows us to work in different universes than **Set**. This possibility turns out to be very relevant in several cases, and in particular for the application we aim at in chapter 11, where Internal Category Theory will be applied to the study of categorical models for the polymorphic lambda calculus. In that case, the possibility of working in more constructive categories than **Set** turns out to be essential, as it is known that the standard set-theoretic interpretation of the first order typed lambda calculus cannot be extended to a model of the second order typed lambda calculus.

In the following, E will always denote a category with all finite limits. Our first step is to mimic within E the presentation, within **Set**, of a small category. Thus the collections of objects and morphisms will be viewed as objects of E.

Notation We write $X\times_0 Y$ (instead of $X\times_Z Y$) for the pullback of X and Y along morphisms with common target Z; $\langle,\rangle_0$ will be used as a "pullback pairing" map, that is, given (suitable) $h:W\to X$ and $k:W\to Y$, we have $\langle h,k\rangle_0:W\to X\times_0 Y$; the pullback projections will be usually (but not always) denoted by the upper case Greek letter Π, indexed with a number or some other symbol.

7.2.1 Definition *$c = (c_0,c_1,DOM,COD,COMP, ID)$ is an **internal category** of E ($c\in Cat(E)$) iff:*

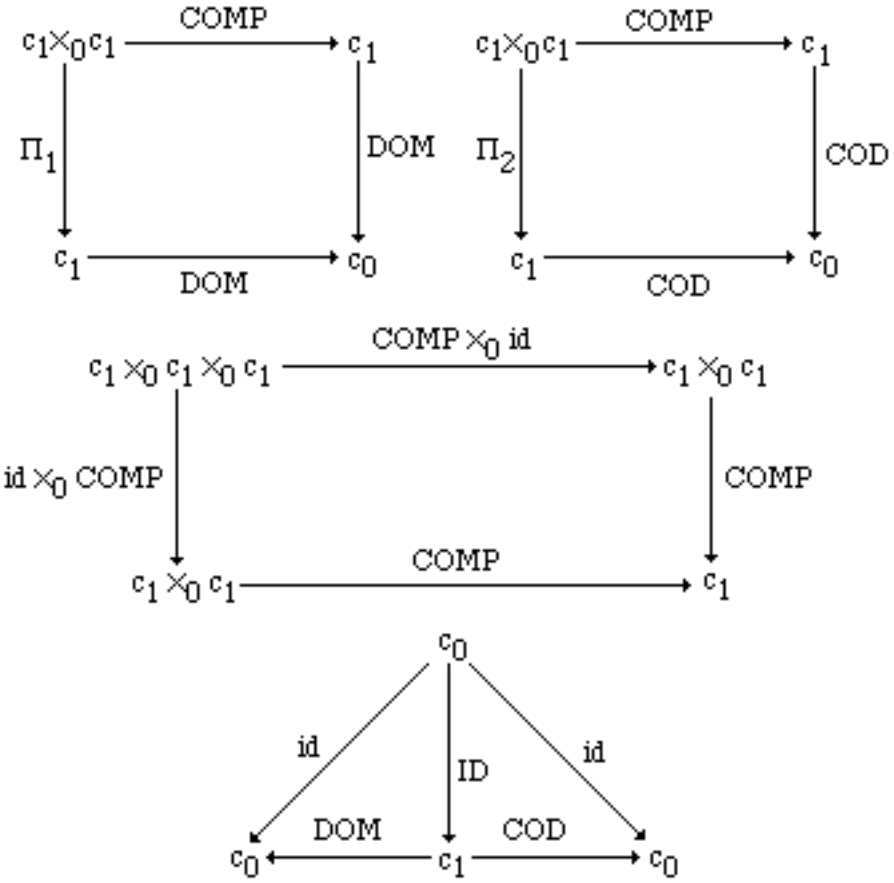
$c_0, c_1\in Ob_E$

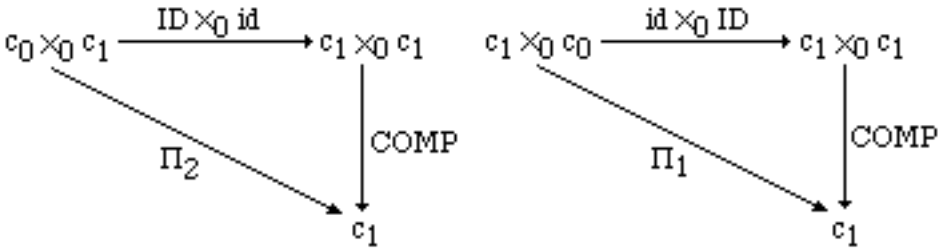
$DOM, COD : c_1 \to c_0$

$COMP : c_1 \times_0 c_1 \to c_1$ *where* $c_1 \times_0 c_1$ *is the pullback of* $DOM, COD : c_1\to c_0$

$ID : c_0 \to c_1$

and moreover





Note that in the diagram expressing the associativity of composition there is an implicit isomorphism between $c_1 \times_0 (c_1 \times_0 c_1)$ and $(c_1 \times_0 c_1) \times_0 c_1$. Indeed,

$$\text{COMP} \circ (\text{COMP} \times_0 \text{id}) : (c_1 \times_0 c_1) \times_0 c_1 \to c_1$$
$$\text{COMP} \circ (\text{id} \times_0 \text{COMP}) : c_1 \times_0 (c_1 \times_0 c_1) \to c_1.$$

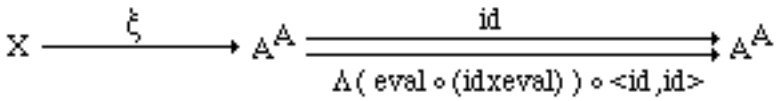
In the following, this isomorphism will be always skipped in order to maintain the notation at a simpler level.

7.2.2 Examples 1. Given an object e in E, the internal **discrete category** associated with e is $(e,e,id_e,id_e,id_e,id_e)$.

2. Let E be a CCC with all finite limits, and let A be an object of E. It is possible to define internally to E a category that plays the role of the category of retractions over A.

Let $m = \Lambda(\text{eval} \circ (\text{id} \times \text{eval})) : A^A \times A^A \to A^A$ the internal composition map, that is let $m = \lambda(f,g).g \circ f$. Since E has all finite limits, it has equalizers for every pair of morphisms. Let then (X, ξ) be the equalizer of

$$\text{id} : A^A \to A^A$$
$$m \circ \langle \text{id},\text{id} \rangle : A^A \to A^A$$



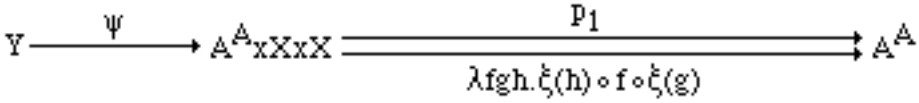
The function $m \circ \langle \text{id},\text{id} \rangle : A^A \to A^A$ is $\lambda f.\, f \circ f$; thus the object X represents the subset of A^A of all those functions f such that $f = f \circ f$, i.e., X is an internalization for the set of retractions in A^A. X plays the role of c_0 in the internal category we are defining.

Intuitively, a morphism between two retractions g and h is a triple (f,g,h), where f is a function from A to A such that $f = h \circ f \circ g$.

In order to internalize this definition we use the equalizer (Y,ψ) of

$$p_1 : A^A \times X \times X \to A^A$$
$$m \circ (m \times \text{id}) \circ \langle \xi \circ p_3 , p_1, \xi \circ p_2 \rangle : A^A \times X \times X \to A^A$$

Note that $m \circ (m \times \text{id}) \circ \langle \xi \circ p_3 , p_1, \xi \circ p_2 \rangle$ is just $\lambda fgh.\, \xi(h) \circ f \circ \xi(g)$.



COD and DOM are obviously defined by the following equations:

$$DOM = p_2 \circ \psi$$
$$COD = p_3 \circ \psi$$

For ID, note first that by definition of ξ, $m \circ \langle \xi, \xi \rangle = id \circ \xi = \xi$ and, therefore,

$$(\lambda fgh.\ \xi(h) \circ f \circ \xi(g)) \circ \langle \xi, id, id \rangle = \xi.$$

Thus $\langle \xi, id, id \rangle: X \to A^A \times X \times X$ equalizes p_1 and $\lambda fgh.\ \xi(h) \circ f \circ \xi(g)$, and $ID: X \to Y$ is the unique arrow such that $\psi \circ ID = \langle \xi, id, id \rangle$. Note that

$$DOM \circ ID = p_2 \circ \psi \circ ID = p_2 \circ \langle \xi, id, id \rangle = id$$
$$COD \circ ID = p_3 \circ \psi \circ ID = p_3 \circ \langle \xi, id, id \rangle = id.$$

Finally, we must define $COMP: Y \times_0 Y \to Y$. The idea is that $(f,g,h) \circ (f',k,g) = (f \circ f', k, h)$. We start defining an arrow $M: Y \times_0 Y \to A^A \times X \times X$ such that $M((f,g,h), (f',k,g)) = (f \circ f', k, h)$; next we prove that M equalizes p_1 and $\lambda fgh.\ \xi(h) \circ f \circ \xi(g)$. Then COMP is the unique arrow from $Y \times_0 Y$ to Y such that $\psi \circ COMP = M$.

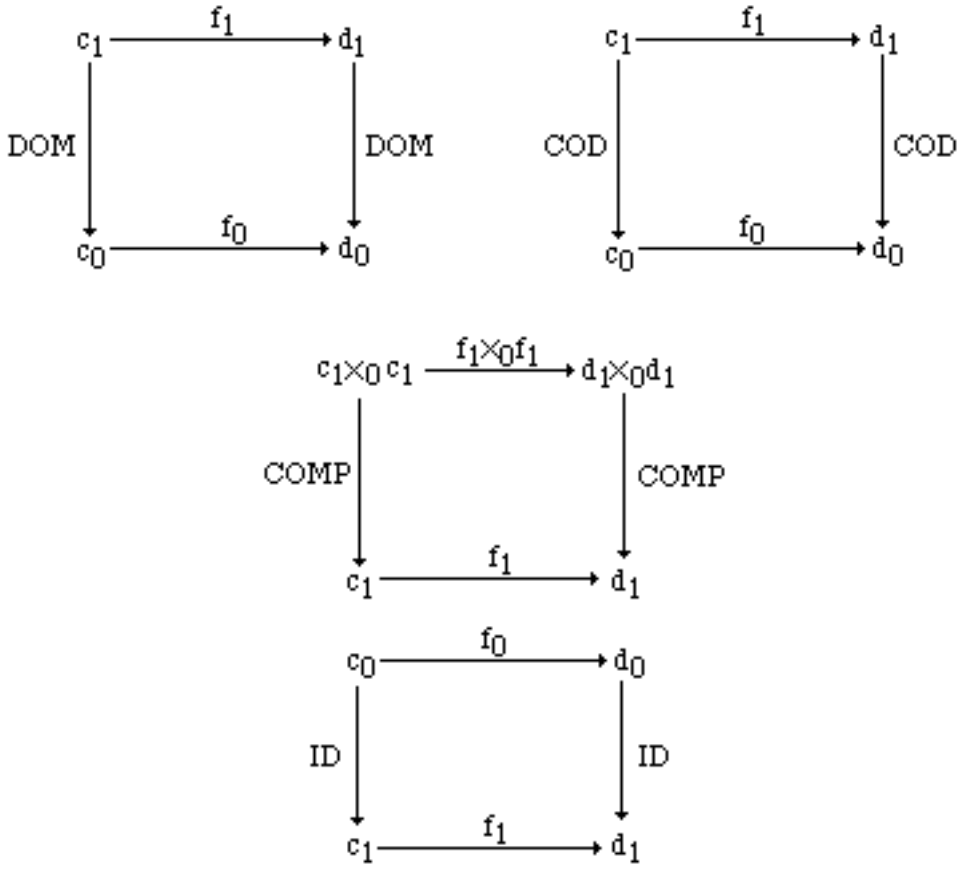
3. Given a function $f: U \to C$, consider the C-indexed collection of sets $\{G(c) = f^{-1}(c)\}_{c \in C}$. We can form a small category $\mathbf{C}$, which has C as set of objects, and with hom-sets $\mathbf{C}[c,c'] = \mathbf{Set}[G(c),G(c')]$. Composition and identities are inherited from **Set**. The previous construction can be generalized to a generic topos E: given $f: U \to C$ in E, there is an internal category Full(f) that plays the role of the full subcategory generated by the fibers of f. $Full(f)_0$ is C; $Full(f)_1$, together with the map $\langle DOM,COD \rangle: Full(f)_1 \to C \times C$, is defined as the exponent $p_1^*(f)p_2^*(f)$ in the slice category $E/C \times C$, where

$$p_1^*(f) = f \times id: U \times C \to C \times C$$
$$p_2^*(f) = id \times f: C \times U \to C \times C$$

are respectively the pullbaks of f along the first and second projections. Composition is obtained from the internal composition map $m: p_2^*(f)p_1^*(f) \times_{p_3^*(f)p_2^*(f)} p_3^*(f)p_1^*(f) \to p_3^*(f)p_1^*(f)$ in the slice category $E/C \times C \times C$. Similarly, the identity morphism $ID: C \to Full(f)_1$ is obtained from the "inclusion of identities" $\Lambda(id_f): id_C \to f^f$ in the slice category E/C.

Our exposition of Internal Category Theory proceeds with the definition of "internal functor." Again, the intuition of a standard functor helps in the understanding of the following definition; a functor F between two small categories C and D is a pair of functions in **Set**, $F = (F_0, F_1)$, where $F_0: Ob_C \to Ob_D$, $F_1: Mor_C \to Mor_D$; moreover F_1 distributes with respect to composition and preserves identity.

7.2.3 Definition *Let $c,d \in Cat(E)$. F is an **internal functor** from c to d ($F: c \to d$) iff $F = (f_0, f_1)$ with $f_0 \in E[c_0,d_0]$, $f_1 \in E[c_1, d_1]$, and F satisfies*



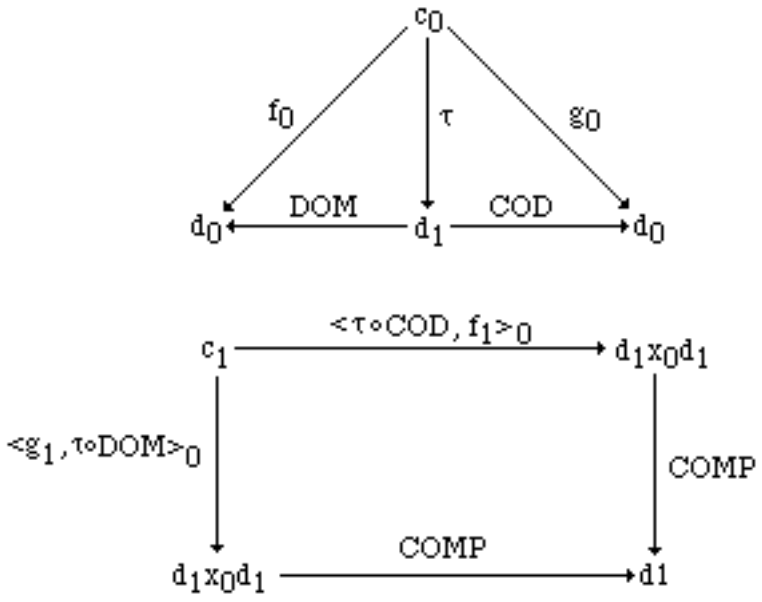
7.2.4 Definition *The category* ***Cat(E)*** *has as objects the internal categories of E and as morphisms the internal functors. Composition of functors is defined in the obvious way; that is, given* $F = (f_0,f_1)$ *and* $G = (g_0,g_1)$, $F \circ G = (f_0 \circ g_0, f_1 \circ g_1)$.

For example, Cat(**Set**) is the category **Cat** of all small categories, i.e., of all those categories whose class of morphisms is a set.

It is easy to carry out the usual constructions on categories inside **Cat(E)**. For example, given $c = (c_0, c_1, \mathrm{DOM}, \mathrm{COD}, \mathrm{COMP}, \mathrm{ID})$, we can define the **dual** category $c^{op} = (c_0, c_1, \mathrm{COD}, \mathrm{DOM}, \mathrm{COMP}\circ\alpha, \mathrm{ID})$, where $\alpha = \langle\Pi_2,\Pi_1\rangle_0 : c_1\times_0 c_1 \leftrightarrow c_1\times_0{}'c_1$. $\_^{op}$: **Cat(E)** → **Cat(E)** is a functor.

The **product** of two internal categories c and d is the category $c\times d = (c_0\times d_0, c_1\times d_1, \mathrm{DOM}_c\times\mathrm{DOM}_d, \mathrm{COD}_c\times\mathrm{COD}_d, (\mathrm{COMP}_c\times\mathrm{COMP}_d)\circ\beta, \mathrm{ID}_c\times\mathrm{ID}_d)$ where β is the isomorphism $(c_1\times_0 c_1)\times(d_1\times_0 d_1) \leftrightarrow (c_1\times d_1)\times_0(c_1\times d_1)$. Clearly, $\_\times\_$: **Cat(E)**×**Cat(E)** → **Cat(E)** is a functor.

7.2.5 Definition *Let* $F = (f_0,f_1)$ *and* $G = (g_0,g_1)$ *be two internal functors from c to d.* τ *is an* ***internal natural transformation*** *from F to G* ($\tau: F\to G$) *iff* $\tau\in E[c_0, d_1]$ *and satisfies*



7.2.6 Definition *Given two internal categories c and d,* ***Nat(c,d)*** *is the category that has internal functors from c to d as objects, and internal natural transformations as arrows. Given* σ*:* $F \to G$ *and* $\tau: G \to H$, $\tau \circ \sigma = COMP_d \circ \langle \tau, \sigma \rangle_0 : F \to H$

7.2.7 Example In this example we define **PER** as an internal category of ω-Set. **PER** is the category of partial equivalence relations constructed over Kleene's applicative structure (ω,·). Remember that the partial application $\cdot : \omega \times \omega \to \omega$ is defined by $m \cdot n = \varphi_m(n)$, where $\varphi: \omega \to PR$ is an acceptable gödel numbering of the partial recursive functions. We will use the following notation:

n A m iff n is related to m by A ,

$\{n\}_A = \{m \mid m\ A\ n\}$ the equivalence class of n with respect to A,

$Q(A) = \{\{n\}_A \mid n \in dom(A)\}$ where $dom(A) = \{n / nAn\}$.

The morphisms of the category are defined by

$f \in PER[A,B]$ iff $f : Q(A) \to Q(B)$ and $\exists n\ \forall p\ (pAp \Rightarrow f(\{p\}A) = \{n.p\}B)$.

Thus the morphisms in **PER** are "computable" in the sense that they are fully described by partial recursive functions, which are total on the domain of the source relation.

Note that **PER** is a small category, as the partial equivalence relations (p.e.r.'s) form a set as well as their morphisms; thus **Set** contains **PER** as an internal category. Though, since a crucial property of **PER** is that its morphisms are "computable," we are interested in introducing a similar notion in the category of sets by a **realizability** relation "⊩" with respect to numbers.

The category **ω−Set** is defined as follows:

objects: $(A, \Vdash) \in$ **ω−Set** iff

A is a set and $\Vdash \subseteq \omega \times A$, such that $\forall a \in A\ \exists n \Vdash a$.

morphisms : $f \in$ **ω−Set**$[A,B]$ iff

$$f : A \to B \text{ and } \exists n\ \forall a \in A\ \forall p \Vdash_A a \quad n \cdot p \Vdash_B f(a)$$

(*notation* : $n \Vdash_{A \to B} f$ and we say that n **realizes** f).

Similarly as for **PER**, each morphism in **ω−Set** is "computed" by a partial recursive function, which is total on $\{ p \mid p \Vdash_A a \}$ for each $a \in A$.

It is not difficult to prove that **ω−Set** is a CCC with all finite limits. The terminal object is simply $(\mathbf{1}, \Vdash_1)$, where $\mathbf{1}$ is the singleton set and $\Vdash_1 = \omega \times \mathbf{1}$. If $[\ ,\ ]$ is a coding of pairs of numbers, then $(A \times B, \Vdash_{A \times B})$ is given by $[n,m] \Vdash_{A \times B} (a,b)$ iff $n \Vdash_A a$ and $m \Vdash_B b$. As for exponents, let $[A \to B] = (\{f: A \to B \mid f \in \mathbf{\omega\text{-Set}}\ [A,B] \}, \Vdash_{A \to B})$, where $\Vdash_{A \to B}$ is given as above.

There is a simple way to embed **Set** into **ω-Set**. Let $\Sigma : \mathbf{Set} \to \mathbf{\omega\text{-Set}}$ be given by

$$\Sigma(S) = (S, \Vdash_S) \quad \text{with} \quad \Vdash_S = \omega \times S \text{ , the "full" relation.}$$

Σ is defined as the identity on morphisms, since by the definition of $\Vdash_S$, all functions are realized by all numbers for total recursive functions. Σ is a full and faithful functor, which preserves all finite limits and exponents.

This embedding suggests how to turn **PER** into an internal category of **ω-Set** (recall that the exponent of A and B in **PER** is given by $m\ (A \to B)\ n \Leftrightarrow \forall p,q\ (p\ A\ q \Rightarrow m \cdot p\ B\ n \cdot q)$). Indeed, $\mathbf{M} = (\mathbf{M_0}, \mathbf{M_1}, \mathrm{dom}^M, \mathrm{cod}^M, \mathrm{id}^M, \mathrm{comp}^M)$ is defined by

1. $\mathbf{M_0} = (\mathbf{PER}, \Vdash_M)$ where $\Vdash_M = \omega \times \mathbf{PER}$;
2. $\mathbf{M_1} = (\{<\{n\}_{A \to B}, A, B)> \mid A,B \in \mathbf{M},\ n\ (A \to B)\ n \},\ \Vdash_1)$
   where $m \Vdash_1 <\{n\}_{A \to B}, A, B>$ iff $m\ (A \to B)\ n$;
3. $\mathrm{dom}^M(<\{n\}_{A \to B}, A, B)>) = A$;
4. $\mathrm{cod}^M(<\{n\}_{A \to B}, A, B)>) = B$;
5. $\mathrm{id}^M(A) = <\{i\}_{A \to A}, A, A)>$ where $i = \lambda x.x$ is a number for the identity function;
6. $\mathrm{comp}^M(<\{n\}_{A \to B}, A, B)>, <\{m\}_{B \to C}, B, C>) = <\{b \cdot m \cdot n\}_{A \to C}, A, C)>$
   where $b = \lambda xyz.x(yz)$.

We have to check that **M** is an internal category of **ω-Set**. It will be easy, in view of the set-theoretic nature of its morphisms. Essentially, one has to prove that the required morphisms are functions that happen to be realized.

Note first that $\mathbf{\omega\text{-Set}}[\underline{A}, \Sigma(S)] = \mathbf{Set}[A,S]$ for any $\underline{A} = (A, \Vdash_A)$ in **ω-Set** and any set S, since $\Vdash_S$ is the full relation and, hence, any function is realized by any index. Thus, the set-theoretic functions dom^M, cod^M are also morphisms in **ω-Set**.

$\mathbf{M_1}$ is a set of triples: equivalence class, domain, and codomain. The realizability relation in $\mathbf{M_1}$ is nontrivial and, hence, one needs to give explicitly the realizers of id^M and comp^M. Indeed, id^M is realized by $\underline{\lambda}x.i$, the constant function equal to an index i for the identity function. As for comp^M, it is defined as usual only on a subset of $\mathbf{M_1} \times \mathbf{M_1}$, namely, where the target of the first morphism coincides with the source of the second. In the general setting, this is expressed by the use

of a pullback as a source object for COMP. In this specific case, that pullback becomes simply the set of pairs such as $(<\{n\}_{A\to B},A,B)>,<\{m\}_{B\to C},B,C>)$. Then the realizer for $comp^M$ is b', for b'[n,m] = bnm, where b is an index for the composition of functions, an operation that may be uniformly and effectively given over $(\omega,\cdot)$.

7.3 Internal Presheaves

We have already remarked that every small category may be regarded as an internal category in **Set.** However, in **Set** we are accustomed to considering not only functors from one small category to another, but also, for example, functors from a small category to a large one and in particular to **Set** itself. A significant example is hom-functor from a small category to **Set**. Surprisingly, it is possible to cope at the internal level also with this problem, by means of the notion of **internal presheaf**.

If F is a functor from $\mathbf{C}^{op}$ to **Set**, then the component F_{Ob} of F is a collection {F(c)} of sets indexed on objects of **C**. Such a collection can be regarded as a function $\rho_0: X\to Ob_{\mathbf{C}}$, where X = {(c,m)/ m∈F(c)} and $\rho_0(c,m) = c$. Then $F_{Ob}(c) \cong \rho_0^{-1}(c)$. Now, given an arrow f: d→c, and an object $(c,m)\in\rho_0^{-1}(c)$, define a function ρ_1 by $\rho_1((c,m),f) = (d,F(f)(m))$. The function ρ_1 describes the behavior of F on morphisms. Note that $\rho_1((c,m),f)$ is defined if and only if cod(f) = $\rho_0(c,m) = c$; thus, the domain of ρ_1 is the pullback Z (in **Set**) of cod: $Mor_{\mathbf{C}}\to Ob_{\mathbf{C}}$ and ρ_0: $X\to Ob_{\mathbf{C}}$. Let $\Pi_2: Z\to Mor_{\mathbf{C}}$ and $\Pi_1: Z\to X$, be the associated projections. Note that

1. $\rho_0(\rho_1((c,m), f)) = \rho_0((d,F(f)(m))) = d = dom(f) = dom(\Pi_2(f,(c,m)))$;
2. $\rho_1((c,m),f\circ f') = (d,F(f\circ f')(m)) = (d,F(f')(F(f)(m))) = \rho_1((d',F(f)(m)),f') =$
   $= \rho_1(\rho_1(f,(c,m)),f')$;
3. $\rho_1((c,m), id_c) = (c,F(id_c)(m)) = (c, m)$.

That is, more concisely:

i. $\rho_0 \circ \rho_1 = dom \circ \Pi_2 : Z\to Ob_{\mathbf{C}}$;

ii. $\rho_1 \circ (id_X \times_0 comp) = \rho_1 \circ (\rho_1 \times_0 id_{Mor\mathbf{C}}) : X\times_0 Mor_{\mathbf{C}}\times_0 Mor_{\mathbf{C}} \to Mor_{\mathbf{C}}$;

iii. $\rho_1 \circ <id_X, ID \circ \rho_0>_0 = id_X$,

where $\times_0$ denotes pullback product and ID: $Ob_{\mathbf{C}}\to Mor_{\mathbf{C}}$ is the function that takes an object c to id_c. Conversely, given a small category **C**, and a triple $(X, \rho_0: X\to Ob_{\mathbf{C}}, \rho_1: X\times_0 Ob_{\mathbf{C}}\to Mor_{\mathbf{C}})$ that satisfies equations i-iii above, it is possible to define a presheaf F: $\mathbf{C}^{op}\to$**Set** by letting

$\forall c\in Ob_{\mathbf{C}}$ $\quad F(c) = \rho_0^{-1}(c)$

$\forall f\in\mathbf{C}[c',c], \forall (c,m)\in F(c)$ $\quad F(f)(c,m) = \rho_1((c,m),f)$.

Equation i states that $\rho_1((c,m),f)$ is in F(c'), indeed c' = dom(f) = $dom(\Pi_2((c,m),f)) = \rho_0(\rho_1((c,m),f))$, and thus, by definition of F, $F(f)(c,m)=\rho_1((c,m),f)\in F(c')$.

Equations ii and iii express the fact that F is a contravariant functor. Indeed,

$F(f\circ g)(c,m) = \rho_1((c,m),comp(f,g))$ by def. of F

$= \rho_1(\rho_1((c,m),f), g)$ by (ii)

$= F(g)(\rho_1((c,m),f))$ by def. of F
$= F(g)(F(f)((c,m)))$ by def. of F

and

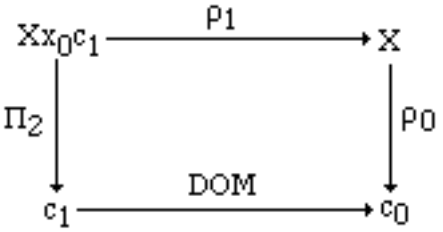
$F(id_c)(c,m) = \rho_1((c,m), id_c)$ by def. of F
$= (c,m)$ by (iii)

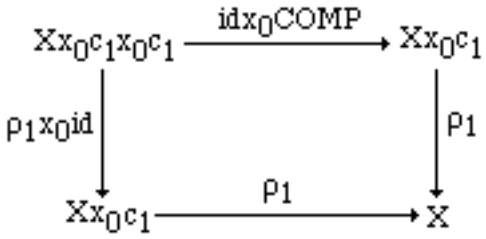
7.3.1 Definition *X is an **internal presheaf** on $c \in Cat(E)$ iff $X = (X, \rho_0, \rho_1)$ with,*

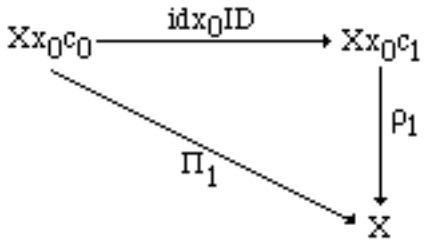
$\rho_0: X \to c_0$

$\rho_1: X \times_0 c_1 \to X$ where $X \times_0 c_1$ is the pullback of $\rho_0: X \to c_0$ and COD: $c_1 \to c_0$,

and X satisfies the following:







Example Let $c \in Cat(E)$, and e an object of E. The **constant-e** diagram is the internal presheaves $(e \times c_0, \text{snd}: e \times c_0 \to c_0, id_e \times DOM: e \times c_1 \to e \times c_0)$. Note that $e \times c_1$ is the pullback of snd: $e \times c_0 \to c_0$ and COD: $c_1 \to c_0$. Moreover, the previous morphism satisfies the requested conditions of definition 7.3.1, since

i. $\text{snd} \circ id_e \times DOM = DOM \circ \text{snd} : e \times c_1 \to c_0$;

ii. $id_e \times DOM \circ (id_e \times COMP) =$
$= id_e \times (DOM \circ COMP)$
$= id_e \times (DOM \circ \Pi_2)$

$= id_e \times DOM \circ (id_e \times \Pi_2)$

$= id_e \times DOM \circ (id_e \times DOM \times_0 id) : e \times c_1 \times_0 c_1 \to e \times c_0$;

iii. $id_e \times DOM \circ (id_e \times ID) = id_{e \times c_0} : e \times c_0 \to e \times c_0$.

The intuition behind the previous definition is that of a collection, indexed by c, of objects e. Indeed, consider the case of an internal category **C** in **Set** (i.e., a small category) and let E be a set. By applying the above "externalization," we obtain

$\forall c \in Ob_{\mathbf{C}}$ $\quad F(c) = \rho_0^{-1}(c) = E \times \{c\}$

$\forall f \in \mathbf{C}[c',c], \forall (e,c) \in F(c)$ $\quad F(f)(e,c) = \rho_1((e,c),f) = (e,DOM(f)) = (e,c')$.

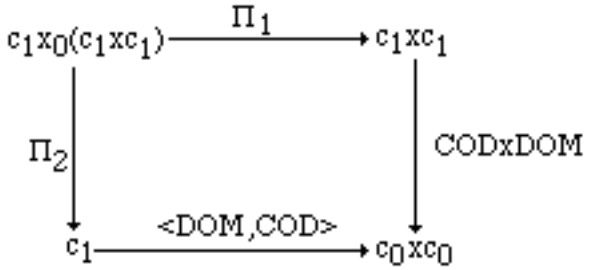
Another major example of a presheaf is given by the hom-functor.

7.3.2 Definition *Let $c \in Cat(E)$. The **internal hom-functor** hom_c is the presheaf (c_1, ρ_0, ρ_1) on $c \times c^{op}$, where*

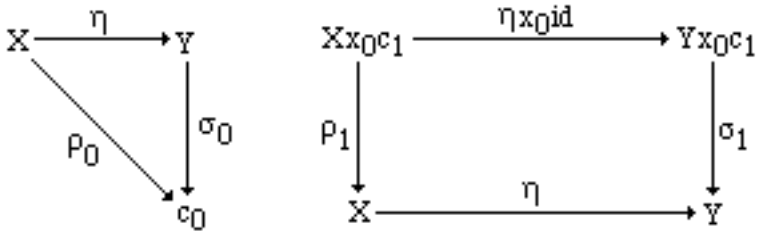
$\rho_0 = \langle DOM, COD \rangle : c_1 \to c_0 \times c_0$

$\rho_1 = COMP \circ \langle p_2 \circ \Pi_1, COMP \circ (id \times_0 p_1) \rangle_0 : c_1 \times_0 (c_1 \times c_1) \to c_1$

(Informally, $\rho_1 = \lambda fgh.\ h \circ f \circ g$), and

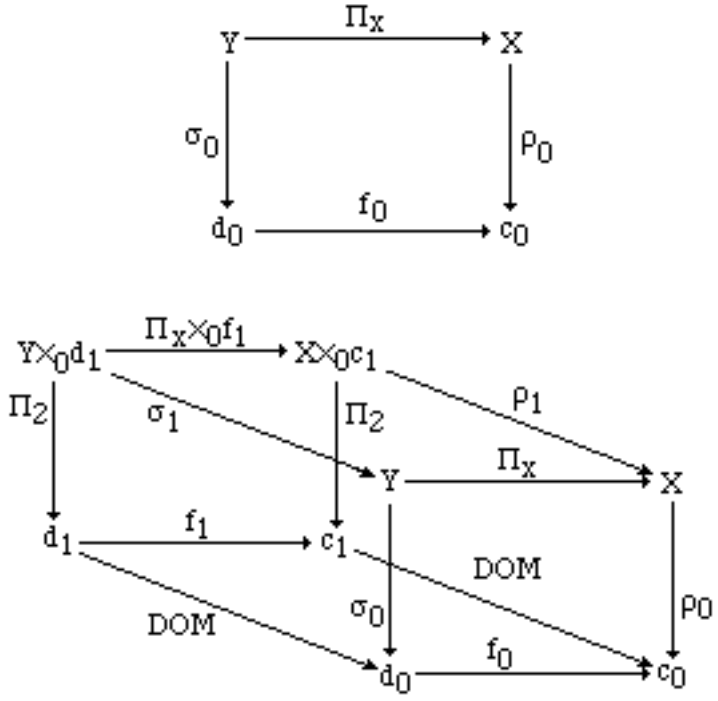


7.3.3 Definition *Let $X = (X, \rho_0, \rho_1)$, $Y = (Y, \sigma_0, \sigma_1)$ be two presheaves on $c \in Cat(E)$. η is a **morphism of presheaves** from X to Y ($\eta : X \to Y$) iff $\eta \in E[X,Y]$ and the following diagrams commute:*



The following definition allows to compose an internal presheaf on c with an internal functor F: d→c, yielding a new presheaf on d.

7.3.4 Definition *Let $X = (X, \rho_0, \rho_1)$ be an internal presheaf on $c \in Cat(E)$, and $F: d \to c$ be an internal functor. The **pullback of X along F** is the presheaf $F^*(X) = (Y, \sigma_0, \sigma_1)$ on d defined by the following commutative diagrams, where the squares are pullbacks:*



Suppose that the internal presheaf X "internalizes" the functor G: $\mathbf{C}^{op}\rightarrow\mathbf{Set}$ (and F: d→c is an "internalization" for F: $\mathbf{D}\rightarrow\mathbf{C}$). Then, $G(F(a)) = \{x\in X \mid \rho_0(x)=f_0(a)\} = \{y\in Y \mid \sigma_0(y)=a\}$ by definition of the pullback for Y, and, if h: a→b, one has $G(F(h)) = \lambda x\in F(b).\rho_1(x,f_1(h)) = \lambda x\in F(b).\sigma_1(x,h)$ by definition of σ_1.

All the definitions given so far were directed towards the following crucial notion, which will enable us to define the concept of internal Cartesian closed category.

7.3.5 Definition *$<F, G, \phi> : c\rightarrow d$ is an **internal adjunction** from c to d iff F is an internal functor from c to d, G is an internal functor from d to c and*

$$\phi : (F\times Id_d^{op})^*(hom_d) \rightarrow (Id_c\times G^{op})^*(hom_c)$$

is an isomorphism between presheaves on $c\times d^{op}$.

The definition of adjunction in 7.3.5 is now easily generalized to the case with parameters.

7.3.6 Definition *$<F, G, \phi> : c\rightarrow d$ is an **internal adjunction** from c to d **with parameters** in a iff F is an internal functor from $c\times a$ in d, G is an internal functor from $a^{op}\times d$ in c and*

$$\phi : (F\times Id_d^{op})^*(hom_d) \rightarrow (Id_c\times G^{op})^*(hom_c)$$

is an isomorphism between presheaves on $c\times a\times d^{op}$.

We can also give an "equational" characterization of internal adjunctions, in the spirit of theorem 5.3.5.

7.3.7 Theorem *Every internal adjunction* $< F, G, \phi > : c \to d$ *is fully determined by the following data in (i) or (ii):*

i. *- the functor* $G: d \to c$

- an arrow $f_0: c_0 \to d_0$

- an arrow Unit: $c_0 \to c_1$ *such that* $DOM \circ Unit = id$, $COD \circ Unit = g_0 \circ f_0$

- an arrow $\phi^{-1}: Y \to X$, *where X and Y are respectively the pullbacks of*

$<DOM,COD> : d_1 \to d_0 \times d_0$, $f_0 \times id: c_0 \times d_0 \to d_0 \times d_0$

$<DOM,COD> : c_1 \to c_0 \times c_0$, $id \times g_0: c_0 \times d_0 \to c_0 \times c_0$

and, moreover, the previous functions satisfy the following equations:

a. $< \rho_0, COMP \circ < g_1 \circ \Pi_X, Unit \circ p_1 \circ \rho_0 >_0 >_0 \circ \phi^{-1} = id_Y$

b. $\phi^{-1} \circ < \rho_0, COMP \circ < g_1 \circ \Pi_X, Unit \circ p_1 \circ \rho_0 >_0 >_0 = id_X$

ii. *- the functor* $F: c \to d$,

- an arrow $g_0: d_0 \to c_0$,

- an arrow Counit: $d_0 \to d_1$ *such that* $DOM \circ Counit = f_0 \circ g_0$, $COD \circ Counit = id$

- an arrow $\phi: X \to Y$, *where X and Y are respectively the pullbacks of*

$<DOM,COD> : d_1 \to d_0 \times d_0$, $f_0 \times id : c_0 \times d_0 \to d_0 \times d_0$

$<DOM,COD> : c_1 \to c_0 \times c_0$, $id \times g_0 : c_0 \times d_0 \to c_0 \times c_0$

and moreover the previous functions satisfy the following equations:

a. $< \rho_0', COMP \circ < Counit \circ p_2 \circ \rho_0', f_1 \circ \Pi_Y >_0 >_0 \circ \phi = id_X$

b. $\phi \circ < \rho_0', COMP \circ < Counit \circ p_2 \circ \rho_0', f_1 \circ \Pi_Y >_0 >_0 = id_Y$

Proof See the appendix to this chapter.

We are finally ready to define internal Cartesian closed categories.

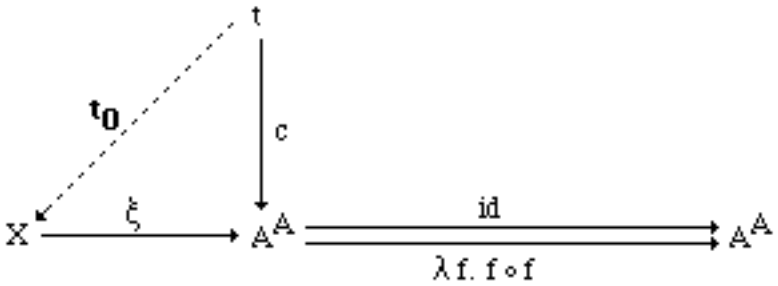
7.3.8 Definition *An* ***internal Cartesian closed category*** *is a category* $c \in Cat(E)$ *with three adjunctions, the third one with parameter in c:*

1. $< O, T, \mathbb{0} > : c \to 1$, where 1 is the internal terminal category.
2. $< \Delta, x, \lessgtr > : c \to c \times c$, where Δ is the internal diagonal functor.
3. $< x, [,] , \mathbb{\Lambda} > : c \to c$, where this adjunction has parameters in c.

7.3.9 Examples 1. In example 2 in 7.2.2, we defined the internal category $\mathbf{Ret_A} \in Cat(E)$ of retractions on a generic object A of E, where E is a CCC with all finite limits. We now prove that if A is a reflexive object, that is, if $A^A < A$, then $\mathbf{Ret_A}$ is Cartesian closed.

Let $A^A < A$ via (in,out). By theorem 2.3.6 we know that $t < A$ and $A \times A < A$. Call these retractions (in',out') and (in",out"), respectively.

Let us begin with the internal terminal object in $\mathbf{Ret_A}$. The idea is that every constant function is a terminal object in a category of retractions. Since $t < A$ via (in',out'), in': $t \to A$ is a point of A and, thus, we can take $in' \circ out'$: $A \to A$ as the constant function we are looking for; moreover, $c = \Lambda(in' \circ out' \circ p_2) : t \to A^A$ is the point in A^A that represent it. Then the internal terminal object $\mathbf{t_0}$: $t \to X$ is defined by the following:

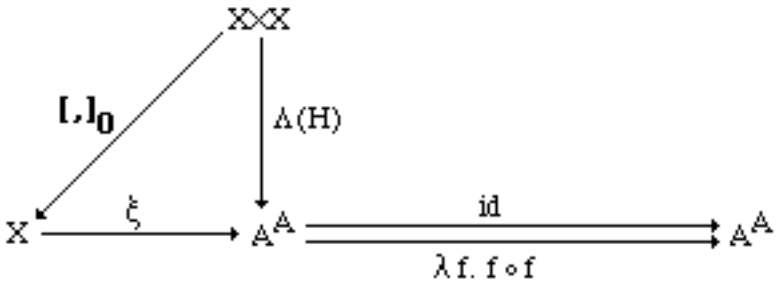


We leave it to the reader to check the soundness of the previous definition, as well as the definition of internal products, and we move on to exponents.

The first notion we must define is the arrow $\mathbf{[,]_0}$: $X \times X \to X$. The idea is that, given two retractions f, g, their exponent is the retraction $\mathbf{[f,g]_0} = \lambda a.in(\xi(g) \circ out(a) \circ \xi(f))$. Let

$$H = \lambda(f,g)\lambda a.\ in(\ \xi(g) \circ out(a) \circ \xi(f)\ ) :\ (X \times X) \times A \to A\ .$$

Then $\mathbf{[,]_0}$: $X \times X \to X$ is formally defined by the following diagram:



The function **EVAL**: $X \times X \to Y$ is the internal Counit of the adjunction; it takes two retractions f and g, and gives a morphism $\mathbf{EVAL}_{f,g}$ from the retraction $\mathbf{[f,g]_0 x_0} f$ to the retraction g (where $\mathbf{x_0}$ is the internal product on objects). More specifically, if

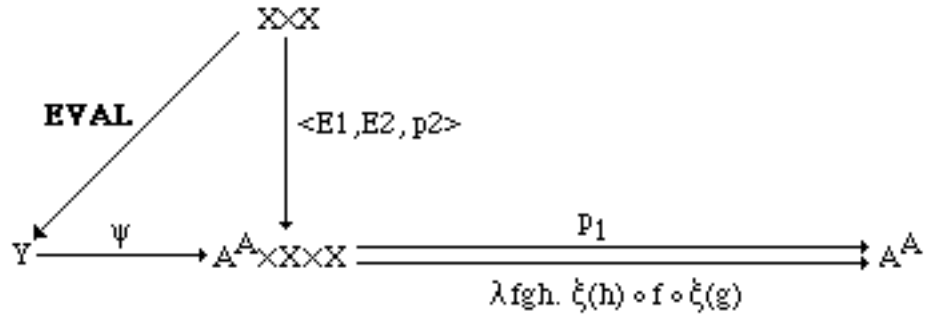
$E = \lambda(f,g)\lambda a{:}[f,g]_0 x_0 f.\ out(\mathbf{FST}(a))(\mathbf{SND}(a)) : X \times X \times A \to A$

(where $\lambda a{:}h.M$ is shorthand for $\lambda a.[h(a)/a]M$, and **FST**, **SND** are the internal projections)

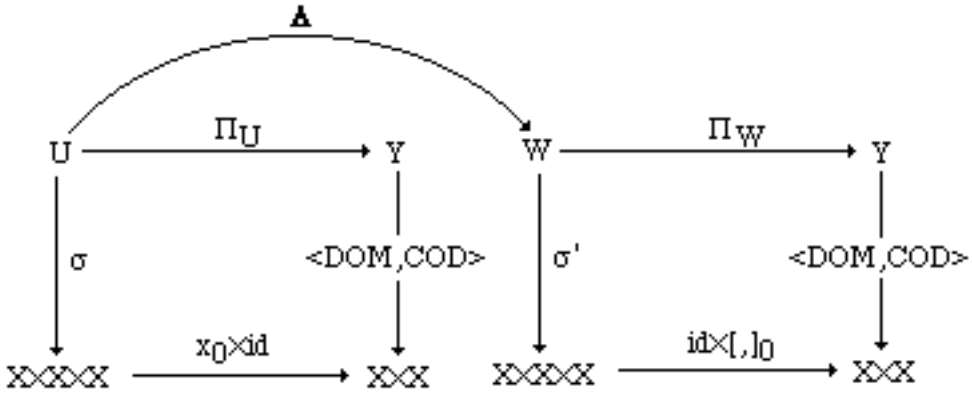
$E_1 = \Lambda(E) : X \times X \to A^A$

$E_2 = \mathbf{x_0} \circ \langle \mathbf{[,]_0}, p_1 \rangle : X \times X \to X.$

Then **EVAL:** $X \times X \to Y$ is defined by the following commutative diagram:



We must now define **Λ**: U → W, where U and W are the pullbacks in the following diagram:



Informally **Λ** works on tuples of the kind (f,g,h, (r, f**x**$_{\mathbf{0}}$g, h)) where f,g,h are retractions and r is a morphism from f**x**$_{\mathbf{0}}$g to h, that is r: A→A such that r = h ∘ r ∘ f**x**$_{\mathbf{0}}$g.

Now, let Curry(r) = λy.in(λz.(r ∘ in")(z,y)): A→A. Then Curry(r) is a morphism from g to **[f,h]**$_{\mathbf{0}}$: indeed, by omitting for simplicity the function ξ: X→A^A, we have

[f,h]$_{\mathbf{0}}$ ∘ λy.in(λz.(r ∘ in")(z,y)) ∘ g =

= λa.in(h ∘ out(a) ∘ f) ∘ λy.in(λz.(r ∘ in")(z,g(y)))

= λy.in(h ∘ λz.(r ∘ in")(z,g(y)) ∘ f)

= λy.in(λz.(h ∘ r ∘ in")(f(z),g(y)))

= λy.in(λz.(h ∘ r ∘ f**x**$_{\mathbf{0}}$g ∘ in")(z,y))

= λy.in(λz.(r ∘ in")(z,y))

Let Curry = λr.λy.in(λz.(r ∘ in")(z,y)): A^A →A^A.

Then F = < Curry ∘ p$_1$ ∘ ψ ∘ Π$_U$, id×**[,]**$_{\mathbf{0}}$ ∘ σ > : U→A^A×X×X .

But we have already verified that

p$_1$ ∘ F = (λfgh. ξ(h) ∘ f ∘ ξ(g)) ∘ F

and, thus, there exists a unique morphism **F**: U→Y such that F = ψ ∘ **F**.

Finally **Λ**= <s, **F**>$_0$: U → W.

2. This example continues example 7.2.7, where we defined **PER** as an internal category of the category **ω-Set**. We still need to check that the internal category **PER** of **ω-Set** is an internal CCC. In general, observe that in order to "internalize" a categorical construction, as we did for the category

of retractions, say, one has to turn implicit set-theoretic functional dependencies into morphisms of the intended global category **E**. For example, consider the map $\mathbf{\Lambda}$ that gives the internal natural isomorphism for Cartesian closure. Externally, $\mathbf{\Lambda}$ is implicitly indexed by objects a, b, for instance, and the map $a,b \vdash \mathbf{\Lambda}_{a,b}$ is simply a function in **Set**. The internal version, requires only that the map $\mathbf{\Lambda}$, depending also on a and b, is a morphism in **E**.

The result, that **M** is an internal CCC of **ω-Set**, then follows by the uniformity and effectiveness of the argument for the Cartesian closure of **PER**. Namely, one only has to observe that $eval_{A,B}$ is realized by any index e of the partial recursive universal function (and hence we could set $e_{A,B} = e$ in the example). Thus, not only $eval_{A,B}$ is realized, but the construction is internal to **ω-Set** as it depends on A,B by a constant function (or e is independent of A, B). Thuis is also the case for $\mathbf{\Lambda}_{A,B}$, since it is uniformly realized by any index of the function s of the s-m-n iteration theorem, independently of A, B.

7.4 Externalization

In this section, we define the process of *externalization* of an internal category via hom-functors that correspond, essentially, to the Yoneda embedding. Since for any object e of E the hom functor $[e,\_]: E \to \mathbf{Set}$ preserves pullbacks, it transforms an internal category $c = (c_0, c_1, DOM, COD, COMP, ID) \in Cat(E)$ into a small category $[e,c] = ([e,c_0], [e,c_1], [e,DOM], [e,COD], [e,COMP], [e,ID])$.

More generally, if $c \in Cat(E)$, then $[\_,c] \in Cat(E^{op} \to \mathbf{Set})$, and, for the uniform behavior with respect to the indexes in E, [_,c] can be also regarded as an E-indexed category, that is, a functor $E^{op} \to \mathbf{Cat}$. In the next section we show that, conversely, every E-indexed category can be regarded as an internal category in $E^{op} \to \mathbf{Set}$.

7.4.1 Definition *Let* $c = (c_0, c_1, DOM, COD, COMP, ID) \in Cat(E)$, *then* $[e,c] = ([e,c_0], [e,c_1], [e,DOM], [e,COD], [e,COMP], [e,ID])$.

The objects of [e,c] are the arrows $\sigma \in E[e, c_0]$. Given two objects σ, τ, a morphism $f: \sigma \to \tau$ in [e,c] is an arrow $f \in E[e, c_1]$ such that $DOM \circ f = \sigma$, $COD \circ f = \tau$. The identity of σ is $id_\sigma = ID \circ \sigma$. Let c_2 be the object of composable maps of c, that is the pullback $c_1 \times_0 c_1$ of COD and DOM. Since the hom-functor $[e,\_]: E \to \mathbf{Set}$ preserves pullbacks, $[e,c_2]$ is the pullback of [e,COD] and [e,DOM], and [e,COMP]: $[e,c_2] \to [e,c_1]$ has the expected type. Given two arrows $f: \sigma \to \tau$, $g: \tau \to \gamma$ in [e,c], their composition by [e,COMP] is $g \circledcirc f = COMP \circ \langle g,f \rangle_0$. In case the ambient category E has small hom-sets, the category [e,c] is obviously small.

Note that, if $c,d \in Cat(E)$, then $[e,c \times d] \cong [e,c] \times [e,d]$ and $[e,c^{op}] \cong [e,c]^{op}$.

In the previous definition, e can be regarded as a parameter, yielding a functor [_,c] : $E^{op}\to$**Cat**, that is, an E-indexed category.

7.4.2 Definition *Let c∈Cat(E). The functor [_,c] : $E^{op}\to$Cat is defined in the following way:*

on objects e∈E [_,c] = [e,c]

on arrows σ: e'→e [_,c](σ) = [σ,c] *is the functor from [e,c] in [e',c] that is defined as [σ,c_0] on objects and as [σ,c_1] on arrows.*

More explicitly, the functor [σ,c] takes every τ∈[e,c] (i.e., τ: e→c_0) to τ ∘ σ, and every g: τ→τ' to g ∘ σ.

We have to prove as follows that the previous definition makes sense:

1. ∀σ: e'→e, [σ,c]: [e,c] → [e',c] is a functor, for
   - 1.1. ∀τ: e→c_0 $[\sigma,c](id_\tau) = [\sigma,c](ID\circ\tau) = ID\circ\tau\circ\sigma = id_{\tau\circ\sigma}$
   - 1.2. ∀f: δ→γ , ∀g: ρ→δ in [e,c]

     $[\sigma,c](f \mathbin{⊙} g) = COMP\circ\langle f, g\rangle\circ\sigma = COMP\circ\langle f\circ\sigma, g\circ\sigma\rangle = [\sigma,c](f) \mathbin{⊙} [\sigma,c](g)$
2. [_,c] : E^{op} → Cat is a functor, for
   - 2.1. ∀e [_,c] (id_e) = I : [e,c] → [e,c] (immediate by definition of [_,c])
   - 2.2. ∀σ: e→e' , ∀τ: e'→e", [_,c](τ ∘ σ) = [_,c](σ) ∘ [_,c](τ) : [e,c] → [e",c]; indeed,
     - 2.2.1. on objects γ∈[e,c]: [_,c](τ ∘ σ)(γ) = γ ∘ τ ∘ σ = [_,c](σ)([_,c](τ)(γ))
     - 2.2.1. on arrows g τ→τ' in [_,c]: [_,c](τ ∘ σ)(g) = g ∘ τ ∘ σ = [_,c](σ)([_,c](τ)(g))

Note that if c = (c_0, c_1, DOM, COD, COMP, ID) is an internal category in E, then ([_,c_0], [_,c_1], [_,DOM], [_,COD], [_,COMP], [_,ID]) is an internal category in $E^{op}\to$**Set**.

Definitions 7.4.3 and 7.4.4 show how to externalize, respectively, an internal functor, an internal natural transformation, and an internal presheaf. Again these definitions, as well as others in the sequel, are parametric with respect to the object e of E .

7.4.3 Definition *Let c,d∈Cat(E), F = (f_0,f_1): c→d be an internal functor, and let e be an object of E. The functor [e,F]: [e,c]→ [e,d] is defined as [e,f_0] on objects, and as [e,f_1] on arrows.*

That is, the functor [e,F]: [e,c] → [e,d] takes every object σ in [e,c] to f_0 ∘ σ in [e,d], and every arrow g: σ→τ in [e,c] to f_1∘ g: (f_0 ∘ σ)→(f_0 ∘ τ) in [e,d].

7.4.4 Definition *Let $c,d \in Cat(E)$ and let $F = (f_0,f_1): c \to d$ be an internal functor. The E-indexed functor $[\_,F]: [\_,c] \to [\_,d]$ is the natural transformation defined by $[\_,F](e) = [e,F]$, for every object e of E.*

We must prove the naturality in e of the previous definition; that is, for any $\sigma: e' \to e$,

$$[e',F] \circ [\sigma,c] = [\sigma,d] \circ [e,F] .$$

We have, for any object τ of $[e,c]$ (i.e, $\tau: e \to c_0$),

$$\begin{aligned} [e,F][\sigma,c](\tau) &= [e',F](\tau \circ \sigma) && \text{by def. of } [\sigma,c] \\ &= f_0 \circ \tau \circ \sigma && \text{by def. of } [e',F] \\ &= [\sigma,d] \circ f_0 \circ \tau && \text{by def. of } [\sigma,d] \\ &= [\sigma,d] \circ [e,F] && \text{by def. of } [e,F]. \end{aligned}$$

7.4.5 Definition *Let $\tau: F \to G$ be an internal natural transformation, where $F,G: c \to d$. The natural transformation $[e,\tau]: [e,F] \to [e,G]$ is defined as the homonimous function $[e,\tau]: [e,c_0] \to [e,d_1]$; that is, it takes every object σ of $[e,c]$ to $[e,\tau](\sigma) = \tau \circ \sigma : (f_0 \circ \sigma) \to (g_0 \circ \sigma)$ (where the last "typing" is in $[e,c]$).*

Exercise Prove that the previous definition makes sense, that is:

1. $[e,\tau](\sigma) : [e,F](\sigma) \to [e,G](\sigma)$
2. for every $h: \sigma \to \gamma$ in $[e,c]$, $[e,G](h) \circledcirc [e,\tau](\sigma) = [e,\tau](\gamma) \circledcirc [e,F](h)$.

7.4.6 Definition *Let $\tau: F \to G$ be an internal natural transformation, where $F,G: c \to d$. The E-indexed natural transformation $[\_,\tau]: [\_,F] \to [\_,G]$ is defined by the following: for any object e of E, $[e,\tau]: [e,F] \to [e,G]$.*

Now we will show how to externalize the notion of *morphism of presheaves*.

7.4.7 Definition *Let $X=(X, \rho_0, \rho_1)$ be an internal presheaf on $c \in Cat(E)$. The functor $[e,X]: [e,c]^{op} \to$ **Set** is defined by:*

$\forall \sigma \in [e,c]$, $\quad [e,X](\sigma) = \{ f \in E[e,X] \,/\, \rho_0 \circ f = \sigma \}$

$\forall g: \tau \to \sigma$ *in* $[e,c]$, $\quad [e,X](g) : [e,X](\sigma) \to [e,X](\tau)$ *is given by:*

$\forall f \in [e,X](\sigma) \quad [e,X](g)(f) = \rho_1 \circ \langle f, g \rangle_0 \in [e,X](\tau)$

(note that $\rho_0 \circ [e,X](g)(f) = \rho_0 \circ \rho_1 \circ \langle f,g \rangle_0 = DOM \circ \Pi_2 \circ \langle f,g \rangle_0 = DOM \circ g = \tau$)

We have chosen the name $[e,X]$ as an analogy for the previous constructions, but in this case it no longer has a direct relation with the Yoneda embedding. The same holds below for the externalization $[e,\eta]$ of a morphism of presheaves η.

Next we check that by externalizing an internal $\hom_c$ on $c\times c^{op}$ we just obtain the hom-functor from $[e,c]^{op}\times[e,c]$ to **Set**.

7.4.8 Proposition *Let $c\in Cat(E)$ and let $hom_c = (c_1, \rho_0, \rho_1)$ be the internal hom-functor on $c\times c^{op}$. Then, for every $e\in Ob_E$, $[e,hom_c] = hom_{[e,c]}: [e,c]^{op}\times[e,c]\to$ **Set** (to within the implicit isomorphism $[e,c]^{op}\times[e,c] \cong [e,c^{op}\times c]$).*

Proof

- on objects: let $\langle\sigma,\tau\rangle: e \to c_0\times c_0$

$$\begin{aligned}[e,\hom_c](\langle\sigma,\tau\rangle) &= \{f: e\to c_1 \;/\; \rho_0 \circ f = \langle\sigma,\tau\rangle\}\\ &= \{f: e\to c_1 \;/\; \langle DOM,COD\rangle \circ f = \langle\sigma,\tau\rangle\}\\ &= \hom_{[e,c]}(\sigma,\tau);\end{aligned}$$

- on morphisms: let $\langle f,g\rangle : \langle\sigma,\tau\rangle\to \langle\gamma,\delta\rangle$ in $[e,c^{op}\times c]$. $\forall h\in[e,\hom_c](\langle\gamma,\delta\rangle)$, i.e., for all $h: e\to c_1$ such that $\langle DOM,COD\rangle \circ h = \langle\gamma,\delta\rangle$, we have

$$\begin{aligned}[e,\hom_c](\langle f,g\rangle)(h) &= \rho_1 \circ \langle h, \langle f,g\rangle\rangle_0\\ &= COMP \circ \langle p_2 \circ \Pi_2, COMP \circ (id \times_0 p_1)\rangle_0 \circ \langle h, \langle f,g\rangle\rangle_0\\ &= COMP \circ \langle g, COMP \circ \langle h, f\rangle_0\rangle_0\\ &= g \mathbin{\circledcirc} h \mathbin{\circledcirc} f\ . \ \blacklozenge\end{aligned}$$

The next definition finally externalizes the notion of morphism of presheaf that simply becomes a natural transformation. Proposition 7.4.10 states that the composition of an internal functor with a morphism of presheaf, given by the pulling back construction of definition 7.3.3, externalizes to the composition of the two associated external functors.

7.4.9 Definition *Let η be a morphism of presheaves from $X = (X, \rho_0, \rho_1)$ to $Y = (Y, \sigma_0,\sigma_1)$, where X an Y are internal presheaves on c. The natural transformation $[e,\eta]: [e,X]\to[e,Y]$ (where $[e,X], [e,Y]: [e,c]^{op}\to$ **Set**) is defined in the following way: $\forall\gamma\in[e,c]$, $\forall f\in[e,X](\gamma)$, $[e,\eta](\gamma)(f) = \eta \circ f$ (note that $[e,\eta](\gamma)(f) \in[e,Y](\gamma)$, since $\sigma_0 \circ \eta \circ f = \rho_0 \circ f = \gamma$).*

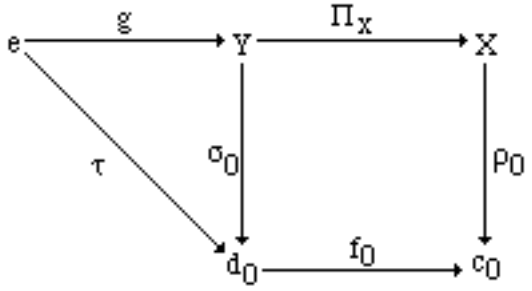
$[e,\eta]$ is indeed a natural transformation, since, $\forall g: \tau\to\gamma$ in $[e,c]$, $\forall f\in[e,X](\gamma)$

$$\begin{aligned}[e,Y](g)\,([e,\eta](\gamma)(f)) &= [e,Y](g)\,(\eta \circ f) && \text{by def. of } [e,\eta]\\ &= \sigma_1 \circ \langle \eta \circ f, g\rangle_0 && \text{by def. of } [e,Y](g)\\ &= \sigma_1 \circ \eta\times_0 id \circ \langle f, g\rangle_0 &&\\ &= \eta \circ \rho_1 \circ \langle f, g\rangle_0 && \text{by the ``naturality'' of } \eta\\ &= \eta \circ ([e,X](g)(f)) && \text{by def. of } [e,X](g)\\ &= [e,\eta](\tau)\,([e,X](g)(f)) && \text{by def. of } [e,\eta]\end{aligned}$$

7.4.10 Proposition *Let $F: d \to c$ be an internal functor, $X = (X, \rho_0, \rho_1)$ an internal presheaf on c, and $F^*(X) = (Y, \sigma_0, \sigma_1)$. For every object e of E, the functors $[e,F^*(X)]$ and $[e,X] \circ [e,F]^{op} : [e,d]^{op} \to$ **Set** are naturally isomorphic. The isomorphism is*

$$\eta_\tau = \lambda g.\, \Pi_X \circ g \;:\; [e,F^*(X)](\tau) \to [e,X]([e,F]^{op}(\tau))$$
$$\eta_\tau^{-1} = \lambda h.\langle \tau, h\rangle_0 : [e,X]([e,F]^{op}(\tau)) \to [e,F^*(X)](\tau)$$

Proof Let us check first that η_τ and η_τ^{-1} have the correct types.
By definition $[e,F^*(X)](\tau) = \{g \in E[e,Y] \,/\, \sigma_0 \circ g = \tau\}$. Let $g \in [e,F^*(X)](\tau)$. Then the following diagram commutes:



Thus $\Pi_X \circ g \in [e,X](f_0 \circ \tau) = [e,X]([e,F]^{op}(\tau))$

Conversely, let $h \in [e,X]([e,F]^{op}(\tau))$. Then the arrow $\langle \tau, h\rangle_0 : e \to Y$ is well defined, because $\rho_0 \circ h = f_0 \circ \tau$. By definition of σ_0, $\sigma_0 \circ \langle \tau, h\rangle_0 = \tau$ which implies $\langle \tau, h\rangle_0 \in [e,F^*(X)](\tau)$.
We now prove the naturality of η_τ and η_τ^{-1}. Let $k: \gamma \to \tau$ in $[e,d]^{op}$; for every $g \in [e,F^*(X)](\tau)$

$$\begin{aligned}
[e,X]([e,F]^{op}(k))(\eta_\tau(g)) &= [e,X](f_1 \circ k)(\Pi_X \circ g) && \text{by def. of } [e,F]^{op}\\
&= \rho_1 \circ \langle \Pi_X \circ g, f_1 \circ k\rangle_0 && \text{by def. of } [e,X]\\
&= \rho_1 \circ \Pi_X \times_0 f_1 \circ \langle g, k\rangle_0 &&\\
&= \Pi_X \circ \sigma_1 \circ \langle g, k\rangle_0 && \text{by def. of } \rho_1\\
&= \Pi_X \circ ([e,F^*(X)](k)(g)) && \text{by def. of } [e,F^*(X)]\\
&= \eta_\gamma([e,F^*(X)](k)(g)) && \text{by def. of } \eta
\end{aligned}$$

Conversely, for every $k: \gamma \to \tau$ in $[e,d]^{op}$ and every $h \in [e,X]([e,F]^{op}(\tau))$:

$$[e,F^*(X)](k)(\eta_\tau^{-1}(h)) = \sigma_1 \circ \langle \eta_\tau^{-1}(h), k\rangle_0 = \sigma_1 \circ \langle \langle \tau, h\rangle_0, k\rangle_0$$

Thus: $\Pi_X \circ ([e,F^*(X)](k)(\eta_\tau^{-1}(h))) = \Pi_X \circ \sigma_1 \circ \langle \langle \tau, h\rangle_0, k\rangle_0$
$= \Pi_X \circ \sigma_1 \circ \langle \langle \tau, h\rangle_0, k\rangle_0$
$= \rho_1 \circ \Pi_X \times_0 f_1 \circ \langle \langle \tau, h\rangle_0, k\rangle_0$
$= \rho_1 \circ \langle \Pi_X \circ \langle \tau, h\rangle_0, f_1 \circ k\rangle_0$
$= \rho_1 \circ \langle h, f_1 \circ k\rangle_0$

and $\sigma_0 \circ ([e,F^*(X)](k)(\eta_\tau^{-1}(h))) = \sigma_0 \circ \sigma_1 \circ \langle \langle \tau, h\rangle_0, k\rangle_0$
$= DOM \circ \Pi_2 \circ \langle \langle \tau, h\rangle_0, k\rangle_0$
$= DOM \circ k$
$= \gamma$

And since $f = \langle \sigma_0 \circ f, \Pi_X \circ f\rangle$, then for every $f : e \to Y$,

$$\begin{aligned}[e,F^*(X)](k)\,(\,\eta_\tau^{-1}(h)\,) &= \langle\gamma,\ \rho_1\circ\langle h,\ f_1\circ k\rangle_0\rangle_0\\ &= \langle\gamma,\ [e,X]([e,F]^{op}(k))\,(h)\rangle_0\\ &= \eta_\gamma^{-1}(\ [e,X]([e,F]^{op}(k))\,(h)\,).\quad\blacklozenge\end{aligned}$$

7.4.11 Proposition *Let* $\langle F, G, \phi\rangle : c\to d$ *be an internal adjunction. For every* e *in* E, *define* $\Theta_e = \eta'\circ[e,\phi]\circ\eta^{-1}$, *where*

$$\eta : [e,(F\times Id_{d^{op}})^*(hom_d)] \to [e, hom_d]\circ[e,F^{op}\times Id]$$
$$\eta' : [e,(Id_c\times G^{op})^*(hom_c)] \to [e, hom_c]\circ[e,Id\times G^{op}]$$

are the isomorphisms of proposition 7.4.10 .

Then $\langle[\_,F], [\_,G], \Theta\rangle : [\_,c]\to[\_,d]$ *is an E-indexed adjunction.*

Proof For every object e of E, we have

$$\begin{aligned}hom_{[e,d]}[[e,F](\_),\ \_\ ] &= [e,\ hom_d]\circ[e,F^{op}\times Id]\\ &\cong [e,(F\times Id_{d^{op}})^*(hom_d)] && \text{via } \eta^{-1}\\ &\cong [e,(Id_c\times G^{op})^*(hom_c)] && \text{via } [e,\phi]\\ &\cong [e,\ hom_c]\circ[e,Id\times G^{op}] && \text{via } \eta'\\ &= hom_{[e,c]}[\_,\ E_{e,G}(\_)\ ].\end{aligned}$$

Moreover, the previous adjunction is "natural in e," that is,

$$\forall f\in E[e',e]\quad \Theta_{e'}\circ[\_,d](f) = [\_,c](f)\circ\Theta_e\ .$$

More explicitly, we must check that, for every $f\in E[e',e]$, σ object of $[e,c]$, τ object of $[e,d]$, and $g: (f_0\circ\sigma)\to\tau$ in $[e,d]$, one has

$$\Theta_{e'}\langle\sigma\circ f,\ \tau\circ f\rangle\,([\_,d](f)\,)\,(g) = ([\_,c](f)\,)\,\Theta_e\langle\sigma,\tau\rangle\,(g)$$

We have

$$\begin{aligned}\Theta_{e'}\langle\sigma\circ f,\ \tau\circ f\rangle\,([\_,d](f)\,)\,(g) &= \Theta_{e'}\langle\sigma\circ f,\ \tau\circ f\rangle\,(g\circ f\,) && \text{by def. of } [\_,d]\\ &= \Theta_{e'}\langle\sigma\circ f,\ \tau\circ f\rangle\,(g\circ f\,)\\ &= \Pi_X\circ\phi\circ\langle\langle\sigma\circ f,\tau\circ f\rangle, g\circ f\rangle_0 && \text{by def. of } \Theta_{e'}\\ &= \Pi_X\circ\phi\circ\langle\langle\sigma,\tau\rangle, g\rangle_0\circ f\\ &= (\Theta_e\langle\sigma,\tau\rangle\,(g)\,)\circ f && \text{by def. of } \Theta_e\\ &= ([\_,c](f)\,)\,\Theta_e\langle\sigma,\tau\rangle\,(g) && \text{by def of } [\_,c].\quad\blacklozenge\end{aligned}$$

7.4.12 Exercise Prove that if $\langle F, G, \phi\rangle : c\to d$ is an internal adjunction, and Unit and Counit are the arrows in theorem 7.3.7, than for every object $\sigma: e\to d_0$ in $[e,d]$, Unit $\circ\,\sigma$, Counit $\circ\,\sigma$ are respectively unit and counit for σ in the associated external adjunction $\langle[e,F], [e,G], \Theta_e\rangle$: $[e,c]\to[e,d]$.

7.5 Internalization

In this section we show how to translate (small) E-indexed notions to internal ones in the topos of presheaves $E^{op}\to$**Set**.

7.5.1 Definition *Let* $A: E^{op}\to$***Cat*** *be an E-indexed category, where all the indexed categories are small. The internal category* $\underline{A} = (\underline{A_0}, \underline{A_1}, \underline{DOM}, \underline{COD}, \underline{COMP}, \underline{ID}) \in Cat(E^{op}\to$***Set***$)$ *is defined as follows: for all objects* e, e' *and arrows* $f: e'\to e$ *in E,*

- $\underline{A_0}: E^{op}\to$***Set*** *is the functor defined by*
$$\underline{A_0}(e) = Ob_{A(e)}$$
$$\underline{A_0}(f) = A(f)_{ob} : Ob_{A(e)}\to Ob_{A(e')}$$
- $\underline{A_1}: E^{op}\to$***Set*** *is the functor defined by*
$$\underline{A_1}(e) = Mor_{A(e)}$$
$$\underline{A_1}(f) = A(f)_{mor} : Mor_{A(e)}\to Mor_{A(e')}$$
- $\underline{DOM}: A_1\to A_0$ *is the natural transformation whose components are the domain maps in the local categories, i.e., for* $e\in Ob_E$, $\underline{DOM}_e: Mor_{A(e)}\to Ob_{A(e)}$ *is defined by* $\underline{DOM}_e(h:\sigma\to\tau) = \sigma$.
- $\underline{COD}$, $\underline{ID}$ *and* $\underline{COMP}$ *are defined analogously, "fiberwise".*

The claimed naturality for $\underline{DOM}$, $\underline{COD}$, $\underline{ID}$, $\underline{COMP}$ is immediate, since A is a functor. For instance, let $f\in E[e',e]$ and $h\in A(e)[\sigma,\tau]$; then $\underline{DOM}_{e'}(A(f)_{mor}(h)) = A(f)_{ob} \circ \underline{DOM}_e(h)$. The reader can check the other cases as an exercise.

7.5.2 Definition *Let A, B be two E-indexed categories, and let* $H: A\to B$ *be an E-indexed functor.The associated* ***internal functor*** $\underline{H} = (\underline{H_0},\underline{H_1}): \underline{A}\to\underline{B}$ *in* $E^{op}\to$***Set***, *is defined in the following way:*

- $\underline{H_0}: \underline{A_0}\to\underline{B_0}$ *is the natural transformation given by* $\underline{H_0}(e) = H(e)_{ob}$
- $\underline{H_1}: \underline{A_1}\to\underline{B_1}$ *is the natural transformation given by* $\underline{H_1}(e) = H(e)_{mor}$

The naturality of $\underline{H}_0$ and $\underline{H}_1$ is an immediate consequence of the "naturality" of H: A→B, that is $H(s) \circ A(f) = B(f) \circ H(s')$. The equations in definition 7.2.3 easily follow from the fact that for every e, H(e) is a functor.

7.5.3 Definition *Let* $H: A\to B$, $K: A\to B$ *be two E-indexed functors, and let* $\tau: H\to K$ *be an E-indexed natural transformation. Then the associated internal natural transformation* $\underline{\tau}: \underline{H}\to\underline{K}$ *in* $E^{op}\to$***Set*** *is the natural transformation* $\underline{\tau}: \underline{A_0}\to\underline{B_1}$ *such that, for any e in E, and any a in A(e),* $\underline{\tau}_e(a) = \tau(e)_a$.

Recall that $\tau: H \to K$ consists of a natural transformation $\tau(e): H(s) \to K(e)$ for any object e of E, such that, for any $f: e \to e'$ in E, and any object a in $A(e')$, $\tau(e)_{A(f)(a)} = B(f)(\tau(e')_a)$.

As a consequence, $\underline{\tau}_e(A_0(f)(a)) = B_1(f)(\underline{\tau}_{e'}(a))$, which gives the naturality of $\underline{\tau}$.

7.5.4 Proposition *Let A, B be E-indexed categories, $H: A \to B$, $K: B \to A$ be E-indexed functors, and $\langle H, K, \phi \rangle : A \to B$ be an E-indexed adjunction. Then $\langle \underline{H}, \underline{K}, \underline{\phi} = \phi \rangle : \underline{A} \to \underline{B}$ is an internal adjunction in $E^{op} \to$* ***Set*** .

Proof Exercise.

The picture is finally completed by the following result, which shows that by applying the externalization process of section 7.4 to an internal category $\underline{A}$, derived from an E-indexed category A, we obtain an indexed category equivalent to A. However, when we externalize $\underline{A}$, we do not want a category indexed over *all* functors from E into **Set**. Since we are interested in a category indexed over E, we must externalize only with respect to a full subcategory of $E^{op} \to$ **Set** equivalent to E. The obvious choice is to consider the image Y(E) of E under the Yoneda embedding Y(e) = E[_, e] (recall that Y(E) is a full subcategory of $E^{op} \to$ **Set**). We will then obtain an indexed category $\underline{A}^{\#} : E^{op} \to$ **Cat**. Recall though that the "internalization" can take place only if the indexed category takes small categories as values, while internal categoies do not need to live in small ambient categories. Thus, the circle is closed by the following theorem, provided that the assumption is made that E is small.

7.5.5 Theorem *Let $A: E^{op} \to$* ***Cat*** *be an E-indexed category, with E small, and let $\underline{A} \in Cat(E^{op} \to$* ***Set****) be its associated internal category. Then the indexed categories $\underline{A}^{\#} = [Y(\_), \underline{A}] : E^{op} \to$* ***Cat*** *and A are equivalent.*

Proof Let $e \in Ob_E$. Then $\underline{A}^{\#}(e) = [Y(e), \underline{A}]$ is, by definition, the category with

Objects: $Nat[E[\_,e], \underline{A}_0]$

Morphisms: $g: \sigma \to \tau$ in $[Y(e), \underline{A}]$ iff

$g \in Nat[E[\_,e], \underline{A}_1]$, $\underline{DOM} \circ g = \sigma$, $\underline{COD} \circ g = \tau$

Now let $f \in E[e',e]$; by definition one has

$\underline{A}^{\#}(f) = [Y(f), \underline{A}] : \underline{A}^{\#}(e) \to \underline{A}^{\#}(e')$

$\underline{A}^{\#}(f)(\sigma) = \sigma \circ Y(f)$ for $\sigma \in Nat[E[\_,e], \underline{A}_0]$

$\underline{A}^{\#}(f)(g) = g \circ Y(f)$ for $g \in Nat[E[\_,e], \underline{A}_1]$

The natural isomorphism between A and $\underline{A}^{\#}$ is given by the Yoneda lemma: for every e in E, we have natural isomorphisms $\Psi_0(e): Nat[E[\_,e], \underline{A}_0] \to \underline{A}_0(e)$ and $\Psi_1(e): Nat[E[\_,e], \underline{A}_0] \to \underline{A}_1(e)$. Ψ_0 and Ψ_1 define the components on objects and morphisms of an indexed functor $\Psi(e): \underline{A}^{\#}(e) \to A(e)$. Explicitly,

$$\Psi(e)(\sigma) = \sigma_e(id_e) \quad \text{for } \sigma \in Nat[E[\_,e], \underline{A}_0]$$

$$\Psi(e)(g) = g_e(id_e) \quad \text{for } g \in Nat[E[\_,e], \underline{A}_1].$$

Then the due diagrams commute, by the usual Yoneda argument.

Our final result shows that, by following the other path (from internal to internal, via external), one obtains equivalent categories:

7.5.6 Theorem *Let $c \in Cat(E)$ be an internal category, $C = [\_,c]: E^{op} \to \mathbf{Cat}$ be as in definition 7.4.2, and $Y: E \to Y(E)$ be the Yoneda embedding. Then $\underline{C} \in Cat(Y(E))$.*

Proof Let $c = (c_0, c_1, DOM, COD, COMP, ID)$; note first that, by definition of C, for the internal category $\underline{C} = (d_0, d_1, DOM', COD', COMP', ID') \in Cat(E^{op} \to \mathbf{Set})$ we have

$d_0 = E[\_, c_0] = Y(c_0)$

$d_1 = E[\_, c_1] = Y(c_1)$

and hence $\underline{C} \in Cat(Y(E))$, since Y is full. That $\underline{C}$ is an internal category, follows by the fact that Y preserves pullbacks. ♦

Appendix

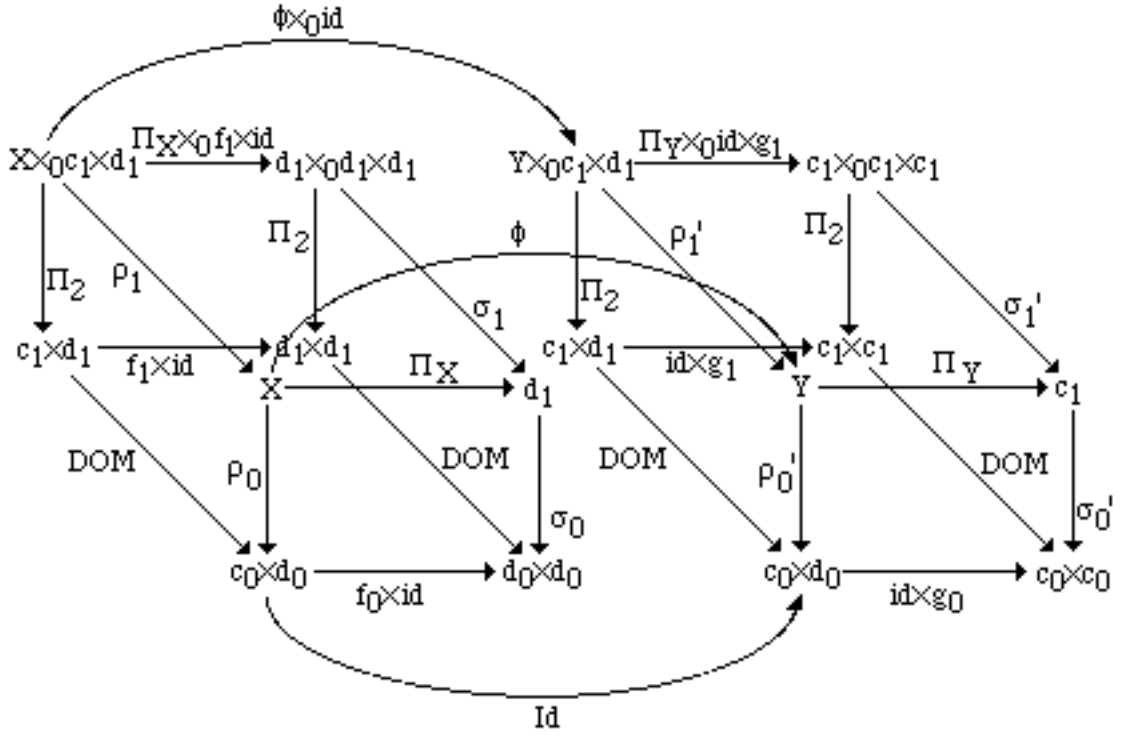
We now study in more details the notions of internal adjunction and internal CCC. The details are rather complex and this appendix may be skipped at first reading.

By definition, an internal adjunction $< F, G, \phi > : c \to d$ is given by two internal functors $F: c \to d$, $G: d \to c$, and an isomorphism

$$\phi : (F \times Id_d^{op})^*(hom_d) \to (Id_c \times G^{op})^*(hom_c)$$

between presheaves on $c \times d^{op}$.

Graphically, the notion of internal adjunction is represented by the following complex diagram:



where (d_1,σ_0,σ_1) is the internal hom-functor of d, and $(c_1,\sigma_0',\sigma_1')$ is the internal hom-functor of c. In particular,

$\sigma_0 = \langle DOM,COD\rangle : d_1 \to d_0\times d_0$ and

$\sigma_0' = \langle DOM,COD\rangle : c_1 \to c_0\times c_0$

respectively represent d_1 and c_1 as indexed collections of morphisms over $d_0\times d_0$ and $c_0\times c_0$. The formal definition of σ_1 and σ_1' is:

$\sigma_1 = COMP \circ \langle p_2 \circ \Pi_2, COMP \circ (id \times_0 p_1)\rangle_0 : d_1\times_0(d_1\times d_1) \to d_1$

$\sigma_1' = COMP \circ \langle p_2 \circ \Pi_2, COMP \circ (id \times_0 p_1)\rangle_0 : c_1\times_0(c_1\times c_1) \to c_1$

More intuitively, they are both described by the lambda term $\lambda fgh.\ h \circ f \circ g$ (recall that $hom[f,g](h) = h \circ f \circ g$).

Note also that $DOM: c_1\times d_1 \to c_0\times d_0 = DOM_c\times COD_d$ because we are working in $c\times d^{op}$.

X and Y are respectively the pullbacks of

$\sigma_0 = \langle DOM,COD\rangle : d_1 \to d_0\times d_0$, $f_0\times id : c_0\times d_0 \to d_0\times d_0$ and

$\sigma_0' = \langle DOM,COD\rangle : c_1 \to c_0\times c_0$, $id\times g_0 : c_0\times d_0 \to d_0\times d_0$

Thus, informally,

$X = \{ (a,b,h) \in c_0\times d_0\times d_1 \mid h : f_0(a) \to b \} = d[f_0(a), b]$

$Y = \{ (a,b,k) \in c_0\times d_0\times c_1 \mid k : a \to g_0(b) \} = c[a, g_0(b)]$.

$\phi : X \to Y$ is the natural isomorphism of the adjunction.

ϕ works on triples of the kind $(a,b,h)\in c_0\times d_0\times d_1$ where $h: f_0(a)\to b$. The first two components a and b are the indexes of the natural trasformation: since $\rho_0'\circ \phi = \rho_0$, these indexes are left unchanged by ϕ, and an "external-like" writing for $\phi(a,b,h)$ would be $\phi_{a,b}(h)$. At the external level, it is common practice to omit these indexes; the formal complexity of the internal theory is mostly due to the necessity of coping with these details.

The naturality of ϕ is expressed by the property,

(†) $\qquad \phi \circ \rho_1 = \rho_1'\circ \phi\times_0 id$.

Still using our informal notation, by (†), for all (a,b,h) in X, k in c_1 and l in d_1, such that :

$cod(k) = a$ (that implies $cod(f_0(k)) = f_0(a) = dom(h)$)

$dom(l) = b = cod(h)$

$cod(l) = b'$

we have

(∗) $\qquad \phi_{a',b'}(l \circ h \circ f_1(k)) = g_1(l) \circ \phi_{a,b}(h) \circ k,$

that is the familiar way the naturality of ϕ is expressed at the external level. Let us show, in this informal notation, that (†) implies (*)

$\phi_{a',b'}(l \circ h \circ f_1(k)) =$

$= (\Pi_Y \circ \phi)(a', b' , l \circ h \circ f_1(k))$

$= (\Pi_Y \circ \phi)(\ a', b'\ ,\ \sigma_1(h,\ f_1(k),\ l\ )\ \ )$ by def. of σ_1

$= (\Pi_Y \circ \phi \circ \rho_1)\ ((a,b,h),\ k,\ l\ )$ by the diagram for the adjunction

$= (\Pi_Y \circ \rho_1' \circ \phi \times_0 id)\ (\ (a,b,h),\ k,\ l\ )$ by (†)

$= (\sigma_1' \circ \Pi_Y \times_0 (id \times g_1) \circ \phi \times_0 id)\ (\ (a,b,h),\ k,\ l\ )$ by the diagram for the adjunction

$= \sigma_1'(\ (\Pi_Y \circ \phi)(a,b,h),\ k,\ g_1(\ l\ )\ )$

$= \sigma_1'(\ \phi_{a,b}(h),\ k,\ \ g_1(\ l\ )\ )$

$= g_1(\ l\ ) \circ \phi_{a,b}(h) \circ k$ by def. of σ_1'.

Given an adjunction $< F, G, \phi > : C \to D$, the arrows $\phi_{a,F(a)}(id_{F(a)})$ and $\phi_{G(b),b}^{-1}(id_{G(b)})$ are respectively called Unit and Counit of the adjunction (for a and b). Units and Counits fully specify the behaviour of ϕ and ϕ^{-1} since:

$$\phi(\ l\ ) = \phi(\ l \circ id\ ) = g_1(\ l\ ) \circ \phi(id) = \ g_1(\ l\ ) \circ \text{Unit}$$
$$\phi^{-1}(k) = \phi^{-1}(id \circ k) = \phi^{-1}(id) \circ F(k) = \text{Counit} \circ F(k)\ .$$

These properties allow to give at the external level the well-known equational characterization of the notion of adjunction. In particular, the definition of Cartesian closed category based on the counits of the adjunctions, plays a central role in the semantic investigation of the lambda calculus, since it provides the underlying applicative structure needed for the interpretation. Remember that the counits of the adjunctions defining products and exponents are respectively the projections associated with the products and the evaluation functions associated with the function spaces.

Now we show how to mimic the same work at the internal level.

7.A.1 Definition *Let $< F, G, \phi > : c \to d$ be an internal adjunction from c to d. Define then:*

$\boldsymbol{ID_F} = \langle\langle id, f_0\rangle, ID \circ f_0 \rangle_0 : c_0 \to X;$

$\boldsymbol{ID_G} = \langle\langle g_0, id\rangle, ID \circ g_0 \rangle_0 : d_0 \to Y;$

$\boldsymbol{Unit} = \ \Pi_Y \circ \phi \circ ID_F : c_0 \to c_1;$

$\boldsymbol{Counit} = \Pi_X \circ \phi^{-1} \circ ID_G : d_0 \to d_1.$

Where X and Y are as in the diagram for the definition of adjunction.

Note that ID_F takes an element a in c_0 and gives the associated identity $id_{F(a)}$ as an element in X. The definition of Unit, is then clear. As one expects, Unit is an internal natural transformation from $I = (id,id)$ to $G \circ F$, and Counit : $d_0 \to d_1$ is an internal natural transformation from $F \circ G$ to $I = (id,id)$. The proof is left as an exercise for the reader.

It is now not difficult to prove that every internal adjunction $< F, G, \phi > : c \to d$ is fully determined by the following data:

the functor $G: d \to c$;

an arrow $f_0: c_0 \to d_0$;

an arrow Unit: $c_0 \to c_1$ such that $DOM \circ Unit = id$, $COD \circ Unit = g_0 \circ f_0$;

an arrow $\phi^{-1}: Y \to X$, where X and Y are respectively the pullbacks of

$\langle DOM,COD \rangle : d_1 \to d_0 \times d_0$, $f_0 \times id: c_0 \times d_0 \to d_0 \times d_0$, and

$\langle DOM,COD \rangle : c_1 \to c_0 \times c_0$, $id \times g_0: c_0 \times d_0 \to c_0 \times c_0$;

and, moreover, the previous functions satisfy the following equations:

a. $\langle \rho_0, COMP \circ \langle g_1 \circ \Pi_X, Unit \circ p_1 \circ \rho_0 \rangle_0 \rangle_0 \circ \phi^{-1} = id_Y$;

b. $\phi^{-1} \circ \langle \rho_0, COMP \circ \langle g_1 \circ \Pi_X, Unit \circ p_1 \circ \rho_0 \rangle_0 \rangle_0 = id_X$.

Indeed the arrow $f_0: c_0 \to d_0$ can be extended to a functor $F = (f_0, f_1): c \to d$ by

$f_1 = \Pi_X \circ \phi^{-1} \circ \langle\langle DOM, f_0 \circ COD \rangle, COMP \circ \langle Unit \circ COD, id \rangle_0 \rangle_0 : c_1 \to d_1$.

The inverse of ϕ^{-1} is

$\phi = \langle \rho_0, COMP \circ \langle g_1 \circ \Pi_X, Unit \circ p_1 \circ \rho_0 \rangle_0 \rangle_0$.

Note that, by (a) and (b), ϕ and ϕ^{-1} define an isomorphism. The non trivial fact is to prove that they are morphisms of presheaves (i.e., to prove their naturality), but again the prof is a mere internal rewriting of the corresponding "external" result.

Dually, if we have the following data:

a functor $F: c \to d$;

an arrow $g_0: d_0 \to c$;

an arrow $Counit: d_0 \to d_1$ such that $DOM \circ Counit = f_0 \circ g_0$, $COD \circ Counit = id$;

an arrow $\phi: X \to Y$, where X and Y are respectively the pullbacks of

$\langle DOM,COD \rangle : d_1 \to d_0 \times d_0$, $f_0 \times id: c_0 \times d_0 \to d_0 \times d_0$, and

$\langle DOM,COD \rangle : c_1 \to c_0 \times c_0$, $id \times g_0: c_0 \times d_0 \to c_0 \times c_0$;

and, moreover, the previous functions satisfy the following equations:

a. $\langle \rho_0', COMP \circ \langle Counit \circ p_2 \circ \rho_0', f_1 \circ \Pi_Y \rangle_0 \rangle_0 \circ \phi = id_X$,

b. $\phi \circ \langle \rho_0', COMP \circ \langle Counit \circ p_2 \circ \rho_0', f_1 \circ \Pi_Y \rangle_0 \rangle_0 = id_Y$,

then we define an adjunction $\langle F, G, \phi \rangle : c \to d$, in the following way.

The arrow $g_0: d_0 \to c_0$ can be extended to a functor $G = (g_0, g_1): c \to d$, by

$g_1 = \Pi_Y \circ \phi \circ \langle\langle g_0 \circ DOM, COD \rangle, COMP \circ \langle id, Counit \circ DOM \rangle_0 \rangle_0 : d_1 \to c_1$.

The inverse of ϕ is

$\phi^{-1} = \langle \rho_0', COMP \circ \langle Counit \circ p_2 \circ \rho_0', f_1 \circ \Pi_Y \rangle_0 \rangle_0 : Y \to X$.

We are now in a position to study internal Cartesian closed categories from an "equational" point of view. This work is needed to exploit the applicative structure underlying the notion of an internal CCC. Recall that an internal Cartesian closed category is a category $c \in Cat(E)$ with three adjunctions

1. $\langle O, T, \mathbb{0} \rangle : c \to 1$, where 1 is the internal terminal category;
2. $\langle \Delta, x, \ll,\gg \rangle : c \to c \times c$, where Δ is the internal diagonal functor;
3. $\langle x, [,] , \mathbb{\Lambda} \rangle : c \to c$, where this adjunction has parameters in c.

By the previous results, we can explicitate the three adjunctions of these definitions by means of their counits:

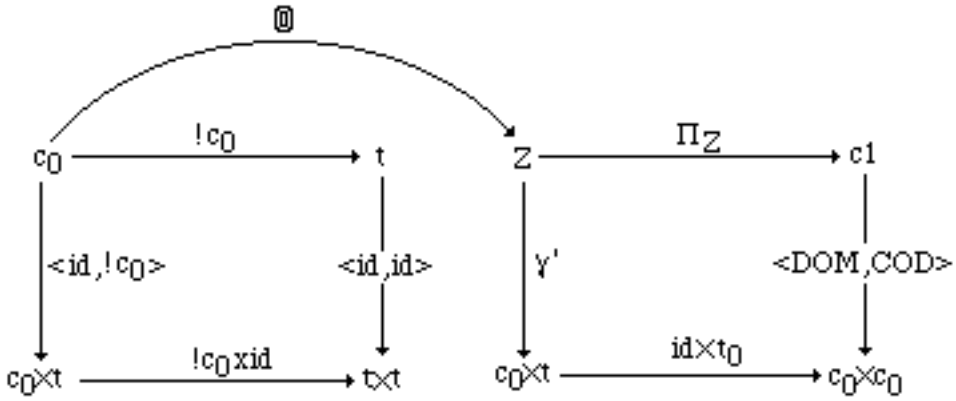
7.A.2 Definition *An **internal terminal object** in $c \in Cat(E)$ is specified by:*

an arrow $t_0 : t \to c_0$;

an arrow $\mathbb{0} : c_0 \to Z$, where Z is the pullback of

$\langle DOM,COD \rangle : c_1 \to c_0 \times c_0$,

$id \times t_0 : c_0 \times t \to c_0 \times c_0$;



and, moreover,

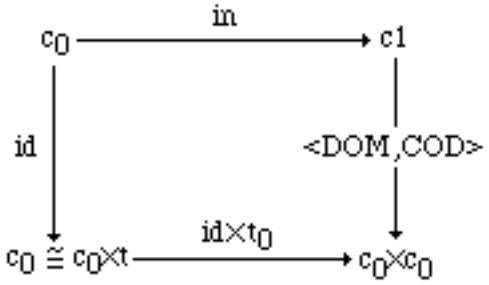
a. $\mathbb{0} \circ \langle \gamma', !Z \rangle_0 = \mathbb{0} \circ p_1 \circ \gamma' = id_Z$;

b. $\langle \gamma', !Z \rangle_0 \circ \mathbb{0} = p_1 \circ \gamma' \circ \mathbb{0} = id_{c_0}$;

where !Z is the unique morphism in E from Z to the terminal object t .

Intuitively $t_0 : t \to c_0$ points to that element in c_0 that is the terminal object. Z is the subset of c_1 of all those morphisms that have the terminal object as target; Z must then be in a bijective relation $\mathbb{0}$ with c_0; $\mathbb{0}$ takes an object a in c_0 to the unique morphism $!_a$ in Z from a to the terminal object.

The previous diagram can be greatly simplified. As a matter of fact, it amounts to say that there is an arrow $t_0 : t \to c_0$ such that the following diagram is a pullback (prove it as an exercise):



The arrow $in : c_0 \to c_1$ is the operation that takes every element a in c_0 to the unique arrow !a in c_1 whose target is the terminal object; in terms of the previous diagram, $in = \Pi_Z \circ \mathbb{0}$.

7.A.3 Definition *An internal category c* ***has products****, iff there exist*

an arrow $x_0: c_0 \times c_0 \to c_0$;

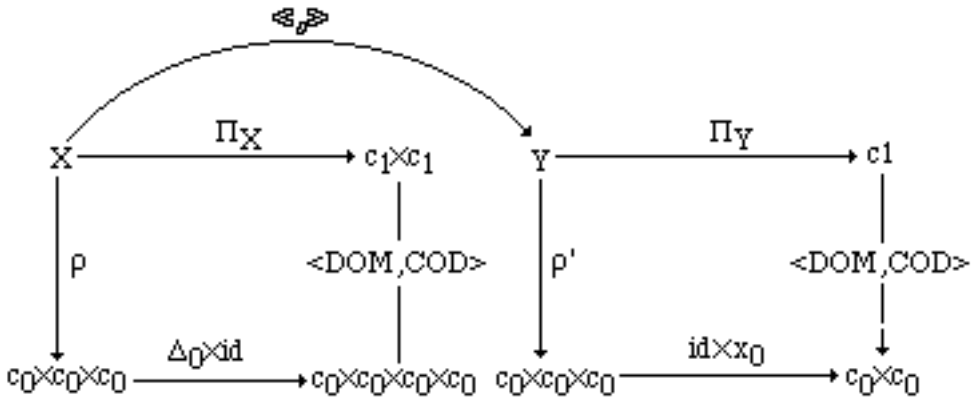
two arrows $FST: c_0 \times c_0 \to c_1$, $SND: c_0 \times c_0 \to c_1$ *such that*

$DOM \circ FST = DOM \circ SND = x_0$,

$COD \circ FST = p_1$; $COD \circ SND = p_2$,

(*Notation:* $FST_{a,b} = FST \circ \langle a,b \rangle$; $SND_{a,b} = SND \circ \langle a,b \rangle$);

an arrow $\ll , \gg : X \to Y$, *where X and Y are the pullbacks in the following diagram* ($\Delta_0 = \langle id,id \rangle$):



and, moreover,

c0. $\rho' \circ \ll , \gg = \rho$;

c1. $(FST \circ p_2 \circ \rho) \mathbin{@} (\Pi_Y \circ \ll , \gg) = p_1 \circ \Pi_X$;

c2. $(SND \circ p_2 \circ \rho) \mathbin{@} (\Pi_Y \circ \ll , \gg) = p_2 \circ \Pi_X$;

d. $\ll , \gg \circ \langle \rho', \langle (FST \circ p_2 \circ \rho') \mathbin{@} \Pi_Y , (SND \circ p_2 \circ \rho') \mathbin{@} \Pi_Y \rangle_0 \rangle_0 = id_Y$,

where $f \mathbin{@} g = COMP \circ \langle f, g \rangle_0$.

7.A.4 Definition *An internal category is* ***Cartesian*** *iff it has a terminal object and products.*

As the definition $f \times g = \langle f \circ p_1, g \circ p_2 \rangle$ extends $\times$ to a functor from C×C to C for any Cartesian C, also the internal x_0 can also be extended to morphisms.

7.A.5 Proposition *Let* $x_1 : c_1 \times c_1 \to c_1$ *be defined by the following:*
$x_1 = \Pi_Y \circ \ll , \gg \circ \langle \langle x_0 \circ DOM_{c \times c}, COD_{c \times c} \rangle, \langle id \mathbin{@} (FST \circ DOM_{c \times c}), id \mathbin{@} (SND \circ DOM_{c \times c}) \rangle_0$
where as above, $f \mathbin{@} g = COMP \circ \langle f,g \rangle_0$. *Then* $x = (x_0, x_1): c \times c \to c$ *is an internal functor.*
Proof: Exercise.

Note that, if f,g: $e \to c_1$, $DOM \circ f = a$, $COD \circ f = c$, $DOM \circ g = b$, $COD \circ g = d$, then: $x_1 \circ \langle f,g \rangle = \Pi_Y \circ \ll , \gg \circ \langle \langle x_0 \circ \langle a,b \rangle, \langle c,d \rangle \rangle, \langle f \mathbin{@} FST_{a,b}, g \mathbin{@} SND_{a,b} \rangle_0 \rangle_0$.

7.A.6 Definition *An internal Cartesian category has* ***exponents*** *iff there exist :*

an arrow $[,]_0 : c_0 \times c_0 \to c_0$;

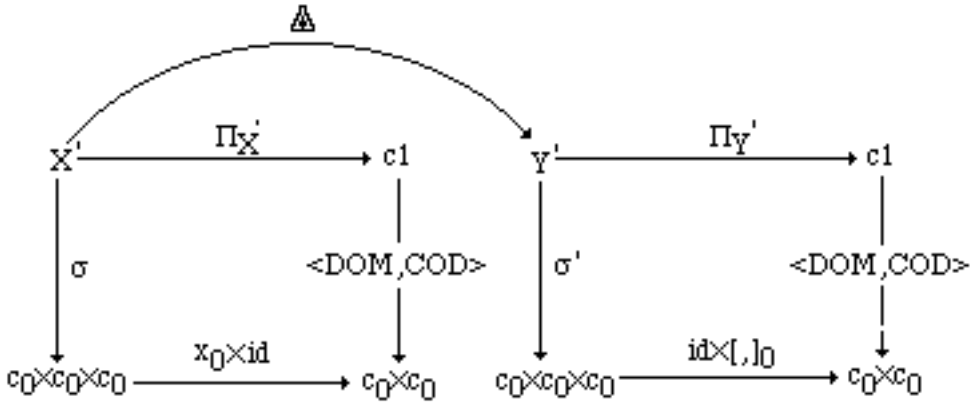
an arrow $EVAL : c_0 \times c_0 \to c_1$ *such that*

$DOM \circ EVAL = \times_0 \circ \langle [,]_0, p_1 \rangle$,

$COD \circ EVAL = p_2$,

(Notation: $EVAL_{a,b} = EVAL \circ \langle a,b \rangle$ *);*

an arrow $\Delta : X' \to Y'$, *where* X' *and* Y' *are the pullbacks in the following diagram:*



and, moreover,

e0. $\sigma' \circ \Delta = \sigma$ *(to within the isomorphism* $(a \times b) \times c \cong a \times (b \times c)$ *);*

e1. $(eval \circ p_1 \circ \sigma) \mathbin{@} (x_1 \circ \langle \Pi_{Y'} \circ \Delta, ID \circ p_2 \circ p_1 \circ \sigma \rangle) = \Pi_{X'}$;

f. $\Delta \circ \langle \sigma', (eval \circ p_2 \circ \sigma') \mathbin{@} (x_1 \circ \langle \Pi_{Y'}, ID \circ p_1 \circ p_2 \circ \sigma' \rangle) \rangle_0 = id_{Y'}$,

where $f \mathbin{@} g = COMP \circ \langle f, g \rangle_0$, *and* x_1 *is the morphism in proposition A.5.*

7.A.7 Definition *An* ***internal Cartesian closed category*** *is an internal Cartesian category with exponents.*

References The introduction of indexed notions in Category Theory has been a slow process, which started to develop into a detailed theory around the beginning of the 1970s, with the independent work of F. W. Lawvere, J. Penon, J. Bénabou. The Theory of Indexed Category owes much of its actual settlement to R. Paré and D. Schumacher (1978), and to their introductory book, written in collaboration with P. T. Johnstone, R. D. Rosebrugh, R. J. Wood and G. C. Wraith.

The first significant study of the Theory of Internal categories is due to R. Diaconescu. Further developments were made by J. Bénabou, though, most of the time, they never appeared as published works. Notions of Internal Category Theory can be found in many books of topos theory, for instance, in those of Johnstone (1977) and Barr and Wells (1985).

Our development of the arguments in this chapter has been essentially inspired by Paré and Schumacher (1978), Johnstone (1977), and some private communications of E. Moggi. The definition of the internal category in definition 7.5.1 has been pointed out to us by B. Jacobs.

Chapter 8

FORMULAE, TYPES, AND OBJECTS

During the last two decades, computer science has turned the Proof Theory of mathematical logic from a philosophical investigation of the foundations of human reasoning into an applied sector of mathematics. Many important logical systems of the beginning of the 1970s, which "fifteen years later" proved of great relevance for computer science, were invented by logicians such as Girard, Martin-Löf and Troelstra, who were mostly interested in constructive approaches to mathematics and foundational problems. Still in 1975, in the first pages of his book on Proof Theory, Takeuti introduced "the formalization of the proofs of mathematics, and the investigation of the structure of these proofs" as a mere "fruitful method in investigating mathematics." Moreover, he himself remarked that "while set theory has already contributed essentially to the development of modern mathematics, it remains to be seen what influence proof theory will have on mathematics".

If not on mathematics, Proof Theory has surely proved since that time his influence on theoretical computer science. The modern theory of functional languages and lambda calculus owes much of his actual settlement to mathematical logic, and what until a few years ago was known as the "denotational semantics of programming languages" has grown under the direct influence of Proof Theory, and together with the understanding of the logical aspects of Category Theory. But far from being only a rich field for applications, computer science has also been for mathematical logic an inexhaustible source of mutual enrichment. The leading theme of this stimulating relation, as for the topic presented in this chapter, has been the so called Curry-Howard correspondence, which exploits the connections between Proof Theory and Type Theory. This correspondence, also known by the suggestive name of "formulae-as-types analogy", has been a main methodological tool both for understanding the relations between intuitionistic logic and typed lambda calculus, and for the design of new functional languages with powerful type systems. The main idea of the Curry-Howard correspondence, is that logical formulae can be interpreted as types in a suitable type theory; a proof of a formula is then associated with a λ-term of the proper type, and the reduction of a proof by cut-elimination corresponds to the normalization of the associated λ-term. As a consequence, if a formula is derivable in a certain logical system, then the corresponding type is inhabited in the associated type theory. The Curry-Howard correspondence works only for logical systems of Intuitionistic Logic. This restriction should be clear, since the constructive, procedural interpretation of the notion of proof was the very basis of Brower's approach to mathematics, which inspired Heyting's formalization of intuitionistic logic. Moreover, although the formulae-as-types analogy can also be applied to logical systems based on axioms and inference rules, such as that of Hilbert, just switching from λ-terms to combinators in the associated type theory, it has a more elegant application to systems of Natural Deduction. Indeed the procedural understanding of a logical proof is more clear in a system like

Natural Deduction, where one proceeds by the method of drawing inferences from assumptions, than in Hilbert's system, where one draw inferences from axioms. Furthermore, especially in Gentzen's variant of Natural Deduction, the inference rules of the calculus are closely related to the intuitive "operational" interpretation of the logical signs, and this fact allows one to proceed in the construction of the proof in a certain direct fashion, affording an interesting normal form for deductions which has no clear counterpart in Hilbert's system.

8.1 λ-Notation

Consider the following mathematical definition of the function f: **N→N** :

$$f(x) = 5*x + 3 .$$

Note that it is not "f" that has been actually defined, but "f(x)," that is the result of the application of f to a formal parameter x which is supposed to range over the integers.

f(x) is not a function: it is a polynomial, that is, an operational description of the behavior of f when applied to x. The mechanism that allows us to "abstract" from the formal parameter x, and thus to pass from the knowledge of "how the function works" to the knowledge of "what the function is," is called **lambda abstraction (λ-abstraction)**. The function f in the example above is defined by the lambda abstraction of 5x + 3 with respect to x, and is denoted by λx. 5∗x+3.

The complementary action to λ-abstraction is called **application**. The application of two terms M and N is usually represented by their justaposition (MN). From the computational point of view, λ-abstraction and application are related by the following rule, known as β-reduction:

$$(\lambda x.M)N \rightarrow [N/x]M$$

where [N/x]M means the substitution of N instead of x in the term M.
For example,

$$(\lambda x.\ 5*x+3)4 \ \rightarrow \ 5*4+3 \ = \ 23$$

The lambda notation was explicitly introduced by the logician Alonzo Church in the **lambda calculus** (λ-calculus), a formalism invented to develop a general theory of computable functions, and to extend that theory with logical notions providing a foundation for (part of) mathematics.

In computer science, the lambda notation made its first appearance in the programming language LISP; this was also the first language to use procedures as objects of the language. Like the early lambda calculus, LISP is an untyped language; that is, programs and data are not distinguished, but they are all elements of a unique untyped universe: the universe of λ-terms for the λ-calculus, the universe of S-expressions for LISP (see next chapter). Anyway, as every computer scientist knows from his or her own programming practice, types arise naturally, even starting from untyped universes, when objects are categorized according to their usage and behavior. The ultimate question is whether it wouldn't be better to consider types as an *a priori* schema of human knowledge, instead of an attempt to organize subsets of objects with uniform properties out of an untyped universe. In

computer science, the debate about typed and untyped languages is very lively today, since it reflects the unresolvable conflict between reliability and flexibility of the language. Nowadays, the practical success of a language is often due to the the more or less successful attempt to compromise security and freedom; in this respect, the language ML is probably one of the most interesting examples. Anyway, since the first appearance of types in Algol 60, when typing of variables was introduced to check at compile time the connections of instances of use with associated declarations, typing has been considered more and more an essential discipline that must not only help, but guide the programmer in the design of code.

8.2 The Typed λ-Calculus with Explicit Pairs ($\lambda\beta\eta\pi^t$)

The collection Tp of type labels, over a ground set At of atomic type symbols, is inductively defined by

i. $At \subseteq Tp$;
ii. if $A,B \in Tp$, then $A \to B \in Tp$;
iii. if $A,B \in Tp$, then $A \times B \in Tp$.

For every type A there exists a denumerable number of **variables**, ranged over by lower case letters near the end of the alphabet. We use upper case letters M, N, P, . . . , as metavariables for terms. The fact that a term M has type A will be denoted with the expression "M: A."

The **well typed (λ-)terms** (w.t.t.) and their associated types, are defined according to the following formation rules:

1. every variable x: A is a w.t.t.;
2. if x: A is a variable, and M: B is a w.t.t, then $\lambda x{:}A.M{:}\ A \to B$ is a w.t.t.;
3. if M: $A \to B$ is a w.t.t and N: A is a w.t.t, then MN: B is a w.t.t.;
4. if M: A is a w.t.t and N: B is a w.t.t, then $\langle M,N\rangle{:}\ A \times B$ is a w.t.t.;
5. if M: $A \times B$ is a w.t.t, then fst(M): A is a w.t.t.;
6. if M: $A \times B$ is a w.t.t, then snd(M): B is a w.t.t.

Given a w.t.t M: B, the set **FV(M)** of the **free variables** of M, is defined as follows:

1. if $M \equiv x$, then $FV(M) = \{x\}$;
2. if $M \equiv \lambda x{:}A.N$, then $FV(M) = FV(N)-\{x\}$;
3. if $M \equiv NP$, then $FV(M) = FV(N) \cup FV(P)$;
4. if $M \equiv \langle N,P\rangle$, then $FV(M) = FV(N) \cup FV(P)$;
5. if $M \equiv fst(N)$, then $FV(M) = FV(N)$;
6. if $M \equiv snd(N)$, then $FV(M) = FV(N)$.

The **substitution [M/x]N: B** of a proof M: A for a generic x: A in a proof N: B is defined in the following way:

1. if $N: B \equiv x: A$ then $[M/x]N: B \equiv M: A$;
2. if $N: B \equiv y: A$, $x \neq y$, then $[M/x]N: B \equiv N: B$;
3. if $N: B \equiv \lambda x{:}C.P : B$, then $[M/x]N: B \equiv \lambda x{:}C.P : B$;
4. if $N: B \equiv \lambda y{:}C.P : B$, $x \neq y$, $y \notin FV(M)$, then $[M/x]N: B \equiv \lambda y{:}C.[M/x]P : B$;
5. if $N: B \equiv PQ : B$, then $[M/x]N: B \equiv [M/x]P[M/x]Q : B$;
6. if $N: B \equiv \langle P,Q\rangle : B$, then $[M/x]N: B \equiv \langle [M/x]P,[M/x]Q\rangle : B$;
7. if $N: B \equiv fst(P) : B$, then $[M/x]N: B \equiv fst([M/x]P) : B$;
8. if $N: B \equiv snd(P) : B$, then $[M/x]N: B \equiv snd([M/x]P) : B$.

As an easy consequence of the definition above, it follows that if $x \notin FV(N)$, then $[M/x]N: B \equiv N: B$. Note also that, if $x \notin FV(N)$, then $[P/x][M/y]N: B \equiv [([P/x]M)/y]N: B$.

Given a sequence $\mathbf{M} = M_1,..., M_n$ of terms, and sequence $\mathbf{x} = x_1, \ldots, x_n$ of variables, $[\mathbf{M}/\mathbf{x}]N$ denotes the simultaneous substitution of every term M_i for the variable x_i in the term N. We also use the notation $[M/\mathbf{x}]N$ to express the simultaneous substitution of the term M for all the variables in $\mathbf{x}$.

We consider an equational theory of proofs, defined as the minimal congruence relation "=" which satisfies the following axiom schemas:

(α) $\lambda x{:}A.M = \lambda y{:}A.[y/x]M$

($\rightarrow\beta$) $(\lambda x{:}A.M)N = [N/x]M$

($\rightarrow\eta$) $\lambda x{:}A.(Mx) = M$, if $x \notin FV(M)$

($\times \beta_1$) $fst(\langle M,N\rangle) = M$

($\times \beta_2$) $snd(\langle M,N\rangle) = N$

($\times \eta$) $\langle fst(P),snd(P)\rangle = P$.

The previous equations may be read from left to right : then one obtains a rewriting system, which defines the operational semantics of lambda calculus as a programming language. Explicitly, we define a **reduction relation** $\Rightarrow$ as the minimal relation between with respect to terms such that

($\rightarrow\beta$) $(\lambda x{:}A.M)N \Rightarrow [N/x]M$

($\times \beta_1$) $fst(\langle M,N\rangle) \Rightarrow M$

($\times \beta_2$) $snd(\langle M,N\rangle) \Rightarrow N$

$M \Rightarrow M'$ implies $MN \Rightarrow M'N$

$M \Rightarrow M'$ implies $NM \Rightarrow NM'$

$M \Rightarrow M'$ implies $\lambda x{:}A.M \Rightarrow \lambda x{:}A.M'$

$M \Rightarrow M'$ implies $fst(M) \Rightarrow fst(M')$

$M \Rightarrow M'$ implies $snd(M) \Rightarrow snd(M')$

($\rightarrow\eta$) $\lambda x{:}A.(Mx) \Rightarrow M$, if $x \notin FV(M)$

($\times \eta$) $\langle fst(P),snd(P)\rangle \Rightarrow P$.

The rewriting rules associated with $(\to\beta)$, $(\times\beta_1)$ and $(\times\beta_2)$ are called β-reductions, and those associated with $(\to\eta)$ and $(\times\eta)$ are called η-reductions. We put the η-reductions aside, because of their lesser computational interest. In the following section we shall see that from the point of view of the Curry-Howard correspondence both β– and η-reductions have an interpretation in terms of proof normalization. Also in this context, however, the β-rules play a major role.

A term is in **normal form** (respectively β-normal form), if it is no more reducible (respectively β-reducible). The (β-)reduction relation is confluent and noetherian.

Let us now look at some simple examples of programs written in the typed lambda calculus. As usual, when considering the formal expressiveness of a calculus, we compare it with the problem of representing integers and arithmetic functions.

Given a type σ, we call $\mathbf{N}_\sigma = (\sigma\to\sigma)\to(\sigma\to\sigma)$ the type of **σ-iterators** (**σ-numerals**). The reason for the name is that in the type $(\sigma\to\sigma)\to(\sigma\to\sigma)$ we have all the terms with the following structure:

$$0_\sigma \equiv \lambda x^{\sigma\to\sigma}\,\lambda y^{\sigma}.y$$
$$1_\sigma \equiv \lambda x^{\sigma\to\sigma}\,\lambda y^{\sigma}.xy$$
$$\dots$$
$$n_\sigma \equiv \lambda x^{\sigma\to\sigma}\,\lambda y^{\sigma}.x^n y, \text{ where } x^n y = x(...(xy)) \text{ “n times”}.$$

The effect of the term n_σ is to take a function x of type $\sigma\to\sigma$, a term y of type σ, and iterate the application of x to y n times. It is possible to show that if σ is an atomic type, then the terms of the form n_σ, together with the identity $\lambda x^{\sigma\to\sigma}.x^{\sigma\to\sigma}$ are the only closed terms in normal form of type σ.

We can now define the function $succ_\sigma \equiv \lambda n{:}\,\mathbf{N}_\sigma\,\lambda x^{\sigma\to\sigma}\,\lambda y^{\sigma}.\,x(nxy)$.

The term $succ_\sigma$ has type $\mathbf{N}_\sigma\to\mathbf{N}_\sigma$ and its effect is to increase the iteration of one unit. For example,

$$\begin{aligned} succ_\sigma\, 0_\sigma &= (\lambda n{:}\,\mathbf{N}_\sigma\,\lambda x^{\sigma\to\sigma}\,\lambda y^{\sigma}.\,x(nxy)\,)\,0_\sigma \\ &= \lambda x^{\sigma\to\sigma}\,\lambda y^{\sigma}.\,x(\,0_\sigma xy) \\ &= \lambda x^{\sigma\to\sigma}\,\lambda y^{\sigma}.\,x(y) \\ &= 1_\sigma \end{aligned}$$

Define now $add_\sigma{:}\,\mathbf{N}_\sigma\times\mathbf{N}_\sigma\to\mathbf{N}_\sigma$ by $add_\sigma \equiv \lambda z{:}\,\mathbf{N}_\sigma\times\mathbf{N}_\sigma\,\lambda f^{\sigma\to\sigma}\,\lambda y^{\sigma}.\,fst\ z\ f\ (snd\ z\ f\ y\,)$

For example, we have:

$$\begin{aligned} add_\sigma\,\langle 1_\sigma,1_\sigma\rangle &= (\lambda z{:}\,\mathbf{N}_\sigma\times\mathbf{N}_\sigma\,\lambda f^{\sigma\to\sigma}\,\lambda y^{\sigma}.\,fst\ z\ f\ (snd\ z\ f\ y\,)\,)\,\langle 1_\sigma,1_\sigma\rangle \\ &= \lambda f^{\sigma\to\sigma}\,\lambda y^{\sigma}.\,fst\,\langle 1_\sigma,1_\sigma\rangle\ f\ (snd\,\langle 1_\sigma,1_\sigma\rangle\ f\ y\,) \\ &= \lambda f^{\sigma\to\sigma}\,\lambda y^{\sigma}.\,1_\sigma\ f\ (1_\sigma\ f\ y\,) \\ &= \lambda f^{\sigma\to\sigma}\,\lambda y^{\sigma}.\,f\,(\,f\ y\,) \\ &= 2_\sigma. \end{aligned}$$

Analogously, $mult_\sigma{:}\,\mathbf{N}_\sigma\times\mathbf{N}_\sigma\to\mathbf{N}_\sigma$ is given by

$$mult_\sigma \equiv \lambda z{:}\,\mathbf{N}_\sigma\times\mathbf{N}_\sigma\,\lambda f^{\sigma\to\sigma}\lambda y^{\sigma}.\,fst\ z\ (snd\ z\ f\,)\ y.$$

As an example of computation, we have

$$\begin{aligned}
\mathrm{mult}_\sigma\ \langle 2_\sigma, 2_\sigma\rangle &= (\lambda z{:}\ \mathbf{N}_\sigma\times\mathbf{N}_\sigma\ \lambda f^{\sigma\to\sigma}\ \lambda y^{\sigma}.\ \mathrm{fst}\ z\ (\mathrm{snd}\ z\ f)\ y)\ \langle 2_\sigma, 2_\sigma\rangle \\
&= \lambda f^{\sigma\to\sigma}\ \lambda y^{\sigma}.\ \mathrm{fst}\ \langle 2_\sigma, 2_\sigma\rangle\ (\mathrm{snd}\ \langle 2_\sigma, 2_\sigma\rangle\ f)\ y \\
&= \lambda f^{\sigma\to\sigma}\ \lambda y^{\sigma}.\ 2_\sigma\ (2_\sigma\ f)\ y \\
&= \lambda f^{\sigma\to\sigma}\ \lambda y^{\sigma}.\ (2_\sigma\ f)(\ 2_\sigma\ f\ \ y\ ) \\
&= \lambda f^{\sigma\to\sigma}\ \lambda y^{\sigma}.\ f\ (\ f(\ f\ (\ f\ \ y\ )\ )\ ) \\
&= 4_\sigma.
\end{aligned}$$

As a final example, we present a test to 0_σ. That is, we want to define a term $\mathrm{test}\ 0_\sigma$ that takes two terms M and N of type σ and a σ-numeral n; if $n = 0_\sigma$ then the function yields M as result; otherwise, it outputs N. Here it is: $\mathrm{test}\ 0_\sigma \equiv \lambda z{:}\ \sigma\times\sigma\ \lambda n{:}\ \mathbf{N}_\sigma.\ n(\ \lambda y^{\sigma}.(\mathrm{snd}\ z)\ )(\mathrm{fst}\ z)$.

It is possible to prove (see references) that these are essentially all the arithmetic functions we can compute by the (pure!) simply typed λ-calculus, that is, all the functions that are defined by composition of the previous ones. For example the predecessor function is not representable.

The simply typed λ-calculus cannot be considered as a real programming language, but only as a paradigm for strongly typed (functional) languages, which allow us to point out their main common properties. In the last section of this chapter we shall consider an extension of the language by fixpoint operators. The combined use of these operators with a reasonable set of arithmetic primitives gives to the language its full computational power.

8.3 The Intuitionistic Calculus of Sequents

Among the many variants of intuitionistic logic expressed in systems of Natural Deduction, Gentzen's calculi of cequents has recently gained much popularity among the computer science community. In the first place, one of the main aspects of the procedural interpretations of proofs is that of making explicit the assumptions on which a formula occurrence in a deduction depends, and this is just what the notion of sequent is meant for. Moreover, since the calculi of sequents can be understood as metacalculi for the deducibility relation in the corresponding system of natural deduction, it seems to be the best instrument for the investigation of the structure and the properties of proofs. Note, however, that even when handling sequents, the dependencies of a formula by its premises can be quite hard to understand, if one adopts some common choices on the structure of the sequent, such as avoiding repetitions of formulae in the antecedent, or fixing a certain alphabetic order for them. This is very clear if we consider a sequent as an effective process for transforming proofs of the premises into a proof of the consequence. It seems not only reasonable, but fair to guarantee the ability of handling two or more different proofs for a same formula. In particular, this means allowing repetitions for a formula in the antecedent of a sequent, each one with an associated hypothetical distinct proof. From the same point of view, even the idea of an ordering among the formulae does not make any sense; what one really needs, are rather some explicit rules in the calculus which allow us “to move formulae around” (exchange rules). The presence of repetitions of formulae requires a

new rule too, stating that two proofs of a formula can actually be assumed to be the same, and thus affording to discharge one of them (contraction rule). These rules, together with the so-called weakening rule, that allows for adding new assumptions to a sequent, are usually referred to as "structural rules." The reason for this name should be clear, since these rules have nothing to do with logical signs and their meanings, but only with the structures of the derivations in the calculi. This property of the structural rules has been often misunderstood, and regarded as a sign of their lesser logical interest: Takeuti called them "weak inferences," with respect to the "strong inferences" of introduction of the logical connectives. The very recent revival of the structural rules, and the analysis of their impact on the logical systems is mostly due to Girard, and to the invention of linear logic (see chapter 4).

The calculi of sequents for classical and intuitionistic logic were introduced by Gentzen at the beginning of the 1930s, who called them L systems. In these systems the elementary statements are statements of deducibility. Each statement $\Gamma \vdash B$ (called a **sequent**) has a conclusion B and a collection Γ, possibly void, of premises, and it is interpreted as stating the existence of a proof leading to the stated conclusion and having no uncanceled premises other than some of those stated.

We consider only the subcalculus that deals with the two connectives of conjunction and implication, and that will be referred to as "positive calculus." The principal connective of the system is implication; the role of conjunction is rather minor, but its analysis will help in clarifying some issues we will be concerned with in the rest of this chapter.

The following presentation of the positive intuitionistic calculus of sequents is slightly different from the usual one. In particular, every formula A will be equipped with an associated proof, formally represented by a lambda term $M: A$. This association between proofs (λ-terms) and formulae is defined in the inference rules of the calculus. If there is a derivation of the sequnt $\Gamma \vdash M: B$, the proof M keeps a trace of the derivation tree. On the other hand, the so-modified inference rules can be understood as metarules for proof manipulation. This understanding of the calculus of sequents as a metacalculus for the deducibility relation is one of the most peculiar aspect of this formal system.

8.3.1 Logical Alphabet

1. atomic propositions, ranged over by $A, B, C, \ldots$;
2. logical symbols: $\times, \rightarrow$.

8.3.2 W.F.Formulae (Types)

1. every atomic proposition is a formula;
2. if A, B are formulae, then $A \times B$ is a formula;
3. if A, B are formulae, then $A \rightarrow B$ is a formula.

There is a bijective correspondence between formulae and types of the typed lambda calculus with explicit pairs. We can go further, and associate every formula B with a λ-term M of the respective type, which intuitively represents its proofs. In particular, if $x_1: A_1,..., x_n: A_n$ are the free variables in M, then $M: B$ is a proof of M depending on the hypothesis $A_1,..., A_n$. The previous approach is sound with the intuitionistic interpretation of the notion of proof as an effective procedure that shows the validity of the conclusion B, as soon as one has a proof of the validity of the premises. Thus, if M is a proof of B, possibly depending on a generic proof $x: A$, one gets a proof of $A \rightarrow B$ by "abstraction on $x: A$," that is $\lambda x{:}A.M: A \rightarrow B$. Juxtaposition of proofs corresponds intuitively to their sequential application. In particular, if we have a proof $M: A \rightarrow B$, and we apply it to a proof $N: A$, then we obtain a proof MN of B. A proof of $A \times B$ is given by a pair of distinct proofs for A and B. Moreover, having a proof of $A \times B$, one can select the two proofs of A and B by means of the projections fst and snd .

The inference rules of the calculus exactly formalize this process of constructiing complex proofs by means of simpler ones.

A (well-typed) term $M: A$, will be called a **proof** when it is regarded from a logical viewpoint. A variable $x: A$, will be called a **generic proof** of A.

An intuitionistic **sequent** has the following syntactic structure:

$$x_1: A_1, \ldots, x_n: A_n \vdash M: B$$

where $x_1: A_1, \ldots, x_n: A_n$ is a finite (possibly empty) list of *distinct* generic proofs, and $M: B$ is a proof of B whose free variables are among $x_1, \ldots, x_n$. Every formula in the left-hand-side (l.h.s.) has an associated distinct variable; thus no confusion can arise between formulae, even if they have the same name.

The intuitive interpretation of the sequent $x_1: A_1, \ldots, x_n: A_n \vdash M: B$ is that of a process which builds a proof M of B, as soon as it has proofs for $A_1, \ldots, A_n$, that is, a function f of type $A_1 \times \ldots \times A_n \rightarrow B$, or equivalently, $A_1 \rightarrow (\ldots (A_n \rightarrow B) \ldots)$, which can be obtained by functional completeness from the polynomial M.

Since the name of a generic proof is not relevant, we assume that sequents which differ only by a renaming of variables are syntactically identified. This is consistent with the α-conversion rule of proofs, in view of the intuitive interpretation of sequents sketched above.

We use Greek capital letters $\Gamma, \Delta, \ldots$ to denote finite sequences of generic proofs in the left hand side of a sequent. A sequent thus has the form $\Gamma \vdash M: B$.

An **inference** is an expression

$$\frac{S_1}{S} \qquad\qquad \frac{S_1 \quad S_2}{S}$$

where S_1, S_2, and S are sequents. S_1 and S_2 are called the upper sequents and S is called the lower sequent of the inference. Intuitively, this means that when S_1 (S_1 and S_2) is (are) asserted, one can infer S from it (them).

We always suppose that the variables in the l.h.s. of the upper sequents are distinct. This is not a problem, since the variables of the upper sequents S_1 and S_2 that contribute to form the l.h.s. of the lower sequent S can be conveniently renamed.

The logical system restricts the legal inferences to those obtained from the following rules.

8.3.3 Axioms and Rules

$$\text{(axiom)} \qquad x: A \vdash x: A$$

$$\text{(exchange)} \quad \frac{\Gamma, x: A, y: B, \Gamma_1 \vdash M: C}{\Gamma, y: B, x: A, \Gamma_1 \vdash M: C}$$

$$\text{(weakening)} \quad \frac{\Gamma \vdash M: B}{\Gamma, x: A \vdash M: B}$$

$$\text{(contraction)} \quad \frac{\Gamma, x: A, y: A \vdash M: B}{\Gamma, z: A \vdash [z/x][z/y]M: B}$$

$$\text{(cut)} \quad \frac{\Gamma \vdash M: A \qquad \Gamma_1, x: A, \Gamma_2 \vdash N: B}{\Gamma_1, \Gamma, \Gamma_2 \vdash [M/x]N: B}$$

$$(\rightarrow, r) \quad \frac{\Gamma, x: A \vdash M: B}{\Gamma \vdash \lambda x{:}A.M: A \rightarrow B} \qquad (\rightarrow, l) \quad \frac{\Gamma_1 \vdash M: A \qquad \Gamma_2, x: B \vdash N: C}{\Gamma_1, \Gamma_2, y: A \rightarrow B \vdash [yM/x]N: C}$$

$$(\times, r) \quad \frac{\Gamma \vdash M: A \qquad \Gamma \vdash N: B}{\Gamma \vdash \langle M,N \rangle : A \times B} \qquad (\times, l, 1) \quad \frac{\Gamma, x: A \vdash M: C}{\Gamma, z: A \times B \vdash [fst(z)/x]M: C}$$

$$(\times, l, 2) \quad \frac{\Gamma, y: B \vdash M: C}{\Gamma, z: A \times B \vdash [snd(z)/x]M: C}$$

Remark Note the identification of the premises of the upper sequents in the rule (×, r).

The formula A in the weakening rule is called the **weakening formula** (of the inference). Similarly, we define the **cut formula**. In the case of the contraction rule we distinguish between the **contracting formulae** x: A, y: A and the **contracted formula** z: A. The formulae A→B and A×B in the logical rules of introduction of the respective connectives, are called the **principal formulae** of the inference; A and B are called the **auxiliary** formulae.

Unlike the usual approach, we intend the previous definition to refer only to the specific occurrences of the formulae pointed out in the inference rules. For instance, consider the following application of the rule (→, l):

$$(\rightarrow, l)\qquad \frac{\Gamma_1 \vdash M: A \qquad w: A\rightarrow B,\ x: B \vdash N: C}{\Gamma_1,\ w: A\rightarrow B,\ y: A\rightarrow B \vdash [yM/x]N: C}$$

The (occurrence of the) formula A→B associated with the proof y is a principal formula of this inference, but that associated with w is not.

It is easy to check that, for every rule of inference, the proof in the right-hand side of the lower sequents is well formed, if those of the upper sequents are as well. A **derivation** D is a tree of sequents satisfying the following conditions:

1. the topmost sequents of D are axioms;
2. every sequent in D except the lowest one (the root) is an upper sequent of an inference whose lower sequent is also in D.

The lower sequent in a derivation D is called the **end-sequent** of D. A path in a derivation D from a leaf to the root is called a **thread**. A derivation without the cut rule is called **cut-free**.

Examples The following is a derivation:

$$\dfrac{\dfrac{x: A \vdash x: A}{z: A\times B \vdash fst(z): A} \qquad \dfrac{y: B \vdash y: B}{z: A\times B \vdash snd(z): B}}{z: A\times B \vdash \langle fst(z),snd(z)\rangle: A\times B}$$

The sequence

x: A |- x: A

z: A×B |- fst(z): A

z: A×B |- <fst(z),snd(z)>: A×B

is a thread.

The tidy display of the introduction of new formulae during a derivation in Gentzen's calculus of sequent, allows us to follow the entire "life" of a formula, from the moment it is "created" (by an axiom, a weakening, or as a principal formula of a logical rule) to the moment it is "consumed" (by a cut, or as an auxiliary formula in a logical rule). Note in particular that a step of contraction must not be regarded as a consumption, but as an identification of a formula A in the lower sequent, with two formulae of the same name in the upper sequent.

Given a thread T and an occurrence of a formula A in the final sequent of T, we call **rank** of A in T the number of consecutive sequents in T, counting upward, that contain the same occurrence of the formula A (or a formula which has been contracted to it).

For instance, consider again the thread

x: A |- x: A

z: A×B |- fst(z): A

z: A×B |- <fst(z),snd(z)>: A×B

of the previous example. The rank of the formula A×B in the l.h.s. of the end sequent is 2, while the rank of the formula A×B in the r.h.s. is 1.

8.4 The Cut-Elimination Theorem

One of the more interesting properties of Gentzen's calculus of sequent is the cut-elimination theorem, also known as Gentzen's Hauptsatz. This result proves that every derivation in the calculus can be effectively transformed in a "natural normal form" which does not contain any application of the cut rule. Moreover we show that this process of reduction of the derivations toward a normal form is in parallel to the β-reduction of the associated proofs.

The identification of the derivations reducible to a same normal form is the base of the so-called "semantics of proofs," and, *a posteriori*, it provides a justification for the equational theory we have introduced over the language of proofs (at least for the β-conversions).

The following presentation of the cut-elimination theorem is meant to study the relations between a derivation ending in the sequent Γ |- M: A and the proof M: A. Indeed, it should be clear that given a proof M: A, with FV(M) ⊆ Γ, it is possible to obtain a derivation of the sequent Γ |- M: A by building a sort of "parse tree" for the term M: A . Anyway, this correspondence between derivations and proofs is not a bijection: the problems arise with the structural rules, since their application is not reflected in the terms. In particular, it is not possible to recover the order in which the structural rules have been applied during the derivation.

The exact formalization of the equivalence of derivations that differ only in "structural details" is not at all trivial, and it is the main goal of Girard's research of "a geometry of interaction". From this respect, the language of proofs is a very handy formalism that allows us to abstract from such structural details.

Before the cut-elimination theorem, we state a first, simple result that relates cut-free derivations and proofs in β-normal form.

8.4.1 Proposition *Let* $\Gamma \vdash M: A$ *be the end sequent of a cut-free derivation. Then the proof* $M: A$ *is in β-normal form.*

Proof Exercise. *Hint:* Show, by induction on the length of the derivation, that the proof M: A cannot contain any of the following subterms:

1. $(\lambda x:B.N)P: C$
2. fst(<N,P>)
3. snd(<N,P>). ♦

One of the aims of this chapter is to put in evidence the logical aspects underlying functional type theories. For this reason we state and prove Gentzen's Hauptsatz and stress the equivalence of the normalization of a derivation D ending in a sequent $\Gamma \vdash M: A$ and the β-reduction of M.

8.4.2 Theorem (The Cut-Elimination Theorem) *Let* D *be a derivation of the sequent* $\Gamma \vdash M: A$ *. Then there exists a cut-free derivation* D' *of the sequent* $\Gamma \vdash N: A$*, where* N *is the β-normal form of* M*.*

It is difficult to appreciate at first sight the cut-elimination theorem in its whole meaning. Indeed, it looks like a mere result about the redundancy of one of the inference rules in the calculus, and one would expect, as a next step, to drop the cut rule from the logical system. This is not at all the case. The fact is that the cut rule is a very relevant and deep attempt to express one of the crucial methodological aspects of reasoning, that is, the decomposition of a complex problem in simpler subproblems: in order to prove $\Gamma \vdash B$, we can prove two lemmas $\Gamma \vdash A$ and $A \vdash B$. This is very clear from the computer science point of view, where the emphasis is not on the results - that is, the terms in normal form - but on the program's synthesis, and its reduction (computation) when applied to data.

An important consequence of the cut-elimination theorem is the following. One can easily see that in any rule of inference except a cut, the lower sequent is no less complicated than the upper sequent(s). More precisely, every formula occurring in an upper sequent is a subformula of some formula occurring in the lower sequent. Hence a proof without a cut contains only subformulae of the formulae occurring in the end sequent (**subformula property**). From this observation, the consistency of the logical system immediately follows. Suppose indeed that the empty sequent $\vdash$ were provable. Then by the cut-elimination theorem, it would be provable without a cut; but this is impossible, by the subformula property of cut-free proofs.

In order to prove theorem 8.4.2, it is convenient to introduce a new rule of inference called a mix. The mix rule is logically equivalent to the cut rule, but it helps deal with the contraction rules, whose behavior during the cut elimination is rather annoying. We shall discuss again this problem at the end of the proof.

A **mix** is an inference of the following form:

$$\frac{\Gamma \vdash M: A \qquad \Gamma_1 \vdash N: B}{\Gamma, \Gamma_1^* \vdash [M/\mathbf{x}]N: B}$$

where $\mathbf{x}$ is a vector of variables in Γ_1, Γ_1^* is obtained from Γ_1 by erasing all the assumption $x: A$ with $x \in \mathbf{x}$, and $[M/\mathbf{x}]N$ denotes the simultaneous substitution of M for all variables in $\mathbf{x}$, inside the term N.

The previous notion of mix rule is slightly different from the usual one. In particular the formula A can still appear in the lower sequent Γ_1^*.

An occurrence of A in the upper sequents of a mix rule is called a **mix formula** (of the inference) if and only if it satisfies one of the two following conditions:

1. it is in the r.h.s. of the left upper sequent;
2. it is in the l.h.s. of the right upper sequent, and it is associated with a proof $x:A$, with $x \in \mathbf{x}$.

Clearly a cut is a particular case of mix, where the vector $\mathbf{x}$ contains only one variable. Moreover, every mix can be simply obtained by a sequence of exchanges and contractions, a cut, and another series of exchanges.

Since a proof without a mix is also a proof without a cut, theorem 8.4.2 is proved if we prove the following:

8.4.3 Theorem (The Mix-Elimination Theorem) *Let D be a derivation of the sequent $\Gamma \vdash M: A$. Then there exists a mix-free derivation D' of the sequent $\Gamma \vdash N: A$, where N is the β-normal form of M.*

Theorem 8.4.3 is easily obtained from the following lemma, by induction on the number of the mix rules occurring in the derivation D.

8.4.4 Lemma *Let D be a derivation of the sequent $\Gamma \vdash M: A$ that contains only one cut rule occurring as the last inference. Then there exists a cut-free derivation D' of the sequent $\Gamma \vdash N: A$, where N is the β-normal form of M.*

The rest of this section is devoted to the proof of this lemma.

We define two scales for measuring the complexity of a proof. The **grade** of a formula A (denoted by g(A)) is the number of logical symbols contained in A. The grade of a mix is the grade of the mix formula. When a derivation D has only one cut as the last inference, we define the grade of D (denoted by g(D)) to be the grade of this mix.

Let D be a derivation which contains a mix

$$\frac{\Gamma \vdash M: A \qquad \Gamma_1 \vdash N: B}{\Gamma, \Gamma_1^* \vdash [M/\mathbf{x}]N: B}$$

as the last inference. We call a thread in D a left (right) thread if it contains the left (right) upper sequent of the mix. The **rank** of a thread is the number of consecutive sequents in D (counting upward from the upper sequents of the cut rule) that contain the mix formula or a formula which has been contracted to it. The rank of every thread is at least 1. The left (right) rank of a derivation D is the maximum among the ranks of the left (right) threads in D. The rank of a derivation D is the sum of its left and right ranks. The rank of a derivation is at least 2.

We prove the lemma by double induction on the grade g and rank r of the derivation D. The proof is subdivided into two main cases, namely $r = 2$ and $r > 2$.

Case 1 : r = 2

We distinguish cases according to the forms of the proofs of the upper sequents of the cut rule.

1.1. The left upper sequent S_1 is an initial sequent. In this case D is of the form

$$\frac{y: A \vdash y: A \qquad \Gamma \vdash N: B}{\Gamma_1^* \vdash [y/\mathbf{x}]N: B}$$

And we obtain the same proof by a series of contractions starting from the sequent $\Gamma \vdash N: B$.

1.2. The right upper sequent S_2 is an initial sequent. Similarly.

1.3. Neither S_1 nor S_2 is an initial sequent, and S_1 is the lower sequent of a structural inference. It is easy to check that, in case $r = 2$, this is not possible.

1.4. Neither S_1 nor S_2 is an initial sequent, and S_2 is the lower sequent of a structural inference. The structural inference must be a weakening, whose weakening formula is A. The derivation D has the structure

$$\dfrac{\Gamma \vdash M: A \qquad \dfrac{x: A, \Gamma_1 \vdash N: B}{\Gamma_1 \vdash N: B}}{\Gamma, \Gamma_1 \vdash [M/x]N: B}$$

Since $x \notin FV(N)$, $[M/x]N \equiv N$, and we obtain the same proof $N: B$ as follows:

$$\dfrac{\dfrac{\Gamma_1 \vdash N: B}{\text{some weakenings}}}{\Gamma, \Gamma_1 \vdash N: B}$$

1.5. Both S_1 and S_2 are the lower sequents of logical inferences. Since the left and right ranks of D are 1, the cut formulae on each side are the principal formulae of the logical inferences, and the mix is actually a cut between these two formulae.

We use induction on the grade and distinguish two cases according to the outermost logical symbol of A.

($\rightarrow$) the derivation D has the structure

$$\dfrac{\dfrac{\Gamma, x: A \vdash M: B}{\Gamma \vdash \lambda x{:}A.M: A \rightarrow B} \qquad \dfrac{\Gamma_1 \vdash N: A \qquad \Gamma_2, y: B \vdash P: C}{\Gamma_1, \Gamma_2, z: A \rightarrow B \vdash [zN/y]P: C}}{\Gamma, \Gamma_1, \Gamma_2 \vdash [\lambda x{:}A.M/z][zN/y]P: C}$$

where, by assumption, the proofs ending with $\Gamma, x: A \vdash M: B$, $\Gamma_1 \vdash N: A$ or $\Gamma_2, y: B \vdash P: C$ do not contain any cut. Note now that

$$[\lambda x{:}A.M/z][zN/x]P: C \equiv [(\lambda x{:}A.M)N/y]P: C = [([N/x]M)/y]P: C,$$

which also suggests how to build the new derivation.

First consider the derivation

$$\dfrac{\Gamma_1 \vdash N: A \qquad \Gamma, x: A \vdash M: B}{\Gamma_1, \Gamma \vdash [N/x]M: B}$$

This proof contains only one mix as its last inference. Furthermore, the grade of the mix formula is less than $g(A \rightarrow B)$. By induction hypothesis, there exists a derivation D' of the sequent $\Gamma_1, \Gamma \vdash M': B$, where M' is the β-normal form of $[N/x]M$. Then, with another mix we get

$$\frac{\Gamma_1, \Gamma \vdash M': B \qquad \Gamma_2, y: B \vdash P: C}{\Gamma, \Gamma_1, \Gamma_2 \vdash [M'/z]P: C}$$

Again we can apply the induction hypothesis, obtaining the requested derivation.

(×) we only consider the case in which the right upper sequent is obtained by a (×,r,1), the other case being completely analogous. The derivation has the structure

$$\frac{\dfrac{\Gamma \vdash M: A \quad \Gamma \vdash N: B}{\Gamma \vdash \langle M,N\rangle : A\times B} \qquad \dfrac{\Gamma_1, x: A \vdash P: C}{\Gamma_1, z: A\times B \vdash [fst(z)/x]P: C}}{\Gamma, \Gamma_1 \vdash [\langle M,N\rangle/z][fst(z)/x]P: C}$$

where by assumption the proofs ending with $\Gamma \vdash M: A$, $\Gamma \vdash N: B$ or $\Gamma_1, x: A \vdash P: C$ do not contain any mix. We have

$$[\langle M,N\rangle/z][fst(z)/x]P: C \equiv [fst(\langle M,N\rangle)/x]P: C = [M/x]P: C.$$

Consider the derivation:

$$\frac{\Gamma \vdash M: A \qquad \Gamma_1, x: A \vdash P: C}{\Gamma, \Gamma_1, z: A\times B \vdash [M/x]P: C}$$

This proof contains only one mix as its last inference. Furthermore the grade of the cut formula is less than $g(A\times B)$. By induction hypothesis, we have the requested cut-free derivation D'.

Case 2 : r > 2

The induction hypothesis is that we can eliminate the cut from every derivation D' that contains only one cut as the last inference, and that satisfies either $g(D') < g(D)$, or $g(D') = g(D)$ and $rank(D') < rank(D)$.

There are two main cases, namely, $rank_r(D) > 1$ and $rank_l(D) > 1$ (with $rank_r(D) = 1$).

2.1. $rank_r(D) > 1$: we distinguish several subcases according to the logical inference whose lower sequent is S_2.

2.1.1. The sequent S_2 is the lower sequent of a weakening rule, whose weakening formula is not the cut formula. The derivation D has the structure

$$\dfrac{\Gamma \vdash M: A \qquad \dfrac{\Gamma_1 \vdash N: B}{y: C, \Gamma_1 \vdash N: B}}{\Gamma, y: C, \Gamma_1^* \vdash [M/\mathbf{x}]N: B}$$

Consider the derivation D'

$$\dfrac{\Gamma \vdash M: A \qquad \Gamma_1 \vdash N: B}{\Gamma, \Gamma_1^* \vdash [M/\mathbf{x}]N: B}$$

where the grade of D' is the same of D, namely g(A). Moreover, the two derivations have the same left rank, while $\text{rank}_r(D') = \text{rank}_r(D) - 1$; thus we can apply the induction hypothesis. With a weakening and some exchanges we obtain the requested mix-free derivation.

2.1.2. The sequent S_2 is the lower sequent of an exchange rule. Similarly.

2.1.3. The sequent S_2 is the lower sequent of a contraction rule. This is the main source of problems with the cut rule (see discussion at the end of the proof). With the mix rule, everything is instead very easy. For instance, let us consider the case when the contracted formula is a cut formula (the other case is even simpler). The derivation has the structure

$$\dfrac{\Gamma \vdash M: A \qquad \dfrac{\Gamma_1, x: A, y: A \vdash N: B}{\Gamma_1, z: A \vdash [z/x][z/y]N: B}}{\Gamma, \Gamma_1^* \vdash [M/\mathbf{x}][z/x][z/y]N: B}$$

where $z \in \mathbf{x}$.

Consider the derivation D'

$$\dfrac{\Gamma \vdash M: A \qquad \Gamma_1, x: A, y: A \vdash N: B}{\Gamma, \Gamma_1^* \vdash [M/\mathbf{x}']N: B}$$

where $\mathbf{x}'$ is obtained from $\mathbf{x}$ by erasing z and adding x, y.

The grade of D' is the same of D, namely g(A). Moreover, the two derivations have the same left rank, while $\text{rank}_r(D') = \text{rank}_r(D) - 1$; thus we can apply the induction hypothesis, and we obtain a cut-free derivation D'' of $\Gamma, \Gamma_1^* \vdash N': B$, where N' is the β-normal form of $[M/\mathbf{x}']N: B$. Since $[M/\mathbf{x}][z/x][z/y]N: B \equiv [M/\mathbf{x}']N: B$, the proof is completed.

2.1.4. The sequent S_2 is the lower sequent of a logical inference J, whose principal formula is not a cut formula.

The last part of the derivation D looks like

$$\dfrac{\Gamma \vdash M: A \qquad \dfrac{\Gamma_1 \vdash N: B}{\Gamma_2 \vdash P: C}\ J}{\Gamma, \Gamma_2^* \vdash [M/\mathbf{x}]P: C}$$

where the derivations of $\Gamma \vdash M: A$ and $\Gamma_1 \vdash N: B$ contains no mixes, and Γ_1 contains all the cut formulae of Γ_2. Consider the following derivation D':

$$\dfrac{\Gamma \vdash M: A \qquad \Gamma_1 \vdash N: B}{\Gamma, \Gamma_1^* \vdash [M/\mathbf{x}]N: B}$$

The grade of D' is the same as that of D, namely g(A). Moreover, the two derivations have the same left rank, while $\mathrm{rank}_r(D') = \mathrm{rank}_r(D) - 1$; thus we can apply the induction hypothesis, and we obtain a cut-free derivation D'' of $\Gamma, \Gamma_1^* \vdash N': B$, where N' is the β-normal form of $[M/\mathbf{x}]N: B$.

Now we can apply the J rule to get:

$$\dfrac{\Gamma, \Gamma_1^* \vdash N': B}{\Gamma, \Gamma_2^* \vdash P': C}\ J$$

and clearly P' is the β-normal form of $[M/\mathbf{x}]P: C$ (the inference rules are sound w.r.t. equality of proofs).

2.1.5. The sequent S_2 is the lower sequent of a logical inference J, whose principal formula is a cut formula. We treat only the case of the arrow.

(→) the derivation D has the structure

$$\dfrac{\Gamma \vdash M: A \to B \qquad \dfrac{\Gamma_1 \vdash N: A \qquad \Gamma_2, y: B \vdash P: C}{\Gamma_1, \Gamma_2, z: A \to B \vdash [zN/y]P: C}}{\Gamma, \Gamma_1^*, \Gamma_2^* \vdash [M/\mathbf{x}][zN/y]P: C}$$

where $z \in \mathbf{x}$.

Let $\mathbf{v_1}, \mathbf{v_2}$ be the variables of $\mathbf{x}$, which are respectively in Γ_1 and Γ_2. Note that $\mathbf{v_1}$ and $\mathbf{v_2}$ cannot be both empty by the hypothesis about rank.

If $\mathbf{v_1}, \mathbf{v_2}$ are both nonempty, consider the following derivations D_1 and D_2:

D_1: D_2:

$$\frac{\Gamma \vdash M: A\to B \quad \Gamma_1 \vdash N: A}{\Gamma, \Gamma_1^* \vdash [M/\mathbf{v_1}]N: A} \qquad \frac{\Gamma \vdash M: A\to B \quad \Gamma_2, y: B \vdash P: C}{\Gamma, \Gamma_2^*, y: B \vdash [M/\mathbf{v_2}]P: C}$$

If $\mathbf{v_1}$ ($\mathbf{v_2}$) is empty, and thus $\Gamma_1 = \Gamma_1^*$ ($\Gamma_2 = \Gamma_2^*$), let D_1 (D_2) be as follows:

D_1: D_2:

$$\frac{\Gamma_1 \vdash N: A}{\dfrac{\text{weakenings}}{\Gamma, \Gamma_1^* \vdash N: A}} \qquad \frac{\Gamma_2, y: B \vdash P: C}{\dfrac{\text{weakenings}}{\Gamma, \Gamma_2^*, y: B \vdash P: C}}$$

The grade of D_1 and D_2 is the same as that of D, namely $g(A\to B)$. Moreover, $rank_l(D_1) = rank_l(D_1) = rank_l(D)$, and $rank_r(D_1) = rank_r(D_1) = rank_r(D) - 1$. Hence, by induction hypothesis, there exist two derivations D_1' and D_2' respectively ending in the sequents

$\Gamma, \Gamma_1^* \vdash N_1: A$ with N_1 β-normal form of $[M/\mathbf{v_1}]N: A$

$\Gamma, \Gamma_2^*, y: B \vdash P_1: C$ with P_1 β-normal form of $[M/\mathbf{v_2}]P: C$.

Let $\mathbf{w}$ be the vector of variables in Γ, and let $\mathbf{w'}$, $\mathbf{w''}$ be two vectors of fresh variables of the same length. Let $\Gamma' = [\mathbf{w'}/\mathbf{w}]\Gamma$, $\Gamma'' = [\mathbf{w''}/\mathbf{w}]\Gamma$ be the sequences of assumptions obtained by renaming the variables in Γ. Let also $N_1' \equiv [\mathbf{w'}/\mathbf{w}]N_1$ and $P_1' \equiv [\mathbf{w'}/\mathbf{w}]P_1$ be the results of the same operation on the two terms N_1 and P_1.

Consider the derivation D'

$$\frac{\Gamma \vdash M: A\to B \qquad \dfrac{\Gamma', \Gamma_1^* \vdash N_1': A \qquad \Gamma'', \Gamma_2^*, y: B \vdash P_1': C}{\Gamma', \Gamma_1^*, \Gamma'', \Gamma_2^*, z: A\to B \vdash [z(N_1')/y](P_1'): C}}{\Gamma, \Gamma', \Gamma_1^*, \Gamma'', \Gamma_2^* \vdash [M/z][zN_1'/y]P_1': C.}$$

The grade of D' is the same of D, namely, $g(A\to B)$. Moreover, $rank_l(D') = rank_l(D)$, and $rank_r(D') = 1$, since $z: A\to B$ is the only cut formula. We can apply again the induction hypothesis, getting a mix-free derivation of the final sequent

$\Gamma, \Gamma', \Gamma_1^*, \Gamma'', \Gamma_2^* \vdash P_2: C$ with P_2 β-normal form of $[M/z][zN_1'/y]P_1'$.

Identifying by means of contractions (and exchanges) the variables in Γ, Γ', Γ'', we get a mix-free derivation of $\Gamma, \Gamma_1^*, \Gamma_2^* \vdash P_3: C$, where P_3 is a β-normal form of $[\mathbf{w}/\mathbf{w'}][\mathbf{w}/\mathbf{w''}]P_2$.

Note now that:

$$\begin{aligned}[\mathbf{w}/\mathbf{w'}][\mathbf{w}/\mathbf{w''}]P_2 &= [\mathbf{w}/\mathbf{w'}][\mathbf{w}/\mathbf{w''}][M/z][zN_1'/y]P_1' \\ &\equiv [M/z][z([\mathbf{w}/\mathbf{w'}]N_1')/y]([\mathbf{w}/\mathbf{w''}]P_1')\end{aligned}$$

$\equiv$ $[M/z][zN_1/y]P_1$
$=$ $[M/z][z([M/\mathbf{v_1}]N)/y]([M/\mathbf{v_2}]P)$
$\equiv$ $[M/z][M/\mathbf{v_1}][M/\mathbf{v_2}][zN/y]P$
$\equiv$ $[M/\mathbf{x}][zN/y]P$: C.

2.2. $\text{rank}_l(D) > 1$ (and $\text{rank}_r(D) = 1$)
This case is proved in the same way as is 2.1 above.

This completes the proof of lemma 8.4.4 and, hence, of the cut-elimination theorem. ♦

8.5 Categorical Semantics of Derivations

In this section we will study the categorical semantics of the intuitionistic proof theory developed in the last section. The main idea is that a formula A is interpreted as an object $|(A)$ in a category **C**. A *provable* sequent $B_1,..., B_n \vdash A$ is then associated with a morphism $f: |(B_1)\otimes...\otimes|(B_n)\rightarrow|(A)$, where $\otimes: \mathbf{C}\times\mathbf{C}\rightarrow\mathbf{C}$ is a suitable bifunctor that gives meaning to the comma in the l.h.s. of the sequent. Since the comma is associative and commutative, $\otimes$ is a symmetric tensor product. Moreover, we need some sort of projections for the weakening rule, and a diagonal map in the contraction rule. Actually, all these morphisms, together with a few reasonable equations, turn $\otimes$ into a categorical product.

Since $C \vdash A\Rightarrow B$ iff $C, A \vdash B$, it is natural to interpret the implication $\Rightarrow$ as the right adjoint to the functor $\otimes$.

Also the binary connective $\times$ is bifunctor of the category. In particular, the fact that $C \vdash A\times B$ iff ($C \vdash A$ and $C \vdash B$) suggests that $\times$ should be interpreted as a categorical product. As a consequence $\otimes$ and $\times$ coincide to within isomorphism, that is consistent with the well-known fact that the comma in the l.h.s. of a sequent has the same logical meaning of a conjunction. Anyway, since they are different concept, we prefer to maintain the distinction at the semantical level.

8.5.1 Definition *A categorical model of the positive intuitionistic calculus is a pair* $(\boldsymbol{C}, |_{At})$, *where* $\boldsymbol{C}$ *is a Cartesian closed category (possibly with two specified products* $\otimes$ *and* $\times$*), and* $|_{At}$ *is a map that takes every atomic formula to an object of* $\boldsymbol{C}$.

Notation In the rest of this chapter, we shall use the same symbols $\Rightarrow$ and $\times$ for indicating both the connectives of the calculus and the associated functors of the category: the intended meaning is clear from the context.

The analysis of the cut-elimination theorem will provide a more convincing justification for the previous notion of model. Now we concentrate on the interpretation.

I is extended to all the formulae in the following way:

$I(A) = I_{At}(A)$ if A is atomic

$I(A \Rightarrow C) = I(A) \Rightarrow I(C)$

$I(A \times C) = I(A) \times I(C)$

The categorical interpretation does not give meanings to sequents, but to derivations. Every derivation D whose final sequent is $B_1, \ldots, B_n \vdash A$ is interpreted by a morphism

$I(D) = f: I(B_1) \otimes \ldots \otimes I(B_n) \rightarrow I(A)$

Before giving the details of the interpretation, we recall a few results about Cartesian closed categories, and fix the notation.

We call $\text{weak-l}_{A,B}: A \otimes B \rightarrow B$, $\text{weak-r}_{A,B}: A \otimes B \rightarrow A$ the projections associated with the product $\otimes$, and $\text{fst}_{A,B}: A \times B \rightarrow B$, $\text{snd}_{A,B}: A \times B \rightarrow A$ the projections associated with the product $\times$.

We have the natural isomorphisms $\text{assoc}_{A,B,C}: (A \otimes B) \otimes C \rightarrow A \otimes (B \otimes C)$ and $\text{exch}_{A,B}: A \otimes B \rightarrow B \otimes A$, given by the symmetric monoidal structure associated with $\otimes$. Moreover, we have a diagonal map $\Delta_A = \langle id_A, id_A \rangle: A \rightarrow A \otimes A$ and an the evaluation map $\text{eval}_{A,C}: (A \Rightarrow C) \otimes A \rightarrow C$.

The interpretation is given by induction on the length of the derivation. We define it in a somewhat informal but suggestive (and concise) way, associating with every inference rule of the calculus in 8.3.4 a respective semantic rule. This will be enough as for the categorical meaning of the intuitionistic system in section 8.3.

8.5.2 Interpretation

(axiom) $\quad id_A : A \rightarrow A$

(associativity)
$$\frac{f: A \otimes (B \otimes C) \rightarrow D}{f \circ \text{assoc}_{A,B,C}: (A \otimes B) \otimes C \rightarrow D}$$

(exchange)
$$\frac{f: B \otimes A \rightarrow C}{f \circ \text{exch}_{A,B}: A \otimes B \rightarrow C}$$

(weakening)
$$\frac{f: B \rightarrow C}{f \circ \text{weak-l}_{A,B}: A \otimes B \rightarrow C}$$

$$\text{(contraction)}\quad \frac{f:\ (A\otimes A)\otimes B \to C}{f \circ \Delta_A \otimes id_B:\ A\otimes B \to C}$$

$$\text{(cut)}\quad \frac{f:\ B \to A \qquad g:\ A\otimes D \to C}{g \circ f \otimes id_D :\ B\otimes D \to C}$$

$$(\to, r)\quad \frac{f:\ B\otimes A \to C}{\Lambda(f):\ B \to A\Rightarrow C} \qquad (\to, l)\quad \frac{f:\ E \to A \qquad g:\ C \to D}{g \circ eval_{A,C} \circ id\otimes f:\ (A\Rightarrow C)\otimes E \to D}$$

$$(\times, r)\quad \frac{f:\ C \to A \qquad g:\ C \to B}{\langle f,g\rangle:\ C \to A\times B} \qquad (\times, l, 1)\quad \frac{f:\ A\otimes C \to D}{f \circ fst_{A,B}\otimes id_C:\ (A\times B)\otimes C \to D}$$

$$(\times, l, 2)\quad \frac{f:\ B\otimes C \to D}{f \circ snd_{A,B}\otimes id_C:\ (A\times B)\otimes C \to D.}$$

8.6 The Cut-Elimination Theorem Revisited

In this section, we look at the cut elimination theorem from the categorical point of view. Our goal is to provide a more convincing justification of the categorical notion of model, not a further proof of cut-elimination. Indeed the notion of CCC imposes some identifications between derivations that are not evident at the logical level of the inference rules. The fact is that we are not interested in giving semantics to the provability relation among formulae, but more generally to the whole proof system, with its associated normalization procedures. The point is that the equalities, which define CCC's, reflect the identity of derivations up to normalization. In particular, we have the following result.

8.6.1 Theorem (The Cut-Elimination Theorem) *Let D be a derivation of the sequent $A \vdash B$. Then there exists a cut-free derivation D' whose final sequent is $A \vdash B$, and such that in every model $\mathcal{I}(D) = \mathcal{I}(D')$.*

We do not prove the previous result, but instead analyze in detail some examples of derivations identified by the cut-elimination process. Our aim is to show that the Cartesian closed structure is imposed by this identification of proofs up to normalization. The examples are instrumental to this.

8.6.2 Example If in the following derivation the left upper sequent $A \vdash A$ is an initial sequent, the cut is eliminated by taking the derivation of the right upper sequent

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

At the semantic level, we have the situation:

$$\frac{id_A: A \to A \qquad f: A \to B}{f \circ id_A: A \to B}$$

thus a model must satisfy the equation $f \circ id_A = f$. Analogously, when the right upper sequent is an axiom, we derive the equation $id_A \circ f = f$

8.6.3 Example Consider a derivation whose final part looks like

$$\frac{D \vdash A \qquad \dfrac{B \vdash C}{A, B \vdash C}}{D, B \vdash C}$$

the cut is eliminated in the following way:

$$\frac{B \vdash C}{D, B \vdash C}$$

At the semantic level we have the following situation:

$$\frac{g: D \to A \qquad \dfrac{f: B \to C}{f \circ \text{weak-l}_{A,B}: A \otimes B \to C}}{f \circ \text{weak-l}_{A,B} \circ g \otimes id_B: D \otimes B \to C}$$

and

$$\frac{f: B \to C}{f \circ \text{weak-l}_{D,B}: D \otimes B \to C}$$

thus, for every $g: D \to A$, we must have

$$\text{weak-l}_{D,B} = \text{weak-l}_{A,B} \circ g \otimes id_B : D \otimes B \to C \qquad (*)$$

The previous equation is clearly true if $\otimes$ is a categorical product, and weak-$l_{D,B}$ is the associated right projection.

8.6.4 Example Consider a derivation whose final part looks like

$$\dfrac{\dfrac{D \vdash A \qquad \dfrac{A, A \vdash B}{A \vdash B}}{D \vdash B}}$$

This is the annoying case of the contraction. With the help of a mix rule, the derivation reduces to

$$\dfrac{D \vdash A \qquad A, A \vdash B}{D \vdash B}$$

Regarding the mix as a simple abbreviation of a series of cuts and structural inferences, the previous derivation is logically equivalent to

$$\dfrac{\dfrac{\dfrac{D \vdash A \qquad A, A \vdash B}{D, A \vdash B}}{D, D \vdash B}}{D \vdash B}$$

At the semantic level we have

$$\dfrac{g: D \to A \qquad \dfrac{f: A \otimes A \to B}{f \circ \Delta_A : A \to B}}{f \circ \Delta_A \circ g: D \to B}$$

which reduces to

$$\dfrac{\dfrac{g: D \to A \qquad f: A \otimes A \to B}{g: D \to A \qquad f \circ g \otimes id_A : D \otimes A \to B}}{\dfrac{f \circ g \otimes id_A \circ id_A \otimes g : D \otimes D \to B}{f \circ g \otimes g \circ \Delta_D : D \otimes D \to B}}$$

This implies that in a model we expect

$$\Delta_A \circ g = g \otimes g \circ \Delta_D \qquad (+)$$

that is the naturality of Δ.

8.6.5 Example Consider the derivation

$$\dfrac{\dfrac{B, A \vdash C}{B \vdash A \Rightarrow C} \qquad \dfrac{E \vdash A \qquad C \vdash D}{(A \Rightarrow C), E \vdash D}}{B, E \vdash D}$$

The last cut is eliminated by introducing two other cuts of lesser grade, namely

$$\dfrac{E \vdash A \qquad \dfrac{B, A \vdash C \qquad C \vdash D}{B, A \vdash D}}{B, E \vdash D}$$

At the semantic level this gives

$$\dfrac{\dfrac{h: B \otimes A \to C}{\Lambda(h): B \to A \Rightarrow C} \qquad \dfrac{f: E \to A \qquad g: C \to D}{g \circ eval_{A,C} \circ id \otimes f: (A \Rightarrow C) \otimes E \to D}}{g \circ eval_{A,C} \circ id \otimes f \circ \Lambda(h) \otimes id: B \otimes E \to D}$$

that reduces to

$$\dfrac{f: E \to A \qquad \dfrac{h: B \otimes A \to C \qquad g: C \to D}{g \circ h: B \otimes A \to D}}{g \circ h \circ id \otimes f: B, E \to D}$$

Note that in a Cartesian closed category, by the β-axiom of exponents, one has

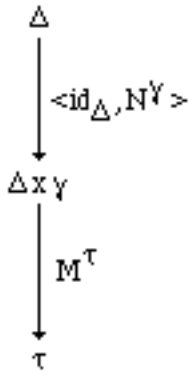
$$g \circ eval_{A,C} \circ id \otimes f \circ \Lambda(h) \otimes id = g \circ eval_{A,C} \circ \Lambda(h) \otimes id \circ id \otimes f = g \circ h \circ id \otimes f.$$

This completes the observation that the Cartesian closure may be described in terms of equivalence of proofs, up to normalization.

8.7 Categorical Semantics of the Simply Typed Lambda Calculus

In sections 8.3 and 8.4, we have investigated the relation between typed lambda calculus and intuitionistic logic, and in sections 8.5 and 8.6 we have established the connection between intuitionistic logic and Cartesian closed categories. In this section we want now to "fill the triangle" among typed lambda calculus, intuitionistic logic and Cartesian closed categories, studying the missing edge - namely the categorical semantics of the Typed Lambda Calculus, over CCC's.

The main idea of the categorical semantics of the typed lambda calculus is to interpret types as objects and terms as morphisms of a category **C**. In particular, the reader has probably noted some analogy between the axioms of $\lambda\beta\eta\pi^t$ and the axioms defining products and exponents in Cartesian closed categories. The relation is quite evident in the case of products: $fst^{\sigma\times\tau\to\sigma}$ and $snd^{\sigma\times\tau\to\tau}$ are easily understood as the projections associated with the categorical product of σ and τ. Apart from the problem of switching from "application" to "composition," the axioms of $\lambda\beta\eta\pi^t$ regarding the product are essentially identical to the axioms in definition 1.3.3 of categorical product. Less immediate is the connection between β (and η) and the rules defining the exponents of a CCC: the main problem is that the definition in the calculus is based on the process of substitution, for which we do not have any immediate equivalent in Category Theory. Let us look a little closer at our tentative interpretation: the intuitive idea is that a term M^τ with free variables among $x_1{}^{\sigma}1, ...,x_n{}^{\sigma}n$ should be interpreted as a morphism M^τ from $\Delta = \sigma_1\times...\times\sigma_n$ to τ. Suppose now that we have two terms M^τ and N^γ, such that the free variables of $[N^\gamma/ x^\gamma]M^\tau$ are in $\Delta= \sigma_1\times...\times\sigma_n$. Then look at M^τ like an arrow from $\Delta\times\gamma$ to τ, and at N^γ like an arrow from Δ to γ. The effect of substituting N^γ for x^γ in M^τ is simply achieved by composing M^τ with $< id_\Delta, N^\gamma >$.



These considerations are formalized in substitution lemma 8.7.6 below, which plays a central role in the semantics of functional languages. Note now that in every CCC, one has

$$(\beta_{cat}) \quad eval \circ <\Lambda(f),g> = eval \circ \Lambda(f)\times id \circ <id,g> = f \circ <id,g>$$

which, by our interpretation of substitution, is just the categorical equivalent of β-conversion in λ-calculus.

We now begin a formal discussion on the categorical interpetation of typed λ-calculus, by outlining a simple and very intuitive, set-theoretic definition of model for $\lambda\beta\eta\pi^t$. This is just a definition and will be instrumental to the categorical characterization below. In particular, it will allow us to relate the category-theoretic approach to the Tarskian view in semantics. For a more direct connection between $\lambda\beta\eta\pi^t$ and CCC's, see the references.

The collection Tp of types over a set At of ground types has been defined in section 8.2. As types will be largely used as indexes (of terms), in this section we go back to the use of small greek letters σ,τ.... as metavariables for types.

8.7.1 Definition *Let* $\mathbf{C} = \{C_i \mid i \in At\}$ *be a collection of sets. Then* $\mathbf{TS_C} = \{C_\sigma\}_{\sigma \in Tp}$ *is a* ***type structure*** *over* $\mathbf{C}$ *iff, for all* $\sigma,\tau \in Tp$,

$$C_{\sigma\to\tau} \subseteq C_\sigma \to C_\tau \ (= \mathbf{Set}[C_\sigma, C_\tau])$$
$$C_{\sigma\times\tau} = C_\sigma \times C_\tau .$$

8.7.2 Definition *A type structure* $\mathbf{TS_A} = (\{A_\sigma\}_{\sigma\in Tp}, [\,])$ *is a* ***model*** *of typed λ-calculus (with products) iff* $[\,]$ *yields an interpretation for* $\lambda\beta\eta(\pi)^t$. *That is, for any* ***environment*** $h: Var \to U_{\sigma\in Tp}(A_\sigma)$ *with* $h(x^\sigma)\in A_\sigma$, *one has* $[M^\rho]_h \in A_\rho$, *for each* $\rho \in Tp$ *and for* $[\,]$ *defined by*

Var. $\quad [x^\sigma]_h = h(x^\sigma)$

App. $\quad [M^{\sigma\to\tau}N^\sigma]_h = [M^{\sigma\to\tau}]_h([N^\sigma]_h)$

β. $\quad [\lambda x^\sigma.M^\tau]_h(a) = [M^\tau]_{h[a/x^\sigma]}$ *for* $a \in A_\sigma$

π. $\quad [fst^{\sigma\times\tau\to\sigma\tau}]_h = p_1$, $[snd^{\sigma\times\tau\to\sigma\tau}]_h = p_2$

where p_1, p_2 *are the set-theoretic projections (of proper types).*

This is a good definition of model, i.e., the axioms and rules are realized and, given a model $\mathbf{TS_A}$ $\lambda\beta\eta(\pi)^t$, one actually has

$$\lambda\beta\eta(\pi)^t \vdash M^\sigma = N^\sigma \ \Rightarrow\ \mathbf{TS_A} \models M^\sigma = N^\sigma.$$

Indeed, the reader will be asked to check the validity of axiom (β) in the exercise before 8.7.7. Note also that, in a model, axiom (η) and rule (ξ) trivially hold, since λ-terms are interpreted as functions in extenso and, hence, one has

$$\eta.\ \forall a\ \ [\lambda y^\sigma.M^{\sigma\to\tau}y^\sigma]_h(a) = [M^{\sigma\to\tau}]_h(a)\ \text{, for } y^\sigma \notin FV(M^{\sigma\to\tau}),\ \text{and}$$
$$\xi.\ \forall a\ \ [M^\tau]_{h[a/x^\sigma]} = [N^\tau]_{h[a/x^\sigma]} \ \Rightarrow\ [\lambda x^\sigma.M^\tau]_h = [\lambda x^\sigma.N^\tau]_h\ ,$$

by (β) and by $[\lambda x^\sigma.P^\tau]_h \in A_{\sigma\to\tau} \subseteq A_\sigma \to A_\tau$.

The properties of projections and pairing, in case one considers $\lambda\beta\eta\pi^t$, are trivially realized by the definition of product.

Thus a type-structure $\mathbf{TS_A}$ is a model of $\lambda\beta\eta(\pi)^t$ iff

$$\lambda\beta\eta(\pi)^t \vdash M^\sigma = N^\sigma \ \Rightarrow\ \mathbf{TS_A} \models M^\sigma = N^\sigma.$$

Since the previous definition of model is set-theoretic, a suitable notion of "concreteness" for categories can help in studying categories as models. The minor loss of generality, in this case, is balanced by the gain in intuition.

In general, concrete categories are widely used in denotational semantics; they are (usually) defined by an "enough points" requirement.

8.7.3 Definition. *Let* $\mathbf{C}$ *be a category.* $t \in Ob_{\mathbf{C}}$ *is a* ***generator*** *iff for all* $a,b \in Ob_{\mathbf{C}}$ *and all* $f,g \in \mathbf{C}[a,b]$, $f \neq g \Rightarrow \exists h \in \mathbf{C}[t,a]\ f \circ h \neq g \circ h$.

$\mathbf{C}$ *has* ***enough points*** *if there exists a generator* t *that is terminal in the given category.*

The terminology derives from the idea that the arrows in $\mathbf{C}[t,a]$, when t is terminal, may be understood as the points (or elements) of a. Indeed, we can associate with each category $\mathbf{C}$ having enough points a category of sets $\mathbf{C}^{set}$ as follows. For $a,b \in Ob_{\mathbf{C}}$ and t terminal

objects: $|a| = \mathbf{C}[t,a]$

morphisms: $f \in \mathbf{C}^{set}[|a|,|b|]$ iff $\exists f' \in C[a,b]\ f = f' \circ \_$.

Observe that f' is unique once f is given.

8.7.4 Proposition. *If* $\mathbf{C}$ *is a CCC with enough points, then* $\mathbf{C}^{set}$ *is also a CCC with enough points.*

Proof If $c_{\sigma\to\tau}$ and $c_{\sigma\times\tau}$ are the exponent and product objects, in $\mathbf{C}$, of c_σ and c_τ, then $|c_{\sigma\to\tau}|$ and $|c_{\sigma\times\tau}|$ are the exponent and product objects, in $\mathbf{C}^{set}$, of $|c_\sigma|$ and $|c_\tau|$. As for the exponent, just set $eval^{set} = eval \circ \_$, and $\Lambda^{set}(f) = \Lambda(f') \circ \_$, where f' defines f as above. The rest is easy. ♦

Note that one obtains type structures from all categories with enough points That is, let $\mathbf{K}$ be a CCC with enough points and $\mathbf{C} = \{k_i \mid i \in At\} \subseteq Ob_{\mathbf{K}}$. Let also $k_{\sigma\to\tau}$ and $k_{\sigma\times\tau}$ be the exponent and product objects, in $\mathbf{K}$, of k_σ and k_τ. Then $\mathbf{K}_{\mathbf{C}} = \{\mathbf{K}[T,k_\sigma] \mid \sigma \in Tp\}$ is the type structure generated by $\mathbf{K}$ and $\mathbf{C}$.

Examples. The simplest type structures are the "full" type-structures and the "term model" of $\lambda\beta\eta(\pi)^t$. That is, $\mathbf{Set}_{\mathbf{C}}$, where $\mathbf{C}$ is a collection of sets, and $\mathbf{Term} = (\{Term_\sigma\}_{\sigma\in Tp}, [\ ])$, where $Term_\sigma$ is the set of terms of type σ modulo $\beta\eta$ convertibility.

8.7.5 Definition *Let* $\mathbf{C}$ *be a CCC, with terminal object* t *and projections fst, snd. Suppose one has a map* I *associating every atomic type* A *with an object of* $\mathbf{C}$. *Then the categorical interpretation is as follows:*

- ***Types:*** $[\sigma] = I(\sigma)$ *if* σ *is atomic*

  $[\sigma\to\tau] = [\tau]^{[\sigma]}$

$$[\sigma\times\tau] = [\sigma] \times [\tau]$$

- ***Terms***: *let* M^σ *be a term of* $\lambda\beta\eta\pi^t$, *with* $FV(M^\sigma) \subseteq \Delta = \{x^\sigma 1,\ldots,x^\sigma n\}$, *and assume that* $A, A_1,\ldots,A_n$ *interpret* $\sigma, \sigma_1,\ldots,\sigma_n$ *(we omit typing, when unambiguous, and write* $p_i \in \mathbf{C}[t\times A_1\times\ldots\times A_n, A_i]$ *for the projection* $p_i : t\times A_1\times\ldots\times A_n \to A_i$ *). Then* $[M^\sigma]_\Delta$ *is the morphism in* $\mathbf{C}[t\times A_1\times\ldots\times A_n, A]$ *defined by*

$$[x^\sigma i]_\Delta = p_i$$
$$[MN]_\Delta = eval \circ \langle [M]_\Delta, [N]_\Delta \rangle$$
$$[\lambda x^\tau.M]_\Delta = \Lambda([M]_{\Delta\cup\{x^\tau\}})$$
$$[\langle M, N\rangle]_\Delta = \langle [M]_\Delta, [N]_\Delta \rangle$$
$$[fst\ (M)]_\Delta = fst \circ [M]_\Delta$$
$$[snd\ (M)]_\Delta = snd \circ [M]_\Delta .$$

8.7.6 Substitution Lemma

i. If $y^\sigma \notin FV(N)$, *then* $[N]_{\Delta\cup\{y^\sigma\}} = [N]_\Delta \circ fst$

ii. $[\,[N/x^\sigma]M\,]_\Delta = [M]_{\Delta\cup\{x^\sigma\}} \circ \langle id,[N]_\Delta\rangle$ *(types are omitted when unambiguous).*

Proof

i. By induction on M. The following is the typical case:

$$\begin{aligned}
[\lambda x^\tau.P]_{\Delta\cup\{y^\sigma\}} &= \Lambda([P]_{\Delta\cup\{y^\sigma,\, x^\tau\}}) \\
&= \Lambda([P]_{\Delta\cup\{x^\tau, y^\sigma\}}\circ\xi) && \text{for } \xi: (a_\Delta\times a_\sigma)\times a_\tau \cong (a_\Delta\times a_\tau)\times a_\sigma \\
&= \Lambda([P]_{\Delta\cup\{x^\tau\}}\circ fst\circ\xi) && \text{by induction} \\
&= \Lambda([P]_{\Delta\cup\{x^\tau\}}\circ fst\times id) && \text{as } fst\circ\xi = fst\times id \\
&= \Lambda([P]_{\Delta\cup\{x^\tau\}})\circ fst && \text{by the naturality of } \Lambda.
\end{aligned}$$

ii. By induction on M:

$M \equiv x^\sigma$

$$\begin{aligned}
[x^\sigma]_{\Delta\cup\{x^\sigma\}} \circ \langle id,[N]_\Delta\rangle &= snd \circ \langle id,[N]_\Delta\rangle \\
&= [N]_\Delta \\
&= [\,[N/\,x^\sigma\,]\,x^\sigma\,]_\Delta
\end{aligned}$$

$M \equiv x^{\sigma_i}$

$$\begin{aligned}
[x^{\sigma_i}]_{\Delta\cup\{x^\sigma\}} \circ \langle id,[N]_\Delta\rangle &= p_i \circ fst^{T\times A1\times\ldots\times An\,,A} \circ \langle id, [N]_\Delta\rangle \quad (\sigma_i \neq \sigma) \\
&= p_i \\
&= [x^{\sigma_i}]_\Delta
\end{aligned}$$

$M \equiv QP$

$$\begin{aligned}
[QP]_{\Delta\cup\{x^\sigma\}} \circ \langle id,[N]_\Delta\rangle &= eval \circ \langle [Q]_{\Delta\cup\{x^\sigma\}}, [P]_{\Delta\cup\{x^\sigma\}} \rangle \circ \langle id,[N]_\Delta\rangle \\
&= eval \circ \langle [Q]_{\Delta\cup\{x^\sigma\}}\circ\langle id,[N]_\Delta\rangle, [P]_{\Delta\cup\{x^\sigma\}}\rangle\circ\langle id,[N]_\Delta\rangle \\
&= eval \circ \langle [\,[N/x^\sigma]\,Q\,]_\Delta, [\,[N/x^\sigma]P\,]_\Delta\rangle \\
&= [\,[N/x^\sigma]\,PQ\,]_\Delta
\end{aligned}$$

$M \equiv \lambda y^{\tau}.P$

$$\begin{aligned}
[\,[N/x^{\sigma}]\,\lambda y^{\tau}.P\,]_{\Delta} &= \Lambda([\,[N/x^{\sigma}]P\,]_{\Delta\cup\{y^{\tau}\}}) \\
&= \Lambda([P]_{\Delta\cup\{y^{\tau},x^{\sigma}\}}\circ \langle id,[N]_{\Delta\cup\{y^{\tau}\}}\rangle) && \text{by induction} \\
&= \Lambda([P]_{\Delta\cup\{y^{\tau},x^{\sigma}\}}\circ \langle id,[N]_{\Delta}\circ fst\rangle) && \text{by (i)} \\
&= \Lambda([P]_{\Delta\cup\{x^{\sigma},y^{\tau}\}})\circ \langle id,[N]_{\Delta}\rangle && \text{by naturality (as in (i))} \\
&= [\lambda y^{\tau}.P]_{\Delta\cup\{x^{\sigma}\}}\circ \langle id,[N]_{\Delta}\rangle.
\end{aligned}$$

As for pairing and projection, the computation is easy and is left for exercise. ♦

The lemma may suffice to give an interpretation of $\lambda\beta\eta\pi^{t}$ over an arbitrary CCC. Indeed, nothing else is required from a purely categorical perspective, i.e., adopting a more suitable notion of model. In particular, it is worth noting that the previous categorical semantics of the typed lambda calculus really "fills the triangle" we talked about at the beginning of this section. We leave it as an exercise for the reader to verify that a term and its categorical interpretation with respect to the previous definition are actually associated with the same derivation in intuitionistic logic.

However, as many, since Tarski, are used to interpreting formal systems into mathematical structures by first assigning values to variables by means of environments as set-theoretic maps, we complete the construction of models (as defined at the beginning of the section) with the following exercise and theorem.

Exercise Prove a version of lemma 8.7.6 in the style of definition 8.7.2. That is, show that in any model and for any environment h, one has (we omit types for simplicity)

$$[M]_{h[[N]h/x^{\sigma}]} = [\,[N/x^{\sigma}]M\,]_{h}\,.$$

This gives axiom (β) in the model.

8.7.7 Theorem *Any CCC with enough points* $\mathbf{C}$ *and any collection* $\{a_i\}_{i\in At}$ *of objects in* $\mathbf{C}$ *yield a model of* $\lambda\beta\eta\pi^{t}$.

Proof Let $\mathbf{C}$ be the given category and let $A = \{|a_i| \,/\, i\in At\}$ be the collection of sets in $\mathbf{C}^{set}$ that interpret the atomic types. Then the model is given by the type structure $\mathbf{TS}_A = \mathbf{C}^{set}{}_A$, i.e., the higher types are interpreted by $|c_{\sigma\times\tau}|$ and $|c_{\sigma\to\tau}|$. As a matter of fact, let $h : Var \to U_{\sigma\in Tp}\{|c_{\sigma}|\}$ be an environment. Fix M^{σ} and let $FV(M^{\sigma}) \subseteq \Delta = \{x^{\sigma}1,\ldots,x^{\sigma}n\}$. By definition, $h(x^{\sigma}i)\in|c_{\sigma i}|$, then set $h_{\Delta} = \langle id,h(x^{\sigma}1),...,h(x^{\sigma}n)\rangle\in\mathbf{C}[t,t\times c_{\sigma 1}\times...\times c_{\sigma n}]$ and define

$$[\![M^{\sigma}]\!]_{h} = [M^{\sigma}]_{\Delta}\circ h_{\Delta}\in |c_{\sigma}|.$$

It is now easy to check that the map F_h, defined by $F_h(e) = h[e/x]_{\Delta}$, for $e\in|c_{\sigma}|$, is in $\mathbf{C}^{set}[|c_{\sigma}|,|c_{\sigma 1}\times...\times c_{\sigma n}|\,]$. Thus, there exists $h'\in\mathbf{C}[c_{\sigma},c_{\sigma 1}\times...\times c_{\sigma n}]$ such that $F_h = h'\circ\_$ and, hence, $h[e/x]_{\Delta} = h'\circ e$. This fact and the substitution lemma guarantee that $[\![\ \_\ ]\!]$ is a well-defined interpretation map and, thus, that $\mathbf{C}^{set}{}_A$ is a model. ♦

We now need to prove the converse and construct a CCC out of every model of $\lambda\beta\eta\pi^t$. Observe first that by functional application, over a type structure, one may define algebraic functions as follows.

8.7.8 Definition *Given a type-structure* $\mathbf{TS_C} = \{C_\sigma\}_{\sigma\in Tp}$, *the* ***typed monomials*** *over* $\mathbf{TS_C}$ *are defined by*

constants $a^\sigma \in C_\sigma$, ... *are typed monomials, for all* $\sigma\in Tp$

variables $x^\sigma, y^\sigma, \ldots$ *are typed monomials, for all* $\sigma\in Tp$

application $M^{\sigma\to\tau}(N^\sigma)$ *is a typed monomial, when* $M^{\sigma\to\tau}$ *and* N^σ *are typed monomials.*

A function $f : C_\sigma \to C_\tau$ *is* ***algebraic*** *if* $f(a^\sigma) = [a^\sigma/x^\sigma]M^\tau$ *for some typed monomial* M^τ.

The intuition is that algebraic functions will give the morphisms of a "rich enough" category as to interpret typed λ-terms. It is clear, for example, that projections are algebraic functions: just set $\mathrm{send}((x^\sigma, y^\tau)) = x^\sigma$, then $\mathrm{snd} : C_{\sigma\times\tau} \to C_\tau$ is the second projection.

8.7.9 Lemma *Given a type-structure* $\mathbf{TS_A} = \{A_\sigma\}_{\sigma\in Tp}$, *one obtains a category with enough points by taking the singleton set, the* A_σ*'s as objects and the algebraic functions as morphisms. Call this category* $\mathbf{CTS}_A$.

Proof Exercise. ♦

8.7.10 Theorem *Let* $\mathbf{TS_A} = \{A_\sigma\}_{\sigma\in Tp}$ *x be a model of* $\lambda\beta\eta\pi^t$ *and let* $\mathbf{CTS}_A$ *be as in the lemma. Then* $\mathbf{CTS}_A$ *is a CCC.*

Proof Products are defined by taking $A_\sigma \times A_\tau = A_{\sigma\times\tau}$. Then, if the typed monomial M defines f and N defines g, <M,N> defines <f,g>, as the formal pairing and projections behave as required in Cartesian categories.

As for exponents, set $A_\tau{}^{A_\sigma} = A_{\sigma\to\tau}$. Then

$$\mathrm{eval} : A_{\tau\to\rho}\times A_\tau \to A_\rho \quad \text{is defined by} \quad x \vdash \mathrm{fst}\, x^{\tau\to\rho\times\tau}(\mathrm{snd}\, x^{\tau\to\rho\times\tau}).$$

Moreover, if $f : A_\sigma \times A_\tau \to A_\rho$ is defined by M,

$$\Lambda(f) : A_\sigma \to A_\rho{}^{A_\tau} \text{ is defined by } \lambda x^\tau.M.$$

Finally observe that axiom (β) gives (β_{cat}) in definition 2.3.1, while (η) gives (η_{cat}). ♦

By the following exercise we let the careful reader relate the two interpretations of λ-terms given so far.

Exercise Compare in detail the categorical and set-theoretic meanings of terms. Namely, start with a $\mathbf{TS_A}$, which is a model, and construct a $\mathbf{CTS_A}$ as in 8.7.9. (Note that in this construction, lots of - irrelevant - morphisms may be lost). Then construct a new $\mathbf{TS'_A}$ as in 8.7.7 (this will be a "substructure" of $\mathbf{TS_A}$, up to an obvious identification). Consider now an interpretation $[\![-]\!]_h$ over

$\mathbf{TS}_\mathbf{A}$, where the range of h, though, is restricted to $\mathbf{TS'}_\mathbf{A}$. Define $[\,-\,]_\Delta$ as in 8.7.5 over $\mathbf{CTS}_\mathbf{A}$ and, finally, give $[\,-\,]'_h$ over $\mathbf{TS'}_\mathbf{A}$ by using h and $[\,-\,]_\Delta$ as in 8.7.7. Prove then that $[\,-\,]_h$ and $[\,-\,]'_h$ coincide. Thus, we moved from $\mathbf{TS}_\mathbf{A}$ to $\mathbf{CTS}_\mathbf{A}$ and to $\mathbf{TS'}_\mathbf{A}$, preserving the meaning of terms.

Conversely, by an even longer and more tedious proof, go from an interpretation $[\,-\,]_\Delta$ over a Cartesian closed category **C**, to a model $\mathbf{TS}_\mathbf{A}$, where the interpretation $[\,-\,]_h$ is given by 8.7.7. Construct then a $\mathbf{CTS}_\mathbf{A}$ as in 8.7.9 and observe that it may be faithfully embedded into **C**. Relate by this $[\,-\,]_\Delta$, over **C**, and the interpretation $[\,-\,]'_\Delta$, given as in 8.7.5, over $\mathbf{CTS}_\mathbf{A}$. (The connection between set-theoretic and categorical meaning of terms will be more closely investigated for the type-free case in the next chapter).

8.8 Fixpoint Operators and CCCs

The typed lambda calculus is strongly normalizable, that is:

1. every computation terminates, and
2. the computation is independent from the evaluation strategy.

The latter property is surely very attractive for a computational language since it greatly simplifies its semantics: the programmer, in the design of the code, does not have to fuss over the operational evolution of reduction, but can concentrate on the *denotation* of the program in its strongest meaning.

Considered by itself, the first property also seems to be quite interesting, but it has the rather annoying corollary that not all "computable" functions (e.g., Turing-computable) will be computable in the language. The problem is not only related to the abstract expressiveness of the language; in today's computer science, the idea of "general purpose" language is no longer considered as central as it was twenty years ago, and the loss of the ability to compute, for example, the Ackermann function, does not worry anybody: primitive recursive functions are surely enough for most interesting applications. What makes the general recursive formalism more attractive than the primitive recursive one, is that it is much easier to write code in the general formalism (also for computing a primitive recursive function).

Coming back to the typed lambda calculus, it is otherwise too poor for any application, so we face the problem of extending the language with more powerful constructs. In chapter 11, we shall study the "second order" or "polymorphic" extension, which still has the nice property of strong normalization together with a great formal expressiveness (although, as a programming language, it imposes on the programmer a completely new approach to the design of the code). We study for the moment a simple extension by means of **fixpoint operators**, which enables us to write recursive definitions of functions.

The reader should be aware, though, that a price must be paid when extending typed languages by fixpoint operators. The problem is related to the Curry-Howard analogy (see section 8.3). Remember

that a (constructive) proof of a formula A corresponds with a closed λ-term of type A in type theory, and, thus, a formula is provable in the logic if and only if the corresponding type is "inhabited" in type theory. The existence of fixpoint operators has as a consequence that all types are inhabited, or, from the logical point of view, that all formulae are provable (see later). Thus fixpoint operators have no logical correspondent; nevertheless, the calculus they originate is not inconsistent from the point of view of the equational theory of programs (that is, not all terms happen to be provably equal). This is shown by the models below.

In conclusion, the programmer, in the present context, must decide whether to acquire the full expressive power and the elegance of programming by "recursive definitions" or to preserve the advantages of the "types as formulae" paradigm. This choice depends on the specific applications he or she has in mind.

8.8.1 Definition *A fixpoint operator of type* σ *is a term* Θ_σ *of type* $(\sigma\to\sigma)\to\sigma$ *such that , for every term* $M^{\sigma\to\sigma}$ *one has*

$$M^{\sigma\to\sigma}(\Theta_\sigma M^{\sigma\to\sigma}) = \Theta_\sigma M^{\sigma\to\sigma}.$$

By fixpoint operators one can simulate recursive definitions in the following way:

letrec f^σ be $M^\sigma[f]$ in $N^\gamma[f]$

becomes

$$(\lambda h^\sigma.N^\gamma)\ (\Theta_\sigma(\lambda\, f^\sigma.M^\sigma))$$

Note that for every type σ we have at least an object of that type, obtained as a fixpoint of the identity

$$\Theta_\sigma\ (\lambda x^\sigma.x^\sigma).$$

We consider next the problem of giving an interpretation to fixpoint operators. As we need to interpret typed λ-calculi, this will be done in suitable CCC's.

8.8.2 Definition *Let* b *be an object of a CCC* $\mathbf{C}$. *A* ***fixpoint operator*** *for* b *is a morphism* $Fix_b: b^b\to b$ *such that* $Fix_b = eval_{b,b}\circ\langle id, Fix_b\rangle$. *A category* ***has fixpoint operators*** *if each object* b *has a fixpoint operator* Fix_b .

In every CCC with fixpoint operators, it is easy to give meaning to the axiom

$$\text{fix.}\quad M^{\sigma\to\sigma}\ (\Theta_\sigma\, M^{\sigma\to\sigma}) = \Theta_\sigma\, M^{\sigma\to\sigma}$$

by letting

$$\{\Theta_\sigma\}_\Delta = \Lambda(\mathrm{Fix}_\sigma\circ \mathrm{snd})\circ !_\Delta,$$

where snd: $t\times\sigma^\sigma\to\sigma^\sigma$, and $!_\Delta$ is the unique arrow from Δ to the terminal object t.
Indeed, one has the following:

$$\begin{aligned}
\{ M(\Theta_\sigma M) \}_\Delta &= \mathrm{eval} \circ \langle \{M\}_\Delta, \{ (\Theta_\sigma M) \}_\Delta \rangle \\
&= \mathrm{eval} \circ \langle \{M\}_\Delta, \mathrm{eval} \circ \langle \Lambda(\mathrm{Fix}_\sigma \circ \mathrm{snd}) \circ !\Delta, \{M\}_\Delta \rangle \rangle \\
&= \mathrm{eval} \circ \langle \{M\}_\Delta, \mathrm{Fix}_\sigma \circ \{M\}_\Delta \rangle \\
&= \mathrm{eval} \circ \langle \mathrm{id}, \mathrm{Fix}_\sigma \rangle \circ \{M\}_\Delta \\
&= \mathrm{Fix}_\sigma \circ \{M\}_\Delta \\
&= \mathrm{eval} \circ \langle \Lambda(\mathrm{Fix}_\sigma \circ \mathrm{snd}) \circ !\Delta, \{M\}_\Delta \rangle \\
&= \{ (\Theta_\sigma M) \}_\Delta .
\end{aligned}$$

It is not difficult to find CCC's with fixpoint operators. The most well-known example is probably the category **CPO**, with complete partial order for objects and continuous functions for morphisms (remember that, in **CPO**, $f: A \to B$ is continuous if and only if f is monotonic and for every directed subset D of A, $f(\cup(D)) = \cup f(D)$; every c.p.o.has a least element, $\perp = \cup\varnothing$).

Observe that, given a c.p.o. C and a continuous function f , we can form a chain

$$\{f^n(\perp_C)\}_{n\in\omega} = \perp_C \leq f(\perp_C) \leq f(f(\perp_C)) \leq \ldots. \leq f^n(\perp_C) \leq \ldots.$$

starting from the bottom element $\perp_C$. The next two results develop this example.

8.8.3 Theorem *Let C be a CPO, let $\perp_C$ be its least element, and let $f: C \to C$ be a continuous function. Then $\cup\{f^n(\perp_C)\}_{n\in\omega}$ is the least fixed point of f .*

Proof: Note that $\cup\{f^n(\perp_C)\}_{n\in\omega} = \cup\{f^{n+1}(\perp_C)\}_{n\in\omega}$. Then

$$\begin{aligned}
f(\cup\{f^n(\perp_C)\}_{n\in\omega}) &= \cup f\{f^n(\perp_C)\}_{n\in\omega}) \\
&= \cup\{f^{n+1}(\perp_C)\}_{n\in\omega}).
\end{aligned}$$

Moreover, if c is another fixed point, then we prove by induction that, for all n, $f^n(\perp_C) \leq c$. Indeed,

$$\perp_C \leq c$$
$$f^n(\perp_C) \leq c \Rightarrow f^{n+1}(\perp_C) \leq f(c) = c.$$

Then, by definition of least upper bound, $\cup\{f^n(\perp_C)\}_{n\in\omega} \leq c$. ♦

8.8.4 Definition *Let C be a c.p.o. Define then $Fix_C: C^C \to C$ the function that takes every continuous function $f: C \to C$ to $\cup\{f^n(\perp_C)\}_{n\in\omega}$.*

8.8.5 Proposition *$FIX_C: C^C \to C$ is continuous.*

Proof Exercise. ♦

In a more general setting, the existence of exponents, in CCC's, suggests the investigation of those "paradoxical" objects that "contain," as a retract, their own function space; namely, the reflexive objects of section 2.3 (see also below). They will turn out to be rather relevant in the next chapter, where examples are given, and in chapter 10, where the idea will be generalized to fixpoint constructions over types, namely to the categorical counterpart of recursive definitions of data types.

Remember (see definition 1.4.2) that in a category **C**, $a < b$ via the retraction (i,j) iff $j \circ i = id_a$. In these assumptions, i turns out to be mono and j epic. Thus a retract a of b is a subobject of b, up to isomorphisms. An object V is reflexive iff $V^V < V$ (see definition 2.3.5). We next show how to construct another simple model of the typed lambda calculus with fixpoint operators, by using the category $\mathbf{Ret_V}$ of retracts over a reflexive object V. Recall that, given an object V in a category **C**, the category $\mathbf{Ret_V}$ is defined as follows (see definition 1.4.4):

$$Ob_{\mathbf{RetV}} = \{ f \in \mathbf{C}[V,V] \mid f \circ f = f \}$$
$$Mor_{\mathbf{RetV}} = \{ (f, k, g) \mid f, g \in Ob_{\mathbf{RetV}} , k \in \mathbf{C}[V,V] , k = g \circ k \circ f \}$$
$$dom((f, k, g)) = f , cod((f, k, g)) = g$$
$$id_f = (f, f, f)$$
$$(f, k, g) \circ (g', k', f) = (g', k \circ k', g)$$

Proposition 8.8.6 below proves that, if V is a reflexive object in a CCC **C**, then $\mathbf{Ret_V}$ is Cartesian closed too. In theorem 8.8.8 we will prove that, given a reflexive object V in an arbitrary CCC, every object f in $\mathbf{Ret_V}$ has a fixpoint operator Fix_f.

8.8.6 Proposition *If* ***C*** *is a CCC and* V *is a reflexive object in* ***C****, then* $\mathbf{Ret_V}$ *is a CCC.*

Proof By proposition 2.3.6, we know that the terminal object t and $V \times V$ are both retracts of V (by definition of reflexive object also $V^V < V$). Suppose the following:

$t < V$ via in, out
$V \times V < V$ via in', out'
$V^V < V$ via in", out".

Let $1 = in \circ out : V \to V$. $1 \circ 1 = in \circ out \circ in \circ out = in \circ out = 1$. 1 is the terminal object of $\mathbf{Ret_V}$. If f is an object in $\mathbf{Ret_V}$, then $!f: f \to 1$ is $(f,1,1)$. Note that $(f,1,1)$ is a well-defined morphism, since $1 = 1 \circ 1 \circ f$ for the terminality of t. Moreover, if $(f,g,1)$ is another morphism, then $g = 1 \circ g \circ f = 1$, again for the terminality of t.

Given two objects f and g in $\mathbf{Ret_V}$, their product is $f \otimes g = in' \circ f \times g \circ out' : V \to V$. (In the present proof only, $\otimes$ has this meaning)

Note that

$$\begin{aligned} f \otimes g \circ f \otimes g &= in' \circ f \times g \circ out' \circ in' \circ f \times g \circ out' \\ &= in' \circ f \times g \circ f \times g \circ out' \\ &= in' \circ f \times g \circ out' \\ &= f \otimes g. \end{aligned}$$

The projections are $(f \otimes g, fst, f)$, $(f \otimes g, snd, g)$ where

$$fst = p_1 \circ out' \circ f \otimes g : V \to V$$
$$snd = p_2 \circ out' \circ f \otimes g : V \to V$$

and p_1, p_2 are the projections associated in **C** to the product $V \times V$.

The pairing operation $\mathbf{<,>_{ret}}$ is defined as follows: given two morphisms (c,h,f) and (c,k,g) set

$$\mathbf{<}\ (c,h,f),\ (c,k,g)\ \mathbf{>_{ret}} = (c,\ in' \circ \langle h,k\rangle, f \otimes g)$$

where <,> is the pairing in **C**.

This is a good definition, since

$$\begin{aligned} f\otimes g \circ in' \circ \langle h,k\rangle \circ\ c &= in' \circ f\times g \circ out' \circ in' \circ \langle h \circ c,\ k \circ\ c\rangle && \text{by def. of } f\otimes g \\ &= in' \circ f\times g \circ \langle h \circ c,\ k \circ\ c\rangle \\ &= in' \circ \langle f \circ h \circ c,\ g \circ\ k \circ\ c\rangle \\ &= in' \circ \langle h,\ k\rangle. \end{aligned}$$

We have still to prove the equations associated with the product; we only prove that

$$(\ f\otimes g,\ fst,\ f) \circ \mathbf{<}\ (c,h,f),\ (c,k,g)\ \mathbf{>_{ret} =}\ (c,h,f)$$

and leave the other proofs as an exercise for the reader. We have the following:

$$\begin{aligned} &(\ f\otimes g,\ fst,\ f) \circ \mathbf{<}\ (c,h,f),\ (c,k,g)\ \mathbf{>_{ret} =} \\ &\quad \mathbf{=}\ (\ f\otimes g,\ fst,\ f) \circ (c,\ in' \circ \langle h,k\rangle, f\otimes g) \\ &\quad \mathbf{=}\ (\ c,\ fst \circ in' \circ \langle h,k\rangle,\ f) && \text{by composition in } \mathbf{Ret_V} \\ &\quad \mathbf{=}\ (\ c,\ p_1 \circ out' \circ f\otimes g\ \circ in' \circ \langle h,k\rangle,\ f) && \text{by def. of fst} \\ &\quad \mathbf{=}\ (\ c,\ p_1 \circ out' \circ in' \circ f\times g \circ out'\ \circ in' \circ \langle h,k\rangle,\ f) && \text{by def. of } f\otimes g \\ &\quad \mathbf{=}\ (\ c,\ p_1 \circ f\times g \circ \langle h,k\rangle,\ f) = (c,h,f). \end{aligned}$$

The functor $\otimes$ is defined on morphisms in the usual way (namely, $f\times g = \langle p_1 \circ f,\ p_2 \circ g\rangle$). Specifically, given two morphisms (c,h,f) and (d,k,g):

$$\begin{aligned} (c,h,f) \otimes (d,k,g) &= \mathbf{<}\ (c,h,f) \circ (c\otimes d,\ fst\ ,\ c),\ (d,k,g) \circ (\ c\otimes d,\ snd,\ d)\ \mathbf{>_{ret}} \\ &= \mathbf{<}\ (c\otimes d,\ h \circ fst\ ,\ f),\ (\ c\otimes d,\ k \circ snd,\ g)\ \mathbf{>_{ret}} && \text{by composition in } \mathbf{Ret_V} \\ &= (\ c\otimes d,\ in' \circ \langle\ h \circ fst,\ k \circ snd\ \rangle,\ f\otimes g\ ) && \text{by def. of } \mathbf{<,>_{ret}} \\ &= (\ c\otimes d,\ in' \circ \langle\ h \circ p_1 \circ out' \circ c\otimes d,\ k \circ p_2 \circ out' \circ c\otimes d\ \rangle,\ f\otimes g\ ) && \text{by def. of fst, snd} \\ &= (\ c\otimes d,\ in' \circ \langle\ h \circ p_1,\ k \circ p_2\rangle \circ out' \circ c\otimes d,\ f\otimes g\ ) \\ &= (\ c\otimes d,\ in' \circ h\times k \circ out' \circ c\otimes d,\ f\otimes g\ ). \end{aligned}$$

We are now in a position to define the exponents. If $h, k : V\to V$, let $[h,k] = \Lambda(k \circ eval_{V,V} \circ (id\times h)) : V^V\to V^V$. Given two objects f and g in $\mathbf{Ret_V}$, their exponent is:

$$g^f = in'' \circ [f,g] \circ out'' : V\to V.$$

This is a good definition, since

$$\begin{aligned} g^f \circ g^f &= in'' \circ [f,g] \circ out'' \circ in'' \circ [f,g] \circ out'' && \text{by def. of } g^f \\ &= in'' \circ [f,g] \circ [f,g] \circ out'' \\ &= in'' \circ \Lambda(g \circ eval_{V,V} \circ (id\times f)) \circ \Lambda(g \circ eval_{V,V} \circ (id\times f)) \circ out'' && \text{by def. of } [f,g] \\ &= in'' \circ \Lambda(g \circ eval_{V,V} \circ (id\times f) \circ \Lambda(g \circ eval_{V,V} \circ (id\times f))\times id\ ) \circ out'' \\ &= in'' \circ \Lambda(g \circ eval_{V,V} \circ \Lambda(g \circ eval_{V,V} \circ (id\times f))\times id \circ (id\times f)\ ) \circ out'' \\ &= in'' \circ \Lambda(g \circ g \circ eval_{V,V} \circ (id\times f) \circ (id\times f)\ ) \circ out'' \quad \text{by } (\beta) \\ &= in'' \circ \Lambda(g \circ eval_{V,V} \circ (id\times f)\ ) \circ out'' && \text{since g, f are retractions} \\ &= in'' \circ [f,g] \circ out'' \quad \text{by def. of } [f,g] \\ &= g^f && \text{by def. of } g^f. \end{aligned}$$

The evaluation function is $(g^f \otimes f, ev, g)$ where

$$ev = eval_{V,V} \circ (out" \times id) \circ out' \circ g^f \otimes f$$

The currying operation $\mathbf{\Lambda_{ret}}$ is so defined: given a morphism $(c \otimes f, h, g)$,

$$\mathbf{\Lambda_{ret}}(c \otimes f, h, g) = (c, in" \circ \Lambda(h \circ in'), g^f).$$

This is a good definition of morphism in $\mathbf{Ret_V}$; indeed

$g^f \circ in" \circ \Lambda(h \circ in') \circ c =$

$= in" \circ \Lambda(g \circ eval_{V,V} \circ (id \times f)) \circ out" \circ in" \circ \Lambda(h \circ in') \circ c$ by def. of g^f

$= in" \circ \Lambda(g \circ eval_{V,V} \circ (id \times f)) \circ \Lambda(h \circ in' \circ c \times id) out" \circ in" = id$

$= in" \circ \Lambda(g \circ eval_{V,V} \circ (id \times f) \circ \Lambda(h \circ in' \circ c \times id) \times id)$

$= in" \circ \Lambda(g \circ eval_{V,V} \circ \Lambda(h \circ in' \circ c \times id) \times id \circ (id \times f))$

$= in" \circ \Lambda(g \circ h \circ in' \circ c \times id \circ id \times f)$ by (β)

$= in" \circ \Lambda(g \circ h \circ in' \circ c \times f)$

$= in" \circ \Lambda(g \circ h \circ c \otimes f \circ in')$ by def. of $c \otimes f$

$= in" \circ \Lambda(h \circ in').$

We only prove the axiom β, and leave as an exercise for the reader to prove η .

$(g^f \otimes f, ev, g) \circ (\mathbf{\Lambda_{ret}}(c \otimes f, h, g) \otimes (f, f, f)) =$

$= (g^f \otimes f, ev, g) \circ ((c, in" \circ \Lambda(h \circ in'), g^f) \otimes (f, f, f))$ by def. of $\mathbf{\Lambda_{ret}}$

$= (g^f \otimes f, ev, g) \circ (c \otimes f, in' \circ (in" \circ \Lambda(h \circ in')) \times f \circ out' \circ c \otimes f , g^f \otimes f)$ by def. of $\otimes$ on arrows

$= (c \otimes f, ev \circ in' \circ (in" \circ \Lambda(h \circ in')) \times f \circ out' \circ c \otimes f , g)$ by composition in $\mathbf{Ret_V}$

$= (c \otimes f, eval_{V,V} \circ (out" \times id) \circ out' \circ g^f \otimes f \circ in' \circ (in" \circ \Lambda(h \circ in')) \times f \circ out' \circ c \otimes f, g)$

by def. of ev

$= (c \otimes f, eval_{V,V} \circ (out" \times id) \circ g^f \times f \circ (in" \circ \Lambda(h \circ in')) \times f \circ out' \circ c \otimes f , g)$ by def. of $g^f \otimes f$

$= (c \otimes f, eval_{V,V} \circ (out" \circ g^f \circ in" \circ \Lambda(h \circ in')) \times f \circ out' \circ c \otimes f , g)$ as $f \circ f = f$

$= (c \otimes f, eval_{V,V} \circ (\Lambda(g \circ eval_{V,V} \circ (id \times f)) \circ \Lambda(h \circ in')) \times f \circ out' \circ c \otimes f , g)$ by def. of g^f

$= (c \otimes f, eval_{V,V} \circ (\Lambda(g \circ eval_{V,V} \circ \Lambda(h \circ in') \times id \circ (id \times f)) \times f \circ out' \circ c \otimes f , g)$

$= (c \otimes f, eval_{V,V} \circ \Lambda(g \circ h \circ in' \circ id \times f) \times f \circ out' \circ c \otimes f , g)$ by β

$= (c \otimes f, g \circ h \circ in' \circ id \times f \circ c \times f \circ out' , g)$ by β

$= (c \otimes f, g \circ h \circ c \otimes f , g)$

$= (c \otimes f, h, g).$

This completes the proof that $\mathbf{Ret_V}$ is a CCC. ♦

8.8.7 Lemma *If V is a reflexive object in a CCC $\boldsymbol{C}$, then there is a fixpoint operator $Fix: V^V \to V$.*

Proof Let

$$F = eval \circ \langle id, in" \rangle : V^V \to V$$

$$H = \Lambda(eval \circ (id \times (F \circ out"))) : V^V \to V^V$$

$Fix = F \circ H$ is a fixpoint operator for V; indeed,

$$\begin{aligned} F\circ H &= \mathrm{eval}\circ\langle \mathrm{id}, \mathrm{in}''\rangle\circ\Lambda(\mathrm{eval}\circ(\mathrm{id}\times(F\circ\mathrm{out}''))) \\ &= \mathrm{eval}\circ\Lambda(\mathrm{eval}\circ(\mathrm{id}\times(F\circ\mathrm{out}''))\times\mathrm{id}\circ\langle \mathrm{id}, \mathrm{in}''\circ\Lambda(\mathrm{eval}\circ(\mathrm{id}\times(F\circ\mathrm{out}''))\rangle \\ &= \mathrm{eval}\circ\mathrm{id}\times(F\circ\mathrm{out}'')\circ\langle \mathrm{id}, \mathrm{in}''\circ\Lambda(\mathrm{eval}\circ(\mathrm{id}\times(F\circ\mathrm{out}''))\rangle \\ &= \mathrm{eval}\circ\langle \mathrm{id}, F\circ\Lambda(\mathrm{eval}\circ(\mathrm{id}\times(F\circ\mathrm{out}''))\rangle \\ &= \mathrm{eval}\circ\langle \mathrm{id}, F\circ H\rangle. \blacklozenge \end{aligned}$$

8.8.8 Theorem *Every object f in $\mathbf{Ret_V}$ has a fixpoint operator Fix_f.*

Proof Let $\mathrm{Fix}: V^V\to V$ be a fixpoint operator for V. Define

$$\mathrm{Fix}_f = (f^f, \mathrm{Fix}\circ\mathrm{out}''\circ f^f, f).$$

We leave it to the reader to prove that this is a good definition, that is:

$$(*)\quad \mathrm{Fix}\circ\mathrm{out}''\circ f^f = f\circ\mathrm{Fix}\circ\mathrm{out}''\circ f^f\circ f^f.$$

We must prove that

$$\mathrm{Fix}_f = (\, f^f\otimes f, \mathrm{ev}, f)\circ\mathbf{<}(f^f, f^f, f^f), \mathrm{Fix}_f\mathbf{>},$$

where $\mathrm{ev} = \mathrm{eval}_{V,V}\circ(\mathrm{out}''\times\mathrm{id})\circ\mathrm{out}'\circ f^f\otimes f$.

Compute then

$$\begin{aligned} \mathrm{Fix}_f &= (f^f, \mathrm{eval}_{V,V}\circ\langle \mathrm{id}, \mathrm{Fix}\rangle\circ\mathrm{out}''\circ f^f, f) && \text{by def. of } \mathrm{Fix}_f \\ &= (f^f, \mathrm{eval}_{V,V}\circ\langle \mathrm{out}''\circ f^f, f\circ\mathrm{Fix}\circ\mathrm{out}''\circ f^f\rangle, f) && \text{by (*)} \\ &= (f^f, \mathrm{eval}_{V,V}\circ(\mathrm{out}''\times\mathrm{id})\circ f^f\times f\circ\langle f^f, \mathrm{Fix}\circ\mathrm{out}''\circ f^f\rangle, f) \\ &= (f^f, \mathrm{eval}_{V,V}\circ(\mathrm{out}''\times\mathrm{id})\circ\ \mathrm{out}'\circ f^f\otimes f\circ\mathrm{in}'\circ\langle f^f, \mathrm{Fix}\circ\mathrm{out}''\circ f^f\rangle, f -) && \text{by def. of } f^f\otimes f \\ &= (f^f, \mathrm{ev}\circ\mathrm{in}'\circ\langle f^f, \mathrm{Fix}\circ\mathrm{out}''\circ f^f\rangle, f) && \text{by def. of ev} \\ &= (\, f^f\otimes f, \mathrm{ev}, f)\circ(f^f, \mathrm{in}'\circ\langle f^f, \mathrm{Fix}\circ\mathrm{out}''\circ f^f\rangle, f^f\otimes f) && \text{by composition in } \mathbf{Ret_V} \\ &= (\, f^f\otimes f, \mathrm{ev}, f)\circ\mathbf{<}(f^f, f^f, f^f), (f^f, \mathrm{Fix}\circ\mathrm{out}''\circ f^f, f)\mathbf{>} && \text{by pairing } \mathbf{<,>} \text{ in } \mathbf{Ret_V} \\ &= (\, f^f\otimes f, \mathrm{ev}, f)\circ\mathbf{<}(f^f, f^f, f^f), \mathrm{Fix}_f\mathbf{>}. \blacklozenge \end{aligned}$$

References. For an introduction to Proof Theory, natural deduction, and intuitionisitic logic, the reader can consult Prawitz (1965) and Takeuti (1975). In particular, the latter inspired our presentation of the cut-elimination theorem. The "formulae as types" analogy is explained in Howard (1980), but the main ideas go back to work of Curry. The connections between λ-calculus and CCC's were first explored by Lambek. The equivalence is shown in Lambek (1980) and Scott (1980). A full account of the relation between λ-calculus, CCC's, and Proof Theory may be found in Lambek and Scott (1986). We tried here to complement that work by relating it to Tarskian semantics and emphasizing the role of "structures," for the convenience of the reader who may begin with our approach and continue with further, more abstract readings. In section 8.7, we dealt only with CCC's, i.e., with the models of $\boldsymbol{\lambda\beta\eta\pi}^t$. Weaker calculi, that is, typed combinatory logic and $\boldsymbol{\lambda\beta\pi}^t$, are discussed in Hayashi (1985) and Martini (1988), which introduce various notions of "weak Cartesian closedness" for this purpose.

Chapter 9

REFLEXIVE OBJECTS AND THE TYPE-FREE LAMBDA CALCULUS

The main aim of this book is to present category theoretic tools for the understanding of some common constructions in computer science. This is largely done in the perspective of denotational semantics, in particular in this second part of the book. It is commonly agreed that a fruitful area of application of denotational semantics has been the understanding and, in some cases, the design of functional languages. This is exactly because the theory of the intended structures is a "theory of functions," indeed Category Theory.

Functional languages are based on the notion of application and functional abstraction. That is programs are "applied," like functions, to data and, given the formal, algebraic definition of a function, it may be turned into an applicative program by "functional completeness" or "lambda abstraction." Observe that the expressive power is mostly based on recursive definitions, even though a different approach is suggested by the higher order calculi discussed in chapter 11.

The aim of this chapter is to clarify the categorical significance of the quoted expressions in the previous paragraph, e.g., "applied", "functional completeness", "lambda abstraction", "uniform", "recursive definition", in the context of a "type-free" programming style. In the previous chapter we dealt with the **typed** λ-calculus, and we discussed typed functional "application" and "abstraction" which have an immediate interpretation in CCC's. As already mentioned, it is easy to conceive a variant of the previous calculus by just erasing all type restrictions in the term formation rules. This defines the (**type-free** or **un(i)typed**) **λ-calculus**, where there is no distinction between functions and data. (In remark 9.5.12 we will suggest some good reasons by which one may better like to consider the type-free λ-calculus as a *typed* calculus with just one type: a *unityped* calculus). The set Λ of terms of the λ-calculus is thus defined by means of the following rules, starting by a given set of (type-free) variables V:

Variables if $x \in V$, then $x \in \Lambda$;
Application if $M \in \Lambda$, and $N \in \Lambda$ then $MN \in \Lambda$;
Abstraction if $M \in \Lambda$, then $\lambda x.M \in \Lambda$.

Free and bound occurences of a variable in a term, and the substitution $[N/x]M$ of a term N for a variable x in M, are defined as for the typed calculus. As usual, we identify terms that differ from each other only for the names of bound variables (α-conversion).

The **λ-theory** deals with the convertibility $M = N$ of two terms M and N. It is axiomatized by the rules

β. $(\lambda x.M)N = [N/x]M$, for x free for N in M
η. $\lambda y.My = M$, for y not free in M (write $y \in FV(M)$)

toghether with the axioms and rules needed for turning "=" into a congruence relation.

The λ-calculus is the prototype of every untyped functional programming language. Many functional languages were directly derived from the λ-calculus, from Landin's ISWIM (a notational variant of λ-calculus with an explicit recursive operator) to Edinburgh ML. Even McCarthy's language LISP, the first running functional programming language, and still one of the most used in several applications of computer science, is greatly endebted to the λ-calculus. Besides the λ-notation, LISP inherits from λ-calculus both the formal elegance and the concise syntax, essentially adding only a few primitives for list manipulation. The main difference is in the binding strategy for variables, which is static for λ-calculus and dynamic for LISP. For example, without taking into account the inessential syntactic differences between the two formalisms, let us see how the following expression is evaluated in the two languages:

$$(\lambda z.\ (\lambda y.\ (\lambda z.yM)N\ )\ (\lambda x.xz)\ )P$$

In λ-calculus, we have the following reduction sequence of reductions:

$$(\lambda z.\ (\lambda y.\ (\lambda z.yM)N\ )\ (\lambda x.xz)\ )P \rightarrow \lambda y.\ (\lambda z.yM)N\ )\ (\lambda x.xP)$$
$$\rightarrow \lambda z.\ (\lambda x.xP)M)N$$
$$\rightarrow (\lambda x.xP)M$$
$$\rightarrow MP$$

In contrast to this, LISP will first bind z to P, then bind y to λx.xz; next z will be rebound to N, and finally yM will be evaluated. This means that x will be bound to M, and then Mz is evaluated. Since LISP uses dynamic binding, the latest active bindings of the variable z is used, i.e., the evaluation of $(\lambda z.\ (\lambda y.\ (\lambda z.yM)N\ )\ (\lambda x.xz)\ )P$ is reduced to the evaluation of MN.

This has been often considered as an anomaly of LISP: in many LISP dialects, there are syntactic constructs for defining functions that guarantee a static binding for their formal parameters and, moreover, some recent LISP-like languages have completely converted to static binding (e.g., Scheme). A first consequence of dynamic binding is that the rule of α-conversion does not hold any more: in the example above, if we replace z with another variable in λz.yM, we obtain a different behavior. LISP violates *referential transparency*, while λ-calculus does satisfy it. This is not only a merely theoretical property: in programming terms, referential transparency means that, in order to understand a structured program, we need only to understand the *denotation* of the subprograms, and not their *connotations* (for example, we do not need to be concerned with the naming of variables used within the programs). These ideas are expressed in the philosophy of *modular programming*, that is of the programming style that requires the construction of program segments as self-contained boxes, or modules, with well-defined interfaces. We shall discuss in the last chapters of this book how this philosophy applies so well to strongly typed polymorphic languages.

The current treatment of both programming concepts of referential transparency and modularity provides a relevant example of an area that is greatly indebted to twenty-odd years work of in denotational semantics. We present in this chapter the categorical understanding of the semantics of

type-free Combinatory Logic and λ-calculus, whose challenging mathematical meaning actually started that work. In section 9.4, we hint at how the categorical approach suggested a new set of combinators and a simple abstract machine for implementig head reduction (CAM).

9.1 Combinatory Logic

Combinatory Logic (CL) is based on an even simpler language than λ-calculus: it just contains variables and two constant symbols K and S. Their operational behaviour is axiomatized by the rules for equality and

k. $Kxy = x$

s. $Sxyz = xz(yz)$

where, as for the λ-calculus, $M_1M_2...M_n$ stands for $(...(M_1M_2)...M_n)$.

The expressive power of λ-calculus and CL is due to their **combinatorial completeness**. That is, for any variable x and term M in their languages, there exists $\langle x\rangle M$ such that

abs. $(\langle x\rangle M)N = [N/x]M$, and $x \notin FV(\langle x\rangle M)$.

For the λ-calculus, this comes with the definition: just set $\langle x\rangle M = \lambda x.M$. As for CL, define inductively

$\langle x\rangle x = I \equiv SKK$;

$\langle x\rangle M = KM$, if M does not contain x;

$\langle x\rangle MN = S(\langle x\rangle M)(\langle x\rangle N)$.

(In general, for $\underline{x} = x_1, ...,x_n$, set $\langle \underline{x}\rangle M = \langle x_1\rangle...(\langle x_n\rangle M)$).

As a matter of fact, CL is the simplest type-free language which is functionally complete; moreover, and surprisingly enough, in 1936 Kleene proved that CL is powerful enough to compute all partial recursive functions.

Note that in type-free universes, there is no distinction between data and functions. In set-theoretic terms, this means that it is possible to apply one to the other in an undistinguished **applicative structure** (X,·), i.e., a set X with a binary operation · .

9.1.1 Definition *A model (X,·, K, S) of CL, called **Combinatory Algebra**, is an applicative structure (X,·) with two distinguished elements K, S∈X such that*

$\forall x,y \quad (K\cdot x)\cdot y = x$

$\forall x,y,z \quad ((S\cdot x)\cdot y)\cdot z = (x\cdot z)\cdot(y\cdot z)$.

As usual, we suppose that the operation · of the applicative structure associate to the left; moreover we shall usually omit it when writing terms. For instance, $(K\cdot x)\cdot y$ will be simply written as Kxy.

9.1.2 Definition *Given an environment ξ, that is a map from the set of variables of CL to X, the* ***interpretation*** *$[M]\xi$ of a combinatory term M in ξ, is inductively defined as follows:*

$$[K]\xi = \mathsf{K}$$
$$[S]\xi = \mathsf{S}$$
$$[x]\xi = \xi(x)$$
$$[MN]\xi = [M]\xi[N]\xi .$$

An interesting semantic consequence of (abs) is the following lemma which will be used later on.

9.1.3 Lemma *Let $(X,\cdot, \mathsf{K}, \mathsf{S})$ be a Combinatory Algebra. For any combinatory term M, any environment ξ and any $a \in X$,*

$$[<x>M]\xi \cdot a = [M]\xi(x=a)$$

where $\xi(x=a)$ is the environment defined by : $\xi(x=a)(z) =$ if $x=z$ then a else $\xi(z)$.

Proof

$[<x>M]\xi \cdot a = [<x>M]\xi \cdot [x]\xi(x=a)$

$= [<x>M]\xi(x=a) \cdot [x]\xi(x=a)$ since x do not occur in <x>M

$= [(<x>M)x]\xi(x=a)$

$= [M]\xi(x=a)$ by (abs). ♦

Clearly, the λ-calculus is at least as expressive as CL, since $\mathsf{K}_\lambda \equiv \lambda xy.x$ and $\mathsf{S}_\lambda \equiv \lambda xyz.xz(yz)$ represent K and S in λ-calculus (and do the same job). By this definition of K and S we obtain a sound translation form CL to λ-calculus, i.e., a translation which preserves term equalities. In the other direction, the abstraction mechanism <x>M described above naturally suggests the following translation.

9.1.4 Definition *Given a λ-term M, the associated term M_{CL} in Combinatory Logic is inductively defined by*

$$x_{CL} = x$$
$$(MN)_{CL} = M_{CL}N_{CL}$$
$$(\lambda x.M)_{CL} = <x>M_{CL}.$$

Unfortunately, this translation is not sound, that is, not all the equations provable in the λ-theory still hold after the translation. Consider for example the two equal terms $M \equiv \lambda y.x$ and $N \equiv \lambda y.(\lambda z.z)x$. Their translation by means of combinators is, respectively:

$M_{CL} = (\lambda y.x)_{CL}$

$= <y>x_{CL}$

$= <y>x$

$= \mathsf{K}x$

$$\begin{aligned} N_{CL} &= (\lambda y.(\lambda z.z)x)_{CL} \\ &= \langle y\rangle((\lambda z.z)x)_{CL} \\ &= \langle y\rangle((\lambda z.z)_{CL}x_{CL}) \\ &= \langle y\rangle((SKK)x) \\ &= S((\langle y\rangle(SKK))\ \langle y\rangle x\ ) \\ &= S((K(SKK))Kx) \end{aligned}$$

and $Kx \neq S((K(SKK))Kx)$.

The problem derives from the fact that in Combinatory Logic $M = N$ does not imply $\langle x\rangle M = \langle x\rangle N$. This fact is independent from the particular abstraction mechanism adopted and it is actually related to the absence of a "canonical" choice for $\langle x\rangle M$ (see references).

From the point of view of computer science, the interest in Combinatory Logic derives more from implementation then from semantics. Indeed, β-conversion, as it is formulated in the λ-calculus, give rise to the well-known, conceptually simple, but syntactically fastidious problem of name clashes. For instance, $M \equiv (\lambda xy.x)y$ does not reduce to $\lambda y.y$, but to $\lambda z.y$. This kind of problems does not sussist in Cominatory Logic, which thus provides a convenient intermediate code where the λ-calculus can be compiled before execution. For example, the previous term M is compiled as:

$$\begin{aligned} M_{CL} &= ((\lambda xy.x)y)_{CL} \\ &= (\lambda xy.x)_{CL}\ (y)_{CL} \\ &= (\langle x\rangle Kx)y \\ &= S(KK)(SKK)y \end{aligned}$$

and its reduction, using, say, an innermost-leftmost strategy, yields:

$$\begin{aligned} S(KK)(SKK)y &= (KKy)(SKKy) \\ &= K(SKKy) \\ &= K((Ky(Ky)) \\ &= Ky\ . \end{aligned}$$

9.2 From Categories to Functionally Complete Applicative Structures

In this section, we suggest how to understand, in categorical terms, the difference between "functional completeness" and "lambda abstraction" and, later, characterize both notions, in absence of type constraints. As mentioned in the introduction, CL is the simplest type-free language that is functionally complete, since, for every term M, there exists $\langle x\rangle M$ that satisfies (abs). In case the choice of $\langle x\rangle M$ is "uniform in M", one has lambda abstraction and λ-calculus: i.e., $\langle x\rangle M$ is canonically given by $\lambda x.M$.

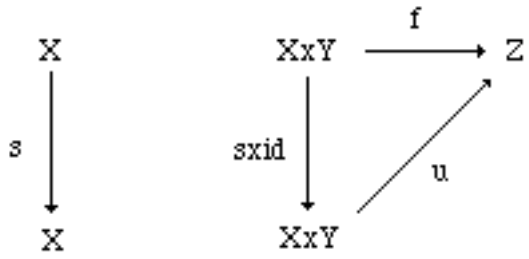
In order to give categorical meaning to this complex situation, we proceed as follows: we start with recovering applicative structures, in particular functionally complete ones, in Cartesian

categories (see 9.2.1-9.2.5); then we shift to the realm of Cartesian closed categories, where the existence of function spaces (exponents) allows a better understanding of the notion of functional completeness (9.2.6-9.2.7) and lambda-abstraction (9.2.8-9.2.12). In section 5, we will give a fully categorical characterization of models of these type-free calculi.

9.2.1 Definition *Let* $\mathbf{C}$ *be a Cartesian category,* T *its terminal object, and* U *an object in* $\mathbf{C}$*, with* $T < U$ *and* $u \in \mathbf{C}[U\times U,U]$*. The* ***applicative structure associated to u****,* $\underline{A}(u)$*, is given by* $\underline{A}(u) = (\mathbf{C}[T,U],\cdot)$*, where* $a\cdot b = u\circ\langle a,b\rangle$*.*

In a category with a terminal object T, $T < U$ simply generalizes the set-theoretic notion that U is "not empty". Clearly, $\underline{A}(u)$ is nontrivial (i.e., it contains at least two elements) iff $T < U$ is strict, i.e., is not an isomorphism.

9.2.2 Definition *Let* $\mathbf{C}$ *be a cartesian category. Then* $u \in \mathbf{C}[X\times Y,Z]$ *is* ***Kleene-universal*** *(K-universal) if* $\forall f \in \mathbf{C}[X\times Y,Z]\ \exists s \in \mathbf{C}[X,X]\ \ f = u\circ(s\times id)$ *, i.e.,*



Kleene-universality is a *weak* (co)universality property, since no unicity of s is required. It has an obvious recursion-theoretic meaning: indeed K-universality generalizes the s-m-n (iteration) theorem, with $X = Y = Z = \omega$ and with f a (total) recursive function, i.e., a morphism from $(\omega,id)\times(\omega,id)$ to (ω,id), in the category **EN** of numbered sets.

9.2.3 Definition *Let* $\mathbf{C}$ *be Cartesian and* $u \in \mathbf{C}[X\times X,X]$*. Then* $u^{(n)} \in \mathbf{C}[X\times X^n,X]$ *is inductively defined by* $u^{(0)} = id$*,* $u^{(n+1)} = u^{(n)}\circ(u\times id^n)$*, that is,*

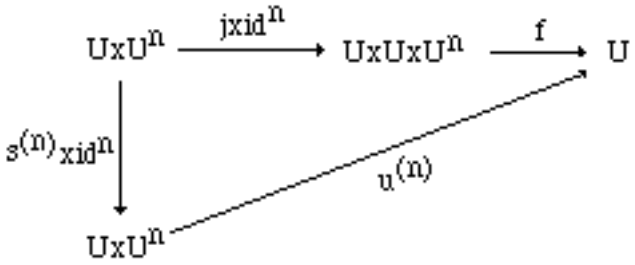
$$u^{(n+1)}\colon X\times X\times X^n \xrightarrow{u\times id^n} X\times X^n \xrightarrow{u^{(n)}} X\ ,\quad \text{where } X^{n+1} = X\times X^n.$$

It is easy to observe that $u^{(n)}$ corresponds exactly to the application of n+1 arguments, from left to right, e.g. $u^{(2)}\circ\langle a,b,c\rangle = u\circ(u\times id)\circ\langle a,b,c,\rangle = u\circ\langle u\circ\langle a.b\rangle,c\rangle$. We write $a\cdot b\cdot c$ for $(a\cdot b)\cdot c$.

9.2.4 Lemma *Let* $\mathbf{C}$ *be Cartesian. Assume that, for some* U *in* $\mathbf{C}$*,* $U\times U < U$ *and there is a K-universal* $u \in \mathbf{C}[U\times U,U]$*. Then* $\forall n\ u^{(n)} \in \mathbf{C}[U\times U^n,U]$ *is K-universal.*

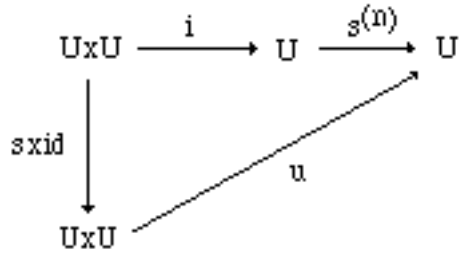
Proof By assumption, this is true for n = 1. Let $U\times U < U$ via (i,j) and $f\in\mathbf{C}[U\times U^{n+1},U]$. Then, by the inductive hypotesis, for some $s^{(n)}\in C[U,U]$ the following diagram commutes:

(1)



By assumption, for some $s\in\mathbf{C}[U,U]$ one also has

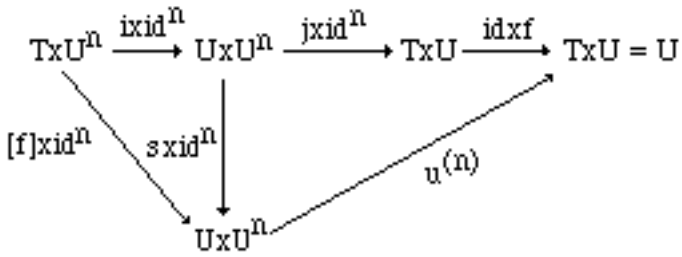
(2)



Then compute
$$\begin{aligned} f &= f\circ(j\times id^n)\circ(i\times id^n) \\ &= u^{(n)}\circ(s^{(n)}\times id^n)\circ(i\times id^n) && \text{by (1)} \\ &= u^{(n)}\circ(u\times id^n)\circ(s\times id^{n+1}) && \text{by (2)} \\ &= u^{(n+1)}\circ(s\times id^{n+1}).\ \blacklozenge \end{aligned}$$

9.2.5 Theorem *Let* ***C*** *be a Cartesian category. Assume that, for some object U, one has* $T < U$, $U\times U < U$ *and there exists a K-universal* $u\in\mathbf{C}[U\times U,U]$. *Then* $\underline{A}(u)$ *is a combinatory algebra.*

Proof Let $T < U$ via (i_T,j_T). Then, by lemma 9.2.4, $\forall n$, $\forall f\in\mathbf{C}[U^n,U]$ $\exists s\in\mathbf{C}[U,U]$ such that the following diagram commutes, with $[f] = s\circ i_T$ (we write i and j for i_T and j_T):



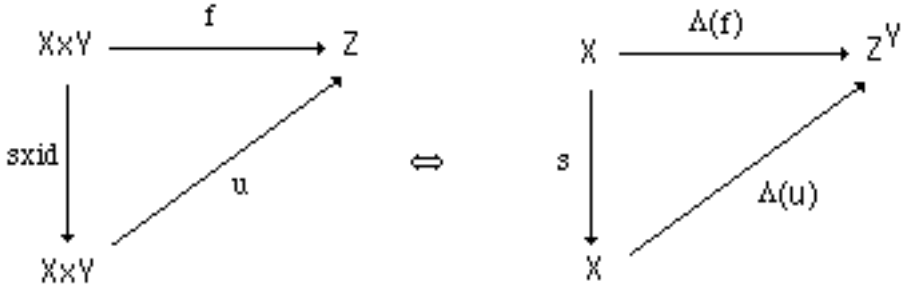
Thus, $u^{(n)}\circ[f]\times d^n = id_T\times f = f$. Since $u^{(n)}$ is the application, from left to right, of its n+1 arguments, [f] "represents" f, with respect to application. By 9.2.1, we only need to define $f\in\mathbf{C}[U^2,U]$ and $g\in\mathbf{C}[U^3,U]$ such that [f] and [g] represent K and S, respectively. For this purpose, take $f = pr^2_1\in\mathbf{C}[U^2,U]$ and $g = u\circ\langle u\circ\langle pr^3_1,pr^3_3\rangle,u\circ\langle pr^3_2,pr^3_3\rangle\rangle\in\mathbf{C}[U^3,U]$. ♦

Theorem 9.2.5 provides sufficient conditions in order to construct "functionally complete" objects. Theorem 9.5.6 will show that these conditions are also necessary.

A further insight, though, into functional completeness may be given in CCC's. The advantage of dealing with CCC's is that for any object X one may consider its "function space" or exponent X^X, as an object. As a matter of fact, functional completeness, in its various forms of increasing strength (combinatory algebras, λ-algebras, λ-models (see below)), expresses some sort of privileged relation between an object in a category and its "function space": its representability, say, in the sense expressed in the proof of theorem 9.2.5. In CCC's K-universal morphisms and principal morphisms are related as follows:

9.2.6 Proposition *Let* $\mathbf{C}$ *be a CCC and* Λ *be the isomorphism* $\mathbf{C}[X\times Y,Z] \cong \mathbf{C}[X,Z^Y]$. *Then* $u\in\mathbf{C}[X\times Y,Z]$ *is K-universal iff* $\Lambda(u)\in\mathbf{C}[X,Z^Y]$ *is principal.*

Proof The isomorphism Λ implies, by definition, the equivalence of the following diagrams:



and we are done. ♦

The connection between K-universal and principal morphisms should remind the reader that the latter correspond to (acceptable) Gödel numberings from ω to PR, the partial recursive functions, when these are viewed as objects in the category **EN** of numbered sets (see section 2.2). Note though that Kleene's $(\omega, \cdot)$ is a *partial* combinatory algebra and does not yield a model of Combinatory Logic. *Total* combinatory algebras turn out to be nontrivial constructions and may be obtained, for instance, in higher types, as it will be shown later. The categorical description, in terms of K-universal maps or principal morphisms, sheds some light on the connections between Gödel numberings and combinatory completeness. As a matter of fact, by proposition 9.2.6 one may then restate theorem 9.2.5 in terms of CCC's and principal morphisms.

9.2.7 Proposition *Let* $\mathbf{C}$ *be a CCC. Assume that, for some* U *in* $\mathbf{C}$, $T < U$, $U\times U < U$ *and there exists a principal* $p\in\mathbf{C}[U,U^U]$. *Then* $\underline{A}(\Lambda^{-1}(p))$ *is a combinatory algebra.*

The previous proposition suggests more explicitly the connection between the definition of application and the morphism eval in CCC's. Just observe that, by definition of "·", one has in a CCC

$$a\cdot b = \Lambda^{-1}(p)\circ\langle a,b\rangle = \mathrm{eval}\circ(p\times id)\circ\langle a,b\rangle = \mathrm{eval}\circ((p\circ a)\times b).$$

Informally, the equation above means "transform a into a morphism, by p, then apply it to b."

This process generalizes the way Gödel numberings associate functions to numbers. Similarly as for the partial recursive functions, there is in general no "canonical" way to go backwards, that is to choose uniformely and effectively a representative for each representable function. That is, this representative does not need to be unique and it is not possible to choose a representative for each representable function in a "uniform" way, i.e., by a morphism in the category. This is, though, possible in λ-models. We define them here in a first order manner, as particular combinatory algebras, with a suitable "choice" operator.

9.2.8 Definition *Let* $A = (X, \cdot)$ *be an applicative structure. Then*

i. A *is a* ***λ-model*** *if for some* $k,s,\varepsilon \in X$ *one has:*

$k.$ $\quad \forall x,y \quad kxy = x$;

$s.$ $\quad \forall x,y,z \quad sxyz = xz(yz)$;

$\varepsilon_1.$ $\quad \forall x,y \quad \varepsilon xy = xy$;

$\varepsilon_2.$ $\quad \forall x,y \quad (\forall z\ xz = yz) \Rightarrow \varepsilon x = \varepsilon y$;

$\varepsilon_3.$ $\quad \varepsilon\varepsilon = \varepsilon$.

ii. A *is an* ***extensional*** *λ-model if one also has* $\forall x \ \varepsilon x = x$.

ε has to be understood as a choice operator that picks up a canonical representative for each representable function. ε coincides with the canonical representative of the function it represents, by axiom (ε_3). In extensional λ-models, there is just one representative and $(\square\varepsilon_1)$, $(\square\varepsilon_3)$ are derived. Note that $A = (X, \cdot)$ is an extensional λ-model iff A is a combinatory algebra and $\forall x,y \ (\forall z\ xz = yz) \Rightarrow x = y$.

There exists an obvious formal system of combinators K, S, ε associated to the previous notion of λ-model, which we shall call **CLε** (we gave priority to the notion of model because we mainly focus here on semantical aspects). The interpretation of CLε in λ-models is straighforward. Note also that CLε may be easily and soundly translated into λ-calculus, by taking ε to λxy.xy.

Conversely, the combinator ε can be used to "clean" the translation of a lambda term by means of combinators described in definition 9.1.4 :

9.2.9 Definition *Given a λ-term* M*, the associated term* $M_{CL\varepsilon}$ *in CLε is inductively defined by*

$$x_{CL\varepsilon} = x$$
$$(MN)_{CL\varepsilon} = M_{CL\varepsilon}N_{CL\varepsilon}$$
$$(\lambda x.M)_{CL\varepsilon} = \varepsilon \cdot \langle x \rangle M_{CL\varepsilon}.$$

This "refinement" is completely worthless from an implementative point of view, since the reduction process is essentially unaffected by the combinator ε, as it is stated by equation (ε_1). On the contrary,

it is relevant in semantics, since it allows a simple definition of a sound interpretation of λ–terms in λ-models, as follows:

9.2.10 Definition *Let $A = (X, \cdot, k, s, \varepsilon)$ be a λ-model. The interpretation $[\![M]\!]\xi$ of a λ-term M in A with respect to an environment ξ is the semantics of the associated combinatorial term $M_{CL\varepsilon}$, i.e.,*

$$[\![M]\!]\xi = [M_{CL\varepsilon}]\xi.$$

We omit the soundness proof, which is technically straightforward and almoust evident from the previous discussions.

In the next two results we show how to derive λ-models from reflexive objects in categories with enough points.

9.2.11 Theorem *Let $\mathbf{C}$ be a CCC with enough points. Assume that, for some U in $\mathbf{C}$, one has $U^U < U$ via (ψ, ϕ). Then, for $\varepsilon = \psi\circ\Lambda(\psi\circ\phi\circ snd)$, $\underline{A} = (\underline{A}(\Lambda^{-1}(\phi)),\varepsilon)$ is a λ-model.*

Proof $\phi \in \mathbf{C}[U, U^U]$ is principal; moreover by 2.3.6, $T < U$ and $U\times U < U$. Thus, for $a\cdot b = eval\circ\langle(\phi\circ a), b\rangle = eval\circ(\phi\times id)\circ\langle a,b\rangle$ and some suitable K and S, $(\underline{A}(\Lambda^{-1}(\phi)),\cdot,K,S)$ is a combinatory algebra, i.e. (k) and (s) in 9.2.8 hold. Define now $\varepsilon = \psi\circ\Lambda(\psi\circ\phi\circ snd) : T\to U$ (that is, informally, $\varepsilon = \psi(\psi\circ\phi)$). Note first that, for any a,

(†) $\qquad \varepsilon \cdot a = \psi \circ \phi \circ a$

indeed:

$\varepsilon\cdot a$	$= (\psi\circ\Lambda(\psi\circ\phi\circ snd))\cdot a$	by def. of ε
	$= eval\circ\langle(\phi\circ\psi\circ\Lambda(\psi\circ\phi\circ snd)),a\rangle$	by def. of "·"
	$= eval\circ\langle(\Lambda(\psi\circ\phi\circ snd)),a\rangle$	since $\phi\circ\psi = id$
	$= eval\circ(\Lambda(\psi\circ\phi\circ snd)\times id)\circ\langle id\times a\rangle$	
	$= \psi\circ\phi\circ snd\circ\langle id\times a\rangle$	
	$= \psi\circ\phi\circ a$	

Then one has:

ε_1. $\varepsilon\cdot a\cdot b$	$= (\psi\circ\phi\circ a)\cdot b$	by (†)
	$= eval\circ\langle(\phi\circ\psi\circ\phi\circ a),b\rangle$	by def. of "·"
	$= eval\circ\langle(\phi\circ a),b\rangle$	since $\phi\circ\psi = id$
	$= a\cdot b$	by def. of "·"

ε_2. Suppose that $\forall z\ az = bz$. Then, since

$a\cdot z = eval\circ\langle(\phi\circ a),z\rangle = eval\circ((\phi\circ a)\times id)\circ\langle id,z\rangle$,

$b\cdot z = eval\circ\langle(\phi\circ b),z\rangle = eval\circ((\phi\circ b)\times id)\circ\langle id,z\rangle$,

and since $\mathbf{C}$ has enough points, we have $eval\circ((\phi\circ a)\times id) = eval\circ((\phi\circ b)\times id)$, and thus $\phi\circ a = \phi\circ b$. Then $\varepsilon\cdot a = \psi\circ\phi\circ a = \psi\circ\phi\circ b = \varepsilon\cdot b$.

ε_3. $\varepsilon\cdot\varepsilon = \psi\circ\phi\circ\psi\circ\Lambda(\psi\circ\phi\circ snd) = \psi\circ\Lambda(\psi\circ\phi\circ snd) = \varepsilon$. ♦

The definition of ε should be clear. Just note that $\psi\circ\phi : U\to U^U\to U$, i.e., ϕ gives a morphism for each point a in U and ψ chooses a "canonical" one, representing $\phi(a)$, as $\phi\circ\psi = id$. Then $\varepsilon = \psi\circ\Lambda(\psi\circ\phi\circ snd): T\to U$, internalizes $\psi\circ\phi$ as a point in U.

9.2.12 Corollary *Let **C** be a CCC with enough points.*

*i. If, for some U in **C**, $U^U < U$ via (i,j) and there exists $u\in \mathbf{C}[U\times U,U]$ K-universal, then also $\underline{A}(u)$ can be turned into a λ-model.*

ii. If $U^U \cong U$ via (ψ,ϕ), then $\underline{A}(u)$ is an extensional λ-model.

Proof

i. By theorems 9.2.6 and 9.2.11 and the definiton of principal morphism.

ii. $\varepsilon\cdot a = \psi\circ\phi\circ a = a$. ♦

9.3 Categorical Semantics of the λ-Calculus

In theorem 9.2.11 we proved that if **C** is a CCC with enough points, and $U\in Ob_{\mathbf{C}}$ is a reflexive object (i.e., $U^U<U$ via (ψ, ϕ)) then, for $\varepsilon=\psi\circ\Lambda(\psi\circ\phi\circ snd) : T\to U$, $(\underline{A}(\Lambda^{-1}(\phi)),\varepsilon)$ is a λ-model. We can thus give an interpretation of the lambda calculus as in definition 9.2.10.

In this section we define a more direct interpretation of the lambda calculus over such an object U, and relate the two interpretations.

9.3.1 Definition *Let **C** be a CCC with terminal object T. Let $U\in Ob_{\mathbf{C}}$ be a reflexive object via the retraction pair $(\psi: U^U\to U, \phi: U\to U^U)$. Let M be a λ-term with $FV(M) \subseteq \Delta = \{x_1,\ldots,x_n\}$. Define then $[M]_\Delta\in\mathbf{C}[U^n,U]$, where $U^n = (\ldots(T\times U)\times\ldots)\times U$ with n copies of U, as follows (we use the two projections fst and snd in a polymorphic fashion, and we omit the indexes):*

$$[x_i]_\Delta = snd\circ fst^{n-i} = pr^n_i$$
$$[MN]_\Delta = eval\circ\langle\phi\circ[M]_\Delta, [N]_\Delta\rangle$$
$$[\lambda x.M]_\Delta = \psi\circ\Lambda([M]_{\Delta\cup\{x\}}).$$

We do not prove the soundness of the interpretation; the reader interested in this result may consult the references.

Examples

1. Let $M = \lambda x.xx$.

$$[\lambda x.xx] = \psi\circ\Lambda([xx]_{\{x\}})$$

$= \psi \circ \Lambda(\text{eval} \circ < \phi \circ [x]_{\{x\}}, [x]_{\{x\}} >)$

$= \psi \circ \Lambda(\text{eval} \circ < \phi \circ \text{snd}, \text{snd} >) : T \to U$.

2. Consider the term $Y = \lambda x.(\lambda y.x(yy)(\lambda y.x(yy))$. This is a fixpoint operator, since for every M,

$YM = (\lambda x.(\lambda y.x(yy)(\lambda y.x(yy)))M$

$= (\lambda y.M(yy)(\lambda y.M(yy))$

$= M((\lambda y.M(yy))(\lambda y.M(yy)))$

$= M(YM)$

Let us interpret Y. We proceed by stages

$[\lambda y.x(yy)]_{\{x\}} = \psi \circ \Lambda([x(yy)]_{\{x,y\}})$

$= \psi \circ \Lambda(\text{eval} \circ < \phi \circ [x]_{\{x,y\}}, [yy]_{\{x,y\}} >)$

$= \psi \circ \Lambda(\text{eval} \circ < \phi \circ \text{snd} \circ \text{fst}, \text{eval} \circ < \phi \circ [y]_{\{x,y\}}, [y]_{\{x,y\}} > >)$

$= \psi \circ \Lambda(\text{eval} \circ < \phi \circ \text{snd} \circ \text{fst}, \text{eval} \circ < \phi \circ \text{snd}, \text{snd} > >)$

Let $P = \psi \circ \Lambda(\text{eval} \circ < \phi \circ \text{snd} \circ \text{fst}, \text{eval} \circ < \phi \circ \text{snd}, \text{snd} > >)$.

Then we have

$[Y] = [\lambda x.(\lambda y.x(yy)(\lambda y.x(yy))]$

$= \psi \circ \Lambda([(\lambda y.x(yy)(\lambda y.x(yy)))]_{\{x\}})$

$= \psi \circ \Lambda(\text{eval} \circ < \phi \circ P \circ \text{snd}, P \circ \text{snd} >): T \to U$

It is not difficult to prove that $\Lambda^{-1}(\phi \circ [Y]) \circ <!U^U, \psi> = \text{eval} \circ < \phi \circ P \circ \psi, P \circ \psi > : U^U \to U$ is a categorical fixpoint operator (see definition 8.8.2).

We now relate the two notions of categorical semantics given in 9.2.10 and 9.3.1. As a matter of fact they are essentially equivalent, the only difference being that the one in definition 9.3.1 does not need the concept of environment.

9.3.2 Definition *Let ξ: Var$\to \underline{A}(\Lambda^{-1}(\phi))$ be an environment . Let $\Delta = \{x_1,\ldots,x_n\}$ be a finite set of variables. Then*

$$\xi'_\Delta = <\ldots<id_T, \xi(x_1)>\ldots ,\xi(x_n)> : T \to T\times U\ldots\times U$$

We shall usually omit the subscript Δ in ξ'_Δ when it will be clear from the context.

9.3.3 Theorem *Let **C** be a CCC with enough points, $U^U < U$ via (ψ, ϕ)), and $\underline{A}$ be the associated λ-model as in proposition 9.2.11. Let [ ] and $[\![\ ]\!]$ be the interpretations respectively defined in 9.2.10 and 9.3.1. Then, for any term M with free variables in $\Delta = \{x_1,\ldots,x_n\}$ and any environment ξ: Var$\to \underline{A}(\Lambda^{-1}(\phi))$, one has*

$$[M]_\Delta \circ \xi'_\Delta = [\![M]\!]\xi.$$

Proof The proof is by induction on the structure of M.

- case M = x. Suppose $\xi(x) = a : T \to U$. Then $\xi'_\Delta = < \text{id}_T, a >$. We have:

$[x]_{\{x\}} \circ \xi'_\Delta = \text{snd} \circ < \text{id}_T, a >$

$= a$

$= \xi(x)$

$= [\![x]\!]\xi$

- case $M = PQ$

$[PQ]_\Delta \circ \xi'\Delta = \text{eval} \circ \langle \phi \circ [P]_\Delta, [Q]_\Delta \rangle \circ \xi'\Delta$

$= \text{eval} \circ \langle \phi \circ [P]_\Delta \circ \xi'\Delta, [Q]_\Delta \circ \xi'\Delta \rangle$

$= \text{eval} \circ \langle \phi \circ [\![P]\!]\xi, [\![Q]\!]\xi \rangle$ by induction hypothesis

$= [\![P]\!]\xi\ [\![Q]\!]\xi$ by def. of application

$= [\![PQ]\!]\xi$

- case $M = \lambda x_{n+1}.N$

Note first that, for any $a : T \to U$,

(*) $[\langle x_{n+1} \rangle N CL\varepsilon]\xi \cdot a = [N]_{\Delta \cup \{x_{n+1}\}} \circ \langle \xi'\Delta, a \rangle$.

Indeed:

$[\langle x_{n+1} \rangle N CL\varepsilon]\xi \cdot a = [N CL\varepsilon]\xi(x_{n+1}=a)$ by lemma 14.1.3

$= [\![N]\!]\xi(x_{n+1}=a)$ by definition of $[\![\ ]\!]$

$= [N]_{\Delta \cup \{x_{n+1}\}} \circ \langle \xi'\Delta, a \rangle$ by induction hypothesis.

Note now that:

$[\langle x_{n+1} \rangle N CL\varepsilon]\xi \cdot a = \text{eval} \circ \langle \phi \circ [\langle x_{n+1} \rangle N CL\varepsilon]\xi, a \rangle$

$= \text{eval} \circ ((\phi \circ [\langle x_{n+1} \rangle N CL\varepsilon]\xi) \times \text{id}) \circ \langle \text{id}_T, a \rangle$,

and

$[N]_{\Delta \cup \{x_{n+1}\}} \circ \langle \xi'\Delta, a \rangle = [N]_{\Delta \cup \{x_{n+1}\}} \circ \xi'\Delta \times \text{id} \circ \langle \text{id}_T, a \rangle$.

Since **C** has enough points, and all the points in $T \times U$ are of the kind $\langle \text{id}_T, a \rangle$, from (*) we have:

$[N]_{\Delta \cup \{x_{n+1}\}} \circ \xi'\Delta \times \text{id} = \text{eval} \circ ((\phi \circ [\langle x_{n+1} \rangle N CL\varepsilon]\xi) \times \text{id})$.

Applyng Λ to both the members, and composing with ψ, we get:

$\psi \circ \Lambda([N]_{\Delta \cup \{x_{n+1}\}} \circ \xi'\Delta) = \psi \circ \phi \circ [\langle x_{n+1} \rangle N CL\varepsilon]\xi$.

By definition, $\psi \circ \Lambda([N]_{\Delta \cup \{x_{n+1}\}} \circ \xi'\Delta = [\lambda x.N]_\Delta \circ \xi'\Delta$

Moreover,

$[\![\lambda x.N]\!]\xi = [(\lambda x_{n+1}.N) CL\varepsilon]\xi$

$= [\varepsilon \cdot \langle x_{n+1} \rangle N CL\varepsilon]\xi$

$= \varepsilon \cdot [\langle x_{n+1} \rangle N CL\varepsilon]\xi$

$= \psi \circ \phi \circ [\langle x_{n+1} \rangle N CL\varepsilon]\xi$.

This conludes the proof. ♦

9.4 The Categorical Abstract Machine

The categorical interpretation in definition 9.3.1 suggests a very simple and nevertheless efficient implementation of the lambda calculus. The implementation is based on a call-by-value, leftmost strategy of evaluation, and it is performed by an abstract environment machine called CAM (see refernces). The first step toward the implementation is the compilation of lambda calculus in a language of **categorical combinators.**

Note that $[MN]_\Delta = eval \circ < \phi \circ [M]_\Delta, [N]_\Delta> = \Lambda^{-1}(\phi) \circ < [M]_\Delta, [N]_\Delta>$. $\Lambda^{-1}(\phi)$: $U\times U \rightarrow U$ is just the application u of the underlyng combinatory algebra. We shall write **app** instead $\Lambda^{-1}(\phi)$. Moreover, let **cur**(f) = $\psi \circ \Lambda(f)$, and write f ; g instead of g $\circ$ f. Then the equations which define the semantic interpretation of the lambda calculus are rewritten as follows:

$[x_i]_\Delta$ = fst; . . . ; fst; snd where fst appears n-i times

$[MN]_\Delta = < [M]_\Delta, [N]_\Delta>$; app

$[\lambda x.M]_\Delta =$ cur($[M]_{\Delta\cup\{x\}}$).

This provides a "compilation" of the λ–calculus in a language where all the variables have been replaced with "access paths" to the information they refer to (note the use of the "dummy" enviroment Δ during the compilation).

One of the main characteristic of the categorical semantical approach is that we can essentially use the same language for representing both the code and the environment. An evaluation of the code C in an environment ξ is then the process of reduction of the term ξ ; C. The reduction is defined by a set of rewriting rules. The general idea is that the environment should correspond to a categorical term in some normal form (typically, a weak head normal form). The reductions preserve this property of the environment, executing one instruction (i.e. one categorical combinator) of the code, and updating at the same time the program pointer to the following instruction.

For fst and snd we have the following rules, whose meaning is clear:

$<\alpha,\beta>$; (fst ; C_1) $\Rightarrow$ α ; C_1

$<\alpha,\beta>$; (snd ; C_1) $\Rightarrow$ β ; C_1

In the left hand side of the previous rules, $<\alpha,\beta>$ is the environment and the rest is the code. We shall use parenthesis in such a way that the main semicolon in the expression will distinguish between the environment at its left, and the code at its right.

For cur(C_1) we use the associative law of composition and delay the evaluation to another time:

ξ ; (cur(C_1); C_2) $\Rightarrow$ (ξ ; cur(C_1)) ; C_2

The structure (ξ ; cur(C_1)) corresponds to what is usually called a **closure**.

The right time for evaluating a term of the kind cur(C) is when it is applied to an actual parameter α. We then have:

$<(\xi$; cur(C_1)), $\alpha >$; (app; C_2) $\Rightarrow$ $<\xi, \alpha>$; (C_1; C_2)

The previous rule is just a rewriting of the equation

$\Lambda^{-1}(\phi) \circ <\psi \circ \Lambda(C_1) \circ \xi, \alpha > = eval \circ <\Lambda(C_1) \circ \xi, \alpha > = C_1 \circ <\xi, \alpha>$

that proves the semantical soundness of the previous rule.
Finally, we must consider the evaluation of a term of the kind $\langle C_1,C_2\rangle; C_3$. We have the formal equation:

$$\xi ; (\langle C_1,C_2\rangle; C_3) = \langle \xi ; C_1, \xi ; C_2\rangle ; C_3$$

but we cannot simply use it for defining a reduction, since we want also to reduce $\xi ; C_1$ and $\xi ; C_2$. We must first carry out independently the reductions of $\xi ; C_1$ and $\xi ; C_2$, and then put them together again building the new environment.
A simple solution on a sequential machine may be given by using a stack and working as follows: first save the actual environment ξ by a *push* operation, then evaluate $\xi ; C_1$ (that yields a new environment ξ_1); next *swap* the environment ξ_1 with the head of the stack (i.e. with ξ); now we can evaluate $\xi ; C_2$ obtaining ξ_2; finally build a pair $\langle \xi_1,\xi_2\rangle$ with the head of the stack ξ_1 and the actual environment ξ_2 (that is a *cons* operation). An interesting and elegant property is that, if we just write at compile time $\langle C_1,C_2\rangle$ as "push; C_1; swap; C_2; cons", then the above behaviour is obtained by a sequential execution of this code.

9.4.1 Definition *The **compilation** by means of categorical combinators of a λ-term M in a "dummy" environment $\Delta = (...(nil, x_1),\ldots), x_n)$ is inductively defined as follows:*

$$[x]_{(\Delta,x)} = snd$$
$$[y]_{(\Delta,x)} = fst; [y]_{\Delta}$$
$$[MN]_{\Delta} = push; [M]_{\Delta}; swap; [N]_{\Delta}; cons; app$$
$$[\lambda x.M]_{\Delta} = cur([M]_{(\Delta,x)}) .$$

Examples

1. The closed term $M = \lambda x.xx$ has the following compilation:

$$\begin{aligned}[\lambda x.xx]_{nil} &= cur([xx]_{(nil,x)}) \\ &= cur(push; [x]_{(nil,x)}; swap; [x]_{(nil,x)}; cons; app) \\ &= cur(push; snd; swap; snd; cons; app).\end{aligned}$$

2. The term $(\lambda x.x)(\lambda x.x)$ is so compiled:

$$\begin{aligned}[(\lambda x.x)(\lambda x.x)]_{nil} &= push; [\lambda x.x]_{nil}; swap; [\lambda x.x]_{nil}; cons; app \\ &= push; cur([x]_{(nil,x)}); swap; cur([x]_{(nil,x)}); cons; app \\ &= push; cur(snd); swap; cur(snd); cons; app.\end{aligned}$$

9.4.2 Definition *The **reduction** of the compiled code is summarized by the following table:*

BEFORE			AFTER		
Environment	Code	Stack	Environment	Code	Stack
$<\alpha,\beta>$	fst; C	S	α	C	S
$<\alpha,\beta>$	snd; C	S	β	C	S
ξ	cur(C_1); C_2	S	ξ; cur(C_1)	C_2	S
$<\xi$;cur(C_1),$\alpha>$	app; C_2	S	$<\xi,\alpha>$	C_1;C_2	S
ξ	push; C	S	ξ	C	ξ.S
ξ_1	swap; C	ξ_2.S	ξ_2	C	ξ_1.S
ξ_1	cons; C	ξ_2.S	$<\xi_2, \xi_1>$	C	S.

Example The code "push; cur(snd); swap; cur(snd); cons; app" corresponding to the λ-term $(\lambda x.x)(\lambda x.x)$ gives rise to the following computation:

ENV. = nil
CODE = push; cur(snd); swap; cur(snd); cons; app
STACK = nil

ENV. = nil
CODE = cur(snd); swap; cur(snd); cons; app
STACK = nil . nil

ENV. = nil; cur(snd)
CODE = swap; cur(snd); cons; app
STACK = nil . nil

ENV. = nil
CODE = cur(snd); cons; app
STACK = nil; cur(snd) . nil

ENV. = nil; cur(snd)
CODE = cons; app
STACK = nil; cur(snd) . nil

ENV. = <nil; cur(snd), nil; cur(snd) >
CODE = app
STACK = nil

ENV. = <nil, nil; cur(snd) >
CODE = snd
STACK = nil

ENV. = nil; cur(snd)
CODE =
STACK = nil

Note that "cur(snd)" is the compilation of $\lambda x.x$.

9.5 From Applicative Structures to Categories

We want now to "go backwards", with respect to section 9.2, where we described how to find models of a type-free language within type structures, i.e., within CCC. Namely, we will see how to construct typed models, in particular a CCC, out of type-free structures. This has an important motivation from Programming Language Theory, since it is the semantic counterpart of the following relevant methodology in functional languages.

We already mentioned, as an example, that one of the main features of Edinburgh ML is its automatic treatment of type assignment. That is, the programmer may write programs without taking care of the tedious details of assigning types. The type checker decides whether the given program is typable and, if so, assigns a type to it (actually, the "most general type").

This effective interactive feature of ML provides a partial check for correctness, as one may automatically control whether type errors occur. This is similar to what physicists call "dimensional analysis" for equations when they verify, say, whether a force faces a force, etc. Of course, a lot must be settled. For example, the actual decidability of the type assignment and the existence of "type schemes" such that all types of a given program are instances of these. The identity function, for example, has type $A \to A$ for all instances of A.

As for the semantics, one must first be able to interpret the type-free language, as handled by the programmer, and then interpret types as objects of suitable CCC constructed over the type-free model. In other words, one must be able to obtain an interpretation of types out of a model for the type-free calculus. "Soundness" then means that a program, once it is assigned a type, is actually interpreted as an "element" of the interpretation of its type. Decidability and soundness have been positively clarified by a mathematical investigation of computability and programming, which goes beyond the scope of this book (see references).

Our present purpose is to survey the main "type structures" (categories) one may construct out of type-free models and to complete, in this way, the categorical understanding of typed versus type-free calculi, as required for the semantics of the type assignment process. Most of the work may be done over an arbitrary combinatory algebra (X, .), i.e., over an arbitrary model of Combinatory Logic. Indeed, it is even not required that "·" , the application in X, is a total operation. As already mentioned, if "·" is not always defined, (X, ·) is no longer a model of CL. However, the categories constructed below still have the same properties (products, exponents, whenever possible...), which the reader should check as an exercise.

9.5.1 Definition *Let* $A = (X,\cdot)$ *be an applicative structure.*

i. The set of ***monomials*** *over* A *is inductively defined by*

$x,y....x_1,x_2...$*(variables)...are monomials*

$a,b....a_1,a_2....$*(constants from* X*)... are monomials*

MN *is a monomial if* M *and* N *are monomials.*

Substitution of constants for variables, i.e. $M[\underline{a}/\underline{x}]$*, in monomials is defined by induction in the usual way.* $M_1M_2...M_n$ *stands for* $(...(M_1M_2)...M_n)$.

ii. $f: X^n \to X$ *is* ***algebraic*** *if* $f(\underline{a}) = M[\underline{a}/\underline{x}]$ *for some monomial* M *and any* $\underline{a} = (a_1,... ,a_n) \in X^n$ *and* $\underline{x}$ *of length* n*. (That is, the set* $\mathbf{P}^n(A) = \mathbf{P}[X^n,X]$ *of algebraic functions of n-arguments is defined by the monomials over* X*, with at most* n *variables, modulo extensional equality.)*

iii. Given $(X,\cdot)$*, call* $f : X^n \to X$ ***representable*** *if* $\exists a \in X\ \forall \underline{b} \in X^n\ f(\underline{b}) = a\cdot b_1\cdot...\cdot b_n$.

By using algebraic functions, one may define a simple category over an arbitrary applicative structure.

9.5.2 Definition *Let* $A = (X, \cdot)$ *be an applicative structure. The category* $\mathbf{P}_A$ *of* ***polynomials over*** A*, has as*

objects*:* $X^n \in \mathbf{P}_A$*, for all* $n \in \omega$*;*

morphisms*:* $f \in \mathbf{P}_A[X^n,X^m]$ *iff* $f: X^n \to X^m$ *and* $\forall i<m\ pr^m_i \circ f \in P^n$*, with* pr^m_i *i-th projection.*

(If there is no ambiguity write $\mathbf{P}[X^n,X^m]$ *for* $\mathbf{P}_A[X^n,X^m]$*).*

For example, f(x,y) = (xb(xax),yxa) for a,b∈X, is in $\mathbf{P}[X^2,X^2]$. By substitution, one may easily show that morphisms are closed under composition; moreover, $pr^n_i \in \mathbf{P}[X^n,X] = \mathbf{P}^n$ and, thus, $\mathbf{P}_A$ is a category.

Exercise (Curry-Shoenfinkel) Prove that exactly in combinatory algebras every algebraic function is representable (*hint:* use the argument which translates a λ-term λx.M into an S-K-term <x>M in the introduction).

If A is a combinatory algebra, then, by the exercise, $\mathbf{P}_A$ may be considered the category of representable morphisms.

9.5.3 Lemma *Let $A = (X,\cdot)$ be an applicative structure. Then $\boldsymbol{P}_A$ is a cartesian category with enough points. Moreover, if $\boldsymbol{C}$ is a CC with enough points and $\boldsymbol{C}$, U and $\underline{A}(u)$ are as in definition 9.2.1, then $\boldsymbol{P}_{\underline{A}(u)}$ is a full sub-cartesian category of $\boldsymbol{C}$.*

Proof. Set $X^n \times X^m = X^{n+m}$ and $T = X^0$ (= a singleton set) for the terminal object. The projections pr_i's are given above. Clearly, $\mathbf{P}_A$ has enough points. The rest easily follows from definition 9.2.1 and the assumption that $\mathbf{C}$ has enough points. The reader may complete the proof as an exercise. ♦

Given a category $\mathbf{C}$, U and $\underline{A}(u)$ as in lemma 9.5.3, we say that $g \in \mathbf{C}[U^n,U]$ **induces** $f : \underline{A}(u) \to \underline{A}(u)$ if $f(h) = g \circ h$ for all $h \in \mathbf{C}[T,U]$. It is straightforward to prove that all algebraic functions defined by a monomial in n variables over $\underline{A}(u)$, and no constants, are induced by morphisms in $\mathbf{C}[U^n,U]$. One only has to interpret variables as projections (see section 9.3) and argue by induction on the structure of the "algebraic term" defining the function. For example, for $f(x_1,x_2,x_3) = (x_3 \cdot x_1) \cdot x_2$ write

$u \circ \langle pr_3, pr_1 \rangle : U^3 \to U \times U \to U$, which is $x_3 \cdot x_1$, and then

$u \circ \langle u \circ \langle pr_3, pr_1 \rangle, pr_2 \rangle : U^3 \to U \times U \to U$, which induces f.

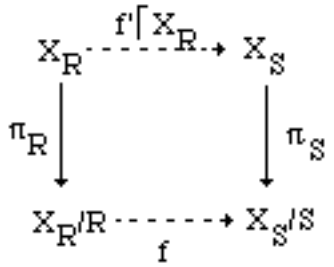
Next, we generalize a definition of category given over a specific applicative structure in example 3.4.1. The definition is slightly different (besides being more general). Since it is an important construction, it is worth seeing it again, under a different and more general viewpoint.

9.5.4 Definition *Let $A = (X,\cdot)$ be an applicative structure. Define then:*

*1. The category $\boldsymbol{PER}_A$ of **partial equivalence relations** given by:*

***objects**: $R \in \boldsymbol{PER}_A$ iff R is an equivalence relation on a subset X_R of X, i.e., $X_R = \text{dom } R = \text{range } R$.*

***morphisms**: for $R \in \boldsymbol{PER}_A$ let $\pi_R(n) = \{m \mid nRm\}$; then $f \in PER[R,S]$ iff $\exists f' \in \boldsymbol{P}[X,X]$ $f \circ \pi_R = \pi_S \circ f'$ on X_R, i.e., the following diagram commutes:*



*(we then say that f' **computes** f).*

2. The category $\boldsymbol{ER}_A$ *of (total)* ***equivalence relations*** *is given as above by using equivalence relations on* X *(i.e.,* $X_R = X$ *in 1.).*

$\mathbf{PER}_\mathrm{A}$ and $\mathbf{ER}_\mathrm{A}$ are clearly categories. Similarly as for $\mathbf{P}_\mathrm{A}$ we write (**P**)**ER**[R,S] for (**P**)$\mathbf{ER}_\mathrm{A}$[R,S] when unambiguous.

Exercise Let A be an applicative structure. Give a terminal object for $\mathbf{PER}_\mathrm{A}$ and $\mathbf{ER}_\mathrm{A}$, and prove that they have enough points. (*Hint*: recall that the constant functions are algebraic).

9.5.5 Proposition *Let* $A = (X,\cdot)$ *be an applicative structure. Then, if* $X\times X < X$ *in* $\boldsymbol{P}_A$, $\boldsymbol{ER}_A$ *and* $\boldsymbol{PER}_A$ *are CCs (with enough points). Moreover,* $\boldsymbol{P}_A$ *and* $\boldsymbol{ER}_A$ *are full sub-CC's of* $\boldsymbol{PER}_A$.

Proof Let $X\times X < X$ via $([-,-],\langle p_1,p_2\rangle)$. Then $R\times S$ may be defined componentwise, by

$$a(R\times S)b \quad \text{iff} \quad (p_1(a))R(p_1(b)) \text{ and } (p_2(a))S(p_2(b)).$$

This turns $\mathbf{ER}_\mathrm{A}$ and $\mathbf{PER}_\mathrm{A}$ into CC's.

Observe now that $\forall n\ X^n < X$ via $([-,\ldots,-],\langle p_1,\ldots,p_n\rangle)$ in $\mathbf{P}_\mathrm{A}$, by iterating $X\times X < X$. Thus $\mathbf{P}_\mathrm{A}$ may be faithfully embedded in $\mathbf{PER}_\mathrm{A}$ by taking, for each X^n, the identity relation restricted to the image of X^n in X via $[-,\ldots,-]$. Call this restricted identity id_n. Moreover, $\mathbf{P}_\mathrm{A}$ is full in $\mathbf{PER}_\mathrm{A}$, since $\mathbf{P}[X^n,X^m] \cong \mathbf{PER}[\mathrm{id}_n, \mathrm{id}_m]$, as sets, by the following isomorphism G (take m = 1, for the sake of simplicity). Let $g\in\mathbf{P}[X^n,X]$, then, for $x = [x_1,\ldots,x_n]$, define $G(g)\in\mathbf{PER}[\mathrm{id}_n, \mathrm{id}]$ by $G(g)(x) = g(p_1(x),\ldots,p_n(x)) = g(x_1,\ldots,x_n)$. G(g) is computed, in the sense of 9.5.4, by $g\circ\langle p_1,\ldots,p_n\rangle : X \to X^n \to X$.

G is an isomorphism, whose reverse map is given as follows: if $h\in\mathbf{PER}[\mathrm{id}_n, \mathrm{id}]$ is computed by $h'\in\mathbf{P}[X,X]$, then $G^{-1}(h) = h'\circ[-,\ldots,-] : X^n \to X \to X$. By definition, $\mathbf{ER}_\mathrm{A}$ is a full sub-CC of $\mathbf{PER}_\mathrm{A}$, and both categories have enough points, by the exercise. ♦

The next theorem proves the converse of theorem 9.2.5 and, moreover, it shows that, by applying the construction in definition 9.2.1 and theorem 9.2.5 to a combinatory algebra, one gets back to the given combinatory algebra.

9.5.6 Theorem *Let* $A = (X,\cdot)$ *be a combinatory algebra and* $\boldsymbol{P}_A$ *be the category of polynomials over A. Then* $T < X$, $XxX < X$ *in* $\boldsymbol{P}_A$ *and, for* $u(x,y) = x\cdot y$, $u\in\boldsymbol{P}[X^2,X]$ *is K-universal in the category* $\boldsymbol{P}_A$. *Moreover,* $\underline{A}(u) = A$.

Proof $T < X$ trivially holds, for $X\neq\emptyset$. Clearly, XxX exists in $\mathbf{P}_\mathrm{A}$, by lemma 9.5.3. Let then $c,c_1,c_2\in X$ represent $\underline{\lambda}xyz.zxy$, $\underline{\lambda}xy.x$, $\underline{\lambda}xy.y$, respectively, in the sense of definition 9.5.1(iii). c is the element that codes pairs (they are commonly coded in this way in λ-calculus), while c_1, c_2 will be used to define projections. Thus, for $[x,y] = cxy$ and $p_i(x) = xc_i$, one has $[-,-]\in\mathbf{P}[X^2,X]$, $p_i\in\mathbf{P}[X,X]$ and $X\times X < X$ via $([-,-],\langle p_1,p_2\rangle)$. Finally, assume that $f\in\mathbf{P}[X^2,X]$ and that $a\in X$

represents f. Then $f = u\circ((\underline{\lambda}x.ax)\times id)$ and, hence, u is K-universal. It is easy to check from the definition that $\underline{A}(u) = A$. ♦

9.5.7 Corollary *Let $A = (X,\cdot)$ be an applicative structure. Then A is a combinatory algebra iff, in $\boldsymbol{P}_A$, one has $T < X$, $X\times X < X$ and, for $u(x,y) = x\cdot y$, u is K-universal.*
Proof ($\Rightarrow$) by theorem 9.5.6; ($\Leftarrow$) by theorem 9.2.6. ♦

As already pointed out, one needs CCC's in order to take care of λ−models. However, also combinatory algebras are tidely characterized within CCC's. The following immediate consequence of theorem 9.5.6 and proposition 9.2.6, plus proposition 9.2.7, fully characterizes the least requirement for functional completeness in CCC's.

9.5.8 Corollary *Let $A = (X,\cdot)$ be a combinatory algebra. Then $\boldsymbol{PER}_A$ is a CCC, where $T < X$, $X\times X < X$ and, for $u(x,y) = x\cdot y$, $\Lambda(u)\in\boldsymbol{PER}[X,X^X]$ is principal. Moreover $\underline{A}(u) = A$.*
Proof. In view of theorem 9.5.6 and proposition 9.5.5, we only need to define the object S^R in $\mathbf{PER}_A$, which represents **PER**[R,S]. Set then

$$aS^Rb \quad \text{iff} \quad \forall x,y\in X\ (xRy \Rightarrow (ax)S(by)).$$

Recall now that, by assumption, each function in **P**[X,X] is representable. The rest of the proof that $\mathbf{PER}_A$ is a CCC is an obvious generalization of example 3.4.1. ♦

9.5.9 Remark While completing the proof that $\mathbf{PER}_A$ is a CCC, in corollary 9.5.8, one may notice that it is only required, for all R, S and all $f\in$**PER**[R,S], that f has a representative in X. This is the point which allows the generalization to the partial case (see section 9.6, besides $\mathbf{PER}_\omega$ in example 3.4.1).

The next result proves the converse of corollary 9.1.12 and completes our categorical understanding of λ-models.

9.5.10 Theorem *Let $A = (X, \cdot, \varepsilon)$ be a λ-model. Then, in the CCC $\boldsymbol{PER}_A$, there exist $\psi\in\boldsymbol{PER}[X^X,X]$ and $\phi\in\boldsymbol{PER}[X, X^X]$ such that $X^X < X$ via (ψ, ϕ). Moreover, $(X,\cdot) \cong \underline{A}(\Lambda^{-1}(\phi))$, and $\varepsilon = \psi(\psi\circ\phi)$. If A is extensional, then $X^X \cong X$.*
Proof Let $f\in$**PER**[X,X] and $a\in X$ be a representative for f. Define then $\psi(f) = \varepsilon a$. By (ε_2), ψ is well defined. Recall also that X^X is the exponent representing **PER**[X,X] in $\mathbf{PER}_A$, hence $\psi\in$**PER**$[X^X,X]$.

As for φ, define $\phi(a) = \underline{\lambda}x.ax$ for any $a\in X$. Clearly $\phi\in$**PER**$[X,X^X]$. Compute then

$$\begin{aligned}\phi(\psi(f)) &= \underline{\lambda}x.\varepsilon ax \qquad \text{if a represents f}\\ &= f \qquad\qquad \text{by } (\varepsilon_1).\end{aligned}$$

Thus $X^X < X$, via (ψ, ϕ).

Finally, $(X,\cdot) \cong \underline{A}(\Lambda^{-1}(\phi))$, since $\phi(a)(b) = ab$. Moreover $\psi(\phi(a)) = \varepsilon a$ and, hence, ε represents $\psi\circ\phi$. Thus

$$\begin{aligned} \psi(\psi\circ\phi) &= \varepsilon\varepsilon && \text{by definition of } \psi \\ &= \varepsilon && \text{by } (\varepsilon 3). \end{aligned}$$

Finally, if $a = \varepsilon a = \psi(\phi(a))$ for all a, then $\psi\circ\phi = id$ and, hence, $X^X \cong X$. ♦

We conclude this part by summarizing the connections between type-free λ-calculus and categories with enough points obtained so far. This provides a unified framework for the topic.

9.5.11 Theorem *Let* $\mathbf{C}$ *be a CCC and* A *an object of* $\mathbf{C}$. *Then*

1. $A^A \cong A \Rightarrow$ *A is an extensional λ-model;*
2. $A^A < A \Rightarrow$ *A is a λ-model;*
3. $\exists p \in \mathbf{C}[A, A^A]$ *principal,* $T < A$ *and* $A\times A < A \Rightarrow$ *A is a combinatory algebra.*

Conversely,

1. *A is an extensional λ-model* $\Rightarrow A^A \cong A$ *in* $\mathbf{PER}_A$;
2. *A is a λ-model* $\Rightarrow A^A < A$ *in* $\mathbf{PER}_A$;
3. *A is a combinatory algebra* $\Rightarrow \exists p \in \mathbf{PER}_A[A, A^A]$ *principal,* $T < A$ *and* $A\times A < A$ *in* $\mathbf{PER}_A$.

9.5.12 Remark In the categorical semantics of lambda calculus, we have to deal with Cartesian (closed) categories, and thus with products and projections. Without much increasing the complexity of the semantics, it is thus possible to consider also a type-free λ-calculus with explicit pairing, λβηπ, in analogy to the typed case, see section 8.2. We leave to the reader the task of defining, as an exercise, this calculus and giving its semantics on a reflexive object U in a CCC $\mathbf{C}$. As a matter of fact, one only needs to add $A\times A \cong A$ to (1) and (2) in theorem 9.5.11 in order to obtain characterizations of the models of λβ(η)π.

In conclusion, the diligent reader will notice that the models of λβηπ are exactly the CCC with e.p. and with a unique object nonisomorphic to the terminal one. It may be fair, then, to call λβηπ the "unityped" λ-calculus.

9.6 Typed and Applicative Structures: Applications and Examples

In the first part of this section, we sketch a recent application of the typed and type-free λ–calculus to category theory. In a sense, this application goes in the other direction with respect to our prevailing perspective, as, so far, we mostly applied categorical tools to the understanding of deductive systems and their calculus of proofs (λ–calculus).

The question we answer here may be simply stated, in categorical terms:

(1) which isomorphisms hold in all CCC's ?

The motivation is clear, as a simple and decidable equational theory of types will allow us to detect provably isomorphic types or, equivalently, valid isomorphisms in all models. In functional programming, for example, when retrieving programs from a library where they are collected according to their types, the search should be done "up to provable isomorphisms", as the same program may have been coded under isomorphic types; for example, a search program for lists of a given length, may be typed by $\mathrm{INT}\times\mathrm{LISTS} \to \mathrm{LISTS}$ or, equivalently, by $\mathrm{LISTS} \to (\mathrm{INT} \to \mathrm{LISTS})$ (see references). The result turns out to be an application of λ–calculus to categories, as we look at the problem from a proof-theoretic view point and, by the work done in chapter 8, we actually answer to the following equivalent question.

Consider the intuitionistic calculus of sequents, in section 8.3, and suppose that proofs of $A \vdash B$ and of $A \vdash B$ are given. Then one may ask

(2) in which cases the composition of $A \vdash B$ and $B \vdash A$ (and of $B \vdash A$ and $A \vdash B$) reduce, by cut-elimination, to the axiom $A \vdash A$ (and $B \vdash B$, respectively) ?

Clearly, (2) corresponds to (1) when types are understood as objects. The point is that the deductions in (2) are coded by λ-terms, M and N, say, as described in section 8.3. By this, and by a lot of (hinted) hacking on λ-terms, we will characterize valid isomorphisms by looking at the structure of M and N such that $M\circ N = I_A$ and $N\circ M = I_B$, by β–conversion.

The second part develops little general Category Theory since it essentially gives more examples of CCC's and of λ–models. The structures presented will be combinatory algebras and models of type-free λβ, since models of λβη may be derived from the general results in chapter 10. In particular, we introduce a few relevant categories of complete partial orders as well as their "effective" versions and hint how they relate to the various categories of quotients or **PER**'s that we largely used in the previous sections.

Part 1: Provable isomorphisms of types

Consider the following equational theory of types. It is given by axiom schemata plus the obvious inference rules that turn "=" into a congruence relation.

9.6.1 Definition Th *is axiomatized as follows, where* T *is a constant symbol:*

1. $A \times T = A$

2. $A \times B = B \times A$

3. $A \times (B \times C) = (A \times B) \times C$

4. $(A \times B) \to C = A \to (B \to C)$

5. $A \to (B \times C) = (A \to B) \times (A \to C)$
6. $A \to T = T$
7. $T \to A = A$.

In remark 3.3.3, we already asked the reader to prove that, when $\to$ and $\times$ are interpreted as cartesian product and exponent, the provable equations of Th hold as isomorphisms in any CCC. Note, though, that there are categorical models which realize Th, but are not CCC's. Take, say, a cartesian category and a bifunctor "$\to$" that is constant in the second argument.

We next hint how to prove the non trivial fact that Th characterizes exactly the valid isomorphisms in all CCC's, by using λ-calculus.

As pointed out in section 8, the typed λ-calculus is, at the same time:
a - the "theory" of CCC's;
b - the calculus of proofs of the intuitionistic calculus of sequents.

Thus the theorem is shown by observing that the isomorphic types in the (closed) term model of typed λ-calculus are provably equal in Th. This answers to (2) in the introduction above and, thus, to (1).

A term model is *closed*, when terms in it contain no free variables.

9.6.2 Definition *Given (an extension of) the typed λ-calculus, $\boldsymbol{\lambda}$, the (closed) term model is the type structure*

$|\boldsymbol{\lambda}| = \{|M: A| \ / \ M$ *is a (closed) term of type* $A\}$
where $|M: A| = \{ N \ / \ \boldsymbol{\lambda} \vdash M = N \}$.

Clearly, the type structure is non trivial, if a collection of ground or atomic types is given. The (pure) λ-calculus may be extended by adding fresh types and constants as well as consistent sets of equations. Consider now the extension of $\lambda\beta\eta\pi^t$ in section 8.3 by adding:
1 - a special atomic type T (the terminal object);
2 - an axiom schema

$$*A: A \to T$$

which gives a constant of that type;
3 - a rule

$$\frac{M: A \to T}{M = *A}$$

that gives the unicity of *A.

Call $\lambda\beta\eta\pi^{*t}$ this extended calculus and Tp^* its collection of types. The point is that the closed term model of $\lambda\beta\eta\pi^{*t}$ (and its extensions) forms a CCC, as the reader may check as an exercise. Then the provable equations of Th are realized in $|\lambda\beta\eta\pi^{*t}|$, as isomorphisms. We give an explicit name to these isomorphisms, as λ-terms provide the basic working tools.

9.6.3 Definition *Let $A,B\in Tp^*$. Then A and B are **provably isomorphic** ($A \cong_p B$) iff there exist closed λ-terms $M : A \to B$ and $N : B \to A$ such that $\lambda\beta\eta\pi^{*t} \vdash M \circ N = I_B$ and $\lambda\beta\eta\pi^{*t} \vdash N \circ M = I_A$, where I_A and I_B are the identities of type A and B. We then say that M and N are **invertible** terms in $\lambda\beta\eta\pi^{*t}$.*

9.6.4 Remark By general categorical facts, we then have the easy implication of the equivalence we want to show; namely, $Th \vdash A = B \Rightarrow A \cong_p B$. It may be worth for the reader to work out the details and construct the λ-terms which actually give the isomorphisms. Indeed, they include the "abstract" verification of cartesian closure; for example, "currying" is realized by $\lambda z.\lambda x.\lambda y.z\langle x,y\rangle$ with inverse $\lambda z.\lambda x.z(p1\ x)(p2\ x)$, that prove $(A \times B) \to C \cong_p A \to (B \to C)$; the term $\lambda z.\langle\lambda x.(p1\ (zx)),\lambda x.(p2\ (zx))\rangle$ with inverse $\lambda z.\lambda x.\langle(p1\ z)x,(p2\ z)\ x\rangle$ prove $A \to (B \times C) \cong_p (A \to B) \times (A \to C)$. The others are easily derived.

The proof of the other implication, i.e., $A \cong_p B \Rightarrow Th \vdash A = B$, roughly goes as follows. As a first step, types are reduced to "type normal forms", in a "type rewrite" system. This will eliminate terminal types and bring products at the outermost level. Then one needs to show that isomorphisms between type normal forms yield componentwise isomorphisms. This takes us to the pure typed λ-calculus (i.e., no products nor T's). Then a characterization of the invertible terms of the pure type-free calculus is easily applied in the typed case, as the invertible type-free terms happen to be typable. The syntactique structure of the invertible terms gives the result.

The axioms of Th suggest the following rewrite system R for types (essentially Th "from left to right ", with no commutativity):

9.6.5 Definition (Type rewriting R) *Let ">" be the transitive and substitutive **type-reduction** relation given by:*

1. $A \times T > A$
1'. $T \times A > A$
3. $A \times (B \times C) > (A \times B) \times C$
4. $(A \times B) \to C > A \to (B \to C)$
5. $A \to (B \times C) > (A \to B) \times (A \to C)$
6. $A \to T > T$
7. $T \to A > A$.

The system R yields an obvious notion of **normal form for types** (type normal form), i.e., when no type reduction can be applied. Note that 1, 1', 6 and 7 "eliminate the T's", while 4 and 5 "bring outside $\times$". It is then easy to observe that each type normal form is identical to T or has the structure $S_1\times ... \times S_n$ where each S_i does not contain T nor "$\times$". We write **nf(S)** for the normal form of S (there is exactly one, see 1.6) and say that a normal form is non trivial if it is not T.

9.6.6 Proposition R *is Church-Rosser and each type has a unique type normal form in* R.
Proof Easy exercise. ♦

By the implication discussed in remark 9.6.4, since $\mathsf{R} \vdash S > R$ implies $\mathsf{Th} \vdash S = R$, it is clear that any reduction $\mathsf{R} \vdash S > R$ is wittnessed by an invertible term of type $S\to R$.

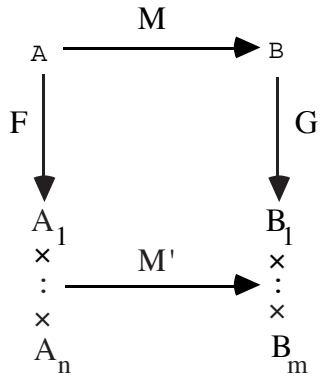
9.6.7 Corollary *Given types S and R, one has:*
1 - $\mathsf{Th} \vdash S = nf(S)$ *and, thus,*
2 - $\mathsf{Th} \vdash S = R \Leftrightarrow \mathsf{Th} \vdash nf(S) = nf(R)$.

In conclusion, when $\mathsf{Th} \vdash S = R$, either we have $nf(S) \equiv T \equiv nf(R)$, or $\mathsf{Th} \vdash nf(S) \equiv (S_1\times ... \times S_n) = (R_1\times ... \times R_m) \equiv nf(R)$. A crucial lemma below shows that, in this case, one also has $n = m$.

The assertion in the corollary can be reformulated for invertible terms in a very convenient way:

9.6.8 Proposition (commuting diagram) *Given types A and B, assume that $F: A\to nf(A)$ and $G: B\to nf(B)$ prove the reductions to type n.f.. Then a term $M: A\to B$ is invertible iff there exist an invertible term $M': nf(A)\to nf(B)$, such that $M = G^{-1}\circ M'\circ F$.*
Proof. $\Leftarrow$) Set $M^{-1} \equiv (G^{-1}\circ M'\circ F)^{-1} \equiv F^{-1}\circ M'^{-1}\circ G$, then M is invertible.
$\Rightarrow$) Just set $M' = G\circ M\circ F^{-1}$. Then $M'^{-1} \equiv F\circ M^{-1}\circ G^{-1}$ and M' is invertible. ♦

A diagram easily represents the situation in the proposition:



We now state a few lemmas that should guide the reader through the basic ideas of this application of λ-calculus to category theory. Most technical proofs, indeed λ-calculus proofs, are omitted (and the reader should consult the references).

Recall first that, when $\mathsf{Th} \vdash S = R$, one has

$$\mathrm{nf}(S) \equiv T \equiv \mathrm{nf}(R), \text{ or } \mathsf{Th} \vdash \mathrm{nf}(S) \equiv (S_1 \times \ldots \times S_n) = (R_1 \times \ldots \times R_m) \equiv \mathrm{nf}(R).$$

Notice that, in the latter case, there cannot be any occurrence of T in either type. Indeed, a non trivial type normal form cannot be provably equated to T, as it can be easily pointed out by taking a non trivial model. Thus it may suffice to look at equations such as $(S_1 \times \ldots \times S_n) = (R_1 \times \ldots \times R_m)$ with no occurrences of T and, hence, to invertible terms with no occurrences of the type constant T in their types. We can show that these terms do not contain any occurrence of $*A$ either, for no type A, via the following lemma.

9.6.9 Lemma *Let M be a term of $\lambda\beta\eta\pi^{*t}$ in n.f..*

1 - (Terms of a product type) *If $M: A \times B$, then either $M \equiv \langle M1, M2 \rangle$, or there is $x:C$ such that $x \in FV(M)$ and $A \times B$ is a type subexpression of C.*

2 - (Every term, whose type contains no T, has no occurrence of *A constants) *Assume that in M there is an occurrence of $*A$, for some type A. Then there is some occurrence of the type constant T in the type of M or in the type of some free variable of M.*

Proof. By induction on the structure of M. ♦

Note now that (the equational theory of) $\lambda\beta\eta\pi^{*t}$ is a conservative extension of (the equational theory of) $\lambda\beta\eta\pi^{t}$. Similarly for $\lambda\beta\eta\pi^{t}$ w.r.t. $\lambda\beta\eta^{t}$. Thus, invertibility in the extended theory, given by terms of a purer one, holds in the latter.

9.6.10 Proposition (Isomorphisms between type normal forms are given by terms in $\lambda\beta\eta\pi^{t}$) *Assume that S and R are non trivial type normal forms. If the closed terms M and N prove $S \cong_p R$ in $\lambda\beta\eta\pi^{*t}$, then their normal forms contain no occurrences of the constants $*A$. (Thus, M and N are actually in $\lambda\beta\eta\pi^{t}$).*

Proof By the previous lemma, as the terms are closed and no T occurs in their type. ♦

So we have factored out the class of constants *A, and we restricted the attention to $\lambda\beta\eta\pi^t$. By the next step, we reduce the problem to the pure calculus, i.e., we eliminate pairing as well, in a sense.

9.6.11 Proposition (Isomorphic type normal forms have equal lenght) *Let* $S \equiv S_1 \times ... \times S_m$ *and* $R \equiv R_1 \times ... \times R_n$ *be type normal forms. Then* $S \cong_p R$ *iff*
$n = m$ *and there exist* $M_1,...,M_n ; N_1,...,N_n$ *such that*

$x_1: S_1, ..x_n: S_n \vdash \langle M_1,...,M_n\rangle: (R_1 \times ... \times R_n)$

$y_1: R_1, ..y_n : R_n \vdash \langle N_1,...,N_n\rangle: (S_1 \times ... \times S_n)$

with $M_i[\underline{x} := \underline{N}] =_{\beta\eta} y_i$, *for* $1 \le i \le n$

$N_j[\underline{y} := \underline{M}] =_{\beta\eta} x_j$, *for* $1 \le j \le n$

and there exist permutations σ, π *over* n *such that*

$M_i = \lambda \underline{u}_i.x_{\sigma i}\, \underline{P}_i$ *and* $N_j = \lambda \underline{v}_j.y_{\pi j}\, \underline{Q}_j$

($\underline{M}$ *is a vector of terms; substitution of vectors of equal lenght is meant componentwise).*

Proof (Not obvious, see references). ♦

By induction, one may easily observe that terms of $\lambda\beta\eta\pi^t$ whose type is arrow-only belong to $\lambda\beta\eta^t$. Thus, one may look componentwise at terms that prove an isomorphism. The next point is to show that each component, indeed a term of $\lambda\beta\eta^t$, yields an isomorphism. This will be done by using a characterization of invertible terms in the pure calculus. The same result will be applied once more in order to obtain the result we aim at.

The characterization below has been given in the type-free calculus, as an answer to an old question of Church on the group of type-free terms. We follow the type-free notation, also for notational convenience.

9.6.12 Definition *Let* M *be a type-free term. Then* M *is a* ***finite hereditary permutation*** *(f.h.p.) iff either*

(i) $\lambda\beta\eta \vdash_u M = \lambda x.x$, *or*

(ii) $\lambda\beta\eta \vdash_u M = \lambda z.\ \lambda \underline{x}.z\underline{N}_\sigma$, *where if* $|\underline{x}| = n$ *then* σ *is a permutation over* n *and* $z\underline{N}_\sigma = zN_{\sigma 1}N_{\sigma 2}\ldots N_{\sigma n}$, *such that, for* $1 \le i \le n$, $\lambda x_i.N_i$ *is a finite hereditary permutation.*

For example, $\lambda z.\lambda x_1.\lambda x_2.zx_2x_1$ and $\lambda z.\lambda x_1.\lambda x_2.zx_2(\lambda x_3.\lambda x_4.x_1x_4x_3)$ are f.h.p.'s. The structure of f.h.p.'s is tidily desplayed by Böhm-trees. The **Böhm-tree** of a term M is (informally) given by:

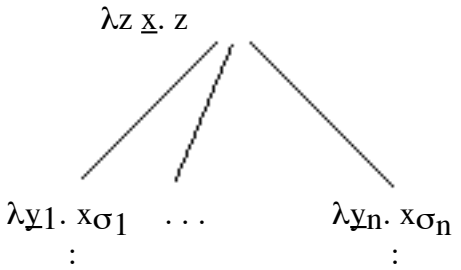
$BT(M) = \Omega$ if M has no head normal form

$BT(M) = \lambda x_1 ... x_n.\ y$ if $M =_\beta \lambda x_1...x_n.\ y\ M_1 ... M_p$

$$\begin{array}{c} / \ \ldots \ \backslash \\ BT(M_1) \quad BT(M_p) \end{array}$$

(see references).

It is easy to observe that a $BT(M)$ is finite and Ω-free iff M has a normal form. Then one may look at f.h.p.'s as Böhm-trees, as follows:



and so on, up to a finite depth (note that $\underline{y}_i$ may be an empty string of variables). Clearly, the f.h.p.'s are closed terms and possess a normal form. In particular, exactly the abstracted variables at level $n+1$ appear at level $n+2$, modulo some permutation of the order (note the special case of z at level 0). The importance of f.h.p.'s arises from the following classic theorem of λ-calculus. (Clearly, the notion of invertible term given in 9.6.3 easily translates to type-free λ-calculi).

9.6.13 Theorem *Let M be an untyped term possessing normal form. Then M is $\lambda\beta\eta$-invertible iff M is a f.h.p..*

Recall now that all typed terms possess a (unique) normal form (see references). Let then M be a typed λ-term and write $e(M)$ for the **erasure** of M, i.e. for M with all type labels erased.

Remark Observe that the erasures of all axioms and rules of the typed lambda calculus are themselves axioms and rules of the type-free lambda calculus. Then, if M and N are terms of $\lambda\beta\eta^t$ and $\lambda\beta\eta^t \vdash M = N$, one has $\lambda\beta\eta \vdash e(M) = e(N)$. Thus, in particular, if $M : \sigma \to \tau$ and $N : \tau \to \sigma$ are invertible terms in $\lambda\beta\eta^t$, $e(M)$ and $e(N)$ are f.h.p.'s.

Exercise Show that the f.h.p.'s are typable terms (*Hint*: Just follow the inductive definition and give z, for instance, type $A1 \to (A2..... \to B)$, where the Ai's are the types of the N_{σ_i}.) Then, by the a small abuse of language, we may talk also of typed f.h.p.'s. Observe that these are exactly the typed invertible terms in definition 9.6.3.

The first application of 9.6.13 we need is the following.

9.6.14 Proposition *Let* $M_1, \ldots M_n$, $N_1, \ldots, N_n$ *and permutation* σ *be as in lemma 9.6.11. Then, for all* i, $\lambda x_{\sigma i}.M_i : S_{\sigma i} \to R_i$ *and* $\lambda y_i.N_{\sigma i} : R_i \to S_{\sigma i}$ *are invertible terms.*

Proof. For a suitable typing of the variables it is possible to build the following terms of $\lambda\beta\eta^t$ (we erase types for convenience):

$M = \lambda z x_1 \ldots x_n . z M_1 \ldots M_n$

$N = \lambda z y_1 \ldots y_n . z\ N_1 \ldots N_n$.

It is an easy computation to check, by the definition of the M_i's and of the N_i's, that M and N are invertible. Moreover, they are (by construction) in normal form, thus, by theorem 9.6.13, M and N are f.h.p.'s. This is enough to show that every M_i has only one occurrence of the x_i's (namely $x_{\sigma i}$); similarly for the N_i's.

Thus we obtain

$$M_i[\underline{x} := \underline{N}] \equiv M_i[\, x_{\sigma(i)} := N_{\sigma(i)}] =_{\beta\eta} y_i, \text{ for } 1 \le i \le n$$

$$N_i[\underline{y} := \underline{M}] \equiv N_i[\, y_{\pi(i)} := M_{\pi(i)}] =_{\beta\eta} x_i, \text{ for } 1 \le i \le n$$

and, hence, for each i, $\lambda x_{\sigma(i)}.M_i : S_{\sigma(i)} \to R_i$ and $\lambda y_i.N_{\sigma(i)} : R_i \to S_{\sigma(i)}$ are invertible. ♦

As a result of the work done so far, we can then focus on invertible terms whose types contain only "$\to$" i.e., investigate componentwise the isomorphisms of type normal forms. Of course, these isomorphisms will be given just by a fragment of theory Th.

Call S the subtheory of Th given by just one proper axiom (plus the usual axioms and rules for "="), namely

$$\text{(swap)} \qquad A \to (B \to C) = (B \to (A \to C)) .$$

S is a subtheory of Th by axioms 2 and 4 of Th.

9.6.15 Proposition *Let* A, B *be type expressions with no occurences of* T *nor* $\times$. *Then*

$$A \cong_p B \ \Rightarrow\ \mathsf{S} \vdash A = B.$$

Proof Suppose $A \cong_p B$ via M and N. As usual, we may assume without loss of generality that M and N are in normal form. By lemma 9.6.9 and the remark after 9.6.11, M and N actually live in $\lambda\beta\eta^t$ and, by theorem 9.6.13, they are f.h.p.'s. We prove $\mathsf{S} \vdash A = B$ by induction on the depth of the Böhm-tree of M.

Depth 1: $M \equiv \lambda z : C.\ z$. Thus $M : C \to C$, and $\mathsf{S} \vdash C = C$ by reflexivity.

Depth n+1: $M \equiv \lambda z : E.\ \lambda \underline{x} : \underline{D}.\ z\underline{N_\sigma}$. Recall $z\underline{N_\sigma} = z N_{\sigma 1} \ldots N_{\sigma n}$ where if the ith abstraction in $\lambda \underline{x} : \underline{D}$ is $\lambda x_i : D_i$ then the erasure of $\lambda x_i : D_i.N_{\sigma_i}$ is a f.h.p.. Thus $\lambda x_i : D_i.N_{\sigma_i}$ gives (half of) a provable isomorphism from D_i to some F_i. Hence the type of N_{σ_i} is F_i. In order to type check, we must have $E = (F_{\sigma 1} \to \ldots \to F_{\sigma n} \to B)$ for some B. Thus the type of M is $(F_{\sigma 1} \to \ldots \to F_{\sigma n} \to B) \to (D_1 \to \ldots \to D_n \to B)$. By induction, since the height of the Böhm tree of (the erasure of) each $\lambda x_i : D_i.\ N_{\sigma_i}$ is less than the height of the Böhm tree of M, ona has $\mathsf{S} \vdash D_i = F_i$ for $1 \le i \le n$. By a repeated use of the rules for "=", we get

$S \vdash (F_{\sigma 1} \to ... \to F_{\sigma n} \to B) = (D_{\sigma 1} \to ... \to D_{\sigma n} \to B)$.

Hence it suffices to show

$S \vdash (D_{\sigma 1} \to ... \to D_{\sigma n} \to B) = (D_1 \to ... \to D_n \to B)$.

This is quite simple to show by a repeated use of axiom (swap) above in conjunction with the rules. ♦

Clearly, also the converse of proposition 9.6.15 holds, since the "⇐" part in 9.6.15 is provable by a fragment of the proof hinted in 9.6.4. Thus one has:

$$S \vdash A = B \iff A \cong_p B \text{ by terms in } \lambda\beta\eta^t.$$

The result we aim at, is just the extension of this fact to Th and $\lambda\beta\eta\pi*^t$.

9.6.16 Main Theorem $S \cong_p R \iff Th \vdash S = R$

Proof. In view of 9.6.4, we only need to prove $S \cong_p R \Rightarrow Th \vdash S = R$. As we know, this is equivalent to proving $nf(S) \cong nf(R) \Rightarrow Th \vdash nf(S) = nf(R)$.

Now, by proposition 9.6.11, for $nf(S) \equiv (S_1 \times ... \times S_n)$ and $(R_1 \times ... \times R_m) \equiv nf(R)$, we have

$nf(S) \cong nf(R) \Rightarrow$ $n = m$ and there exist $M_1, ..., M_n, N_1, ..., N_n$

and a permutation σ such that $\lambda x_{\sigma i}.M_i : S_{\sigma i} \to R_i$ and $\lambda y_i.N_{\sigma i} : R_i \to S_{\sigma i}$.

By 9.6.14, these terms are invertible, for each i. Thus, by 9.6.15, $S \vdash R_i = S_{\sigma i}$ and, hence, by the rules, $Th \vdash S = R$. ♦

This concludes the proof of the main theorem of this part.

9.6.17 Corollary *Given types A and B, it is decidable whether they are (their interpretation yields) isomorphic (objects) in all CCC's.*

Proof *(Hint)* Reduce A and B to type normal form. Check that these have an equal number of factors. If so, observe that theory S does not change the lenght of types and perfom the required swaps to check the equality, in that theory, of each component. ♦

Exercise Check the complexity of the theory of provable isomorphisms.

Part 2: Higher type objects as models of the type-free λ-calculus

We give here some examples of categories and objects with the properties mentioned in the previous sections and discuss connections to Higher Type Recursion Theory, a highly developed topic to which denotational semantics of programming languages is greatly indebted. This theory suggested the early structures for a "generalized theory of computation," stressed the role of CCC's and, jointly

with category theory, set the basis for the construction of the early models of (type-free) λ-calculus and, thus, of functional programming languages.

We first mention a simple way to obtain lots of type-free models in CCC's with reflexive objects. Then we apply this construction to the main type structures for higher type recursion: the partial continuous and computable functionals in all finite higher types. Well-established results allow to recover from those structures the various hierarchies of total functionals, which actually started the topic (see references).

In section 2.4 we already gave two examples of reflexive objects in different CCC and, thus, of λ-models. When presenting the first, Pω, in 2.4.1, we promised to show the reflexivity of another very familiar Scott domain: the collection $P(R)$ of the partial (recursive) functions from ω to ω. Its reflexivity, see theorem 9.5.2, will be a consequence of a stronger property, with respect to a suitable category.

Exercise Let **C** be a CCC and T be a terminal object. Then $\forall X,Y \in Ob_{\mathbf{C}}$, if $T < Y$, one has $X < X^Y$.
(*Solution*: The retraction (i,j) is given by $i = \Lambda(pr_1) \in \mathbf{C}[X,X^Y]$ and $j = eval\circ(id\times t) \in \mathbf{C}[X^Y,X]$ for some fixed $t \in \mathbf{C}[T, X]$. Indeed, $j\circ i = j\circ(i\times d_T) = eval\circ(id\times t)\circ(i\times d_T) = eval\circ(i\times id)\circ(id\times t) = eval\circ(\Lambda(pr_1)\times id)\circ(id\times t) = pr_1\circ id\times t = id$.)

Let the set of type symbols, Tp, contain at least the atomic type 1. Then, for X in a CCC **C**, set

$$X^1 = X, \text{ and, for } A = X^\sigma \text{ and } B = X^\tau, \text{ set } X^{\sigma\to\tau} = B^A \text{ and } X^{\sigma\times\tau} = A\times B.$$

9.6.18 Lemma *Let U be a reflexive object in a CCC **C**. Then, for $\{U^\sigma\}_{\sigma\in Tp}$ as above*

$$\forall \sigma, \tau \in Tp \quad U^\sigma < U^\tau \text{ in } \mathbf{C}$$

In particular, then, $\forall \sigma \in Tp \; U^{\sigma\to\sigma} < U^\sigma$.

Proof Assume, by induction on the syntactic structure of types, that $U < U^\sigma$ and $U < U^\tau$. Clearly, $U\times U < U^{\sigma\times\tau}$. It is also easy to check that $U^U < U^{\sigma\to\tau}$. Similarly, from the inductive assumptions $U^\sigma < U$ and $U^\tau < U$, one has $U^{\sigma\to\tau} < U^U < U$, as U is a reflexive object, and $U^{\sigma\times\tau} < U\times U < U$, by proposition 2.3.6. Finally, $U < U\times U$ and $U < U^U$, by $T < U$ and the exercise. Therefore, $\forall \sigma,\tau \in Tp \; U^\sigma < U < U^\tau$ in **C**. ♦

As already shown, Scott domains, coherent domains and other categories of continuous functions have nontrivial reflexive objects. By the lemma, in these categories there are lots of λ–models, one in each higher type σ, over the reflexive object. We consider here yet another simple category with a reflexive object, namely the category **pcD** of ω-algebraic pair-consistent c.p.o.'s, and the object P of partial functions from integer to integer.

Call first a subset D of a p.o.set $(X,\leq)$ **pairwise consistent** if any pair of elements in D has an upper bound in X (and write $x \Uparrow y$ for $\exists z \in X\ x,y \leq z$). $(X,\leq)$ is **pair-consistent** if any pairwise consistent subset has a l.u.b. Call **pcD** the category of ω-algebraic pair-consistent c.p.o.'s (cf. examples in 2.4.1), with continuous maps as morphisms.

Exercise Prove that **pcD** is a full subCCC of the category **D** of Scott domains in 2.4.1.

9.6.19 Theorem *Let P be the set of the partial number-theoretic functions. Then for any object $\underline{X}$ in **pcD**, $\underline{X} < P$. In particular, P is reflexive.*

Proof. Let $\underline{X} = (X, X_0, \leq)$ be in **pcD**, and let $e: \omega \to X_0$ be an enumeration of the compact elements of $\underline{X}$. Define $\varphi: X \to P$ by setting, for all $n \in \omega$,

$$\varphi(x)(n) = \text{ if } e(n) \leq x \text{ then } 0 \text{ else if } \sim(x \Uparrow e(n)) \text{ then } 1 \text{ else } \perp .$$

Equivalently, $\varphi(x) \in P$ is uniquely determined by the ordered pair in which its domain splits

$$\langle \{ n \in \omega \mid \varphi(x)(n) = 0\}, \{n \in \omega \mid \varphi(x)(n) = 1\} \rangle = \langle \{n \in \omega \mid e(n) \leq x\}, \{n \in \omega \mid \sim(x \Uparrow e(n))\ \} \rangle .$$

It is easy to prove that φ is continuous.

In order to define $\psi: P \to X$, for any $f \in P$, set

$$X_f = \{e(i) \mid f(i) = 0 \ \underline{\text{and}}\ \forall j \leq i, \sim(e(i) \Uparrow e(j)) \to f(j) \neq 0\}.$$

First, X_f is pairwise consistent. Let i, j be such that $e(i), e(j) \in X_f$. Then $f(i) = f(j) = 0$. Suppose that $\sim(e(i) \Uparrow e(j))$. Then $i < j \ \Rightarrow\ f(i) \neq 0$ and $j \leq i \ \Rightarrow\ f(j) \neq 0$, which is impossible. Thus $e(i) \Uparrow e(j)$, so X_f is pairwise-consistent. By the consistency property of $\underline{X}$, we can define $\psi: P \to X$ by $\psi(f) = \sup_{\underline{X}}(X_f)$. It is a simple exercise to prove that ψ is continuous and that $\psi \circ \varphi(x) = x$ for all $x \in X$. Therefore $\underline{X}$ is a retract of P. ♦

It is clear that computability is at hand. As a matter of fact, the categories and results described so far can be "effectivized," in analogy to the examples in 2.4.1, e.g., the category **ED**. Denote by X_o the collection of the compact elements of $(X,\leq)$ in **pcD**.

9.6.20 Definition *Let $\underline{X} = (X, X_O, e_O, \leq)$ be in **pcD** and $e_O: \omega \to X_O$ (bijective). Then $\underline{X}$ is **effectively given** if*

1. *$e_O(n) \Uparrow e_O(m)$ is a decidable predicate*
2. *$\exists g \in R\ \forall n,m\ (\ e_O(n) \Uparrow e_O(m) \ \Rightarrow\ e_O(g(n,m)) = \sup\{e_O(n), e_O(m)\}\ )$.*

It is easy to show that if $\underline{X}$ and $\underline{Y}$ are effectively given, then also the space of continuous functions from $\underline{X}$ to $\underline{Y}$ is effectively given. Indeed, the category of effectively given ω-algebraic pair-consistent c.p.o.'s and continuous functions is cartesian closed (similarly to **ED**).

Recall that ideals are downward closed directed subsets of a poset $(X,\leq)$. As in the definition of the category **CD** of constructive domains in 2.4.1, the idea now is to take, within an effectively given $(X,X_o,e_o,\leq)$, only the l.u.b of those ideals of X_o, which are indexed over a recursively enumerable set. Call **computable** elements the l.u.b of the r.e. indexed ideals. Clearly, the computable elements of P are exactly the partial recursive functions, PR.

9.6.21 Definition *A sub-p.o.set X_c of an effectively given $\underline{X} = (X,X_o,e_o,\leq)$ is a* ***constructive and pair-consistent domain (ccd)*** *if for any ideal $D \subseteq X_o$ one has:*

D is principal in X_c iff $e_o^{-1}(D)$ is a recursively enumerable set.

Thus X_c contains exactly the computable elements of $\underline{X}$, e.g., $P_c = PR$. By the following exercise, this gives yet another interesting CCC.

Exercises

i. Let **CCD** be the category whose objects are ccd's and whose morphisms are the continuous and computable functions (computable as elements of the function spaces). Prove that **CCD** is cartesian closed.

ii. Prove 9.6.19 above for PR instead of P, i.e., prove that also PR is reflexive in **CCD**.

Consider now the type structure $\{PR^\sigma\}_{\sigma\in Tp}$ constructed over PR in **CCD**. These are known as the (higher type) partial recursive functionals. By the exercise and lemma 9.6.18, they yield a (type-free) λ-model in any finite type, as $PR^{\sigma\to\sigma} < PR^\sigma$.

CCD tidily relates to categories defined in the previous section, provided that a minor generalization is made. So far, we have only been dealing with **total** applicative structures, i.e., where "$\cdot$" is everywhere defined, as combinatory algebras are total structures. There exist, though, interesting **partial** applicative structures: for example, Kleene's $K = (\omega,\cdot)$, where $n\cdot m = \phi_n(m)$ for some acceptable Gödel numbering $\phi: \omega\to PR$ of the partial recursive functions.

In general, given a *partial* applicative structure $B = (X,\cdot)$, i.e., "$\cdot$" may be a partial binary operation, one can define the categories $\mathbf{P}_B$, $\mathbf{ER}_B$ and $\mathbf{PER}_B$ as in definitions 9.5.2 and 9.5.4, with a minor caution. Since we deal here with categories of total morphisms, we consider only total polynomials in each $\mathbf{P}[X^n,X^m]$; in particular in $\mathbf{P}[X,X]$, when defining $\mathbf{ER}[R,S]$ in definition 9.5.4. As for $\mathbf{PER}[R,S]$, each $f\in\mathbf{PER}[R,S]$ is "computed," in the sense of 9.5.4, by a (possibly partial) $g\in\mathbf{P}[X,X]$ which must be total on domR, though. In conclusion, by the exercise before 9.5.3 and remark 9.5.9, if $X\times X < X$ in $\mathbf{P}_B$, then proposition 9.5.5 applies similarly and $\mathbf{P}_B$ and $\mathbf{ER}_B$ are full sub-CC of the CC $\mathbf{PER}_B$. Moreover, if B is a partial combinatory algebra, then $\mathbf{PER}_B$ is a CCC by corollary 9.5.8 and what follows. The remaining results carry on similarly.

This remark has already been (implicitly) applied when defining **PER** over ω. As a matter of fact, Kleene's $K = (\omega,\cdot)$ is a partial combinatory algebra, as it contains (indices for) partial k and s. Thus, the properties of **PER** (= $\mathbf{PER}_K$ and **ER** = $\mathbf{ER}_K$) could be derived also by the work in this chapter.

Exercise The reader has already checked that the category **EN** of numbered sets (see the examples in section 2.2) is equivalent to $\mathbf{ER}_K$. He or she may try to give now an extension of **EN** which is cartesian closed and equivalent to $\mathbf{PER}_K$.

We already mentioned, in example 2.4.1, an important theorem, the Generalized Myhill-Shepherdson Theorem, which related constructive domains and **EN**. In this frame, it may be restated as "**CCD** is equivalent, as a category, to a full subCCC of $\mathbf{ER}_K$." Jointly to the exercise, this provides interesting embeddings, up to equivalence, of **CCD** into $\mathbf{ER}_K$ into $\mathbf{PER}_K$.

It should be clear why Kleene's K is only a partial combinatory algebra and not a total one. If ω^ω represents $\mathbf{PER}[\omega,\omega] = \mathbf{P}[\omega,\omega]$, then there is no principal $p \in \mathbf{PER}[\omega,\omega^\omega]$, as there is no Gödel-numbering or effective enumeration of $\mathbf{P}[\omega,\omega] = R$, the recursive functions.

One may try another way to turn K into a combinatory algebra. Consider first the category $\mathbf{CCD}_p$ of ccd's and partial morphisms, i.e., partial continuous maps with open domain. Let $\_^\perp = \_^\circ$ be the lifting functor defined in example 5.2.4. $\mathbf{CCD}[\omega^\perp,\omega^\perp]$, then, coincides with PR plus the everywhere constant function on $\omega^\perp$. From any acceptable Gödel-numbering of PR it is easy to construct a principal $p' \in \mathbf{CCD}[\omega^\perp,PR]$; however, $\omega^\perp \times \omega^\perp < \omega^\perp$ in **CCD** fails, by a simple continuity argument. Thus also $(\omega^\perp,\cdot)$ does not yield a combinatory algebra. However, by $PR^{\sigma\to\sigma} < PR^\sigma$ in **CCD** and 9.5.10, λ-models may be found at any finite higher type.

Note that p' above or the Gödel-numberings are principal morphisms which cannot be turned into retractions, by the latter observation or because, given a partial recursive function, there is no uniform effective choice of one of its indices, by the Rice theorem. More examples could be given by taking combinatory algebras which cannot be turned into λ-models. Indeed, a simple example is given by the term model of Combinatory Logic, i.e., by a "model" constructed by purely syntactic tools.

Remark *(Partial vs. total maps)* The reader may have noticed that there are various notions of partiality mentioned in these sections. As for the last, it poses no problem: a partial applicative structure has a (possibly) partial application. One may construct over it, though, categories of total maps, such as **EN**, $\mathbf{ER}_K$ (cf. the definition of morphisms in these categories).

Note now that we called $\{PR^\sigma\}_{\sigma\in Tp}$ the *partial* continuous and computable functionals, even though, at any type higher then 1, these are *total*, i.e., always defined, continuous (and computable) maps. Why are they called partial, in the literature? The intuition should be clear, but the categorical

notion of complete object (see definition 2.5.6) may provide a better or more rigorous understanding. As for the intuition, PR contains partial maps, in the ordinary sense, and each of the higher types contains a least element: the empty set in PR, the constantly empty map in PR^{PR} and so on. Intuitively, these least elements give the "undefined value" in the intended type. Categorically, this is understood by observing that, by this, each higher type, except for type 0 (i.e., ω) is a complete object in the intended category of partial maps, in the sense of section 2.5. The point is that these objects, by theorem 2.5.9, are exactly those such that the partial morphisms may be extended, and indeed viewed, as total ones.

Remark *(On total functionals)* Consider now ω and the total maps from ω to ω. We hint now at how to move to higher types and preserve totality of morphisms, in a categorical environment where it is possible to give a good notion of computability.

In sections 3.4 and 5.3 we presented the CCC (**sep-**)**FIL** of (separable) filter spaces and discussed some of its categorical properties. We also mentioned that this category is essentially nontopological, as some relevant types are not in the range of the embedding functor H: **Top**$\rightarrow$**FIL**, defined in example 5.3.7. In particular, **FIL**-ω, the sub-CCC of **sep-FIL** generated by ω, endowed with (the convergence induced by) the discrete topology, contains non topological types. The objects in **FIL**-ω are the sets of total continuous functionals.

On separable filter spaces, with an enumeration $\{U_i\}_{i\in\omega}$ of the base, one may consider the effectively given objects by generalizing the technique for domains in example 2.4.1 and definition 9.6.3. In short, one has to require that the base is decidable in the sense that

$$\{\, i \mid U_i = \varnothing \,\} \text{ and } \{\, (i_1,...,i_m,k_1,...,k_n) \mid U_{i1}\cap...\cap U_{im} \subseteq U_{k1}\cap...\cap U_{kn} \,\}$$

are recursive (uniformely in n and m). Then the computable elements are defined as limits of r.e. indexed filters. More precisely, let $\downarrow^s$ be the notion of convergence over filter spaces given in (s-conv.) in 5.3.7. Then set, for $(X,F)\in$**sep-FIL** with a decidable base,

$$x\in X \text{ is } \textbf{computable} \text{ iff } \exists\Phi\downarrow^s x \;\; \{\, i \mid U_i\in\Phi \,\} \text{ is r.e..}$$

In higher types, this defines total computable functionals.

Exercise Embedd the category **ED**, as topogical spaces, in 2.4.1 into **sep-FIL** by the functor H : **Top**$\rightarrow$**FIL** in 5.3.7 and describe how H and its left adjoint behave on computable elements. The computable elements of **FIL**-ω are the (Kleene-Kreisel) total continuous and computable functionals (see references).

As a final connection to the other categories of effective maps we used here, we just mention that a complex quotienting technique, which hereditarily defines total elements as the functions that take total elements to total elements, allows us to establish adjoint equivalences between **ED-**$\boldsymbol{\omega}^{\perp}$, the sub-CCC generated by $\boldsymbol{\omega}^{\perp}$ in **ED**, and $\mathbf{PER}_K$-ω, the sub-CCC generated by (ω,id) in $\mathbf{PER}_K$.

Similarly, the subCCC $\mathbf{PER}_{P\omega}$-ω in $\mathbf{PER}_{P\omega}$, the category of p.e.r.'s over the reflexive object $P\omega$, corresponds to **FIL-**ω. (See references).

References The original construction in Scott (1972) gives a non trivial isomorphism $A^A \cong A$ in a CCC of lattices. The general notion of categorical model of the type-free calculus $\boldsymbol{\lambda\beta\eta}$ and $\boldsymbol{\lambda\beta}$ are investigated in Berry (1979), Obtulowicz and Wiweger (1982), Koymans (1982), by using, though, the category of retractions, which does not need to have e.p., instead of **PER** (see Barendregt (1984) for a survey). By this one may deal with a weaker notion of model, the λ-algebras. λ-algebras, originally called pseudo-λ-models in Hindley and Longo (1980), are exactly the models of Combinatory Logic with β-equality (see also Barendregt (1984) or Hindley and Seldin (1986)). They are characterized as in theorem 9.5.11(2), by dropping the assumption that **C** has enough points and they are in between combinatory algebras and λ-models (in Barendregt and Koymans (1980) it is shown that the term model of CL cannot be turned into a λ-algebra.) However, retractions do not form a CCC on combinatory algebras and do not help in the categorical understanding of CL Thus we used **PER**, as in Longo and Moggi (1990), which we largely followed and where the models of CL were first characterized. The use of a choice operator ε in order to extend combinatory algebras to λ-models, which we followed here, is formalized in Meyer (1982).

Cousineau and al. (1985) and Curien (1986) introduce the categorical abstract machine.

The valid isomorphisms in Cartesian Closed categories are characterized in Bruce and DiCosmo and Longo (1990) (by a different proof, they where also given in Soloviev (1983)). The invertibility theorem 9.6.13, for the type-free λ-calculus, is due to Dezani (1976). Rittri (1989) applies isomorphisms in all CCC's to retrieval methods in libraries of programs.

Finally, pairwise consistent domains and the properties of $T\omega$ are investigated in Plotkin (1978). Filter spaces and related categories are used, for higher type recursion theory, in Hyland (1979). Ershov (1973-76) and Longo and Moggi (1984-84a) establish the relation between various classes of total and partial functionals.

Chapter 10

RECURSIVE DOMAIN EQUATIONS

One of the early applications of Category Theory to computer science was the solution of recursive domain equations. This kind of equation is typical of every language that allows an explicit or implicit form of self-application of its data types (such as a recursive procedure call, say).

For example, by theorem 9.5.10 we already know that in order to give semantics to the pure λ-calculus we need a domain isomorphic to its own function space; moreover, if we are interested in computing over a fixed domain A of atoms, we need a solution to the equation

$$(*) \quad X \cong A+(X \to X).$$

In general, recursive specification of domains can be seen as a particular case of recursive definition of data types that is an even more important topic in computer science. For example every programmer is used to considering the data type of all the lists of objects of type A as a solution to the following recursive equation:

$$A\_List = Nil + A \times (A\_list)$$

where Nil is a one-element data type.

In many respects, the general and unified theory of this kind of equation, mainly developed in the framework of Category Theory, has provided the base for the elimination, in most modern languages, of many unreasonable restrictions in the definition of recursive data types that existed in older languages. Note that these restrictions were not always motivated by implementation problems, but often by a real misunderstanding of the semantics of recursive definitions.

The first mathematical difficulty in giving a meaning to recursive specifications of data types is that they do not always have a set-theoretic solution. For example, equation (*) above has no solution in **Set** by obvious cardinality reasons: the arrow-domain $A \to B$ cannot be interpreted as the collection of all functions between A and B. The natural choice is to consider categories other than **Set**, with fewer morphisms but sufficiently many as to include all "computable functions" over the intended data types. The relevance of morphisms suggests why the categorial framework arises so naturally in this context. Note that the intended category **C** must be still Cartesian closed, in order to give the correct interpretation to the function space constuction. Once **C** is fixed, the idea for solving recursive domain equations is to use some sort of *fixed point technique*. To be more specific, in the categorical approach, the general form of equations such as (*) above, looks like

$$(**) \quad X \cong F(X)$$

where X ranges over the object of a category **C** and $F: \mathbf{C} \to \mathbf{C}$ is a covariant endofunctor of the category. If **C** is ω-cocomplete, **C** has an initial object, and F is ω-continuous, then theorem 6.5.2

gives a solution to (**). The main problem is that an obvious definition does not always exist for such an F, as we are going to show in the next section.

10.1 The Problem of Contravariant Functors

Consider again the equation

$$(*) \quad X \cong A+(X \to X)$$

in the introduction. In order to apply theorem 6.5.2, we need to work in a ω-cocomplete category **C**. Moreover, the category must be Cartesian closed and have coproducts, so that we can give the expected interpretation to the operators $\to$ and +, which appear in (*).

Such a category is not difficult to find: an example is the category **CPOS** of c.p.o.'s with a least (bottom) element and strict (bottom-preserving) continuous functions for morphisms. The next step is to define a covariant functor $F: \mathbf{C} \to \mathbf{C}$ such that, for any object X of **C**, $F(X) = A+(X \to X)$. The first idea would be to express F as a composition of the functors of sum and exponentiation associated to the closure properties of **C**. For example, if $A+\_: \mathbf{C} \to \mathbf{C}$ is the functor which takes B to A+B, and f to id_A+f, exp is the exponentiation functor, and Δ is the diagonal functor, we could be tempted to define

$$F = (A+\_) \circ \exp \circ \Delta$$

Unfortunately, this is not possible since exp is contravariant in the first component and cannot be composed with the diagonal function; in other words, since $\exp: \mathbf{C}^{op} \times \mathbf{C} \to \mathbf{C}$, and $\Delta: \mathbf{C} \to \mathbf{C} \times \mathbf{C}$, the previous equation is ill typed.

We would like to have a way of transforming every functor F in an associated functor F^* with the same behavior on objects, but covariant on morphisms, in all its components.

Unfortunately this is not possible in general, but still there is a very simple way, as we will see, to turn a contravariant endofunctor $F: \mathbf{C} \to \mathbf{C}$ into a covariant endofunctor F^* in an associated category $\mathbf{C}^*$, which has the same objects of **C**, and such that isomorphisms in $\mathbf{C}^*$ give isomorphisms in **C**.

Then, if $\mathbf{C}^*$ is ω-cocomplete and has a terminal object, and F^* is ω-continuous, we can apply theorem 6.5.2, find a solution in $\mathbf{C}^*$, and then derive a solution in **C.** Note that the category $\mathbf{C}^*$ does not need to be cartesian closed, neither it must be closed under coproducts, since we already know how to give a meaning to the constuctions of new objects in **C**.

We shall define $\mathbf{C}^*$ as a suitable subcategory of the following category $\mathbf{C}^{+-}$.

10.1.1 Definition *Given a category* $\mathbf{C}$*, the category* $\mathbf{C}^{+-}$ *has the same object of* $\mathbf{C}$ *and*

$$f \in \mathbf{C}^{+-}[A, B] \quad \text{iff} \quad f = (f^+, f^-) \text{ with } f^+ \in \mathbf{C}[A, B] \text{ and } f^- \in \mathbf{C}[B, A].$$

Composition is defined by $(f^+, f^-) \circ (g^+, g^-) = (f^+ \circ g^+, g^- \circ f^-)$.

An intuitive way to regard the category $\mathbf{C}^{+-}$ is the following. Consider the objects as data types; there is a morphism from A to B if and only if a pair of "coercions" is given in order to go from one to the other. Note also that two objects are isomorphic in $\mathbf{C}^{+-}$ iff they are isomorphic in **C.**

We next show how to define covariant functors on $\mathbf{C}^{+-}$ from arbitrary functors on **C**. For simplicity, we reduce the definition to the case of a bifunctor F covariant in the first component and contravariant in the second.

10.1.2 Definition *Given a category* $\mathbf{C}$*, and a functor* $F: \mathbf{C}\times\mathbf{C} \to \mathbf{C}$ *contravariant in the first component and covariant in the second, the covariant functor* $F^{+-}: \mathbf{C}^{+-}\times\mathbf{C}^{+-} \to \mathbf{C}^{+-}$ *is defined by*

$$F^{+-}(A,B) = F(A,B).$$
$$F^{+-}((f^+,f^-), (g^+,g^-)) = (F(f^-,g^+), F(f^+,g^-)).$$

One problem with the category $\mathbf{C}^{+-}$ is that it is very unlikely to have colimits for all ω-chains. The interesting fact, though, is that the idea upon which definition 10.1.2 is based, works in *every* subcategory $\mathbf{C}^*$ of $\mathbf{C}^{+-}$, provided that *only those functors F such that* $(F(f^-,g^+), F(f^+,g^-))$ *is still a morphism in* $\mathbf{C}^*$ are considered.

Our goal, now, is to find a subcategory $\mathbf{C}^*$ of $\mathbf{C}^{+-}$ such that simple and common properties on **C** (such as, for example, the existence of limits for all diagrams) are enough to guarantee the existence of colimits for all ω-chains in $\mathbf{C}^*$.

In the search for such a category $\mathbf{C}^*$, we can be helped by some intuition.

When in theorem 6.5.2 we considered the ω-diagram $(\{F^i(0)\}_{i\in\omega}, \{F^i(z)\}_{i\in\omega})$, we had in mind that it is a chain of increasingly finer approximations of the limit. Thus, a morphism $F^i(0)$, in a sense, must explain *why* $F^i(z)$ is *less than* $F^{i+1}(z)$: no information must be lost passing from $F^i(z)$ to $F^{i+1}(z)$. Reasonably, we may try to define a subcategory **D** of $\mathbf{C}^{+-}$ whose morphisms express this kind of relation between objects.

Among the subcategories of $\mathbf{C}^{+-}$ that seem to satisfy this condition, an important one is $\mathbf{C}^{\mathbf{Ret}}$, whose morphisms (f^+,f^-) have the property that $f^-\circ f^+ = id$ (in **C**).

$\mathbf{C}^{\mathbf{Ret}}$ is also very attractive because*every* functor F on **C** may be still turned into a covariant functor F^{+-} on $\mathbf{C}^{\mathbf{Ret}}$ by means of definition 10.1.2.

Indeed, consider for example a bifunctor F contravariant in the first component and covariant in the second one. Then one has

$$\begin{aligned} F(f^+,g^-)\circ F(f^-,g^+) &= F(\,(f^+,g^-)\circ(f^-,g^+)\,) \\ &= F(f^-\circ f^+,\, g^-\circ g^+) && \text{by composition in } \mathbf{C}^{\mathbf{op}}\times\mathbf{C} \\ &= F(id, id) && \text{as } (f^+,f^-), (g^+,g^-) \text{ are morphisms of } \mathbf{C}^{\mathbf{Ret}} \\ &= id. \end{aligned}$$

Thus $F^{+-}((f^+,f^-), (g^+,g^-)) = (F(f^-,g^+), F(f^+,g^-))$ is well defined as $(F(f^-,g^+), F(f^+,g^-))$ is a retraction pair.

Unfortunately, $\mathbf{C}^{\mathbf{Ret}}$ does not seem to suffice for our purposes. For example, up to now, there is no known nontrivial Cartesian closed category $\mathbf{C}$ such that $\mathbf{C}^{\mathbf{Ret}}$ has colimits for every ω-chain. Indeed, this is poses a very interesting problem: whether it is possible to find such a category, or prove that it cannot exist.

It is very instructive to understand where the difficulty arises in general.

Suppose that the category **C** has limits for every diagram (this property holds in many interesting CCC's; see chapter 6 for some examples) and let $(\{D_i\}_{i\in\omega}, \{f_i\}_{i\in\omega})$ be an ω-chain in $\mathbf{C}^{\mathbf{Ret}}$. Then $(\{D_i\}_{i\in\omega}, \{f_i^-\}_{i\in\omega})$ is an ω-chain in **C**, and it has a limit $(L,\{\gamma_i\}_{i\in\omega})$. The object L seems to be a good candidate as a limit also for the chain $(\{D_i\}_{i\in\omega}, \{f_i\}_{i\in\omega})$ in $\mathbf{C}^{\mathbf{Ret}}$. Indeed, the following theorem holds:

10.1.3 Theorem *Let $\{D_i\}_{i\in\omega}, \{f_i\}_{i\in\omega})$ be a ω-chain in $\boldsymbol{C}^{\boldsymbol{Ret}}$. If $(L, \{\gamma_i\}_{i\in\omega})$ is a limit for $(\{D_i\}_{i\in\omega}, \{f_i^-\}_{i\in\omega})$ in **C**, then there is a cone $(L, \{(\delta_i,\gamma_i)\}_{i\in\omega})$ for $(\{D_i\}_{i\in\omega}, \{f_i\}_{i\in\omega})$ in $\boldsymbol{C}^{\boldsymbol{Ret}}$ (that is, every γ_i is a right member of a retraction pair).*

Proof: Fix D_j. For every i define $f_{j,i} : D_j \to D_i$ by:

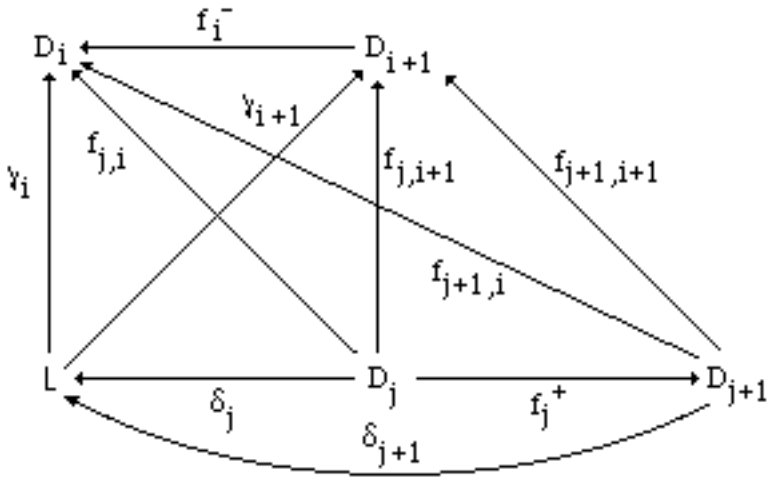
$$f_{j,i} = f_i^- \circ f_{i+1}^- \circ \ldots \circ f_{j-1}^- \quad \text{if } i < j$$
$$f_{i,i} = id$$
$$f_{j,i} = f_{i-1}^+ \circ \ldots \circ f_{j+1}^+ \circ f_j^+ \quad \text{if } i > j .$$

$(D_j, \{f_{j,i}\}_{i\in\omega})$ is a cone for $(\{D_i\}_{i\in\omega}, \{f_i^-\}_{i\in\omega})$, since $f_i^- \circ f_{j,i+1} = f_{j,i}$ as it is easy to check. Thus there exists a unique morphism $\delta_j: D_j \to L$ such that $\forall i\in\omega\ \gamma_i \circ \delta_j = f_{j,i}$. In case $i = j$, $\gamma_j \circ \delta_j = f_{j,j} = id$.

We still have to check that $\forall j\in\omega\ (f_j^+, f_j^-)\circ(\delta_{j+1},\gamma_{j+1}) = (\delta_j,\gamma_j)$.

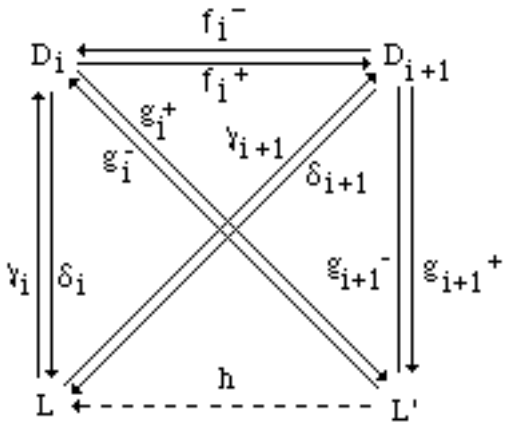
Now, $f_j^- \circ \gamma_{j+1} = \gamma_j$ by the definition of cone in **C**. In order to prove that $\delta_{j+1}\circ f_j^+ = \delta_j$, note that $\forall i\in\omega\ \gamma_i \circ \delta_{j+1} \circ f_i^+ = f_{j+1,i} \circ f_j^+ = f_{j,j} = \gamma_i \circ \delta_j$, and the result follows by unicity.

By a diagram, for $i > j$:



◆

Unfortunately, we have no way to prove that the cone $(L, \{\delta_i,\gamma_i\}_{i\in\omega})$ is universal. Indeed let $(L', \{g_i\}_{i\in\omega})$ be a cone for the ω-chain $(\{D_i\}_{i\in\omega}, \{f_i\}_{i\in\omega})$ in $\mathbf{C}^{\mathbf{Ret}}$. Then $(L',\{g_i^-\}_{i\in\omega})$ is a cone in **C** for $(\{D_i\}_{i\in\omega}, \{f_i^-\}_{i\in\omega})$ and there exists a unique morphism h in **C** from L' to L such that $\forall i\in\omega$ $h \circ \gamma_i = g_i^-$ and $\gamma_i \circ h = g_i^+$. In other words:



However, there is in general no reasonable way to define a morphism k from L to L' such that $h \circ k = id$. In the next section, we will see a way out of this problem.

10.2 0-Categories

Consider the class of morphisms $\{g_i^+\circ \gamma_i\}_{i\in\omega}$ from L to L' defined by the diagram at the end of the previous section. The morphism $g_i^+ \circ \gamma_i$: L→ L' describes *how the approximation of L up to the ith-level may be represented inside L'*. Intuitively, one may expect that, in some sense, $g_i^+ \circ \gamma_i \le g_{i+1}^+ \circ \gamma_{i+1}$. Moreover, if

1. the hom-sets in the category **C** are c.p.o.'s *and*
2. the class $\{g_i^+ \circ \gamma_i\}_{i\in\omega}$ is an ω-chain,

then we could soundly define $k = \cup_{i\in\omega}\{g_i^+ \circ \gamma_i\}$, and k would play the role required at the end of the previous section. This takes us to the notion of "0-category."

10.2.1 Definition *A category* ***C*** *is a* ***0-category*** *iff*

i. every hom-set ***C****[a,b] is a cpo, with a least element* $0_{a,b}$ *;*

ii. composition of morphisms is a continuous operation with respect to the partial order;

iii. for every $f\in$***C****[a,b] ,* $0_{b,c} \circ f = 0_{a,c}$.

Then, by definition, every 0-category satisfies (1) above. Our next problem is to ensure condition (2). Note first that $g_i^+ \circ \gamma_i = g_{i+1}^+ \circ f_i^+\circ f_i^- \circ \gamma_{i+1}$. Moreover, if $f_i^+\circ f_i^- \le id$, then $g_i^+ \circ \gamma_i \le g_{i+1}^+ \circ \gamma_{i+1}$. This suggests the refinement we need in the category $\mathbf{C}^{\mathbf{Ret}}$.

10.2.2 Definition Let **C** be a 0-category, and let i: D→E, j: E→D be two morphisms in C. Then (i,j) is a **projection pair** (from D to E) iff $j \circ i = id_D$ and $i \circ j \leq id_E$. (If (i,j) is a projection pair, i is an **embedding** and j is a **projection**.)

Exercises Prove the following statements:
1. If (i,j) is a projection pair from D to E, and (i',j') is another projection pair for the same two objects, then $i \leq i'$ iff $j \geq j'$.
2. Every embedding i has a unique associated projection $j = i^R$; conversely every projection j has a unique associated embedding $i = j^L$.

10.2.3 Definition *Let **C** be a 0-category. The category* $\mathbf{C^{Prj}}$ *has the same objects as **C**, and projection pairs as morphisms.*

Remark $\mathbf{C^{Prj}}$ is a subcategory of $\mathbf{C^{Ret}}$, and thus also of $\mathbf{C^{+-}}$. By the fact that every embedding i has a unique associated projection $j = i^R$ (and, conversely, every projection j has a unique associated embedding $i = j^L$), $\mathbf{C^{Prj}}$ is isomorphic to a subcategory $\mathbf{C^E}$ of **C** that has embeddings as morphisms (as well to a subcategory $\mathbf{C^P}$ of **C** which has projections as morphisms). We prefer to work in $\mathbf{C^{Prj}}$ since it looks more natural and carries more explicit information. Note, however, the following dualities: $(\mathbf{C^E})^{op} \cong \mathbf{C^P} \cong (\mathbf{C}^{op})\mathbf{^E}$ (and, of course, $(\mathbf{C^P})^{op} \cong \mathbf{C^E} \cong (\mathbf{C}^{op})\mathbf{^P}$).

Exercise Let **C** be a 0-category with terminal object t. Prove that t is terminal in $\mathbf{C^{Prj}}$ too. (Use property 10.2.1.(iii) in the definition of 0-category).

10.2.4 Theorem *Let **C** be a 0-category with all ωop-limits. Let* $(\{D_i\}_{i\in\omega}, \{(f_i^+, f_i^-)\}_{i\in\omega})$ *be an ω-chain in* $\mathbf{C^{Prj}}$ *(and thus in* $\mathbf{C^{Ret}}$*) and* $(L, \{(\delta_i,\gamma_i)\}_{i\in\omega})$ *be the cone in* $\mathbf{C^{Ret}}$ *defined by theorem 10.1.3. Then* $(L, \{(\delta_i,\gamma_i)\}_{i\in\omega})$ *is a cone also in* $\mathbf{C^{Prj}}$*. Moreover it is universal in this category.*
Proof In order to prove that $(L, \{(\delta_i,\gamma_i)\}_{i\in\omega})$ is a cone in $\mathbf{C^{Prj}}$ we must show that, $\forall i\in\omega$, $\delta_i \circ \gamma_i \leq id$. Note that $\forall i\in\omega$ $\delta_i \circ \gamma_i = \delta_{i+1} \circ f_i^+ \circ f_i^- \circ \gamma_{i+1} \leq \delta_{i+1} \circ \gamma_{i+1}$. Thus, $\{\delta_i \circ \gamma_i\}_{i\in\omega}$ is a chain and its limit $\Theta = \cup_{i\in\omega}\{\delta_i \circ \gamma_i\}$ must exist. We prove that $\Theta = id$ and, thus, that $\forall i\in\omega$ $\delta_i \circ \gamma_i \leq \Theta = id$. $\Theta = id$ since Θ is a mediating morphism between $(L, \{\gamma_i\}_{i\in\omega})$ and itself in **C**. Indeed, $\forall j\in\omega$,

$$\begin{aligned}\gamma_j \circ \Theta &= \gamma_j \circ \cup_{i\in\omega}\{\delta_i \circ \gamma_i\}\\ &= \gamma_j \circ \cup_{i \geq j}\{\delta_i \circ \gamma_i\}\\ &= \cup_{i \geq j}\{(\gamma_j \circ \delta_i) \circ \gamma_i\}\\ &= \cup_{i \geq j}\{f_{i,j} \circ \gamma_i\}\end{aligned}$$

$$= \cup_{i \geq j}\{f_{i,j} \circ \gamma_i\}$$
$$= \gamma_j.$$

We prove next that the cone $(L, \{(\delta_i,\gamma_i)\}_{i\in\omega})$ is universal in $\mathbf{C^{Prj}}$.

Let $(L',\{(g_i^+,g_i^-)\}_{i\in\omega})$ be another cone for $(\{D_i\}_{i\in\omega},\{(f_i^+,f_i^-)\}_{i\in\omega})$. That is,

$\forall i\in\omega \quad g_i^+ \circ \gamma_i = g_{i+1}^+ \circ f_i^+ \circ f_i^- \circ \gamma_{i+1} \leq g_{i+1}^+ \circ \gamma_{i+1}$

$\forall i\in\omega \quad \delta_i \circ g_i^- = \delta_{i+1} \circ f_i^+ \circ f_i^- \circ g_{i+1}^- \leq \delta_{i+1} \circ g_{i+1}^-.$

Define then

$h = \cup_{i\in\omega}\{g_i^+ \circ \gamma_i\} : L \to L'$

$k = \cup_{i\in\omega}\{\delta_i \circ g_i^-\} : L' \to L.$

Observe that (h,k) is a projection pair, for

$$k \circ h = \cup_{i\in\omega}\{\delta_i \circ g_i^-\} \circ \cup_{i\in\omega}\{g_i^+ \circ \gamma_i\}$$
$$= \cup_{i\in\omega}\{\delta_i \circ (g_i^- \circ g_i^+) \circ \gamma_i\}$$
$$= \cup_{i\in\omega}\{\delta_i \circ \gamma_i\}$$
$$= \Theta = id$$

and

$$h \circ k = \cup_{i\in\omega}\{g_i^+ \circ \gamma_i\} \circ \cup_{i\in\omega}\{\delta_i \circ g_i^-\}$$
$$= \cup_{i\in\omega}\{g_i^+ \circ (\gamma_i \circ \delta_i) \circ g_i^-\}$$
$$= \cup_{i\in\omega}\{g_i^+ \circ g_i^-\}$$
$$\leq id.$$

Moreover, (h,k) is a mediating morphism between $(L, \{(\delta_i,\gamma_i)\}_{i\in\omega})$ and $(L',\{(g_i^+,g_i^-)\}_{i\in\omega})$, since $\forall j\in\omega$

$$(h,k) \circ (\delta_j,\gamma_j) = (h \circ \delta_j , \gamma_j \circ k)$$
$$= (\cup_{i\in\omega}\{g_i^+ \circ \gamma_i\} \circ \delta_j , \gamma_j \circ \cup_{i\in\omega}\{\delta_i \circ g_i^-\})$$
$$= (\cup_{i \geq j}\{g_i^+ \circ \gamma_i \circ \delta_j\} , \cup_{i \geq j}\{\gamma_j \circ \delta_i \circ g_i^-\})$$
$$= (\cup_{i \geq j}\{g_i^+ \circ f_{j,i}\} , \cup_{i \geq j}\{f_{i,j} \circ g_i^-\})$$
$$= (g_j^+,g_j^-)$$

(h,k) is unique, because, if (h',k') is another mediating morhism, then

$$(h',k') = (h' \circ id, id \circ k')$$
$$= (h' \circ \Theta, \Theta \circ k')$$
$$= (h' \circ \cup_{i\in\omega}\{\delta_i \circ \gamma_i\}, \cup_{i\in\omega}\{\delta_i \circ \gamma_i\} \circ k')$$
$$= (\cup_{i\in\omega}\{h' \circ \delta_i \circ \gamma_i\}, \cup_{i\in\omega}\{\delta_i \circ \gamma_i \circ k'\})$$
$$= (\cup_{i\in\omega}\{g_i^+ \circ \gamma_i\}, \cup_{i\in\omega}\{\delta_i \circ g_i^-\})$$
$$= (h,k) \ . \ \blacklozenge$$

A useful characterization of ω-colimits in the category $\mathbf{C^{Prj}}$ is the following:

10.2.5 Proposition *The cone* $(L, \{(\delta_i,\gamma_i)\}_{i\in\omega})$ *for the* ω*-chain* $(\{D_i\}_{i\in\omega}, \{(f_i^+,f_i^-)\}_{i\in\omega})$ *in* $\mathbf{C}^{\mathbf{Prj}}$ *is universal iff* $\Theta = \cup_{i\in\omega}\{\delta_i \circ \gamma_i\} = id$.

Proof Exercise. ♦

Up to now, we have shown that, if **C** is a 0-category with all ωop-limits, then the category $\mathbf{C}^{\mathbf{Prj}}$ has colimits for every ω-chain.

The next point is to understand what we have lost with regard to the possibility of applying the construction in definition 10.1.2, which turns contravariant functors into covariant ones. Indeed, there is no reason to believe that the functor F^{+-} of this definition transforms projection pairs within projection pairs.

Recall now that a two-argument endofunctor F over **C**, which is contravariant in the first argument and covariant in the second one, has type $F: \mathbf{C}^{op}\times\mathbf{C} \to \mathbf{C}$.

10.2.6 Definition *Let* **C** *be a 0-category. A functor* $F: \mathbf{C}^{op}\times\mathbf{C} \to \mathbf{C}$ *is* ***locally monotonic*** *iff it is monotonic on the hom-sets; that is, for* $f, f' \in \mathbf{C}^{op}[A, B]$ *and* $g, g' \in \mathbf{C}[C, D]$, *one has*

$$f \le f' \ , \ g \le g' \ \Rightarrow \ F(f,g) \le F(f',g').$$

10.2.7 Proposition *If* $F: \mathbf{C}^{op}\times\mathbf{C} \to \mathbf{C}$ *is locally monotonic and* (f^+,f^-) , (g^+,g^-) *are projection pairs, then also* $F^{+-}((f^+,f^-),(g^+,g^-))$ *is also a projection pair.*

Proof: By definition $F^{+-}((f^+,f^-),(g^+,g^-)) = (F(f^-,g^+), F(f^+,g^-))$. Compute then

$$\begin{aligned} F(f^+,g^-) \circ F(f^-,g^+) &= F(\ (f^+,g^-) \circ (f^-,g^+)\ ) \\ &= F(\ f^-\circ f^+,\ g^- \circ g^+) \\ &= F(\ id,\ id\ ) \\ &= id \end{aligned}$$

and

$$\begin{aligned} F(f^-,g^+) \circ F(f^+,g^-) &= F(\ (f^-,g^+) \circ (f^+,g^-)\ ) \\ &= F(\ f^+\circ f^-,\ g^+ \circ g^-) \\ &\le F(\ id,\ id\ ) \\ &= id. \ \blacklozenge \end{aligned}$$

The last step is to see if we can find some simple condition on the functor F in **C** such that the associated functor F^{+-} in $\mathbf{C}^{\mathbf{Prj}}$ is ω-continuous.

10.2.8 Definition *Let* **C** *be a 0-category. A functor* $F: \mathbf{C}^{op}\times\mathbf{C} \to \mathbf{C}$ *is* ***locally continuous (0-functor)*** *iff it is* ω*-continuous on the hom-sets. That is, for every directed set* $\{f_i\}_{i\in\omega}$ *in* $\mathbf{C}^{op}[A, B]$, *and every directed set* $\{g_i\}_{i\in\omega}$ *in* $\mathbf{C}[C, D]$, *one has*

$$F(\cup_{i\in\omega}\{f_i\}, \cup_{i\in\omega}\{g_i\}) = \cup_{i\in\omega} F(f_i,g_i) .$$

Of course, if F is locally continuous, then it is also locally monotonic.

Exercise Prove that the composition of two locally nonotonic (continuous) functors is still monotonic (continuous).

10.2.9 Theorem *Let* $\mathbf{C}$ *be a 0-category with all ωop-limits. Let also* $F: \mathbf{C}^{op}\times\mathbf{C}\to\mathbf{C}$ *be a locally continuous functor. Then the functor* $F^{+-}: \mathbf{C}^{\mathbf{Prj}}\times\mathbf{C}^{\mathbf{Prj}}\to\mathbf{C}^{\mathbf{Prj}}$ *is ω-continuous.*

Proof Let $(\{A_i\}_{i\in\omega}, \{(f_i^+,f_i^-)\}_{i\in\omega})$ and $(\{B_i\}_{i\in\omega}, \{(g_i^+,g_i^-)\}_{i\in\omega})$ be two ω-chains in $\mathbf{C}^{\mathbf{Prj}}$ and let $(L, \{(\rho_i^+,\rho_i^-)\}_{i\in\omega})$ and $(M, \{(\sigma_i^+,\sigma_i^-)\}_{i\in\omega})$ be the respective limits. Then, by proposition 10.2.5,

$$\Theta = \cup_{i\in\omega}\{\rho_i^+ \circ \rho_i^-\} = id;$$
$$\Psi = \cup_{i\in\omega}\{\sigma_i^+ \circ \sigma_i^-\} = id.$$

We must show that

$$(F^{+-}(L,M), \{F^{+-}((\rho_i^+,\rho_i^-), (\sigma_i^+,\sigma_i^-))\}_{i\in\omega}) = (F(L,M), \{F(\rho_i^-,\sigma_i^+), F(\rho_i^+,\sigma_i^-))\}_{i\in\omega})$$

is a limit in $\mathbf{C}^{\mathbf{Prj}}$ for the ω-chain

$$(\{F^{+-}(A_i,B_i)\}_{i\in\omega}, \{F^{+-}((f_i^+,f_i^-), (g_i^+,g_i^-))\}_{i\in\omega}).$$

It is clearly a cone, by the property of functors. We show that it is universal by proving that $\cup_{i\in\omega}\{ F(\rho_i^-,\sigma_i^+) \circ F(\rho_i^+,\sigma_i^-) \} = id$. The result then follows by proposition 10.2.5. We have the following:

$$\begin{aligned}\cup_{i\in\omega}\{ F(\rho_i^-,\sigma_i^+) \circ F(\rho_i^+,\sigma_i^-)\} &= \cup_{i\in\omega}\{ F(\rho_i^+ \circ \rho_i^-, \sigma_i^+ \circ \sigma_i^-) \}\\ &= F(\cup_{i\in\omega}\{\rho_i^+ \circ \rho_i^-\}, \cup_{i\in\omega}\{ \sigma_i^+ \circ \sigma_i^- \})\\ &= F(\Theta, \Psi)\\ &= F(id, id)\\ &= id. \blacklozenge\end{aligned}$$

In conclusion, we have described a way to turn arbitrary functors into covariant functors on suitably derived categories. Then we set the condition under which it is possible to obtain ω-continuous functors. The solution of equations, such as (*) at the beginning of this chapter, is thus immediately found for these functors.

Example In the introduction to this section we mentioned the important categorical equation $X \cong A+(X\to X)$. If we wish to find a solution to equations of this kind in some category $\mathbf{C}$ based on c.p.o.s (such as **CPO**, **CPOS**, **Scott Domains**, and so on), we are generally forced to relax the interpretation of at least one of the two symbols + and →. Indeed, for their nature, all these categories usually have fixpoints for all objects, and we know that this is inconsistent with having at the same time coproducts and cartesian closedness. A typical way for avoiding the problem is to content

ourselves with the interpretation of + as a weak coproduct. A **weak** coproduct is essentially defined as a coproduct in definition 2.2.6, but no unicity is requested for the commuting arrow h.
The category **CPO** of c.p.o.'s with least (bottom) element and continuous functions for morphisms is a 0-category with respect to the pointwise ordering of morphisms. **CPO** is a CCC with weak coproducts A+B given by the coalesced sum (i.e. by identifying the two bottom elements of A and B). Since it has coequalizers for every pair of objects, it has limits for every diagram.(see chapters 2 and 6). The functors $A+\_ : \mathbf{CPO}\to\mathbf{CPO}$ and $\to : \mathbf{CPO}\times\mathbf{CPO}\to\mathbf{CPO}$, respectively defined by:

$$A+\_\,(B) = A+B,\quad A+\_\,(f) = id_A + f,$$
$$\to(A,B) = B^A,\quad \to(f,g) = \lambda h.\, g \circ h \circ f,$$

are both locally continuous.

The diagonal functor $\Delta : \mathbf{CPO}\to\mathbf{CPO}\times\mathbf{CPO}$, defined by $\Delta(A) = (A,A)$ and $\Delta(f) = (f,f)$, is locally continuous too. Thus we can apply theorem 10.2.9 and conclude that the associated functors

a. $(A+\_)^{+-} : (\mathbf{CPO})^{\mathbf{Prj}} \to (\mathbf{CPO})^{\mathbf{Prj}}$
   $(A+\_)^{+-}(f^+, f^-) = (id_A+f^+, id_A+f^-)$
b. $(\to)^{+-} : (\mathbf{CPO})^{\mathbf{Prj}}\times(\mathbf{CPO})^{\mathbf{Prj}} \to (\mathbf{CPO})^{\mathbf{Prj}}$,
   $(\to)^{+-}((f^+, f^-), (g^+,g^-)) = (\lambda h.\, g^+ \circ h \circ f^-, \lambda h.\, g^- \circ h \circ f^+)$
c. $(\Delta)^{+-} : (\mathbf{CPO})^{\mathbf{Prj}} \to (\mathbf{CPO})^{\mathbf{Prj}}\times(\mathbf{CPO})^{\mathbf{Prj}}$
   $(\Delta)^{+-}(f^+, f^-) = ((f^+, f^-), (f^+, f^-))$

are ω-continuous. But composition of ω-continuous functors is still a ω-continuous functor; thus, the functor $F = (A+\_)^{+-} \circ (\to)^{+-} \circ (\Delta)^{+-} : (\mathbf{CPO})^{\mathbf{Prj}} \to (\mathbf{CPO})^{\mathbf{Prj}}$ is ω-continuous. Explicitly, F is defined by

$$F(X) = A+X^X$$
$$F(f^+,f^-) = (id_A+\lambda h.\, f^+ \circ h \circ f^-,\ id_A+\lambda h.\, f^- \circ h \circ f^+).$$

Thus, for every A, there exists X such that $X \cong A+X^X$.

References For an early computer scientific introduction to recursive domain equations, the reader should consult Stoy (1977). The first solution to the problem of finding a domain isomorphic with its own function space, as required for the type-free λ-calculus, was given in Scott (1972) which basically started the mathematical discussion on recursive definitions of data types and, more generally, the so called area of "denotational semantics". The categorical approach exposed in the present chapter is a direct generalization of Scott's method and is essentailly due to Wand (1979), Lehmann and Smith (1981) and Smith and Plotkin (1982). An introductory presentation may be also found in Plotkin (1978). Gunter (1985) investigates the notion of embedding as a particular case of adjunction and, thus, sets the base for an interesting generalization of the categorial approach.

Chapter 11

SECOND ORDER LAMBDA CALCULUS

The system λ_2, or second order λ-calculus, has been introduced by Girard for the sake of Proof Theory. It was meant to prove, in particular, a normalization theorem for second order arithmetic, PA_2 (also considered a sound formalization of analysis, by proof theorists). The key points are that (second order) λ-terms code proofs in (second order) systems based on natural deduction and, quite generally, that cut elimination corresponds to β reduction for λ-terms. Thus normal proofs for and consistency of PA_2 follow from the normalization theorem for λ_2 (see chapter 8 and references).

This calculus was later rediscovered by Reynolds and, since then, it has received great attention, mostly within the computer science community, from both the syntactic (typing, consistent extensions, etc.) and semantic points of view. The main novelty of λ_2, over the simply typed calculus, is the possibility of abstracting a term with respect to *type* variables; by this, λ_2 represents "polymorphic" functions, that is, functions that may have several types or, more precisely, that may update their own type.

It is largely agreed that this specific formalization of the broad notion of polymorphism in programming focuses most of the main concerns of the programmers who apply these methods and suggests relevant implementations of modularity in functional programming.

The type system of λ_2 is an extension of the simple types in section 8.2, and it is meant to provide a type for polymorphic functions (i.e., terms obtained by type abstraction). As we said, this is achieved by introducing *type variables* and allowing a quantification (informally, a product) over all types. The type $\forall X{:}Tp.T$ is the type of all those terms that, when applied to a type S, yield a new term of type $[S/X]T$. Types are defined impredicatively since in the inductive definition of the collection Tp of all types one has to use Tp itself, which is being defined. For example, Tp is used when defining $\forall X{:}Tp.T$ as a type or, also, $\forall X{:}Tp.T$ is said to be a type, while it contains a quantification over *all* types, including itself. The "dimensional clash" (and the semantic difficulty) which derives from impredicativity is evident when considering that a term of type $\forall X{:}Tp.T$ can be applied to its own type. This circularity, very common in mathematics (e.g., for least upper bounds and related notions) comes with the expressive power of the system and is reflected in the difficulty of finding sound mathematical models for it. It is our opinion that Internal Category Theory provides significant tools for this semantic investigation. First, it allows the explicit use of "constructive" universes, as an alternative to the (usually intended) set-theoretic frame for (small) categories, where the constructions below would provide trivial models. Second, it reflects and gives meaning to the circularity of impredicativity by a mathematically clear closure property of some internal categories.

11.1 Syntax

Types and (rough) terms are first defined in BN-form. The typing rules will pick up, among the rough terms, the legal ones. Types are built up from type variables, ranged by X, Y, Z…; terms are built up from (term) variables, ranged by x, y,…:

Type expressions: $T := X \mid (T \to S) \mid (\forall X{:}Tp.T)$

Term expressions: $e := x \mid (ee) \mid (eT) \mid (\lambda x{:}T.e) \mid (\Lambda X{:}Tp.e)$.

We use capital letters T,S,... as metavariables for type expressions.

Conventions: λ, Λ and $\forall$ are all variable binders. An unbound variable x in e is **free** in e (*notation:* $x \in FV(e)$). The **substitution** of a for x in e ([a/x]e) is defined by induction, provided that a is free for x in e, as usual.

A **context** is a finite set Γ of type variables; ΓX stands for $\Gamma \cup \{X\}$. A type T is **legal** in Γ iff $FV(T) \subseteq \Gamma$. A **type assignment in** Γ is a finite list $E = (x_1{:}T_1),\ldots, (x_n{:}T_n)$ such that any T_i is legal in Γ.

The typing relation $\Gamma;E \vdash e: T$, where E is a type assignment legal in Γ, e is a term expression and T is a type expression, is defined as follows:

(assumption) $\quad \Gamma;E \vdash x:T \quad$ if $(x:T) \in E$

(→I)
$$\frac{\Gamma;E(x:T) \vdash e: S}{\Gamma;E \vdash (\lambda x{:}T.e) : (T \to S)}$$

(→E)
$$\frac{\Gamma;E \vdash f: (T \to S) \quad \Gamma;E \vdash e: T}{\Gamma;E \vdash (fe) : S}$$

(∀I)
$$\frac{\Gamma X; E \vdash e : T}{\Gamma;E \vdash (\Lambda X{:}\, Tp.\, e) : (\forall X : Tp.\, T)} \quad (*)$$

* if there is no variable in FV(e) whose type depends on X

(∀E)
$$\frac{\Gamma;E \vdash f : (\forall X : Tp\,.\, T) \qquad \Gamma \vdash S : Tp}{\Gamma;E \vdash (fS) : [S/X]T}$$

Conversion Equations among well-typed terms are defined by the following axioms:

β.	$(\lambda x{:}A\ .b)\ e = [e/x]b$	
β_2.	$(\Lambda X{:}Tp\ .b)\ A = [A/X]b$	
η.	$\lambda x{:}A\ .bx = b$	if $x \notin FV(b)$
η_2.	$\Lambda X{:}Tp\ .bX = b$	if $X \notin FV(b)$

and by the usual rules that turn "=" into a congruence relation.

Before starting the formal definition of the models for such a system, it is worthwhile to say a few words about its interpretation, and to try to explain the following work by a naive presentation of the categorical meaning of λ_2. Note also that this chapter is followed by one entirely dedicated to examples of the abstract categorical treatment which we follow here. The reader may find convenient to check hi/her understanding of it against the structures presented in chapter 12.

As pointed out in chapter 8, any Cartesian closed category **C** can be used to give a categorical semantics to the simply typed lambda calculus: types (which are just constant types or "arrow types") are interpreted by objects of **C**; terms of type T, with free variables $x_1{:}T_1,\ldots, x_n{:}T_n$, are interpreted as morphisms from $T_1\times\ldots\times T_n$ to T. The categorical interpretation of the second order calculus generalizes this semantics; in this case, however, the collection of types Tp must be closed not only under the arrow construction, but also under universal quantification. If we write $\forall X{:}Tp.T$ as $\forall(\underline{\lambda}X{:}Tp.T)$, where $\underline{\lambda}$ is an informal lambda notation for functions, $\forall$ may be readily understood as a map from $(Tp \to Tp)$ to Tp, as it turns the map $\underline{\lambda}X{:}Tp.T$ in $(Tp\to Tp)$ into a type. Thus, the interpretation of $\forall$ should be a map from $Ob_C\to Ob_C$ to Ob_C. The problem is that this map has to be represented *internally* in some ambient category. A natural choice is to have some sort of metacategory E such that **C** may be regarded as an internal category of E, in the sense of section 7.2. Recall, in short, that **C** must consist of a pair (c_0, c_1) of objects of E such that, informally, $Ob_C = c_0$ and $Mor_C = c_1$. If E is Cartesian closed, then $\forall$ may typed as $\forall: c_0{}^{c0}\to c_0$.

Objects of the kind $Tp\to Tp$ (i.e. "points," or "elements" of $c_0{}^{c0}$) are usually called *variable types*. As we have already seen, if σ is a variable type, the type $\forall(\sigma)$ represents intuitively the collection of all the polymorphic terms e such that, for all types T, $(eT) : \sigma(T)$. This is equivalent to saying that $\forall(\sigma)$ is a *dependent product* that is, a product of different copies of c_0 indexed by σ on elements of c_0 itself. The projections of this dependent product yield the instances of the polymorphic terms in $\forall(\sigma)$ with respect to particular types. In other words, there will be in the model an operation $proj: (c_0\to c_0)\times c_0\to c_1$ that takes a variable type $\sigma: c_0\to c_0$, a type T, and gives a morphism $proj_\sigma(T): \forall(\sigma)\to\sigma(T)$; $proj_\sigma(T)$ describes how a polymorphic term of type $\forall(\sigma)$ can be instantiated into the type $\sigma(T)$, thus modeling the application of a term e in $\forall(\sigma)$ to a type T.

By the definition of a dependent product, we will also have an isomorphism between the polymorphic terms in $\forall(\sigma)$ and the collection of all the families $\{e_T: \sigma(T)\}_{T\in c0}$ of terms indexed

over all types. Let us call Δ this isomorphism, which relates a family of terms $\{e_T: \sigma(T)\}_{T\in c0}$ to the polymorphic term $\Delta(\{e_T: \sigma(T)\}_{T\in c0}) = \Lambda T{:}Tp.e_T : \forall(\sigma)$. The functions proj and Δ satisfy the following equations:

1. $proj_\sigma(S)\ (\ \Delta(\{e_T\}_{T\in c_0})\ ) = e_S$;
2. $\Delta(\{proj_\sigma(T)(e)\}_{T\in c_0}) = e$;

whose meaning may be sketched as follows:

1. if we define a polymorphic function from the collection of functions $\{e_T\}_{T\in c0}$, and then we consider the particular instance relative to the type S, this is equal to the original function e_S;

2. if, given a polymorphic function e, we consider the collection of all its instances $\{proj_\sigma(T)(e)\}_{T\in c_0}$ and then we use this collection to rebuild a polymorphic function, we obtain e again.

Equations 1. and 2. above are the key facts allowing the interpretation of rules β and η, respectively, for second order abstraction and application.

Exercise Compare equations 1 and 2 above with the equations for the categorical Cartesian product

$p_i \circ \langle f_1, f_2\rangle = f_i$ for $i = 1,2$;

$\langle p_1 \circ f, p_2 \circ f\rangle = f$.

11.2 The External Model

The informal discussion of the previous section should have motivated the use of internal concepts in describing the semantics of λ_2. The model definition inspired by these ideas will be the main object of study in this chapter and it is presented in the following section. We introduce here a different notion of categorical model that does not require the use of internal concepts. It is based on an algebraic generalization of the semantics of the simply typed lambda calculus in a bidimensional universe of Cartesian closed categories indexed over another (global) CCC. We will call this model "external." In this model the collection of types is represented by a single object of the global category, say c_0 (or Ω, as it is usually denoted in this approach), but no requirement is made in order to have an internal category with c_0 as an object of objects. This fact, however, must be heavily compensated for by a number of particular conditions that relate "on the nose" categorical properties of indexed categories, which are not very intuitive. Thus, on one hand, the external model is more manageable then the internal one; on the other, it is less limpid and, in a sense, less suggestive. We claim that both these properties of the external model are due to the fact that it is a particularly simple instance of the internal notion. More specifically, we will show that an external model is just an internal one whose ambient category is a topos of presheaves, and whose object of objects c_0 is a representable functor $[\_,\Omega]$. The understanding we propose of the external model also sheds some

light on the interplay among the different conditions in its definition and gives a new justification for some apparently ad hoc requirements.

At the base of the external notion of a model for λ_2, there is the notion of a class of small categories indexed over another (global) category E - essentially a contravariant functor G from E to **Cat**. E is a Cartesian category with a distinguished object Ω, which interprets the collection of types. Products Ω^n are used to give meaning to contexts. Arrows in $E[\Omega^n,\Omega]$ represent types with at most n free variables in the context Ω^n.

The functor G: E→**Cat** takes every context Ω^n in E to a (local) category $G(\Omega^n)$ whose objects are the types legal in that context. Thus these types appear both as arrows in E and as objects in the local categories, and it is natural to require $Obj(G(e)) = hom_E(e, \Omega)$. The arrows between two types σ and τ in a local category $G(\Omega^n)$ correspond to terms of type τ and free variables in σ. Every local category is required to be a model of a simply typed lambda calculus and, thus, it is Cartesian closed. As for the interpretation of the polymorphic product it is described by an adjoint situation between local categories; moreover this adjointness must be natural with respect to the global parameter given by the context.

11.2.1 Definition *An **external λ_2 model** (PL category) is a triple (E, G, Ω) where:*

1. E is a Cartesian closed category (global category);

2. Ω is a distinct object in E;

*3. $G: E^{op} \to$ **Cat** is a functor such that*

i. for each object e in E, $Obj(G(e)) = hom_E(e, \Omega)$, and, for each morphism $\sigma\in E[e',e]$, the functor $G(\sigma): G(e) \to G(e')$ acts on the objects of G(e) as $hom_E(\sigma,\Omega)$.

ii. for each object e in E, the (local) category G(e) is Cartesian closed; for every $\sigma\in E[e',e]$, the functor $G(\sigma): G(e) \to G(e')$ preserves the Cartesian closed structure "on the nose" (and not just up to isomorphism); that is, for $a,b\in Obj_{G(e)} = hom_E(e, \Omega)$ it satisfies:

a. $G(\sigma)(t_{G(e)}) = t_{G(e')}$, where $t_{G(e)}$ is the terminal object in G(e)
$G(\sigma)(!_a) = !_{G(\sigma)(a)}$

b. $G(\sigma)(a \times_{G(e)} b) = G(\sigma)(a) \times_{G(e')} G(\sigma)(b)$, where $\times_{G(e)}$ is the product in G(e)
$G(\sigma)(fst_{a,b}) = fst_{G(\sigma)(a),G(\sigma)(b)}$
$G(\sigma)(snd_{a,b}) = snd_{G(\sigma)(a),G(\sigma)(b)}$

c. $G(\sigma)([a,b]_{G(e)}) = [G(\sigma)(a), G(\sigma)(b)]_{G(e')}$, where $[,]_{G(e)}$ is the exponent in G(e)
$G(\sigma)(eval_{a,b}) = eval_{G(\sigma)(a),G(\sigma)(b)}$

iii. an E-indexed adjunction $< \mathbf{Fst}, \forall, \Delta > : G \to G^{\Omega}$, where

*a. $G^{\Omega} : E^{op} \to$ **Cat** (see definition 7.1.2) is the functor defined by*
$\forall e\in Ob_E \quad G^{\Omega}(e) = G(e\times\Omega)$
$\forall\sigma\in E[e',e] \quad G^{\Omega}(\sigma) = G(\sigma\times id_{\Omega})$

b. $\forall e\in Ob_E$, $\mathbf{Fst}(e) = G(fst_{e,\Omega}) : G(e) \to G^{\Omega}(e) = G(e\times\Omega)$ (with $fst_{e,\Omega}: e\times\Omega\to e$).

By definition 7.1.6 of an E-indexed adjunction, we have, for every object e in E, an adjunction

$$< G(fst_{e,\Omega}), \forall(e), \triangle(e) > : G(e) \to G(e\times\Omega)$$

and, moreover, $\triangle(e') \circ G(\sigma\times id_\Omega) = G(\sigma) \circ \triangle(e)$

11.2.2 Remark If (E,G) is an external λ_2 model, then we have the following natural transformations:

$$\times_{G(\_)}: \quad hom_{E\times E}(K^2(\_), (\Omega ,\Omega)) \to hom_E(\_, \Omega),$$
$$[,]_{G(\_)}: \quad hom_{E\times E}(K^2(\_), (\Omega ,\Omega)) \to hom_E(\_, \Omega),$$
$$\forall: \quad hom_E(\_\times\Omega, \Omega) \to hom_E(\_, \Omega),$$

where K^2 is the diagonal functor.

Indeed conditions 3.ii.b and 3.ii.c in definition 11.2.1 express exactly the naturality of $\times_{G(\_)}$ and $[,]_{G(\_)}$, while by definition $\forall$ is natural from G^Ω to G and, a fortiori, also from $hom_E(\_\times\Omega, \Omega)$ to $hom_E(\_, \Omega)$.

11.2.3 Lemma *For every object a in E there are morphisms*

$$x_0 : \Omega x\Omega \to \Omega$$
$$[,]_0 : \Omega x\Omega \to \Omega$$
$$\forall_0: \Omega^\Omega \to \Omega$$

such that, for each object e of E and for all objects σ,τ of G(e),

$$x_0 \circ <\sigma,\tau> = \sigma \times_{G(e)} \tau$$
$$[,]_0 \circ <\sigma,\tau> = [\sigma,\tau]_{G(e)}$$

and, for each object ρ of $G(e\times\Omega)$,

$$\forall_0 \circ \Lambda(\rho) = \forall(e)(\rho)$$

Proof We have the following natural transformations

$$\varepsilon_1 = \times_{G(\_)} \circ <,>^{-1} : hom_E(\_, \Omega\times\Omega) \to hom_E(\_, \Omega)$$
$$\varepsilon_2 = [,]_{G(\_)} \circ <,>^{-1} : hom_E(\_, \Omega\times\Omega) \to hom_E(\_, \Omega)$$
$$\varepsilon_3 = \forall \circ \Lambda^{-1} : hom_E(\_, \Omega^\Omega) \to hom_E(\_, \Omega)$$

where

$$\times_{G(\_)}: \quad hom_{E\times E}(K^2(\_), (\Omega,\Omega)) \to hom_E(\_, \Omega)$$
$$[,]_{G(\_)}: \quad hom_{E\times E}(K^2(\_), (\Omega,\Omega)) \to hom_E(\_, \Omega)$$
$$\forall: \quad hom_E(\_\times\Omega, \Omega) \to hom_E(\_, \Omega)$$

are the natural transformations of remark 11.2.2 and

$$<,>^{-1}: \quad hom_E(\_, \Omega\times\Omega) \to hom_{E\times E}(K^2(\_), (\Omega,\Omega))$$
$$\Lambda^{-1}: \quad hom_E(\_, \Omega^\Omega) \to hom_E(\_\times\Omega, \Omega)$$

are the natural isomorphisms given by the Cartesian closure of E.

Then, by the Yoneda lemma, the arrows x_0, $[,]_0$, $\forall_0$ with the requested properties are obtained by setting

$$x_0 = \varepsilon_1(\Omega\times\Omega)(id_{\Omega\times\Omega})$$
$$[,]_0 = \varepsilon_2(\Omega\times\Omega)(id_{\Omega\times\Omega})$$
$$\forall_0 = \varepsilon_3(\Omega^{\Omega})(id_{\Omega}\Omega)$$

For example, we have, for σ,τ: $e \to \Omega$,

$$\begin{aligned} x_0 \circ <\sigma,\tau> &= \varepsilon_1(\Omega\times\Omega)(id_{\Omega\times\Omega}) \circ <\sigma,\tau> \\ &= hom_E[<\sigma,\tau>, \Omega\times\Omega] (\varepsilon_1(\Omega\times\Omega)(id_{\Omega\times\Omega})) \\ &= \varepsilon_1(e)(hom_E[<\sigma,\tau>, \Omega\times\Omega](id_{\Omega\times\Omega})) \\ &= \varepsilon_1(e) <\sigma,\tau> \\ &= \sigma \times_{G(e)} \tau. \end{aligned}$$

The other equations are proved similarly. ♦

11.3 The External Interpretation

In this section we define, in several steps, the "external" interpretation of the second order lambda calculus.

11.3.1 Type Expressions A type expression T legal in a context $\Gamma = \{X_1,...,X_n\}$ is interpreted by a morphism $[T]_\Gamma$: $[\Gamma]\to[A]$ in E (where $[\Gamma] = c_0^n=((t\times c_0)\times\ldots\times c_0)$, inductively defined as follows:

1. $[X_i]_\Gamma = snd\circ fst^{n-i}$;
2. $[S\to T]_\Gamma = [,]_0 \circ < [S]_\Gamma, [T]_\Gamma >$;
3. $[\forall X{:}Tp.T]_\Gamma = \forall_0 \circ \Lambda([T]_{\Gamma X})$.

Note that, as in the simply typed λ-calculus, variables are projections and arrows are exponents. Morover, impredicative types are interpreted by formalizing (externally) the informal discussion at the end of 11.2 (see also 11.5.1).

11.3.2 Type assignment A legal type assignment $E = (z_1: S_1)... (z_n: S_n)$, in a context Γ, is interpreted by the product (local in $G([\Gamma])$)

$$[E]_\Gamma = (...(t_{G([\Gamma])} \times [S_1]_\Gamma)...)\times[S_n]_\Gamma = x_o^n \circ <...<t_{G([\Gamma])},[S_1]_\Gamma>...,[S_n]_\Gamma >$$

where $t_{G([\Gamma])}$ is the terminal object in the local category $G([\Gamma])$.

11.3.3 Terms A legal term M such that

$$\Gamma = X_1,...,X_n \ ; \ E = (z_1: S_1)... (z_n: S_n) \ \vdash \ M : T$$

is interpreted by a morphism

$[M]_{\Gamma E}$: $[E]_\Gamma \to [T]_\Gamma$ in the local category $G([\Gamma])$.

The inductive definition is

1. $[z_i]_{\Gamma E} = snd \circ fst^{n-i}$;
2. $[MN]_{\Gamma E} = eval \circ \langle [M]_{\Gamma E}, [N]_{\Gamma E} \rangle$;
3. $[\lambda x{:}S.M]_{\Gamma E} = \Lambda([M]_{\Gamma E[x:S]})$;
4. $[\Lambda X{:}Tp.M]_{\Gamma E} = \mathbb{\Delta}([M]_{\Gamma X;E})$;
5. $[M[T]]_{\Gamma E} = G(\langle id, [T]_\Gamma \rangle)(Proj_{([\Gamma])}) \circ [M]_{\Gamma E}$,

where $Proj_e$ is the counit of the adjunction $\langle G(fst_{e,\Omega}), \forall(e), \mathbb{\Delta}(e) \rangle : G(e) \to G(e \times \Omega)$, i.e., $Proj_e = \mathbb{\Delta}(e)^{-1}(id)$.

11.3.4 Remark The interpretation we have just given is somewhat informal. Indeed we should always specify in which local category we are working in, and we should add a lot of indices for the natural transformations. Note also that the interpretation is not, as it could seem, by induction on the syntactical structure of terms, but on the lenghth of the proof of their derivation (the proof that the terms are well typed). For example, a fully specified interpretation for $[\Lambda X{:}Tp.M]_{\Gamma E}$ would be: if $\Gamma X; E \vdash M: T$ and $\tau = [T]_{\Gamma X}$ then $[\Lambda X{:}Tp.M]_{\Gamma E} = (\Delta([\Gamma])([E]_\Gamma, \tau))([M]_{\Gamma X;E})$.

11.4 The Internal Model

In this section we define the notion of internal λ_2 model. The intuition is to require for an internal Cartesian closed category $c \in Cat(E)$ the existence of the arrow $\forall_0: c_0^{c0} \to c_0$, which gives the depended product, in a such a way that equations 1 and 2 of section 11.1 are verified. We obtain by this a characterization of internal models by means of *ground* equations, with the consequence that internal models are preserved by limit- and exponent-preserving functors.

In what follows, we always assume that the ambient category E is Cartesian closed and has finite limits.

11.4.1 Definition *Let a be an object of E.* $|a|$ *is the internal category* (a, a, id, id, id, id).

The internal $|a|$ represents (internalizes) the discrete category with exactly one morphism (the identity) for each point in a.

11.4.2 Definition *Let* $c = (c_0, c_1, DOM, COD, COMP, ID) \in Cat(E)$. *Define then:*
$\boldsymbol{c*} = (c_0^{c0}, c_1^{c0}, DOM*, COD*, COMP*, ID*)$ *with*

$DOM* = \Lambda(DOM \circ eval)$
$COD* = \Lambda(COD \circ eval)$
$COMP* = \Lambda(COMP \circ eval \times_0 eval \circ p)$

$$ID* = \Lambda(ID \circ eval)$$

where $p = \langle\langle \Pi_1 \circ p_1, p_2\rangle, \langle \Pi_2 \circ p_1, p_2\rangle\rangle_0 : (c_1{}^{c_0} \times_0 c_1{}^{c_0}) \times c_0 \to (c_1{}^{c_0} \times c_0) \times_0 (c_1{}^{c_0} \times c_0)$.

The idea behind the previous definition is that the object $c*$ represents internally the category of functors from $|c_0|$ to c. Note that, as $|c_0|$ is a discrete category, the functors F from $|c_0|$ to c are fully determined by their functions f_0 on objects. In effect, this informal idea may be formalized by the following remark: if E is Cartesian closed, so it is Cat(E). The object $c*$ of the previous definition is isomorphic in Cat(E) to the exponent of $|c_0|$ and c. The category $c*$ may be also regarded as the collection of all tuples of elements of c indexed by elements in c_0, for $|c_0|$ is a discrete category.

11.4.3 Definition *The constant internal functor* $K : c \to c*$ *is* $K = (k_0, k_1)$ *with*

$k_0 = \Lambda(fst) : c_0 \to c_0{}^{c_0}$ *where* $fst: c_0 \times c_0 \to c_0$ *is the projection*

$k_1 = \Lambda(fst) : c_1 \to c_1{}^{c_0}$ *where* $fst: c_1 \times c_0 \to c_1$ *is the projection.*

$K : c \to c*$ must be considered as a sort of diagonal functor. Informally, given an object b in c_0, its image under K is the tuple (indexed on c_0) of all-b elements (i.e., the constant-b function from c_0 to c_0). As a right adjoint to the diagonal functor $K^2: C \to C^2$ yields the categorical product, similarly, a right adjoint to the functor $K : c \to c*$ yields the (categorical) dependent product indexed over c_0.

11.4.4 Definition *A* ***model for*** λ_2 *is given by*

1. a Cartesian closed category E with all finite limits (global category);

2. an internal Cartesian closed category $c = (c_0, c_1, DOM, COD, COMP, ID) \in Cat(E)$;

3. a right (internal) adjoint to $K: c \to c*$.

The requirements on the global category E in the previous definition could be slightly relaxed: the notion of internal category can be also given in interesting ambient categories without *all* limits (see the next chapter for an example). Similarly, for a model of λ_2 we actually need only exponents of the form e^{c_0} in E. Our requirements are very close to those needed for models of the stronger calculus Fω. In this case the only further condition is that of having a right (internal) adjoint to $K_e: c \to c_e$ for every object e of E, where $c_e = (c_0{}^e, c_1{}^e, DOM_e, COD_e, COMP_e, ID_e)$ represents the category of functors from $|e|$ to c, and $K_e: c \to c_e$ is the internal functor defined by

$$K_e = (k_{e,0} = \Lambda(fst): c_0 \to c_0{}^e,\ k_{e,1} = \Lambda(fst): c_1 \to c_1{}^e).$$

By theorem 7.3.7, a right adjoint to $K: c \to c*$ is fully determined by

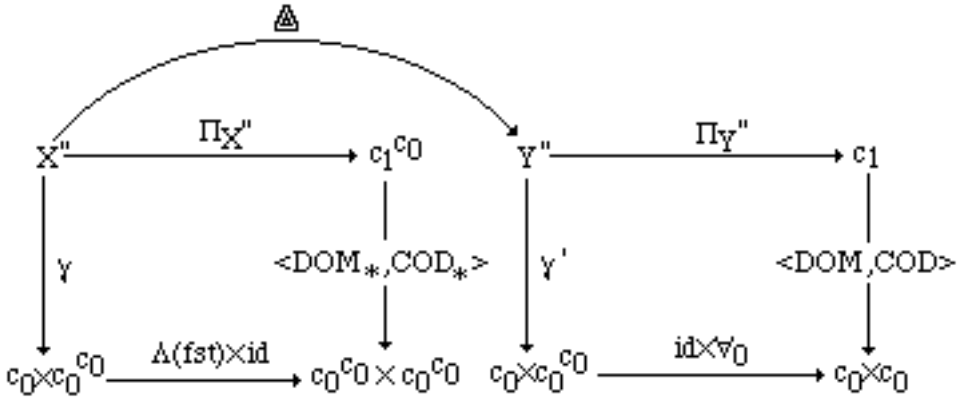
i. an arrow $\forall_0: c_0{}^{c_0} \to c_0$,

ii. an arrow PROJ: $c_0{}^{c0} \to c_1{}^{c0}$ such that $DOM* \circ PROJ = k_0 \circ \forall_0$, $COD* \circ PROJ = id$.
(if $h: e \to c_0{}^{c0}$, we use the abbreviation $PROJ_h$ for $PROJ \circ h$)

iii. an arrow $\Delta: X'' \to Y''$ where X" and Y" are respectively the pullbacks of

$< DOM* , COD* > : c_1{}^{c0} \to c_0{}^{c0} \times c_0{}^{c0}$, $k_0 \times id : c_0 \times c_0{}^{c0} \to c_0{}^{c0} \times c_0{}^{c0}$

$< DOM,COD > : c_1 \to c_0 \times c_0$, $id \times \forall_0 : c_0 \times c_0{}^{c0} \to c_0 \times c_0$



such that:

g0. $\gamma' \circ \Delta = \gamma$

g1. $(PROJ \circ p_2 \circ \gamma) * (\Lambda(fst) \circ \Pi_{Y''} \circ \Delta) = \Pi_{X''}$

h. $\Delta \circ < \gamma', (PROJ \circ p_2 \circ \gamma') * (\Lambda(fst) \circ \Pi_{Y''}) >_0 = id_{Y''}$

where $h*k = COMP* \circ <h,k>_0 = \Lambda(\Lambda^{-1}(h) \circledcirc \Lambda^{-1}(k))$, and $fst: c_1 \times c_0 \to c_1$ is the projection.

PROJ is the counit of the adjunction. In order to understand its meaning, it is useful to compare it with the counit of the Cartesian product. In that case, the counit is

$$(p_{(a1,a2),1}, p_{(a1,a2),2}) \in \mathbf{C}^2[(K^2(a_1 \times a_2), (a_1,a_2)]$$

where K^2 is the diagonal functor. That is, the counit is a collection of morphisms $p_{(a1,a2),i}$ in **C**, indexed over objects (a_1,a_2) of $\mathbf{C}^2$ and objects $i = 1,2$ in the category **2**, such that each $p_{(a1,a2),i}$ has domain $a_1 \times a_2$ and codomain a_i. Analogously, $PROJ: c_0{}^{c0} \to c_1{}^{c0} \cong c_0{}^{c0} \times c_0 \to c_1$ is a collection of morphisms $PROJ_\sigma(T)$ in c_1 indexed over objects σ of $c*$ and objects T of c (which now corresponds to the category **2** above), such that each projection $PROJ_\sigma(T)$ has domain $\forall_0(\sigma)$ and codomain $\sigma(T)$.

Consider now two points $f: t \to c_1{}^{c0}$, $g: t \to c_1$ in $c_1{}^{c0}$ and c_1. Informally, f is a family of terms in c_1 indexed by objects in c_0, and g is a term in c_1. If there exists some a such that $DOM \circ f = K_0 \circ a$, that is if all the terms represented by f have a common domain $a : t \to c_0$, then we can "apply" the isomorphism Δ and obtain the polymorphic term $\Delta \circ f$ (note the usual confusion between application and composition resulting from reasoning about points). Conversely, if there exists b such that $COD \circ g = \forall_0 \circ b$, then g is a polymorphic term of type $\forall_0 \circ b$.

Formally, for every e in E, given $f: e \to c_1{}^{c0}$ and $g: e \to c_1$ such that

$DOM \circ f = K_0 \circ a$ $\qquad$ $COD \circ f = b$

$DOM \circ g = a$ $\qquad$ $COD \circ g = \forall_{a0} \circ b$

equations (g1) and (h) above give

g1. $PROJ_b * (\Lambda(fst) \circ \Pi_{Y''} \circ \mathbb{\Delta} \circ <<a,b>,f>_0) = f$

h. $\Delta \circ < <a,b>, PROJ_b * (\Lambda(fst) \circ g) >_0 = <<a,b>,g>_0$

and with easy manipulations, recalling that $h*k = COMP* \circ <h,k>_0 = \Lambda(\Lambda^{-1}(h) \circledcirc \Lambda^{-1}(k))$, one obtains

g1'. $\Lambda((eval \circ PROJ_b \times id_{c0}) \circledcirc (\Pi_{Y''} \circ \Delta \circ <<a,b>,f>_0 \circ fst)) = f$

h'. $\Delta \circ < <a,b>, \Lambda((eval \circ PROJ_b \times id_{c0}) \circledcirc (g \circ fst)) >_0 = <<a,b>,g>_0$

which are the formalization of equations 1 and 2 of our informal discussion about second order models in section 11.1.

11.5 The Internal Interpretation

Let us now summarize some of the data that come with an internal model. All these objects, morphisms, and functions will be used to give an explicit definition of the interpretation for second order terms as follows:

i. For the global category E:

- a terminal object T, products and exponents
- projections: fst, snd
- evaluation morphism: eval (used for defining COMP*)
- pairing function: <,>
- "currying" function: Λ

ii. For the internal category:

- an arrow $t_0: T \to c_0$ defining the internal terminal object
- an arrow $x_0 : c_0 \times c_0 \to c_0$ defining the internal product
- an arrow $[,]_0 : c_0 \times c_0 \to c_0$ defining the internal exponent
- an arrow $\forall_0: c_0^{c0} \to c_0$ defining the dependent product
- internal projections: FST: $c_0 \times c_0 \to c_1$, SND: $c_0 \times c_0 \to c_1$
- internal evaluation morphism: EVAL: $c_0 \times c_0 \to c_1$
- instantiation morphism: PROJ: $c_0^{c0} \to c_1^{c0}$ of the dependent product
- internal pairing: $\ll,\gg: (c_0 \times c_0 \times c_0) \times_0 (c_1 \times c_1) \to (c_0 \times c_0 \times c_0) \times_0 c_1$

  $\ll,\gg(a,b,c, f: a \to b, g: a \to c) = (a,b,c, pairing(f,g): a \to b \times c)$
- internal currying function: $\mathbb{\Lambda}: (c_0 \times c_0 \times c_0) \times_0 c_1 \to (c_0 \times c_0 \times c_0) \times_0 c_1$

  $\mathbb{\Lambda}(a,b,c, f: a \times b \to c) = (a,b,c, curry(f): a \to c^b)$
- dependent pairing: $\mathbb{\Delta}: (c_0 \times c_0^{c0}) \times_0 c_1^{c0} \to (c_0 \times c_0^{c0}) \times_0 c_1$

  $\mathbb{\Delta}(a, \sigma, \{e_T: a \to \sigma(T)\}) = (a, \sigma, dep\_pairing(\{e_T\}): a \to \forall(\sigma))$.

We point out that an internal model is completely determined by (pullbacks and) a set of *ground* equations, that is, equations without (quantified) free variables; this contrasts with external models (or with standard, "external" Cartesian closed categories, for that matter). An important consequence of this is that *internal models are preserved by limit- and exponent- preserving functors* (that is functors preserving the structure for sources and targets of the data defining the model). This fact will be used later on to relate internal and external models.

Notation For $e \in Ob_E$, $e^n = (((t \times e) \times e) \times \ldots \times e)$ where t is the terminal object of E and e appears n times.

11.5.1 Type Expressions A type expression T legal in a context $\Gamma = \{X_1, ..., X_n\}$ is interpreted by a morphism $[T]_\Gamma : c_0^n \to c_0$ in E. In particular:

1. $[X_i]_\Gamma = snd \circ fst^{n-i}$
2. $[S \to T]_\Gamma = [,]_0 \circ < [S]_\Gamma, [T]_\Gamma >$
3. $[\forall X{:}Tp.T]_\Gamma = \forall_0 \circ \Lambda([T]_{\Gamma X})$

11.5.2 Type Assignments A type assignment $E = (z_1 : S_1)...(z_m : S_m)$ legal in a context $\Gamma = \{X_1,...,X_n\}$ is interpreted by the product

$$[E]_\Gamma = x_0^m \circ <...<t_0 \circ !c_0^n, [S_1]_\Gamma>..., [S_m]_\Gamma> : c_0^n \to c_0$$

where $x_0^1 = x_0$ and, for $i>1$, $x_0^i = x_0 \circ (x_0^{i-1} \times id)$.

11.5.3.Terms A legal term e such that

$$\Gamma = \{X_1,...,X_n\};\ E = [z_1 : S_1]... [z_m : S_m] \vdash e : T$$

is interpreted by a morphism

$$[e]_{\Gamma E} : c_0^n \to c_1$$

such that $DOM \circ [e]_{\Gamma E} = [E]_\Gamma : c_0^n \to c_0$

$COD \circ [e]_{\Gamma E} = [T]_\Gamma : c_0^n \to c_0$

In particular,

1. $[z_i]_{\Gamma E} = SND \odot FST^{n-i}$
   (where for simplicity we omit the "indexes" for FST and SND);
2. if $\Gamma;E \vdash f: S \to T$, $\Gamma;E \vdash e:S$, $\sigma = [S]_\Gamma$, $\tau = [T]_\Gamma$, then
   $[fe]_{\Gamma E} = EVAL_{\sigma,\tau} \odot (\leqslant, \geqslant \circ < <<[E]_\Gamma, [S \to T]_\Gamma>, \sigma>_0 , <[f]_{\Gamma E}, [e]_{\Gamma E}> >_0)$;
3. if $\Gamma;E(x:S) \vdash e: T$, $\sigma = [S]_\Gamma$, $\tau = [T]_\Gamma$, then
   $[\lambda x{:}S.e]_{\Gamma E} = \Pi_{y'} \circ \mathbb{\Lambda} \circ < <[E]_\Gamma, \sigma, \tau>, [e]_{\Gamma E(x:S)} >_0$;
4. if $\Gamma X; E \vdash e: T$ and $\tau = [T]_{\Gamma X}$, then
   $[\Lambda X{:}Tp.e]_{\Gamma E} = \Pi_{Y''} \circ \mathbb{\Delta} \circ < <[E]_\Gamma, \Lambda(\tau)>, \Lambda([e]_{\Gamma X;E})>_0$;

5. if $\Gamma,E \vdash e: \forall X{:}Tp.S$ and $\sigma = \Lambda([S]_{\Gamma X})$, then
$[eT]_{\Gamma E} = \Lambda^{-1}(PROJ_\sigma \ast (\Lambda(fst) \circ [e]_{\Gamma E})) \circ \langle id, [T]_\Gamma \rangle$
$= (\Lambda^{-1}(PROJ_\sigma) \mathbin{\circledcirc} ([e]_{\Gamma E} \circ fst)) \circ \langle id, [T]_\Gamma \rangle$
(where $h \ast k = COMP \ast \circ \langle h,k \rangle$; $h \mathbin{\circledcirc} k = COMP \circ \langle h,k \rangle$).

Given the above definitions, the proof of a soundness theorem, with the required substitution lemmas, is a routine check (as straightforward as it is tedious and laborious).

11.6 Relating Models

In the previous sections, two different notions of model have been introduced. We are now interested in the relation between them. It will turn out that the two notions are not as distant as they may seem.

We start by an analysis of how we can define an external model from an internal one. The construction is based on the *externalization* process of an internal category via hom-functors presented in chapter 7, which corresponds, essentially, to the Yoneda embedding. Since the hom-functor preserves pullbacks and exponents, we will be able to show that any internal model yields an "equivalent" external one.

Suppose that $c = (c_0,c_1,DOM,COD,COMP,ID) \in Cat(E)$ is an internal model. As the reader has probably imagined, the functor $[\_,c] : E^{op} \to \mathbf{Cat}$ of definition 7.4.2 plays the role of G in the external approach. For ease of reference, we recall here that definition.

11.6.1 Definition *Let $c \in Cat(E)$. The functor $G = [\_,c] : E^{op} \to Cat$ is defined in the following way:*

on objects $e \in E$ $[\_,c] = [e,c]$;

on arrows $\sigma: e' \to e$ $[\_,c](\sigma) = [\sigma,c]$ *is the functor from $[e,c]$ in $[e',c]$ which is defined as $[\sigma,c_0]$ on objects and as $[\sigma,c_1]$ on arrows.*

11.6.2 Lemma *$\forall\, \sigma: e' \to e$, $G(\sigma) = [\sigma,c]: [e,c] \to [e',c]$ acts on the objects of $[e,c]$ (i.e., on $E[e,c_0]$) as $E[\sigma,c_0]$.*

Proof By definition. ♦

11.6.3 Lemma *If c is (internally) Cartesian closed, then, for every e in E, $E_{e,c}$ is Cartesian closed.*

Proof (sketch)

Let 1. $\langle O, T, \mathbin{\circledcirc} \rangle : c \to 1$

2. $\langle \Delta, x, \ll,\gg \rangle : c \to c \times c$

3. $\langle x, [,] , \mathbb{\Lambda} \rangle : c \to c$

be the internal adjunctions given by the Cartesian closure of c. Then $\langle [\_,F], [\_,G], \Theta \rangle : [\_,c] \to [\_,d]$ is an E-indexed adjunction. By proposition 7.4.11 there are three E-indexed adjunctions

1'. $\langle [\_,O], [\_,T], o' \rangle : [\_,c] \to [\_,1]$

2'. $\langle [\_,\Delta], [\_,x], \langle,\rangle' \rangle : [\_,c] \to [\_,c \times c] \cong [\_,c] \times [\_,c]$

3'. $\langle [\_,x], [\_, [,]], \Lambda' \rangle : [\_,c] \to [\_,c]$ with parameters in $[\_,c]$.

Hence for every e in E, $[e,c]$ is Cartesian closed, since $[e,1]$ is the terminal category in Cat and $[e,\Delta] : [\_,c] \to [\_,c] \times [\_,c]$ is the diagonal functor. ♦

Propositions 7.4.10 and 7.4.11 allow us to give an explicit definition for the natural isomorphisms in (1')-(3') above. In particular,

given $\sigma,\tau,\gamma : e \to c_0$, and $f: e \to c_1 \times c_1$ such that $DOM \circ f = \Delta_0 \circ \sigma$, $COD \circ f = \langle \tau, \gamma \rangle$

$$\langle,\rangle'(f) = \Pi_y \circ \Subset_{,\Subset} \circ (\langle \langle \sigma, \langle \tau, \gamma \rangle \rangle, f \rangle_0) : e \to c_1;$$

given $\sigma,\tau,\gamma : e \to c_0$, and $g: e \to c_1$ such that $DOM \circ g = \sigma$, $COD \circ f = x_0 \circ \langle \tau, \gamma \rangle$

$$\langle,\rangle'^{-1}(g) = \Pi_x \circ \Subset_{,\Subset}^{-1} \circ (\langle \langle \sigma, \langle \tau,\gamma \rangle \rangle, g \rangle_0) : e \to c_1 \times c_1;$$

given $\sigma,\tau,\gamma : e \to c_0$ and $f: e \to c_1$ such that $DOM \circ f = x_0 \circ \langle \sigma,\tau \rangle$, $COD \circ f = \gamma$

$$\Lambda'(f) = \Pi_y' \circ \mathbb{\Lambda} \circ (\langle \langle\langle \sigma,\tau \rangle, \gamma \rangle, f) : e \to c_1;$$

given $\sigma,\tau,\gamma : e \to c_0$ and $g: e \to c_1$ such that $DOM \circ g = \sigma$, $COD \circ g = [,]_0 \circ \langle \tau, \gamma \rangle$

$$\Lambda'^{-1}(g) = \Pi_x' \circ \mathbb{\Lambda}^{-1} \circ (\langle \langle \sigma,\langle \tau, \gamma \rangle, g) : e \to c_1.$$

By exercise 7.4.12, given $\sigma,\tau : e \to c_0$, the projections associated to $\langle,\rangle'$ are derived from the internal projections FST and SND by

$$FST_{\sigma,\tau} = FST \circ \langle \sigma,\tau \rangle : e \to c_1$$

$$SND_{\sigma,\tau} = SND \circ \langle \sigma,\tau \rangle : e \to c_1.$$

Note that

$$DOM \circ FST_{\sigma,\tau} = x_0 \circ \langle \sigma,\tau \rangle$$

$$COD \circ FST_{\sigma,\tau} = \sigma$$

$$DOM \circ SND_{\sigma,\tau} = x_0 \circ \langle \sigma,\tau \rangle$$

$$COD \circ SND_{\sigma,\tau} = \tau.$$

Analogously, given $\sigma,\tau : e \to c_0$, the counit $EVAL_{\sigma,\tau}$ of Λ' for the object $[,]_0 \circ \langle \sigma,\tau \rangle$ is

$$EVAL_{\sigma,\tau} = EVAL \circ \langle \sigma,\tau \rangle : e \to c_1$$

where EVAL is the internal evaluation map.

Note that

$$DOM \circ EVAL_{\sigma,\tau} = x_0 \circ \langle [,]_0 \circ \langle \sigma,\tau \rangle, \sigma \rangle$$

$$COD \circ EVAL_{\sigma,\tau} = \tau .$$

11.6.4 Lemma *Let c be (internally) Cartesian closed. $\forall \sigma: e' \to e$, $[\sigma,c]: [e,c] \to [e,c']$ preserves the Cartesian closed structure "on the nose".*

Proof We only consider the product; the other cases are similar.

$\forall\tau,\gamma$ in [e,c] $[\sigma,c](\tau\times\gamma) = [\sigma,c]([e,x](\tau,\gamma))$

$= [\sigma,c](x_0 \circ <\tau,\gamma>)$ by def. of [e,x]

$= x_0 \circ <\tau,\gamma> \circ \sigma$

$= x_0 \circ <\tau \circ \sigma, \gamma \circ \sigma >$

$= x_0 \circ < [\sigma,c](\tau), [\sigma,c](\gamma) >$

$= [e,x]([\sigma,c](\tau), [\sigma,c](\gamma))$

$= [\sigma,c](\tau) \times [\sigma,c](\gamma)$ by def. of [e,x]. ♦

11.6.5 Lemma *For every e, a objects of E,* $[e,c^{|a|}] \cong [e\times a,c]$.

Proof E is Cartesian closed, thus there are the isomorphisms

$\Lambda_{e,c_0}: E[e\times a, c_0] \cong E[e, c_0{}^a]$

$\Lambda_{e,c_1}: E[e\times a, c_1] \cong E[e, c_1{}^a]$.

Λ_{e,c_0} and Λ_{e,c_1} are respectively the functions on objects and on arrows of a functor $\mathbf{\Lambda}$ from [e×a,c] to $[e,c^{|a|}]$. Indeed, for every σ: $e\times a \to c_0$

$\Lambda_{e,c_1}(id_\sigma) = \Lambda_{e,c_1}(ID \circ \sigma)$

$= \Lambda_{e,c_1}(ID \circ eval \circ \Lambda_{e,c_0}(\sigma)\times id)$

$= \Lambda_{e,c_1}(ID \circ eval) \circ \Lambda_{e,c_0}(\sigma)$

$= ID_{c^{|a|}} \circ \Lambda_{e,c_0}(\sigma)$

$= id_{\Lambda(\sigma)}$

and for every f, g: $e\times a \to c_1$

$\Lambda_{e,c_1}(g \odot f) = \Lambda_{e,c_1}(COMP \circ <g,f>)$

$= \Lambda(COMP \circ <eval \circ \Lambda_{e,c_1}(g)\times id, eval \circ \Lambda_{e,c_1}(f)\times id>)$

$= \Lambda(COMP\circ eval\times eval \circ <\Lambda_{e,c_1}(g)\times id, \Lambda_{e,c_1}(f)\times id>)$

$= \Lambda(COMP\circ eval\times eval \circ p \circ <\Lambda_{e,c_1}(g), \Lambda_{e,c_1}(f) >\times id)$

$= \Lambda(COMP\circ eval\times eval\circ p) \circ <\Lambda_{e,c_1}(g), \Lambda_{e,c_1}(f) >$

$= COMP_{c^{|a|}} \circ <\Lambda_{e,c_1}(g), \Lambda_{e,c_1}(f) >$

$= \Lambda_{e,c_1}(g) \odot \Lambda_{e,c_1}(f)$

Similarly, ${\Lambda_{e,c_0}}^{-1}$ and ${\Lambda_{e,c_1}}^{-1}$ define the functions on objects and on arrows of $\mathbf{\Lambda}^{-1}$, respectively. ♦

11.6.6 Lemma *Let K: c→c∗ be the functor of definition 11.4.3. For every e in E,* $\mathbf{\Lambda}^{-1} \circ [e,K] = G(fst) = [fst,c] : [e,c] \to [e,c^{|a|}]$.

Proof On objects σ: $e \to c_0$

$(\Lambda^{-1} \circ [e,K])(\sigma) = \Lambda^{-1}([e,K](\sigma))$

$= \Lambda^{-1}(k_0 \circ \sigma)$ by def. of [e,K]

$= \Lambda^{-1}(k_0) \circ \sigma\times id$

$= \mathrm{fst} \circ \sigma \times id$ by def. of k_0

$= \sigma \circ \mathrm{fst}$

$= [\mathrm{fst},c](\sigma)$ by def. of [fst,c].

On arrows $f: e \to c_1$

$(\Lambda^{-1} \circ [e,K])(f) = \Lambda^{-1}([e,K](f))$

$= \Lambda^{-1}(k_1 \circ f)$ by def. of [e,K]

$= \Lambda^{-1}(k_1) \circ f \times id$

$= \mathrm{fst} \circ f \times id$ by def. of k_1

$= f \circ \mathrm{fst}$

$= [\mathrm{fst},c](f)$ by def. of [fst,c]. ♦

11.6.7 Corollary *Let* $\Omega = c_0$ *. Then for every* e *in* E, $\langle [fst,c], [e,\forall] \circ \Lambda, \Delta' \circ \Lambda_{e,c_1} \rangle$ *:* $[e,c] \to [e \times \Omega, c]$ *is an adjunction.*

Proof By lemma 11.6.6, $[\mathrm{fst},c] = \Lambda^{-1} \circ [e,K]$. Then we have the isomorphisms

$$[e\times\Omega,c][\Lambda_{e,c_0}{}^{-1}([e,K](\sigma)), \tau] \overset{\Lambda_{e,c_1}}{\cong} [e,c^*][[e,K](\sigma), \Lambda_{e,c_0}(\tau)] \overset{\Delta'}{\cong} [e,c][\sigma, [e,\forall](\Lambda_{e,c_0}(\tau))]. \blacklozenge$$

Note that, given $\sigma: e \to c_0$, $\tau: e \times a \to c_0$, and $f: e \times a \to c_1$ such that

$\mathrm{DOM} \circ f = \Lambda^{-1}([e,K](\sigma)) = \sigma \circ \mathrm{fst}$

$\mathrm{COD} \circ f = \tau$,

we have $\Delta' \circ \Lambda_{e,c_1}(f) = \Pi_Y{}'' \circ \Delta \circ (\langle\langle \sigma, \tau \rangle, \Lambda(f) \rangle_0) : e \to c_1$, where $\Pi_Y{}''$ is as in the diagram after definition 11.4.4.

Analogously, given $\sigma: e \to c_0$, $\tau: e \times a \to c_0$, and $g: e \to c_1$ such that

$\mathrm{DOM} \circ g = \sigma$

$\mathrm{COD} \circ f = [e,\forall](\Lambda(\tau)) = \forall_0 \circ \Lambda(\tau)$

we have the following:

$(\Delta' \circ \Lambda_{e,c_1})^{-1}(g) =$

$= \Lambda_{e,c_1}{}^{-1}(\Delta'^{-1}(g))$

$= \Lambda^{-1}(\Pi_X{}'' \circ \Delta'^{-1} \circ (\langle\langle \sigma, \tau \rangle, g \rangle_0))$

$= \mathrm{eval} \circ (\Pi_X{}'' \circ \Delta'^{-1} \circ (\langle\langle \sigma, \tau \rangle, g \rangle_0)) \times id: e \times a \to c_1$.

In particular, given $\sigma : e \to c_0{}^{c_0}$, the counit $(\Delta' \circ \Lambda_{e,c_1})^{-1}(id_{[e,\forall](\Lambda(\tau))})$ is

$\mathrm{Proj}_{\sigma,\tau} = \mathrm{eval} \circ (\Pi_X{}'' \circ \Delta'^{-1} \circ (\langle \langle \sigma, \tau \rangle, ID \circ \forall_0 \circ \Lambda(\sigma) \rangle_0)) \times id: e \times c_0 \to c_1$

$= \mathrm{eval} \circ (\mathrm{PROJ} \circ \langle s,t \rangle) \times id$

$= \Lambda^{-1}(\mathrm{eval} \circ (\mathrm{PROJ} \circ \langle s,t \rangle))$

where PROJ is the internal counit.

11.6.8 Lemma *The isomorphism of the adjunction in corollary 11.6.7 is also natural in e; that is, for every* $\gamma: e \to e'$, $[\gamma,c] \circ (\Delta' \circ \Lambda_{e,c_1}) = (\Delta' \circ \Lambda_{e,c_1}) \circ [\gamma \times id,c]$.

Proof For every $\gamma: e \to e'$, and $f: e \times a \to c_1$ such that

$$\mathrm{DOM} \circ f = \Lambda^{-1}([e,K](\sigma)) = \sigma \circ \mathrm{fst}$$
$$\mathrm{COD} \circ f = \tau \quad (\text{where } \sigma: e \to c_0, \tau: e \times a \to c_0)$$

$$\begin{aligned}
([\gamma,c] \circ (\Delta' \circ \Lambda_{e,c_1}))(f) &= [\gamma,c]\,(\Delta'(\Lambda_{e,c_1}(f))) \\
&= [\gamma,c]\,(\Pi_Y'' \circ \Delta \circ \langle\langle \sigma, \tau \rangle, \Lambda(f) \rangle_0) \\
&= \Pi_Y'' \circ \Delta \circ \langle\langle \sigma, \tau \rangle, \Lambda(f) \rangle_0 \circ \gamma \\
&= \Pi_Y'' \circ \Delta \circ \langle\langle \sigma \circ \gamma, \tau \circ \gamma \rangle, \Lambda(f) \circ \gamma \rangle_0 \\
&= \Pi_Y'' \circ \Delta \circ \langle\langle \sigma \circ \gamma, \tau \circ \gamma \rangle, \Lambda(f \circ \gamma \times id) \rangle_0 \\
&= (\Delta' \circ \Lambda_{e,c_1})\,(f \circ \gamma \times id) \\
&= ((\Delta' \circ \Lambda_{e,c_1})\,([\gamma \times id,c](f)) \\
&= ((\Delta' \circ \Lambda_{e,c_1}) \circ [\gamma \times id,c])\,(f). \blacklozenge
\end{aligned}$$

11.6.9 Theorem *If (E,c) is an internal* λ_2*-model, then (E,* c_0*, G=[_,c]) is an external* λ_2*-model. Moreover, for any legal expression Q of* λ_2*, the internal interpretation of Q in (E,c) coincides with the external interpretation of Q in (E,* c_0*, G=[_,c]); t hat is, they are the same arrow in E.*

Proof Easy, by the previous lemmas. ♦

Now we prove that, using the "internalization" technique of chapter 7, we obtain from any external model $G: E^{op} \to \mathbf{Cat}$ an internal model in the topos of presheaves $E^{op} \to \mathbf{Set}$. The translation shows that, essentially, any PL-category is nothing else but an internal category in the category of presheaves having as object of objects the contravariant hom-functor. Recall (see definition 7.5.1) that given an E-indexed category $G: E^{op} \to \mathbf{Cat}$, we can build an internal category $\underline{G} = (\underline{G}_0, \underline{G}_1, \underline{\mathrm{DOM}}, \underline{\mathrm{COD}}, \underline{\mathrm{COMP}}, \underline{\mathrm{ID}}) \in \mathrm{Cat}(E^{op} \to \mathbf{Set})$ in the following way:

for all objects e, e' and arrows $f: e' \to e$ in E:

- $\underline{G}_0: E^{op} \to \mathbf{Set}$ is the functor defined by
  $$\underline{G}_0(e) = \mathrm{Ob}_{G(e)}$$
  $$\underline{G}_0(f) = G(f)_{ob} : \mathrm{Ob}_{G(e)} \to \mathrm{Ob}_{G(e')}$$
- $\underline{G}_1: E^{op} \to \mathbf{Set}$ is the functor defined by
  $$\underline{G}_1(e) = \mathrm{Mor}_{A(e)}$$
  $$\underline{G}_1(f) = G(f)_{mor} : \mathrm{Mor}_{G(e)} \to \mathrm{Mor}_{G(e')}$$
- $\underline{\mathrm{DOM}}: G_1 \to G_0$ is the natural transformation whose components are the domain maps in the local categories, i.e., for $e \in \mathrm{Ob}_E$, $\underline{\mathrm{DOM}}_e: \mathrm{Mor}_{G(e)} \to \mathrm{Ob}_{G(e)}$ is defined by
  $$\underline{\mathrm{DOM}}_e(h: \sigma \to \tau) = \sigma.$$
- $\underline{\mathrm{COD}}$, $\underline{\mathrm{ID}}$ and $\underline{\mathrm{COMP}}$ are defined analogously, "fiber-wise."

Note in particular that if (E, c_0, G) is a PL category, then $G_0 = E[\_, \Omega]$.

11.6.10 Proposition *If (E, c_0, G) is a PL category, then $\underline{G}$ is an internal Cartesian closed category.*

Proof By proposition 7.5.4. ♦

Before showing that $\underline{G}$ also has an internal dependent product, it is useful to take a closer look at the structure of the involved exponents in $E^{op} \to \mathbf{Set}$.

11.6.11 Lemma *Let $H: E^{op} \to \mathbf{Set}$ be any functor, and let $G_0 = E[\_, \Omega]$. Then their exponent $H^{G_0}: E^{op} \to \mathbf{Set}$ is given, up to isomorphisms, by the following data:*

a. $H^{G_0}(e) = H(e \times \Omega)$
$H^{G_0}(f) = H(f \times id_\Omega)$;

b. $eval : H^{G_0} \times G_0 \to H$
$eval_e(m,f) = H(\langle id_e, f \rangle)(m)$, *for* $e \in Ob_E$, $m \in H(e \times \Omega)$, $f \in E[e, \Omega]$;

c. $\Lambda : Nat[F \times G_0, H] \cong Nat[F, H^{G_0}]$
$\Lambda(\tau)(e)(m) = \tau_{e \times \Omega}(F(fst)(m), snd)$,
where $\tau: F \times G_0 \to H$, $e \in Ob_E$, $m \in F(e)$, $fst: e \times \Omega \to e$, $snd: e \times \Omega \to \Omega$.

Proof We use the usual definition of exponents in the category of presheaves (see section 3.5) and prove that the one given above is equivalent up to isomorphisms. Remember that

$$H^F(e) = Nat[E[\_,e] \times F, H]$$
$$H^F(f: e' \to e)(\sigma) = \sigma \circ E[\_,f] \times id_F$$

where $E[\_,f]$ is the natural transformation from $E[\_,e']$ into $E[\_,e]$ defined by $E[\_,f] = f \circ \_$. When $F = G_0 = E[\_, \Omega]$, we can use Yoneda's lemma and have

$H^{G_0}(e) = Nat[E[\_,e] \times E[\_, \Omega], H] \cong Nat[E[\_,e \times \Omega], H] \cong H(e \times \Omega)$.

Let now $f \in E[e',e]$:

$H^{G_0}(f)(\sigma) = \sigma \circ E[\_,f] \times E[\_,id_\Omega] \cong \sigma \circ E[\_,f \times id_\Omega] \in Nat[E[\_,e' \times \Omega], H] \cong H(e' \times \Omega)$

Hence, the Yoneda isomorphism yields $H^{G_0}(f) \cong H(f \times id_\Omega)$.

Let us check that the above expressions for eval and Λ satisfy the equations for the exponents. We have to prove that $eval \circ \Lambda(\tau) \times id = \tau$ and $\Lambda(eval \circ h \times id) = h$; let $m \in F(e)$ and $f \in c_0(e)$:

$eval_e((\Lambda(\tau)_e \times id_e)(m,f)) =$

$= eval_e(\tau_{e \times \Omega}(F(fst)(m), snd), f)$	by def. of Λ
$= H(\langle id_e, f \rangle)(\tau_{e \times \Omega}(F(fst)(m), snd))$	by def. of eval
$= \tau_e(F(\langle id_e, f \rangle)(F(fst)(m)), \; snd \circ \langle id_e, f \rangle)$,	by naturality of τ
$= \tau_e(F(fst \circ \langle id_e, f \rangle)(m), f)$	for F functor
$= \tau_e(m, f)$	

$\Lambda(\text{eval} \circ h\times id)_e\ (m) =$

$= (\text{eval}_{e\times\Omega} \circ h_{e\times\Omega}\times id_{e\times\Omega})\ (F(fst)(m), snd)$ by def. of Λ

$= \text{eval}_{e\times\Omega}(\ h_{e\times\Omega}(F(fst)(m)), snd))$

$= H(<id_{e\times\Omega}, snd>)(\ h_{e\times\Omega}(F(fst)(m))\ )$ by def. of eval

$= H(<id_{e\times\Omega}, snd>)(H(fst\times id_\Omega)(h_e(m)))$ by naturality of h

$= H(\ fst\times id_\Omega \circ\ <id_{e\times\Omega}, snd>)(\ h_e(m)\ )$ for G functor

$= H(<fst,snd>)\ (\ h_e(m)\ )$

$= h_e(m).$ ♦

The following lemma exploits the results above in order to give an explicit definition of the constant internal functor $K: \underline{G} \to \underline{G}*$, whose right adjoint will give the depended product:

11.6.12 Lemma *The internal functor* $K = (k_0, k_1): \underline{G} \to \underline{G}*$ *of definition 11.4.3 is given in* $E^{op}\to$***Set*** *by*

$k_0(e) = G(fst)_{obj}$

$k_1(e) = G(fst)_{mor}$

where $fst: e\times\Omega \to e$ *in E.*

Proof Definition 11.4.3 gives the following for K

$k_0 = \Lambda(fst) : G_0 \to G_0{}^{G_0}$ where $fst: G_0\times G_0 \to G_0$ in $E^{op}\to$**Set**

$k_1 = \Lambda(fst) : G_1 \to G_1{}^{G_0}$ where $fst: G_1\times G_0 \to G_1$ in $E^{op}\to$**Set**.

Note first that, for $e\in Ob_E$, the components of the natural transformations fst above behave as the first projections. Now let $h\in c_0(e)$; then

$k_0(e)(h) = \Lambda(fst)(e)(h)$

$= fst_{e\times\Omega}(\ G_0(fst)(h), snd\ )$ by lemma 11.6.12, where $fst: e\times\Omega \to e$ in E

$= G_0(fst)(h)$

$= G(fst)(h).$

Analogously, for any $g\in c_1(e)$:

$k_1(e)(g) = \Lambda(fst)(e)(g)$

$= fst_{e\times\Omega}(\ c_1(fst)(g), snd\ )$

$= G_1(fst)(g)$

$= G(fst)(g).$ ♦

11.6.13 Theorem *Let* (E, G, Ω) *be an external model; then* $\underline{G}$ *is an internal model. Moreover, for any legal expression* Q *of* λ_2, *the external interpretation of* Q *in* (E, G, Ω) *coincides with the internal interpretation of* Q *in* $\underline{G}$; *that is, they are the same arrow in E.*

Proof $\underline{G}$ is Cartesian closed, by proposition 11.6.11. By definition of an external model, the functor G(fst) has a right adjoint $\forall: G^{\Omega} \to G$. In view of lemma 11.6.12, this is all we need for the proof.♦

By the previous theorem, and by the particularly simple way the category $\underline{G}$ is defined from the indexed category (E, G, Ω), every external model can be thought of as an internal model. We could even say that external models *are* the internal categories in the topos of presheaves that have the (contravariant) hom-functor as object of objects (and that have the required internal structure, of course). In this sense, *external models are less general than internal ones*, since they result from fixing some data in an internal model. Note that we have also obtained a posteriori a justification of the apparent simplicity of the external model. This is due to the choice of the well-known topos of presheaves as ambient category and of the hom-functor as canonical object of objects for the internal categories in this topos. This approach, though not fully general, allows a great simplification in the definitions of the involved exponents.

A final comparison between the two approaches is suggetsted by the following remark. Note first that any internal model in a presheaves topos "is" an indexed category; thus, one can think as well of a definition of indexed category model in which also the indexing functor is not representable. On the other hand, if the indexing functor is chosen to be representable, as in the external model, one may wonder why only the object of objects should enjoy this privileged condition. Note that if we suppose that also the object of morphisms is representable, i.e., $c_1 = E[\_, \Omega_1]$, then by the Yoneda embedding, we have an internal model $c = (\Omega, \Omega_1, ...)$ in E.

References The polymorphic lambda calculus was defined in Girard (1971) in his investigation of foundational problems in mathematics. Three years later it was reinvented by Reynolds (1974), who was mainly interested in the type structure of programming languages, testifying the relevance of this formalism for computer science. References to protoype programming languages, where polymorphism is formalized in terms of second order λ-calculi, and a recent application may be found in Cardelli and Longo (1990).

The model definition based on the internal approach is due to Moggi (1985). Unfortunately, since at that time there was no known concrete model that could be described "internally," his idea was never published, and for some years it remained known only to a restricted number of specialists and collaborators (see Hyland (1987)). Meanwhile a different and in a sense simpler notion of model based on indexed categories was proposed in Seely (1987). Both models are based on the idea in Lawvere (1970) of expressing logical quantifications by means of categorical adjunctions. Further discussions on categorical models of λ_2 may be found in Reynolds (1984), Bainbridge et al. (1987),

Hyland and Pitts (1987), Pitts (1987), Longo and Moggi (1988), Scedrov (1988), Reynolds and Plotkin (1988) and Meseguer (1988), among others.

This chapter is derived largely from Asperti and Martini (1989).

Chapter 12

EXAMPLES OF INTERNAL MODELS

In this chapter we present three examples of internal models: provable retractions inside a PER model, PER inside **ω-Set**, and PL-Categories inside their Grothendieck completion. The general categorical investigation of internal models developed in the previous chapter allows a deeper and unified analysis of the different aspects of these models, which brings, in our view, to in an original understanding of all of them.

12.1 Provable Retractions

This example continues the analysys of an internal model of retractions developed in chapter 7 (see examples 7.2.2.2, 7.3.9.1.)

12.1.1 Definition *$\lambda\beta(\eta)p$ is $\lambda\beta(\eta)$ plus a fresh constant p such that*

1. $pp = p$

2. $(px)\circ(px) = px$ *where* $s\circ r = \lambda x.s(rx)$

R. $$\frac{a \circ a = a}{pa = a}$$

1+2 imply $p \circ p = p$ and, hence, p is a retraction whose range contains exactly the provable retractions: range(r) = {a | $\lambda\beta\eta p \vdash$ ra = a }. The model constructed over this simple type-free theory is the "distilled" (syntactic) version of the various models in the literature based on categories of retractions: closures, finitary projections, etc. (see the references). It focuses on the general idea of "types as retractions" and, thus, clarifies the basic constructions in these models where "type is a type" (i.e., p itself is a retraction). This extension of type-free λ-calculus is provably non Church-Rosser, w.r.t. the obvious reduction relation, but it is consistent (see references). Consider then the term model of $\lambda\beta\eta p$: it is an applicative structure $\mathsf{T} = (\mathsf{T}, \cdot)$, where $[M]\in\mathsf{T}$ iff $[M] = \{ N \mid \lambda\beta\eta p \vdash M = N \}$ and $[M]\cdot[N] = [MN]$. Sometimes, for semplicity, we will make no distinction between a term and is equivalence class.

Of course, T is an extensional λ-model and, thus, by theorem 9.5.10, $\mathsf{T}^{\mathsf{T}} \cong \mathsf{T}$ in $\mathrm{PER}_{\mathsf{T}}$ (remember that by definition $[M]\,\mathsf{T}\,[N]$ iff $\lambda\beta\eta p \vdash M = N$) .

It is easy to show that this isomorphism is actually an identity, that is $\mathsf{T}^{\mathsf{T}} = \mathsf{T}$, indeed:

$$\begin{aligned}
[M]\ \mathsf{T}^{\mathsf{T}}\ [N] \quad &\text{iff} \quad \forall[P],\forall[Q]\ \ [P]\ \mathsf{T}\ [Q] \ \ \text{implies} \ \ [MP]\ \mathsf{T}\ [MQ] \\
&\text{iff} \quad \forall P,\forall Q \ \ \lambda\beta\eta p \vdash P = Q \ \ \text{implies} \ \ \lambda\beta\eta p \vdash MP = NQ \\
&\text{iff} \quad \lambda\beta\eta p \vdash Mx = Nx \\
&\text{iff} \quad \lambda\beta\eta p \vdash M = N \\
&\text{iff} \quad [M]\ \mathsf{T}\ [N]
\end{aligned}$$

As already pointed out, $\mathrm{PER}_{\mathsf{T}}$ is a Cartesian closed category with all finite limits.

Let $\mathrm{RET}_{\mathsf{T}} = (R_0, R_1, \mathrm{dom}, \mathrm{cod}, \mathrm{comp}, \mathrm{id}) \in \mathrm{Cat}(\mathrm{PER}_{\mathsf{T}})$ be the internal category of retractions on T. As we proved in the example 7.3.9.1, $\mathrm{RET}_{\mathsf{T}}$ is Cartesian closed; we want to prove that $\mathrm{RET}_{\mathsf{T}}$ yields an (internal) λ_2-model.

We start by giving the explicit definition of R_0 and R_1. Remember that in $\mathrm{PER}_{\mathsf{T}}$ the equalizer of a pair of morphisms $f,g: P \to Q$ is $(E, h: E \to P)$ where aEb iff (aPb and $f(a)Qg(b)$) and h is the injection from E to P. Note also that $\mathrm{dom}(E) = \mathrm{dom}(P) \cap \{a \mid f(a)\ Q\ g(a)\}$, and on this domain E coincides with P. Then it is easy to verify that $[M]\ R_0\ [N]$ iff ($[M]\ \mathsf{T}\ [N]$ and $\lambda\beta\eta p \vdash M \circ M = M$.) In other words R_0 coincides with T but is restricted to those terms which are provable retractions.

The domain of R_1 are ($\lambda\beta\eta p$-equivalence classes of) triples $\langle M, F, G\rangle$, where $\langle\ ,\ \rangle$ is the standard encoding of tuples in λ-calculus, F and G are provable retractions, and M is a morphism from F to G, that is, $\lambda\beta\eta p \vdash G \circ M \circ F = M$. Formally:

$$\mathrm{dom}(R_1) = \{\ [\langle M, F, G\rangle] \mid F \circ F =_{\lambda\beta\eta p} F\ ;\ \ G \circ G =_{\lambda\beta\eta p} G\ ;\ \ G \circ M \circ F =_{\lambda\beta\eta p} M\ \},$$

$$[\langle M, F, G\rangle]\ R_1\ [\langle M', F', G'\rangle] \quad \text{iff} \quad [\langle M, F, G\rangle]\ \mathsf{T}\ [\langle M', F', G'\rangle]$$

Note moreover that $[\langle M, F, G\rangle]\ \mathsf{T}\ [\langle M', F', G'\rangle]$ iff $M =_{\lambda\beta\eta p} M'$, $F =_{\lambda\beta\eta p} F'$, $G =_{\lambda\beta\eta p} G'$.

Let $\mathrm{RET}_{\mathsf{T}}{}^{*} = (R_0{}^{R_0}, R_1{}^{R_0}, \mathrm{dom}^{*}, \mathrm{cod}^{*}, \mathrm{comp}^{*}, \mathrm{id}^{*})$ be as in definition 11.4.2. We must prove that the costant functor $K: \mathrm{RET}_{\mathsf{T}} \to \mathrm{RET}_{\mathsf{T}}{}^{*}$ has a right adjoint $\forall = (\forall_0: R_0{}^{R_0} \to R_0, \forall_1: R_1{}^{R_0} \to R_1)$. $\forall_0$ is the function realized by $\lambda F \lambda z t.F(pt)(z(pt))$. Thus, we need to show that

1. if F is in the domain of $R_0{}^{R_0}$ then $\forall_0(F)$ is in the domain of R_0 ;
2. $\forall_0$ takes elements that are equivalent in $R_0{}^{R_0}$ to elements that are equivalent in R_0.

As for (1) we have

$$\begin{aligned}
\forall_0 F \circ \forall_0 F &= \lambda x(\ \lambda z t.F(pt)(z(pt))\ )(\ \lambda t.F(pt)(x(pt))\ ) \\
&= \lambda x(\ \lambda t.F(pt)(\ \lambda t.F(pt)(x(pt))\ (pt))\ ) \\
&= \lambda x(\ \lambda t.F(pt)(\ F(pt)(x(pt)))\ ) \\
&= \lambda x \lambda t.\ F(pt)(x(pt)) \qquad \text{because } F(pt) \text{ is a retraction by hypothesis} \\
&= \forall_0 F
\end{aligned}$$

(2) is evident.

An element in $R_1{}^{R_0}$ is a triple $\langle T, F, G\rangle$, where F and G are in the domain of $R_0{}^{R_0}$, T is in the domain of T^{R_0}, and for every retraction M, one has $TM = GM \circ TM \circ FM$.

$\forall_1$ is the function that takes (an equivalence class in R_1^{R0} of) a triple $< T, F, G >$ in (the equivalence class in R_1 of) the triple $< \lambda t\lambda m .T(pm)(\forall_0 Ft), \forall_0(F), \forall_0(G) >$.

Note that $\lambda t\lambda m .T(pm)(\forall_0 Ft))$ is a morphism from the retraction $\forall_0 F$ to the retraction $\forall_0 G$; indeed,

$$\begin{aligned}
&\forall_0 G \circ \lambda t\lambda m .T(pm)(\forall_0 Ft) \circ \forall_0 F = \\
&\quad = \lambda z.\forall_0 G(\lambda m .T(pm)(\forall_0 Fz)) \\
&\quad = \lambda z. (\lambda x\lambda t.G(pt)(x(pt))) (\lambda m .T(pm)(\forall_0 Fz)) \\
&\quad = \lambda z. \lambda t.G(pt)((\lambda m .T(pm)(\forall_0 Fz)) (pt)) \\
&\quad = \lambda z. \lambda t.G(pt)((T(pt)(\forall_0 Fz)) \\
&\quad = \lambda z. \lambda t.T(pt)(\forall_0 Fz) \qquad \text{because } G(pt) \circ T(pt) = T(pt) \text{ by hypothesis}
\end{aligned}$$

It should be clear that $\forall_1$ is indeed a morphism in PER_{T} its realizer is simply obtained by abstraction on $< T, F, G >$.

Given a variable type G in R_0^{R0} and a type N in R_0, the projection $proj_G(N)$ from $\forall_0 G$ to $G(N)$ is realized by the term $\lambda x.xN$. Indeed, if S has type $\forall_0 G$ that is, $S = (\forall_0 G)S$, then $SN = \forall_0 GSN = G(N)(SN)$, and thus SN has type $G(N)$.

We define now the isomorphism Δ of the adjunction.

Let $< T, \lambda x.M, G >$ be an element in R_1^{R0}, where M is a retraction and $\lambda x.M = K_0(M)$.

Define then $\Delta_{M,G}(< T, \lambda x.M, G >) = < \lambda t\lambda m .T(pm)t, M, \forall_0 G >$.

Conversely, given $< S, M, \forall_0(G) >$ in R_1, define $\Delta_{M,G}^{-1}(< S, M, \forall_0(G) >) = < \lambda m\lambda t.Stm, \lambda x.M, G >$. Note that $\lambda m.\lambda t.Stm = \lambda m.(\lambda x.xm \circ S) = \lambda m.(proj_G(m) \circ S)$. Thus, for every retraction N, one has

$$G(N) \circ \lambda t.StN \circ M = proj_G(N) \circ S \circ M = G(N) \circ proj_G(N) \circ S \circ M = \lambda t.StN .$$

Moreover,

$$\begin{aligned}
&\Delta_{M,G}^{-1}(\Delta_{M,G}(< T, \lambda x.M, G >)) = \\
&\qquad = \Delta_{M,G}^{-1}(< \lambda t\lambda m .T(pm)t , M, \forall_0(G) >) \\
&\qquad = < \lambda m\lambda t.T(pm)t, \lambda x.M, G > \\
&\qquad = < \lambda m.T(pm), \lambda x.M, G >
\end{aligned}$$

and clearly T and $\lambda m.T(pm)$ are equivalent in T^{R0}.

As for the converse, note first that if $S = \forall_0(G) \circ S$, then

$$St(pm) = \forall_0(G)(St)(pm) = (\lambda m.G(pm)(St(pm)))(pm) = G(pm)(St(pm)) = Stm$$

Thus:

$$\begin{aligned}
&\Delta_{M,G}(\Delta_{M,G}^{-1}(< S, M, \forall_0(G) >)) = \\
&\qquad = \Delta_{M,G}(< \lambda m\lambda t.Stm, \lambda x.M, G >) \\
&\qquad = < \lambda t\lambda m .St(pm) , M, \forall_0 G > \\
&\qquad = < \lambda t\lambda m .Stm , M, \forall_0 G > \\
&\qquad = < S, M, \forall_0 G >
\end{aligned}$$

Define then

$$\Delta (< M,G, < T, \lambda x.M, G > >) = < M, G, < \lambda t \lambda m .T(pm)t , M, \forall_0 G > >$$

and

$$\Delta^{-1}(< M,G, < S, M, \forall_0 G > >) = <M,G, < \lambda m \lambda t.Stm, \lambda x.M, G > >.$$

12.1.2 Remark It is possible to give an elegant categorical characterisation of the models of $\lambda\beta(\eta)p$. Let A be a reflexive object ($A^A \cong A$) in some category C, and let $RET_A = (R_0, R_1,$ dom, cod, comp, id) be the internal category of retractions on A in PER_A. Let $\xi: R_0 \to A^A$ be as usual the equalizer of the identity and $\lambda f.\ f \circ f$. All we need to turn RET_A in a model for $\lambda\beta(\eta)p$ is that there exist $p: A^A \to R_0$ such that R_0 is a retract of A^A via (ξ,p). Then, in a sense, since R_0 represents all the retractions on A, we can say that the collection of objects of RET_A is itself an object of RET_A: more formally, that RET_A is internal to RET_A w.r.t. PER_A. We have thus the following facts:

1. ($A^A < A$ in a CCC C *and* RET_A internal to RET_A with respect to PER_A)
   imply $A \models \lambda\beta p$.
2. $A \models \lambda\beta p$ *implies*
   ($A^A < A$ in PER_A *and* RET_A internal to RET_A w.r.t. PER_A).

The reader should complete the details for exercise.

12.2 PER inside ω-Set

This example continues our presentation of **PER** an an internal category **M** of **ω-Set** (see examples 7.2.7 and 7.3.9.2, where it is shown that the construction gives also an internal model.)

Let $\mathbf{M}*$ be defined as $\mathbf{c}*$ in 11.4.2. We next define an internal right adjoint to the constant functor $K : \mathbf{M} \to \mathbf{M}*$, that by def. 11.4.4 will complete the construction of a model for λ_2.

Again, we shall take advantage of the wise blend of set-theory and computability on which **ω-Set** is based, in order to avoid the most formal details.

The following lemma motivates the definition of $\forall$ below.

12.2.2 Lemma *If* $<\tau,F,G> \in \boldsymbol{\omega\text{-}Set[M_0,M_1]}$ *then* $\exists n\ \forall A \in \boldsymbol{M_0}\ \tau(A) = \{n\}_{F(A)\to G(A)}$.

Proof. Suppose that $m \Vdash <\tau,F,G>$. Since $0 \Vdash A$ for any $A \in \mathbf{PER}$, then take $n = m \cdot 0$ and observe that $n \Vdash <\tau(A),F(A),G(A)>$. Thus $\tau(A) = \{n\}_{F(A)\to G(A)}$. ♦

As for the definition of $\forall_0$, observe that the intersection of any collection $\{A_i\}_{i\in I}$ of objects in **PER** is still in **PER**, by viewing them as sets of pairs (of numbers). That is, set

$$n \ (\cap_{i\in I} A_i)\ m \quad \text{iff} \quad \forall i \in I \ (n\ A_i\ m).$$

12.2.3 Definition

1. $\forall_O : [\mathbf{M_o} \to \mathbf{M_o}] \to \mathbf{M_o}$*, is given by* $\forall_O(F) = \cap_{A \in \mathbf{Mo}} F(A)$*,*

2. $\forall_1 : [\mathbf{M_o} \to \mathbf{M_1}] \to \mathbf{M_1}$ *is defined as follows. If* $m \Vdash \langle \tau, F, G \rangle \in \boldsymbol{\omega}\textbf{-Set}[\mathbf{M_o}, \mathbf{M_1}]$*, set* $\forall_1(\langle \tau, F, G \rangle) = \langle \{m \cdot 0\} \forall_O(F) \to \forall_O(G) , \forall_O(F), \forall_O(G) \rangle$.

12.2.4 Proposition. $\forall$ *in definition 12.2.3 is well defined. In particular,* $\forall_1$ *is realized by any number p such that , for all m ,* $p \cdot m = m \cdot 0$.

Proof. By definition, if $F : \mathbf{M_o} \to \mathbf{M_o}$, then $\forall_o(F)$ is in $\mathbf{M_o}$. $\forall_o$ is realized by (any index of) any total recursive function. By lemma 12.2.2, $\forall_1$ is well defined as its definition does not depend on the choice of the realizer m for $\langle F, \tau, G \rangle$. Therefore we only need to show that $\forall_1$ is realized by p. Namely, that if

1. $m_1, m_2 \Vdash \langle \tau, F, G \rangle$
2. $n_1 \; \forall_o(F) \; n_2$
3. $A \in \mathbf{M_o}$

then one has $(p \cdot m_1 \cdot n_1) \; G(A) \; (p \cdot m_2 \cdot n_2)$. Indeed,

4. $p \cdot m_1 = m_1 \cdot 0 \; (F(A) \to G(A)) \; m_2 \cdot 0 = p \cdot m_2$, by (1)
5. $n_1 \; F(A) \; n_2$, by (2) and (3),

and thus $(p \cdot m_1 \cdot n_1) \; G(A) \; (p \cdot m_2 \cdot n_2)$ by (4), (5) and the definition of $(F(A) \to G(A)) \in \mathbf{M_o}$. ♦

12.2.5 Remark. It should be clear that lemma 12.2.2 is a very simple but crucial lemma. Note first that the morphisms in $\boldsymbol{\omega}\textbf{-Set}[\mathbf{M_o}, \mathbf{M_1}]$ are described as triples, $\langle \tau, F, G \rangle$, where $\tau: \mathbf{M_o} \to \cup_{A \in \mathbf{Mo}} Q(F(A) \to G(A))$ is such that $\tau(A) \in Q(F(A) \to G(A))$ and $F, G: \mathbf{M_o} \to \mathbf{M_o}$ give the source and target of $\tau(A)$. This is a sound description, since $\mathbf{M}* = ([\mathbf{M_o} \to \mathbf{M_o}], [\mathbf{M_1} \to \mathbf{M_1}], ...)$ may be viewed as the internal category of functors from the discrete category, whose object of objects is $\mathbf{M_o}$, to $\mathbf{M}$. Thus $[\mathbf{M_o} \to \mathbf{M_1}]$ or $\boldsymbol{\omega}\textbf{-Set}[\mathbf{M_o}, \mathbf{M_1}]$ are internal natural transformations. By lemma 12.2.2, now, there is a uniform n which realizes $\tau(A)$ for all A. In a sense these internal natural transformations are "almost" constant maps and only depend on the source and target objects.

We need now to prove that there exists an internal natural isomorphism Δ, which gives the adjunction between $\forall$ and K. We can use the simplicity of the intended universe and perform the construction directly. As in the proof of the internal Cartesian closure of $\mathbf{M}$, the point is to show that the functional dependencies, usually implicit in the external world (or given by "indexes": recall $a, b \Vdash \Lambda_{a,b}$), can be turned into internal constructions. Once more, this will be straightforward to check within $\boldsymbol{\omega}\textbf{-Set}$, as the realizers for the natural isomorphism and its inverse will not depend on their "indexes."

Let K, with components k_o and k_1, be the internal constant functor in definition 11.4.3.

12.2.6 Definition. *For* $m \Vdash\!\!- < \tau,k_O(B),G> \in \boldsymbol{\omega\text{-}Set[M_0,M_1]}$, *set*

$$\Delta_{B,G}(< \tau,k_O(B),G>) = < \{m\cdot 0\}_{B\to \forall O(G)} , B, \forall_O(G)> \in \boldsymbol{M_1} .$$

and $$\Delta^{-1}{}_{B,G}(<\{m\}_{B\to \forall O(G)} , B, \forall_O(G)>)$$
$$= < \underline{\lambda} X\in \boldsymbol{M_0}.\{m\}_{B\to G(X)} , k_O(B), G> \in \boldsymbol{\omega\text{-}Set[M_0,M_1]}.$$

12.2.7 Proposition. *$\Delta_{B,G}$ and $\Delta^{-1}{}_{B,G}$ in 12.2.6 are well defined, for all $B\in \boldsymbol{M_0}$. and $G: \boldsymbol{M_0}\to \boldsymbol{M_0}$ $(= [\boldsymbol{M_0} \to \boldsymbol{M_0}])$. Moreover, they are realized by p and k (respectively), such that $p\cdot m = m\cdot 0$ and $k\cdot m\cdot n = m$.*

Proof: By lemma 12.2.2, $\Delta_{B,G}$ does not depend on the choice of the particular realizer m. As for $\Delta^{-1}{}_{B,G}$, note that, if $m(B\to \forall_O(G))m$, then $\forall n,q\ (\ nBq \Rightarrow m\cdot n\ (\cap_{A\in \mathbf{M_0}} G(A))\ n\cdot q)$, by definition of $\forall_O$, and hence $m\cdot n\ G(A)\ n\cdot q$, for all $A\in \mathbf{M_0}$. Therefore, $m\ (B \to G(A))\ m$ for all $A\in \mathbf{M_0}$ and $\Delta^{-1}{}_{B,G}$ too is well defined. By definition, they are uniformly realized by p and k as above. In particular, p and k compute $\Delta_{B,G}$ and $\Delta^{-1}{}_{B,G}$ for all B and G. ♦

12.3 PL-Categories Inside Their Groethendiek Completion

This example show another way for obtaining an internal model from a PL-category, different from the internalization process of chapter 11.

12.3.1 Definition. *Given any E-indexed category G, define the category $\int G$, the* ***Grothendieck completion of G,*** *having as objects the pairs (e, σ) with $e\in Obj_E$ and $\sigma\in Obj_{G(e)}$ and as morphisms pairs (α, f) such that:*

$$(\alpha, f)\in \int G[(e, \sigma), (e', \tau)] \quad iff \quad \alpha\in E[e,e'] \ and \ f\in G(e)[\sigma, G(\alpha)(\tau)].$$

The identity of (e, σ) is (id_e, id_σ); the composition of $(\alpha, f)\in \int\mathbf{G}[(e,\sigma),(e',\tau)]$ and $(\beta, g)\in \int\mathbf{G}[(e',\tau), (d,\rho)]$ is $(\beta, g)\circ(\alpha, f) = (\beta\circ\alpha, G(\alpha)(g)\circ f)$.

Let (E, G, Ω) be an external model (that is an indexed category with the additional requirements on functor G in 12.2.1); then the Grothendieck completion $\int\mathbf{G}$ assumes a particularly simple form

Objects: $(e, \sigma)\in \int\mathbf{G}$ iff $\sigma\in Obj_{G(e)} = E[e, \Omega]$

(hence we can identify objects of $\int G$ with arrows $\sigma: e\to\Omega$)

Morphisms: $(\alpha,f)\in \int\mathbf{G}[\sigma:e\to\Omega,\tau:e'\to\Omega]$ iff

$$\alpha\in E[e,e'] \ and \ f\in G(e)[\sigma,G(\alpha)(\tau)] = G(e)[\sigma,\tau\circ\alpha].$$

The point is that $\int\mathbf{G}$ contains an internal category which, in a sense, internalizes the external model (E, G, Ω).

Warning We are here forcing our terminology, since the category $\int\mathbf{G}$ does *not* need to have all finite limits, at least not in general, and hence we could not speak of "internal categories in $\int\mathbf{G}$." However, all the needed pullback diagrams exist in $\int\mathbf{G}$, as pointed out below.

The internal category $\Gamma = (c_0, c_1, DOM, COD, COMP, ID) \in Cat(\int\mathbf{G})$ is defined as follows:

$c_0 = t_{G(\Omega)} : \Omega \to \Omega$ — the terminal object in $G(\Omega)$

$c_1 = [,]_0 : \Omega\times\Omega \to \Omega$ — $[,]_0$ is given in lemma 11.2.3

$DOM = (fst, !)$ — ! is the unique arrow from c_1 to $c_0 \circ fst$ in $G(\Omega\times\Omega)$

$COD = (snd, !)$

$ID = (\langle id_\Omega, id_\Omega\rangle, \Lambda_{G(\Omega)}(snd))$ — $snd \in G(\Omega)[t_{G(\Omega)}\times id_\Omega, id_\Omega]$, hence $\Lambda(snd) \in G(\Omega)[t_{G(\Omega)}, id_\Omega \to id_\Omega]$, where $id_\Omega \to id_\Omega$

The situation for COMP is more delicate, since $\int\mathbf{G}$ does not have all finite limits. However, the pullback of $DOM: c_1 \to c_0$, $COD: c_1 \to c_0$ does exist, and it is given by

$c_2 = (p_2 \to p_3)\times(p_1 \to p_2)$,

where product and exponents are in the fiber $G(\Omega\times\Omega\times\Omega)$ and $p_i \in E[\Omega\times\Omega\times\Omega, \Omega]$; the pullback projections $\prod_1, \prod_2 : c_2 \to c_1$ are

$\prod_1 = (\langle p_2, p_3\rangle, id)$

$\prod_2 = (\langle p_1, p_2\rangle, id)$.

In order to define COMP, remember that in any CCC, given three objects A, B, C, there exists a morphism $cmp \in Hom[C^B\times B^A, C^A]$ that internalizes the composition, namely

$cmp = \Lambda(eval \circ id\times eval \circ p)$ — for $p: (C^B\times B^A)\times A \to C^B\times(B^A\times A)$ an isomorphism.

Define then

$COMP = (\langle p_1, p_3\rangle, cmp)$ — for $cmp \in G(\Omega\times\Omega\times\Omega)[c_2, p_1 \to p_3]$

(recall that, in $G(\Omega\times\Omega\times\Omega)$, $p_1 \to p_3 = [,]_0 \circ \langle p_1, p_3\rangle$).

The verification that these data define an internal category is straightforward.

We now prove that if (E, G, Ω) is an external model, than $\int\mathbf{G}$ is Cartesian closed.

The terminal object is given by $t_{G(t)}$, the terminal object of the local category G(t) (for t terminal in E). As for products, they can be defined by using the Cartesian structure of the local categories: $(f:e\to\Omega)\times_{\int G}(g:d\to\Omega) := x_0 \circ f\times_E g$, where $x_0: \Omega\times\Omega\to\Omega$ is the arrow obtained in lemma 11.2.3; hence $f \times_{\int G} g = (f \circ fst)\times_{G(e\times d)}(g \circ snd)$. Projections are obtained as follows. We need FST: $c_0\times c_0 \to c_1$, where $FST = (\alpha, f)$ for some $\alpha:\Omega\times\Omega\to\Omega\times\Omega$ and $f \in G(\Omega\times\Omega)[c_0\times c_0, c_1 \circ \alpha]$. As for α, which intuitively takes a pair of objects (σ, τ) to $(\sigma\times\tau, \sigma)$, we can take $\alpha = \langle x_0, fst\rangle$: $\Omega\times\Omega\to\Omega\times\Omega$.

In order to define $f \in G(\Omega\times\Omega)[t_{G(\Omega\times\Omega)}, x_0 \to fst]$, we can use again the Cartesian closed structure of $G(\Omega\times\Omega)$ and the fact that, by lemma 11.2.3, one has

$$x_0 = x_0 \circ id = x_0 \circ \langle fst, snd\rangle = fst \times_{G(\Omega\times\Omega)} snd.$$

Define then $f = \Lambda(fst)$ where $fst \in G(\Omega\times\Omega)[fst\times snd, fst]$. SND is defined analogously.

It remains to give the pairing isomorphism $\ll,\gg$; note first of all that the required pullbacks (see definition 7.3.6, and also appendix A at the end of chapter 7) are given by the following data:

$X = [,]_0 \times_{\int G} [,]_0 \circ \langle id\times fst, id\times snd\rangle : \Omega\times(\Omega\times\Omega)\to\Omega$
$\prod_X : X \to c_1\times c_1$ (that is $X \to [,]_0 \times_{\int G} [,]_0$)
$\prod_X = (\langle id\times fst, id\times snd\rangle, id)$
$\rho : X \to c_0\times c_0\times c_0$ (that is $X \to t_{G(\Omega\times\Omega\times\Omega)}$)
$\rho = (id, !)$

$Y = [,]_0 \circ id\times x_0 : \Omega\times(\Omega\times\Omega)\to\Omega$
$\prod_Y : Y \to c_1$ (that is $Y \to [,]_0$)
$\prod_Y = (id\times x_0, id)$
$\rho' : Y \to t_{G(\Omega\times\Omega\times\Omega)}$
$\rho' = (id, !)$

Observe now that, as we are dealing with objects of $G(\Omega\times(\Omega\times\Omega))$, by interpreting all products and exponentials locally in $G(\Omega\times(\Omega\times\Omega))$), one has

$X = (fst \to fst \circ snd)\times(fst \to snd \circ snd)$
$Y = fst \to ((fst \circ snd)\times(snd \circ snd))$

and hence $X \cong Y$, since $G(\Omega\times(\Omega\times\Omega))$ is Cartesian closed.

The isomorphism $\ll,\gg: X\to Y$ (in $\int\mathbf{G}$) is then given by $\ll,\gg = (id,f)$, where f is the isomorphism between X and Y in $G(\Omega\times(\Omega\times\Omega))$. The required equations are now easily verified.

Let us now come to the exponents $\Rightarrow_0 = ([,]_0 , !) : c_0\times c_0\to c_0$, where $[,]_0$ is given again by lemma 11.2.3.

EVAL : $c_0\times c_0 \to c_1$ is given by a pair: EVAL = (α,f), for $\alpha:\Omega\times\Omega\to\Omega\times\Omega$ and $f\in G(\Omega\times\Omega)[c_0\times c_0, c_1\circ\alpha]$. As for α, whose intuitive content is to send a pair of objects (σ, τ) into $(\tau^\sigma\times\sigma, \tau)$, we can set $\alpha = \langle x_0 \circ \langle [,]_0, fst\rangle, snd\rangle$.

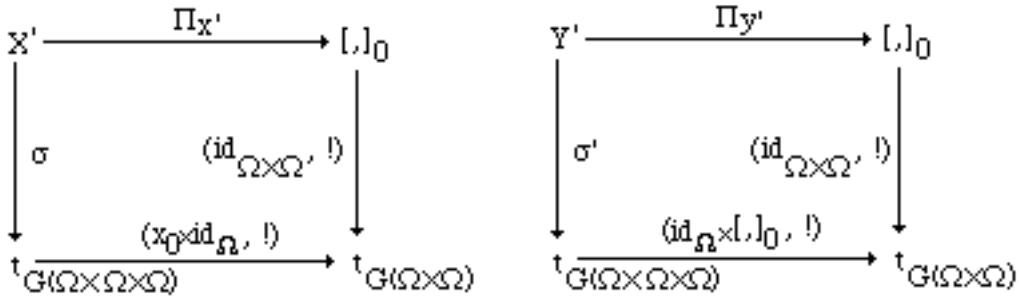
We need now $f\in G(\Omega\times\Omega)[t_{G(\Omega\times\Omega)}, [,]_0 \circ \langle x_0 \circ \langle [,]_0, fst\rangle, snd\rangle]$; observe that

$[,]_0 \circ \langle x_0 \circ \langle [,]_0, fst\rangle, snd\rangle =$
$= (x_0 \circ \langle [,]_0, fst\rangle) \to snd$ in $G(\Omega\times\Omega)$
$= ([,]_0 \times fst) \to snd$
$= ((fst\to snd) \times fst) \to snd.$

Take then $eval_{fst,snd}\in G(\Omega\times\Omega)[(fst\to snd) \times fst, snd]$ and set

$f = \Lambda(eval_{fst,snd})\in G(\Omega\times\Omega)[t_{G(\Omega\times\Omega)}, ((fst\to snd) \times fst) \to snd]$.

Before giving the isomorphism Λ, we need to express the pullback diagrams of theorem 7.3.7 (see also appendix A). They can be instantiated as



where:

$X' = [,]_0 \circ (x_0 \times id_\Omega)$ $\qquad$ $Y' = [,]_0 \circ (id_\Omega \times [,]_0)$

$\prod X' = (x_0 \times id_\Omega, id)$ $\qquad$ $\prod Y' = (id_\Omega \times [,]_0, id)$

$\sigma = (id_{\Omega\times\Omega\times\Omega}, !)$ $\qquad$ $\sigma' = (id_{\Omega\times\Omega\times\Omega}, !)$

As before, observe that

$X' = fst \times (fst \circ snd) \to (snd \circ snd)$

$Y' = fst \to ((fst \circ snd) \to (snd \circ snd))$.

Therefore $X' \cong Y'$, by the Cartesian closure of $G(\Omega\times(\Omega\times\Omega))$.

The internal Cartesian closed category $\Gamma \in Cat(\int G)$ is actually an internal model when (E, G, Ω) is an external one. In order to define the required adjunction, we need first to show that we can indeed construct the internal category $\Gamma*$; that is, the exponents $c_0{}^{c}0$ and $c_1{}^{c}0$ exist in $\int G$ (as for the limits, not all the exponents exist in $\int G$).

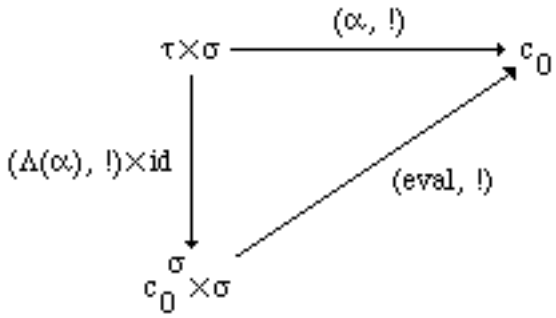
12.3.2 Lemma

i. For any object e of E, $c_0{}^{t}G(e)$ exists in $\int G$ and it is given by $t_{G(\Omega^e)}$.

ii.For any object e of E, $c_1{}^{t}G(e)$ exists in $\int G$ and it is given by $\forall([,]_0 \circ eval)$.

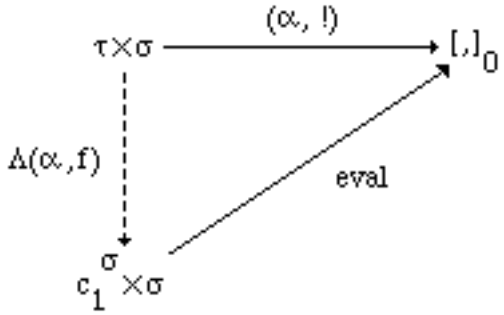
Proof Set $\sigma = t_{G(e)}$.

i. Note first that in $\int G$, for any e' and τ, every arrow $(\beta,g):\tau \to t_{G(e')}$ has $g = !$. We must then show that, for $c_0{}^{\sigma} = t_{G(\Omega^e)}$, the following diagram commutes, for any object τ and arrow $(\alpha,!)$:



which is immediate.

ii. Let $\tau: e' \to \Omega$ be any object of $\int \mathbf{G}$; we are looking for the unique $\Lambda(\alpha, f)$ such that the following diagram commutes:



Since $\alpha: e' \times e \to \Omega \times \Omega$, if we set $\Lambda(\alpha,f) = (\underline{\alpha},\underline{f})$, a natural choice for $\underline{\alpha}$ is $\underline{\alpha} = \Lambda(\alpha): e' \to (\Omega \times \Omega)^e$. Moreover, for $(\alpha, f): \tau \times \sigma \to [,]_0$, we have the following:

$f \in G(e' \times e)[\tau \times \sigma, [,]_0 \circ \alpha\ ] \cong$

$\cong G(e' \times e)[\tau \circ fst, [,]_0 \circ \alpha]$ by $\sigma = t_{G(e)}$ and the definition of product in $\int \mathbf{G}$

$\cong G(e')[\tau \circ fst, \forall([,]_0 \circ \alpha)]$ via the isomorphism Δ of the adjuction giving the external model

$\cong G(e')[\tau \circ fst, \forall([,]_0 \circ eval) \circ \Lambda(\alpha)]$ by naturality of $\forall$.

We can then define

$c_1{}^{\sigma} = \forall([,]_0 \circ eval) : (\Omega \times \Omega)^e \to \Omega$

and take $\underline{f} = \Delta(f)$; the previous diagram then commutes for $eval = (eval, proj_{c0}\sigma)$.

We can eventually give the data for the adjunction

$\forall_0 : c_0{}^{c0} \to c_0$

$\forall_0 = (\forall_0 : \Omega^{\Omega} \to \Omega, !)$.

As for $PROJ = (\alpha, f)$: $c_0{}^{c0} \to c_1{}^{c0}$, we need $\alpha : \Omega^{\Omega} \to (\Omega \times \Omega)^{\Omega}$ and $f \in G(\Omega^{\Omega})[c_0{}^{c0}, c_1{}^{c0} \circ \alpha]$. For $eval : \Omega^{\Omega} \times \Omega \to \Omega$ and $fst: \Omega^{\Omega} \times \Omega \to \Omega^{\Omega}$, set $\alpha = \Lambda(<\forall_0 \circ fst, eval>)$. As for f, note first that

$c_0{}^{c0} = t_{G(\Omega^{\Omega})}$

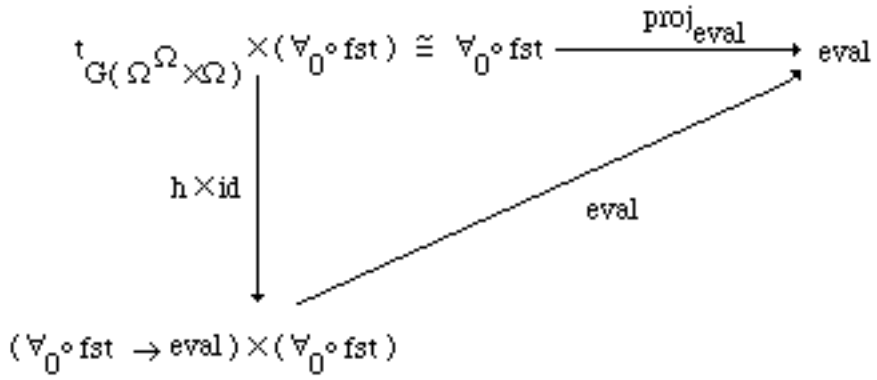
and

$c_1{}^{c0} \circ \alpha = \forall([,]_0 \circ eval) \circ \Lambda(<\forall_0 \circ fst, eval>)$

$= \forall([,]_0 \circ <\forall_0 \circ fst, eval>)$ by naturality of $\forall$

$= \forall((\forall_0 \circ fst) \to eval)$ by the usual isomorphims.

We are then looking for $f \in G(\Omega^{\Omega})[t_{G(\Omega^{\Omega})}, \forall((\forall_0 \circ fst) \to eval)\ ]$; as we did in the proof of the previous lemma, let us look for an arrow $h \in G(\Omega^{\Omega} \times \Omega)[t_{G(\Omega^{\Omega})} \circ fst, (\forall_0 \circ fst) \to eval]$. Then the f we need will then be $\Delta(h)$. Observe now that $proj_{eval} \in G(\Omega^{\Omega} \times \Omega)[\forall_0 \circ fst, eval]$; this immediately gives the required h in the following diagram:



In conclusion, the Grothendieck completion provides yet another example of the general categorical construction of model in chapter 11. Moreover, it gives further insight into the external versus internal approach by giving a setting where the external models may be viewed as internal constructions.

References The examples in this chapter may be found, in part, in the many papers mentioned at the end of the previous chapter. Since Moggi's hint for the small completeness of the **PER** construction, in particular, these models have been thoroughly explored by many authors. The three sections above develop the ideas in Amadio and Longo (1986) and in Longo and Moggi (1988), and follow Asperti and Martini (1989), respectively. By this, the first two sections extend to higher order the understanding of simple types as partial equivalence relations and retractions in Scott (1976). The consistency of $\lambda\beta\eta p$ is shown in Berardi (1988).

BIBLIOGRAPHY

Adachi T. [1983] "A categorical characterization of lambda-calculus models" Dept. of Info. Sci. Tokyo Inst. Tech., n. C-49.

Amadio R., Bruce K., Longo G. [1986] "The finitary projections model and the solution of higher order domain equations" **IEEE Conference on Logic in Computer Science**, **LICS'86** Boston.

Amadio R., Longo G. [1986] "Type free compiling of parametric types", IFIP Conference, in **Formal Description of Programming Concepts-III,** M. Wirsing (ed.), Ebberup (DK), North-Holland, 1987.

Arbib M., Manes E. [1975] **Arrows Structures and functors: The Categorical Imperative**, Academic Press, London.

Arbib M., Manes E. [1975a] "Adjoint Machines, State-Behavior Machines and Duality" **J. Pure and Applied Algebra**, 6, (313-344).

Asperti A. [1988] "Stability and Computability in Coherent domains" **Info&Comp**. (to appear).

Asperti A., Longo G. [1987] "Categories of partial morphism and the relation between type-structures" **Semester on Theory of Computation**, Banach Center Publications, vol. 21, Warsaw.

Asperti A., Martini S. [1989] "Categorical models of polimorphism" Note interne, Dip. di Informatica, Università di Pisa.

Bainbridge E., Freyd P., Scedrov A., Scott P.J. [1990] "Functorial Polymorphism" **Theoretical Comp. Sci. 70**, 35-64.

Barendregt H.P., Koymans K. [1980] **Comparing some classes of lambda calculus models,** in Hindley and Seldin, (287-302).

Barendregt H. [1984] **The Lambda Calculus; its syntax and semantics**, Revised and expanded edition, North Holland.

Barr M. [1979] ***-Autonomous Categories**, Springer Lecture Notes in Mathematics 752, Berlin.

Barr M. [1990] "*-Autonomous Categories and Linear Logic", **Mathematical Structures in Computer Science**, to appear.

Barr M., Wells C.F. [1985] "Toposes and Theories" **Gundlehren der Mathematiscgen Wissenschaffen**, 273, Springer-Verlag, New York.

Beeson M. [1980] **Foundations of Constructive Mathematics**, Springer -Verlag.

Bénabou J. [1985] "Fibered Categories and the Foundations of Naive Category Theory" **J. Symbolic Logic 50**, 1, (1o-37).

Berger U. [1986] **Berechenbarkeit in Hoheren Typen Nach Einem Ansatz Von Ju.L. Ersov**, dissertation Munchen.

Berry G. [1978] **Stable models of typed λ-calculi**. Proc. 5th ICALP. **LNCS 62**, (72-89).

Berry G. [1979] "Some Syntactic and categorical Constructions of Lambda Calculus Models" **INRIA**, Valbonne.

Berry G. [1981] "On the definition of lambda-calculus models" Proc. Int. Coll. on **Formalization of Programming Concepts, LNCS 107**, (218-230).

Bruce K., Di Cosmo R., Longo G. [1990] "Provable Isomorphisms of Types" **Symposium on Symbolic Computation**, E.T.H., Zuerich (CH), March 1990; *to appear*.

Bruce K., Longo G. [1988] "Modest models for inheritance and explicit polymorphism" CMU report CS-88-126, **IEEE Conference on Logic in Computer Science**, **LICS '88,** Edinburgh, (**Info.&Comp**., to appear).

Carboni A., Freyd P., Scedrov A. [1988] "A categorical approach to realizability and polymorphic types" **3rd ACM Symp. on Math. Found of Language Semantics**, New Orleans, Main (ed), Springer LNCS 298, (23-42).

Cartmell J. [1978] "Generalised Algebraic Theories and Contextual Categories", PhD Thesis, Oxford.

Coquand T., Gunter C., Winskel G. [1989] "Domain theoretic models of polymorphism" **Information and Computation 81**,123--167.

Cousineau G., Curien P.L., Mauny M. [1985] "The categorical Abstract machine" in **Functional Programming Languages and Computer Architecture**, J.P. Jouannaud (ed.), Lecture Notes in Computer Science 201, (Springer, Berlin), (130-139).

Curien P.L. [1986] **Categorical Combinators and Functional Programming**, Pitman.

Curien P.L., Obtulowicz A. [1988] "Partiality, Cartesian Closedness and Toposes", **Info&Comp**., Vol. 80, (50-95).

Degano P., Meseguer J., Montanari U. [1989] "Axiomating Net Computation and Processes", **IEEE Conference on Logic in Computer Science, LICS'89**.

Dezani M. [1976] "Characterization of normal forms possessing an inverse in the λβη-calculus", **Theor. Comp. Sci.**, 2 (323-337).

Di Paola A., Heller A. [1984] “Dominical Categories”, City Univ. New York.

Ehrhard T. [1988] “A Categorical Semantics of Constructions” **IEEE Conference on Logic in Computer Science**, **LICS'88,** Edinburgh.

Ehrig H, Kiermeier K.D., Kreowski H.J., Kuehnel W. [1974] “Universal Theory of Automata: A Categorical Approach” B.G. Teubner, Stuttgard.

Eilenberg S., Moore J.C. [1965] “Adjoint Functors and Triples”, **Illinois J. Math**. 9, (381-398).

Eilenberg S., Kelly M.G. [1966] “Closed Categories” **Proc. Conf. Categorical Algebra**, in S. Eilemberg et al. (eds.) (La Jolla 1965). Springer-Verlag.

Elgot C.C. [1971] “Algebraic Theories and programs Schemes”, **Symp. on the Semantics of Algorithmic Languages**, in Engeler (ed.) **LNM 188**, (71-88).

Ershov, Ju.L. [1973] “Theorie der Numerierungen I”, **Zeischr. f. math. Logik un Grundl. d. math.**, Bd. 19.

Ershov, Ju.L. [1975] “Theorie der Numerierungen II”, **Zeischr. f. math. Logik un Grundl. d. math.**, Bd. 21.

Ershov, Ju. [1976] “Hereditarily Effective Operations”, **Algebra i Logika,** Vol. 15.

Feferman S. [1969] “Set-Theoretical Foundations of Category Theory”, **Reports of the Mid-West Category Seminar III**, in S. Mac Lane (ed.), **LNM 106**, (201-247).

Fourman M.P., Scott D.S. [1979] “Sheaves and Logic”, Applications of Sheaves, Springer, **LNM 753**, (302-401).

Freyd P. [1964] **Abelian Categories: An Introduction to the Theory of Functors**, Harper & Row, New York.

Freyd P., Girard J.Y., Scedrov A., Scott P.J. [1988] “Semantic Parametricity in Polymorphic Lambda-Calculus” **IEEE Conference on Logic in Computer Science**, **LICS'88,** Edinburgh.

Germano G., Mazzanti S. [1987] “Loose Diagrams, Semigroupoids, Categories, Groupoids and Iteration”, **1st Workshop on Computer Science Logic**, E. Börger et al. (eds.), **LNCS 273**.

Giannini P., Longo G. [1984] “Effectively given domains and lambda calculus sematics”, **Information and Control, 62**, 1 (36-63).

Gierz G., Hofmann K.H., Keimel K., Lawson J.D., Mislove M., Scott D.S. [1980] **A compendium of continuous lattices**, Springer-Verlag.

Girard J.Y. [1971] “Une extension de l'interpretation de Gödel a l'analyse, et son application a l'elimination des coupures dans l'analyse et la theorie des types”. In **2nd Scandinavian Logic Symposium**, J.E. Festand (ed.), North-Holland, Amsterdam, (63-92).

Girard J.Y. [1972] “Interpretation fonctionelle et elimination des coupure dans l'arithmetic d'odre superieur”, These de Doctorat d'Etat, paris.

Girard J.Y. [1986] “The system F of variable types, fifteen years later”, **Theor. Comp. Sci.**, 45, (159-192).

Girard J.Y. [1987] “Linear Logic” **Theor. Comp. Sci**., 50, (1-102).

Girard J.Y., Lafont Y. [1987] “Linear Logic and Lazy Computation” **Tapsoft '87**, Pisa, **LNCS 250**, Springer Verlag.

Goguen J.A., Thatcher J.W., Wagner E.G., Wright J.B. [1973] “A junction between computer science and category theory: I, Basic definitions and concepts” **Technical Report RC-4526**, IBM Research, September, (part 1).

Goguen J.A., Thatcher J.W., Wagner E.G., Wright J.B. [1976] “A junction between computer science and category theory: II, Basic definitions and concepts” **Technical Report RC-5908,** IBM Research, March, (part 2).

Goguen J.A., Thatcher J.W., Wagner E.G., Wright J.B. [1975] “An introduction to Categories, Algebraic Theories and Algebras” **Technical Report RC-5369**, IBM Research, April. Yorktown Heights.

Goguen J.A., Thatcher J.W., Wagner E.G., Wright J.B. [1977] “Initial Algebra Semantics and Continuous Algebras” **J. ACM**, 24(1), (68-95).

Goguen J.A., Ginali S. [1978] “A Categorical Approach to General Systems Theory”. **Applied General Systems Research** (257-270) in G. Klir (ed.), Plenum, New York.

Goguen J.A., Thatcher J.W., Wagner E.G. [1978] “An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract data Types”. **Current Trends in Programming Methodology**, in R. Yeth (ed.), Prentice Hall, NJ (80-149).

Goguen J.A., Burstall R.M., [1984] “Some Foundamental Tools for the Semantics of Computation, Part 1: Comma categories, Colimits, Signatures and Theories”. **Theor. Comp. Sci.,** 31, (175-209).

Goldblatt R. [1979] **Topoi-The Categorial Analysis of Logic**, North Holland, Amsterdam.

Gunter C. [1985] “Profinite solutions for recursive Domain Equations”, Ph.D. Thesis, Comp. Sci. Dept., C.M.U..

Hayashi S. [1985] “Adjunction of semifunctors: categorical structures in non-extensional lambda-calculus”, **Theor. Comp. Sci.**, 4.

Herrlich H., Strecker G.E. [1973] **Category Theory**, Allyn and Bacon.

Hindley R., Longo G. [1980] "Lambda-calculus models and extensionally", **Zeit. Math. Logik Grund. Math**. n.2, Vol. 26 (289-310).

Hindley R., Seldin J., (ed.) [1980] **To H.B. Curry: Essays in Combinatory Logic, Lambda calculus and formalism**, Academic Press.

Hindley R., Seldin J. [1986] **Introduction to Combinators and Lambda-Calculus**, London Mathematical Society.

Hyland J.M.E. [1977] "Filter Space and Continuous Functionals" **Annals of Mathematical Logic 16**, (101-143).

Hyland J.M.E. [1982] "The effective Topos", *in* **The Brouwer Symposium**, (Troelstra, Van Dalen eds.), North Holland.

Hyland J.M.E. [1987] "A small complete category", Lecture delivered at the Conference **Church's Thesis after 50 years**, Zeiss (NL), June 1986 (**Ann. Pure Appl. Logic**, to appear).

Hyland J.M.E., Johnstone P., Pitts A. [1980] "Tripos Theory", **Math. Proc. Camb. Phil. Soc**., 88 (205-232).

Hyland J.M.E., Pitts A. [1987] "The Theory of Constructions: categorical semantics and topos theoretic models", **Categories in Computer Science and Logic**, Boulder (AMS notes).

Hyland J.M.E., Robinson E., Rosolini P. [1990] "The descrete objects in the effective topos", **Proc. London Math. Soc.** (3) 60,1-36.

Huet G. [1986] "Formal Structures for Computation and Deduction" Lecture Notes, **C.M.U.**.

Jacobs B. [1989] "Some Notes on Fibred Categories and the Semantics of Dependent Types", Università di Pisa, Projects ST2-0374-C of the European Communitee.

Johnstone P.T. [1977] **Topos Theory**. Academic Press, London.

Jones N. [1980] (ed.) **Semantics-Directed Compiler Generation**, LNCS 94, Springer-Verlag.

Kan D.M. [1958] "Adjoint Functors" **Trans. Am. Math. Soc.**, 87, (294-329).

Kasangian S., Labella A. [1988] "Enriched Categorical Semantics for Distributed Calculi", Univ. di Roma "La Sapienza", (preprint).

Kelly G.M. [1964] "On MacLane's Conditions for Coherence of Natural Associativities, Commutativities, ect." **J. Algebra 1** (397-402).

Kelly G.M. [1982] **Basic Concepts of Enriched Category Theory**, Cambridge University Press.

Kock A. [1972] "Strong Functors and Monoidal Modas", **Archiv. der Mathematik**, 23.

Koymans K. [1982] "Models of the lambda calculus", **Information and Control,** 52, (306-332).

Kreisel G. [1959] "Interpretation of analysis by means of constructive functionals of finite type", **Constructivity in Mathematics,** A. Heyting N-H (ed.), (101-128).

Kuratowski C. [1952] **Topologie**, Vol. 1, Warsaw.

Lafont Y. [1988] "The Linear Abstract Machine", **Theoretical Computer Science** 59, 157-180.

Lafont Y. [1988] "Introduction to Linear Logic" Lecture Notes of the **School on Constructive Logics and Category Theory** (Isle of Thorns, August 1988).

Lambek J. [1968] "Deductive systems and categories", **I. J. Math. Systems Theory**, 2, (278-318).

Lambek J. [1974] "Functional completeness of cartesian categories", **Ann. Math. Logic**, 6, (252-292).

Lambek J. [1980] "From lambda-calculus to cartesian closed categories", *in* J.R. Hindley and J.P. Seldin (eds.) to H.B. **Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism**, Academic Press, (375-402).

Lambek J. [1985] "Cartesian Closed Categories and typed lambda calculi", *in* Guy Cousineau, Pierre-Luis Curien and Bernard Robinet (eds.) **Combinators and Functional Programming languages, LNCS 242**, Springer-Verlag.

Lambek J., Scott P.J. [1974] "Aspects of higher order categorical logic" **Contemporary Mathematics**, 30, (145-174).

Lambek J., Scott P.J. [1980] "Intuitionist type theory and the free topos", **J. Pure Appl. Algebra**, 19, (215-257).

Lambek J., Scott P.J. [1986] **Introduction to higher order Categorical Logic**, Cambridge University Press.

Lawvere F.W. [1966] "The Category of categories as a Foundation for Mathematics", *in* **Proc. Conf. on Categorical Algebra**, S. Eilemberg et al. (eds.), La Jolla, 1965, Springer-Verlag.

Lawvere F.W. [1976] "Variable quantities and variable structures in topoi", *in* **Algebra Topology and Category Theory**: a collection of papers in honor of Samuel Eilenberg, A. Heller and M. Tierney (eds.), Academic Press, (101-131).

Lehman D., Smyth M. [1981] "Algebraic specification of data types: a synthetic approach", **Mathematical Systems Theory**, 14, (97-139).

Longo G. [1983] "Set-Theoretical Models of Lambda-Calculus: Theories, Expansions, Isomorphisms", **Annals Pure Applied Logic**, 24, (153-188).

Longo G. [1984] "Limits, higher computability and type free languages", **MFCS'84**, Prague (Chytil, Koubek eds.), **LNCS 176**, Springer-Verlag, (96-114).

Longo G. [1986] "On Church's Formal Theory of functions and functionals", **Church's Thesis after 50 years**, Zeiss (NL), June 1986, (**Ann. Pure Appl. Logic.**, **40**, 1988, (93-133)).

Longo G. [1988] **From type-structures to Type Theories**, Lecture Notes, Spring semester 1987/8, Computer Science dept., C.M.U..

Longo G., Moggi E. [1984] "The Hereditary Partial Recursive Functionals and recursion Theory in higher types", **J. Symb. Logic**, vol. 94, 4 (1319-1332).

Longo G., Moggi E. [1984a] "Cartesian Closed Categories of Enumerations and effective Type Structures", **Symposium on Semantics of Data Types** (Khan, MacQueen, Plotkin eds.) LNCS 173, Springer-Verlag, (235-247).

Longo G., Moggi E. [1990] "A category-theoretic characterization of functional completeness" **Theoretical Computer Science** vol. 70, 2 (193-211)

Longo G., Moggi E. [1988] "Constructive Natural Deduction and its ω-Set Interpretation" CMU Report CS-88-131, **Mathematical Structures in Computer Science**, vol.1, n.2, 1991.

Mac Lane S. [1971] **Categories for the Working Mathematician**, Springer-Verlag, New York.

Mac Lane S. [1982] "Why Commutative Diagrams Coincide with Equivalent Proofs?", **Contemporary Mathematics**, 13, (387-401).

Manes E.G. [1976] **Algebraic Theories**, Springer-Verlag, New York.

Makkai M., Reyes G.E. [1977] "First order Categorical Logic", **Lecture Notes in Math.**, 611.

Margharia I., Zacchi M. [1983] "Right and left invertibility in λβ-calculus", **R.A.I.R.O.**, 17, no.1, (71-88).

Marti-Oliet N., Meseguer J. [1989] "From Petri Nets to Linear Logic", **Category Theory and Computer Science**, Manchester, Pitt et al. (eds.) **LNCS 389**, Springer Verlag.

Marti-Oliet N., Meseguer J. [1990] "Duality in Closed and Linear Categories", **Mathematical Structures in Computer Science**, to appear.

Martini S. [1987] "An interval model for second order lambda calculus", **Category Theory and Computer Science**, Edimburgo, Pitt et al. (eds.) **LNCS 283**, Spriger-Verlag.

Martini S. [1988] “Modelli non estensionali del polimorfismo in programmazione funzionale”, Tesi di Dottorato, Dipartimento di Informatica, Pisa.

Martini S. [1988] “Bounded quantifiers have interval models”, **ACM Conference on Lisp and Functional Programming Languages**, Snowbird, Utah.

Mauny M., Suarez A. [1986] “Implementig functional languages in the categorical abstract machine”, **Proc. Lisp and Functional Programming Conf.**, ACM, Boston.

Meseguer J., Montanari U. [1988] “Petri Nets are Monoids: A New Algebraic Foundation for net Theory”, **IEEE Conference on Logic in Computer Science , LICS'88**, Edinburgh.

Meseguer J. [1988] “Relating Models of Polymorphism”, SRI-CSL-88-13, October, SRI Projects 2316, 4415 and 6729, Comp. Sci. Lab., **SRI International.**

Meyer A.R. [1982] “What is a model of the lambda calculus?”, **Information and Control**, 52, (87-122).

Mints G.E. [1981] “Closed Categories and the Theory of Proofs”, **Journal of Soviet Math.**, N.V., 15, (45-63).

Mitchell B. [1965] **Theory of Categories**, Academic Press.

Mitchell J. [1984] “Semantic Models for Second-Order Lambda Calculus”, *in* **Proc. 25th IEEE Symp. on Foundations of CS**, (289-299).

Mitchell J., Harper R. [1988] “The essence of ML”, **Proc. ACM-POPL 88**.

Moggi E. [1987] “Interpretation of second order lambda-calculus in categories” unpublished manuscript, underground Edinburgh/Pisa.

Moggi E. [1988] “Partial Morphism in Categories of Effective Objects”, **Info&Comp**., 76, 2/3, (250-277).

Moggi E. [1988a] “The partial lambda-calculus”, Ph.D. Thesis, Edinburgh.

Moggi E. [1989] “Computational Lambda-Calculus and Monads”, **IEEE Conference on Logic in Computer Science**, **LICS'89**, Asilomar (Ca).

Obtulowicz A., Wiweger A. [1982] “Categorical Functorial and Algebraic Aspects of the Type-Free Lambda-Calculus”, Universal Algebra and Applications, Banach Center Publications, 9, (399-422) **PWN**-Polish Scientific Publishers, Warszawa.

Ohori A. [1987] “Orderings and types in databases”, **Proc. of the Workshop on Database Programming Languages**, Roscoff, France, September.

Oles F.J. [1985] “Type algebras, functor categories, and block structure”. *In* **Algebraic Methods in Semantics**, Maurice Nivat and John C. Reynolds (Eds.), Cambridge University Press.

de Paiva V.C.V. [1987] "The Dialectica Categories", **Conference in Category Theory**, **Computer Science and Logic**, Boulder (AMS notes).

Pitt D.H., Abramsky S., Poigné A., Rydeheard D.E., (Eds.), [1985] **Category Theory and Computer Programming**, **LNCS 240**, Springer-Verlag.

Pitt D.H., Abramsky S., Poigné A., Rydeheard D.E., (Eds.) [1987] **Category Theory and Computer Science, LNCS 283**, Springer-Verlag.

Pitts A. [1987] "Polymorphism in Set Theoretic Constructively" Symposium on Category Theory and Comp. Sci., **LNCS 283** (Pitt et al. eds.), Edinburgh.

Plotkin G. [1978] "Tω as a universal domain", **J. Comp. Syst. Sci.**, 17, (209-236).

Plotkin G. [1980/4] "Domains", Lecture Notes, C.S. Dept., Edinburgh.

Reynolds J. [1980] "Using category theory to design implicit conversion and generic operators". *In* **Proceedings of the Aarhus Workshop on Semantics Directed Compiler Generation**, N.D. Jones (ed.), Springer-Verlag, January, LNCS 94.

Reynolds J. [1984] "Polymorphism is not set-theoretic", **Symposium on Semantics of Data Types,** Kahn, MacQueen, Plotkin (eds.) **LNCS 173**, Springer-Verlag.

Reynolds J.C., Plotkin G. [1988] "On functors expressible in the polymorphic typed lambda calculus", CMU report CS 88-125, in **Logical Foundations of Functional programming**, G. Huet (ed.), Addison-Wesley, 1990, 127-152.

Rittri M. [1990] "Using types as Search Keys in Function Libraries" **Journal of Functional Programming**, vol. 1.

Robinson E., Rosolini P. [1988] "Categories of partial maps", **Info&Comp.**, 79, (95-130).

Rosolini G. [1986] "Continuity and Effectiveness in Topoi" D. Phil. Thesis, Oxford University.

Rydeheard D.E., Burstall R.M. [1988] "Computational Category Theory", **Prentice Hall International Series in Computer Science**, C.A.R. Hoare Series Editor.

Scedrov A. [1988] "A Guide to Polymorphic Types", **CIME Lectures**, Montecatini Terme, June 1988 (revised version).

Scott D. [1972] "Continuous lattices", **Toposes, Algebraic Geometry and Logic**, Lawvere (ed.) **SLNM** 274, (97-136) Springer-Verlag.

Scott D. [1976] "Data types as lattices", **SIAM Journal of Computing**, 5, (522-587).

Scott D. [1979] "Identity and Existence in Intutionistic Logik", *In* **Applications of Sheaves**, **LNM 753**, Springer-Verlag (660-696).

Scott D. [1980] "Relating theories of the lambda-calculus", *In* Hindley/Seldin .

Scott D. [1980a] "Lambda-calculus, some models, some philosophy", **The Kleene Symposium**, Barwise et al. (eds.), North-Holland.

Scott D. [1980b] "A space of retracts", manuscript, Bremen.

Scott D. [1981] "Lectures on a mathematical theory of computation", Oxford Univ. Comp. Lab., **Tech. Mon. PRG-19**.

Scott D. [1982] "Some ordered sets in Computer Science", *In* **Ordered Sets**, Rival (ed.), Reidel.

Scott D. [1982] "Domain for denotational semantics", (preliminary version), Proceedings **ICALP 82, LNCS 140**, Springer-Verlag.

Scott D. and Gunter C. [1990] "Semantic Domains" **Handbook of Theoretical Computer Science**, J. van Leeuwen (ed), North Holland, to appear.

Scott P.J. [1978] "The "dialectica" interpretation and categories", **Zeitschr. f. Math. Logik und Grundlagen d. Math.**, 24, (553-573).

Seely R.A.G. [1987] "Categorical semantics for higher order polymorphic lambda calculus", **Journal Symb. Logic**, 52, n.4, (969-989).

Seely R.A.G. [1987] "Linear Logic*-Autonomous Categoires and Cofree Coalgebras", **Categories in Computer Science and Logic**, Boulder (AMS notes).

Smyth M. [1977] "Effectively Given Domains", **Theoret. Comp. Sci**., **5**, (255-272).

Smyth M., Plotkin G. [1982] "The category theoretic solution of recursive domain equations", **SIAM Journal of Computing**, **11**, (761-783).

Soloviev S.V. [1983] "The category of finite sets and cartesian closed categories" **Journal of Soviet Math.**, 22, 3.

Stoy J. [1977] **Denotational Semantics**, M.I.T. Press.

Tatsuya H. [1987] "A typed lambda calculus with categorical type constructors", In D.H. Pitt et al. (eds.)

Taylor P. [1986] "Recursive Domains, Indexed Category Theory and Polymorphism", Ph.D. Thesis, Cambridge University, London.

Troelstra A.S. [1973] "Notes in intutionistic second order arithmetic", **Summer School in Math Logic**, **LNM 337**, Cambridge, Springer-Verlag, (171-203).

Wand M. [1979] "Fixed-point Construction in Order-enriched Categories", **Theor. Comp. Sci.**, (13-30).

Winskel G. [1986] “Category Theory and Models of Parallel Computation”, **Proc. Summer Workshop on Category and Computer Programming**, Surrey, LNCS 240, (266-281).

Wiweger A. [1984] “Pre-adjunctions and lambda-algebraic theories”, **Coll. Math. XLVIII**, Warszava, (153-165).