Categories and Computability

J.R.B. Cockett *
Department of Computer Science, University of Calgary,
Calgary, T2N 1N4, Alberta, Canada

August 12, 2014

Contents

*Partially supported by NSERC, Canada.

1 Introduction

It is impossible to give a complete view of category theory in a short sequence of lectures such as these and at the same time begin to illustrate an application. It is, therefore, inevitable that I will present a very biased and selective sampling. Since its inception in 1945 (with Eilenberg and Mac Lane's paper [13]) category theory has found use as a fundamental tool in almost all areas of mathematics, it has had a profound effect on logic and philosophy, and it has been seriously applied both in the foundations of physics and, increasingly since the mid 1980's, to the foundations of computer science in programming semantics.

The second part of these notes describe another important area of application to computer science: computability. Despite the obvious potential for categorical methods to provide new insights in this area, computability (and complexity) has still not received the level of categorical attention it deserves. This makes it a particularly appropriate target for a series of lectures as there are plenty of opportunities to contribute!

1.1 Categories and foundations

Category theory starts with a beguilingly simple notion: a category is just a directed graph with a composition defined on the edges. However, as more structure (mediated by functors, natural transformations, limits, and colimits) is added to this modest beginning, a fundamentally new and powerful mathematical perspective begins to emerge .... the reverberations of which are still working their way through the body of mathematics.

Prior to the 1950's work in the foundations of mathematics was almost all reductionist. This approach to foundations originated in Cantor's creation of set theory. It is easy to forget that shortly after its formulation around the turn of the 20^{th} century this foundation was beset by "paradoxes" and was widely criticized. Indeed, it was not until the work of Russell, Whitehead, and Zermelo which established axiomatic set theory, that Cantor's sets were viewed as laying a foundation for mathematics. The antinomies which had plagued the foundations of mathematics had been quite embarrassing, and it was almost, with relief that the Mathematical establishment embraced the solution offered by axiomatic set theory. In 1926, when some mathematicians – following the "intuitionistic" views of Brouwer – continued to question this newly lain foundation, it was hardly surprising that Hilbert, representing the establishment, famously declared: "No one shall expel us from the paradise that Cantor has created for us".

Ironically it was only a few years later in 1931 that Gödel's work left Hilbert's program in tatters. Brouwer's "intuitionistic" views, although in some sense vindicated, never became mainstream. In fact, the focus in Mathematics had by that time already shifted away from foundational issues toward new and more applied frontiers – analysis, topology, algebraic geometry, number theory, combinatorics, etc. – and to other developments, such as Bourbaki's program to organize and systematize mathematics at a more abstract level. By the 1950's large tracts of new mathematics had been developed and documented. The relationships between these diverse developments had begun to assume an importance which, on the one hand, could not have been foreseen and, on the other, was simply not addressed at all by a reductionist approach to foundation.

The development of category theory brought to the table a potent antidote to the reductionist program. Not only did it allow for the expression of relationships between areas of mathematics but also it provided methods to directly encode behavior in terms of admissible transformations.

It embodies an approach to mathematics very much in the tradition of Klein's Erlangen Program (involving transformation groups in geometry), and Noether's abstract approach to algebra. Most importantly category theory allowed the systematic discussion of behavior in a manner which avoided the detailed bottom-up development of internal structure. This enabled mathematicians to focus on structural interactions without actually worrying about exactly how the structures might arise. Considering that, famously, it took 360 pages of Principia Mathematica to prove $1 + 1 = 2$, this was an immensely important innovation. Mathematical structures had become increasingly sophisticated and a reductionist approach to all structure had become a millstone around the neck of progress. It was simply too much work to connect securely to increasingly distant and irrelevant foundations.

1.2 Categories and computer science

In 1987 Andre Scedrov and John Gray arranged a research conference at the University of Boulder (Colorado) entitled "Categories in Computer Science and Logic." If one were to try to pinpoint exactly when category theory became a respectable tool in computer science this meeting (and its aftermath) stands out. John Reynolds, already a prominent computer scientist, and Jean-Yvres Girard, a young French logician, had independently invented the polymorphic λ-calculus. John Reynold had naively suggested a set theoretic semantics and had discovered somewhat to his dismay that not only was his semantics flawed but that he could prove that there could be no such semantics. The fact that one could compute perfectly well with this calculus displayed a serious gap between set theoretic semantics and what was apparently required for computation. Consequently Reynolds had asked both logicians and category theorists where the semantics of such systems lay. It turned out that category theorists already had an answer and part of the meeting was dedicated to an exposition (by Andy Pitts and Edmund Robinson) of that answer. The answer involved topos theory and altogether too many categorical steps for John Reynolds to digest at the meeting itself, however, he undertook to learn category theory and subsequently made substantial use of it in his work.

The seeds of a number of other burgeoning interactions between these subjects (logic, category theory, and computer science) were also latent at that meeting. To mention just one: Jean-Yvres Girard was present and gave an exposition of linear logic and its proof theory. Subsequently in the proceedings Robert Seely [28] provided a categorical semantics for this system in Barr's $*$-autonomous categories. This developed into a continuing interaction between categorical proof theory, logic, and computer science which is still central as it has important implications for the semantics of concurrency and more generally in systems in which resources are limited.

1.3 Categories and computability

An area in which one might have expected computer science and category theory to have had a potentially significant interaction is right at the foundations of computer science, namely, at computability and complexity theory. Realizability topoi (due to Martin Hyland and Dana Scott) are built on top of a model of computability, and these were instrumental in answering Reynolds question. Thus, already computability issues were indirectly present. In the same year as the Boulder Conference, a very influential paper by Robert Di Paola and Alex Heller on "recursion theory without elements" [12] was published in the Journal of Symbolic Logic. It outlined a categorical approach to computability theory. Their work set in motion a rather different program to understand computability categorically.

Di Paola and Heller's paper was based on a categorical formulation of partiality and almost immediately Pino Rosolini and Edmund Robinson [26] pointed out that there was a cleaner foundation using what they called P-categories. Their work indicated that there were still foundational issues underlying Di Paola and Heller's approach which needed to be resolved. A little later Eugenio Moggi, in his thesis [24], undertook an exhaustive study of the logics of partiality: he linked half a dozen available approaches to partiality in an impressive tour de force. However, Moggi in the course of this work also realized that partial map classifiers, viewed as cartesian monads, could be used to describe partiality and, indeed, other computational effects. This led him to subsequently concentrate on the theory and logics for these monads. This had the unfortunate effect of sidelining the original program of Di Paola and Heller as it became submerged in the somewhat different program of investigating the role of monads in computation.

The purpose of these notes is to return to Di Paola and Heller's program and to provide an elementary categorical development of the subject starting much further back than they started but, eventually, arriving at a setting very much as they had envisaged it. A satisfying aspect of this development is that it is suitable abstract: it encompasses not only the traditional classical setting of recursive function theory but also the –somewhat different in feel – *total* theories of computability. These include Church's λ-calculus and Curry's combinatory logic. Significantly the underlying mathematical machinery, restriction categories – while certainly being well-suited to this application – are much more broadly applicable (see the introduction of [9] and [23]) and this repositions the subject more centrally in mathematics.

A defect of these notes is that I have not presented any non-standard examples. Instead I have drawn the examples, such as they are, almost entirely from traditional sources. This is a pity as the value of a more abstract approach is exactly that it accommodates unexpected examples. That there are, indeed, highly non-standard examples of Turing categories and recursion categories is known. The notes, in fact, do include a brief mention of one such example: the classifying category for partial combinatory algebras. This Turing category is in a strong sense the mother of all computation. It is also the tip of an iceberg. There are many fascinating Turing and recursion categories out there which have yet to be thoroughly investigated and each has the potential to add to – or even upset – our preconceptions of computation.

Acknowledgements:

Thanks to Geoff Crutwell and Brian Redmond for providing feedback throughout the development of these notes. Thanks also to Steve Awodey for passing his philosophically more experienced eyes over my introduction.

The first part of the notes are a selection from my Category Theory class notes. The section on restriction categories is based on a term paper of Johnathan Gallagher and, of course, on [8]. The section on Turing categories was borrowed – virtually verbatim – from work with Pieter Hofstra [7]. The last section on recursion categories was based on a talk I gave at FMCS in 2007.

Part I
Basic Categories

This part introduces category theory. It starts with various ways to define a category and introduces some examples of categories. It then provides a brief introduction to functors and natural transformations. It ends by giving a brief description of adjunctions and limits.

2 Categories and examples

2.1 The definition of a category

There are various ways to define a category which are naively equivalent – but also subtlety quite different. Each definition presents a different perspective on what a categoryhood really is .... We shall start with the most common presentation of a category which is as a directed graph with a composition. From there we mention three further formulations, as an internal category, as a partial algebra, and as an enriched category.

2.1.1 Categories as graphs with composition

A **category**, $\mathbb{C}$ consists of a directed graph, that is a collection of **objects**, $\mathbb{C}_0$, and a collection of **maps**, C_1, such that each map $f \in \mathbb{C}_1$ has an associated object $\partial_0(f) \in \mathbb{C}_0$ which is its **domain** and an associated object $\partial_1(f) \in \mathbb{C}_0$ which is its **codomain**. We shall indicate that f has domain A and codomain B by writing $f : A \longrightarrow B$.

The maps of a category can be composed: that is given any pair of maps $f, g \in \mathbb{C}_1$ with the codomain of f being the same as the domain of g, that is they are a **composable** pair of the form $f : A \longrightarrow B$ and $g : B \longrightarrow C$, then there is an associated **composite** $fg : A \longrightarrow C$.[1] The operation of forming composites is called **composition**:

$$\frac{A \xrightarrow{f} B \quad B \xrightarrow{g} C}{A \xrightarrow[fg]{} C}$$

Categories are also required to have **identity maps** for the composition. Thus, given any object $C \in \mathbb{C}$ there is associated to it an **identity** map

$$\frac{}{C \xrightarrow[1_C]{} C}$$

denoted by $1_C : C \longrightarrow C$. This data must then satisfy the following axioms:

[C.1] (Identity laws) if $f : A \longrightarrow B$ then $1_A f = f = f 1_B$,

[C.2] (Associative law) if $f : A \longrightarrow B$, $g : B \longrightarrow C$, and $h : C \longrightarrow D$ then $(fg)h = f(gh)$.

Now when checking that something is actually a category one must first establish that all the data is present. Thus, one must first identify a collection of objects. It will often be the case that the objects will be known by some other name such as nodes, types, sets, or algebras. Next one must have a collection of maps. Again the maps will often be known by some other name such as arrows, terms, functions, or homomorphisms. Finally one must have a notion of composition for these maps: this involves not only having the operation which allows the formation of composites for maps which juxtapose but also the presence of identities.

We remark that having identities is a property rather than extra structure as an identity (if it exists at an object) is uniquely determined by the composition. This is because if 1_A and $1'_A$ are identities on A then $1_A = 1_A 1'_A = 1'_A$; so there can be at most one identity map.

[1] Please note that "diagrammatic" as opposed to "applicative" order will be used in this text. Applicative order is particularly common in many of the more mathematical text books on category theory so you should be able to read both.

The identity map is an example of a map which starts and ends at the same place: these we call **endo-maps**. A category can have – and often will have – many endo-maps. More generally a category can have many different parallel maps from one object A to another B: these maps can be collected into a **hom-object**, which in this case is a set denoted $\mathbb{C}(A, B)$

A **subcategory** $\mathbb{C}'$ of a category $\mathbb{C}$ is given by any subcollection of the objects and maps which is "closed" to the domain, codomain, identity, and composition structure.

Please notice that it has *not* been required that the objects or arrows of a category should form a *set*: this is because some categories are too large for either their objects or their arrows to form sets. A typical "big" category is the category of sets Sets itself: Russell's paradox[2] informs us that the set of all sets cannot possibly be a set. So it is convenient to allow the objects to belong to something bigger than a set (a class).

However, it is also sometimes convenient to require that the maps and arrows all belong to one world: this gives rise to the notion of an **internal category**. Thus, a **small** category is a category whose arrows and objects do both form sets a **finite** category is a category whose objects and maps are finite sets, a **topological category** is a category whose arrows and objects are topological spaces, an **ordered category** is a category whose objects and arrows are partially ordered sets, etc.....

2.1.2 Categories as partial semigroups

Another way to view a category is as collection of maps which have a partial associative multiplication and a system of units. This view is interesting as it indicates that the objects are actually redundant structure and their role can be replaced by the identity maps. However, the cost of making them redundant is that one needs some additional axioms.

This time a category $\mathbb{C}$ just consists of a collection of maps on which there is a partial associative composition and two assignments of arrows to arrows ∂_0 and ∂_1 such that:

[C'.1] $\partial_0(f)f$ is always defined and is f,

[C'.2] $f\partial_1(f)$ is always defined and is f,

[C'.3] fg defined if and only if $\partial_1(f) = \partial_0(g)$ and $\partial_0(fg) = \partial_0(f)$ and $\partial_1(fg) = \partial_1(g)$,

[C'.4] (Associative) $(fg)h = f(hg)$ whenever each side is defined,

[C'.5] $\partial_j(\partial_i(x)) = \partial_i(x)$, for $i, j = 0, 1$.

To return to the definition above it is necessary to provide the collection of objects: these are the maps for which $\partial_0(x) = x = \partial_1(x)$ which are just the identity maps.

[2]Russell's paradox concerns the set of all sets: this should be a set because as one can define sets by properties (such as being a set) but consider the set of all sets which are not members of themselves - a subset of the set of all sets. If it were a member of itself then it could not be a member of itself; on the other hand if it were not a member of itself then it would be a member of itself! This paradox forced set theorists to declare that the set of all sets cannot be a set. As one can hardly deny the existence of the collection of all sets, the awkward question then arises: what is the set of all sets? This forced set theorists – Russel – into considering class hierarchies.

2.1.3 Categories as enriched categories

Recall the notation $\mathbb{C}(A,B)$ for hom-objects mentioned above for this leads naturally into our third view of what a category is: if $\mathbb{C}$ is a category then by $\mathbb{C}(A,B)$ will be denoted the collection of all arrows with domain A and codomain B. These are called the **hom-objects** of the category. If these collections are actually *sets* then we call these "hom-sets" and say that the category is **enriched** in Sets.

Notice that when a category is enriched in Sets, the composition can be described as a family of maps, one for each triple of objects in $\mathbb{C}$:

$$m_{A,B,C} : \mathbb{C}(A,B) \times \mathbb{C}(B,C) \to \mathbb{C}(A,C)$$

(where m is for multiplication). In addition, there must be an identity map $1_A : 1 \to \mathbb{C}(A,A)$, where 1 as an object represents a generic one element set $1 = \{*\}$. These maps must now satisfy the two requirements imposed on a category (associativity and the identity laws):

$$\begin{tikzcd}
\mathbb{C}(A,B) \times \mathbb{C}(B,C) \times \mathbb{C}(C,D) \arrow[r, "m_{ABC}\times 1"] \arrow[d, "1\times m_{BCD}"'] & \mathbb{C}(A,C) \times \mathbb{C}(C,D) \arrow[d, "m_{ACD}"] \\
\mathbb{C}(A,B) \times \mathbb{C}(B,D) \arrow[r, "m_{ABD}"'] & \mathbb{C}(A,D)
\end{tikzcd}$$

$$\begin{tikzcd}
1 \times C(A,B) \arrow[r, "1_A\times 1"] \arrow[dr, "\pi_1"'] & \mathbb{C}(A,A) \times \mathbb{C}(A,B) \arrow[d, "m_{AAB}"] \\
& \mathbb{C}(A,B)
\end{tikzcd}
\qquad
\begin{tikzcd}
C(A,B) \times 1 \arrow[r, "1\times 1_A"] \arrow[dr, "\pi_0"'] & \mathbb{C}(A,B) \times \mathbb{C}(B,B) \arrow[d, "m_{ABB}"] \\
& \mathbb{C}(A,B)
\end{tikzcd} .$$

These requirements have been expressed as commuting diagrams - a style which we shall see much more of as we progress.

This enriched category style of expressing the structure of a category is important as often the hom-objects of categories are not simply unstructured collections of maps but themselves have structure (e.g. Abelian categories, poset enriched categories, Cat-enriched categories, etc.). By regarding a category in this manner we will be able to explain how the additional structure of the hom-objects must interact with the composition of the category itself.

For enriched categories a very natural notion of subcategory is determined by taking a subcollection of the objects and leaving the hom-objects and composition structure the same. This is called a **full-subcategory**.

2.1.4 The opposite category and duality

Category theory is full of symmetries which are called **dualities**. The basic source of symmetry is the ability to reverse arrows. Thus, given any category we may obtain a new category by keeping everything the same except to switch the direction of the arrows. If we start with a category $\mathbb{C}$ and flip the direction of the arrows we obtain a new category written $\mathbb{C}^{\mathrm{op}}$. Observe now that anything which is true of $\mathbb{C}$ now holds in the **dual** form in $\mathbb{C}^{\mathrm{op}}$. Thus, when we prove a result there is always another result, obtained by reversing the sense, of the arrows which will also be true. This principle of duality allows us to get double the bang for our buck!

Often the prefix "co" (as in colimit, coequalizer, coproduct) is an indication that this is the dual concept to that with this prefix removed.

Some categories are actually **self-dual**, thus $\mathbb{C}^{\rm op}$ is in some sense the same as $\mathbb{C}$. In this case there is often an explicit translation (this is an example of a contravariant functor see section 3.2) $(\_)^* : \mathbb{C} \longrightarrow \mathbb{C}^{\rm op}$ this has the following properties:

(i) If $f : A \longrightarrow B$ then $f^* : B^* \longrightarrow A^*$;

(ii) If $f : A \longrightarrow B$ and $g : B \longrightarrow C$ then $(fg)^* = g^* f^*$ and $1_A^* = 1_{A^*}$.

We shall say that such a translation is an **involution** in case $(f^*)^* = f$ for each arrow. Sometimes it is the case that for objects $A = A^*$ in this case we shall refer to the involution as a **converse**.[3]

2.2 Examples of categories

Below we outline a series of sources of examples of categories.

2.2.1 Preorders

Categories enriched in sets of cardinality at most 1 (in other words the category containing, as objects, the empty set and the one element set, and, as maps, all set maps between these objects) are called **preorders**. They are important as they account for both equivalence relations and partially ordered sets. There is a considerable body of knowledge about partially ordered sets and this provides an important source of examples and, often, a pattern to follow when developing categorical structures.

Notice when there is at most one arrow between any two objects the value of the composite of any two maps is forced. Thus, it is simply a matter of whether maps exists between objects or not. Thus we may view such a category as a relation on the objects.

A **preordered set** may more prosaically be described as a set with a reflexive, transitive relation . A relation is **reflexive** on a set X in case whenever $x \in X$ we require (x, x) to be in the relation: this gives the identity map on that object. A relation is **transitive** in case whenever (x, y) and (y, z) are in the relation then (x, z) must be in the relation: this gives composition.

A relation is an **equivalence relation** in case in addition it is **symmetric** that is whenever (x, y) is in the relation (y, x) is also in the relation. This is equivalent to asking that every arrow is an isomorphism (see section 3.1: a category in which all the maps are isomorphisms is called a groupoid).

Clearly an equivalence relation viewed as a category has a converse involution.

A **partially order set** is a preorder with the addition **anti-symmetry** property that whenever (x, y) and (y, x) are in the relation then $x = y$. This is equivalent categorically to asking either that the only isomorphisms are the identity maps or, less stringently, that all isomorphisms are endo-maps. This is a small illustration of how one notion for a poset can be generalized in different ways for arbitrary categories indicating a shortcoming of viewing categories as simply generalization of posets.

[3]Notice that an involution cannot in general be stationary on maps. If an involution is stationary on maps one can easily show that there is only one object in each connected component of the category and that the endomorphism monoid on each object is a commutative monoid.

2.2.2 Finite categories

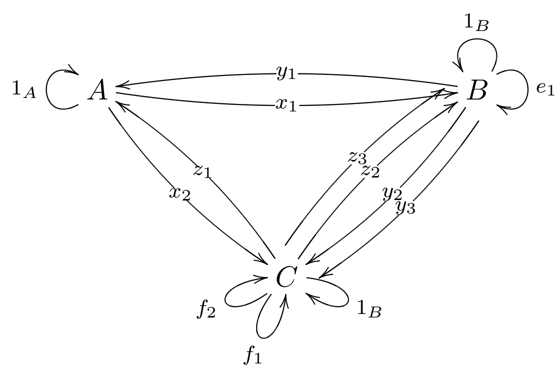
The simplest category of all has no objects and no maps. This is called (for reasons which will be explained later) the **initial** category. The initial category is certainly finite and there is not much else one can say about it!

The next most simple category is the category with one object and exactly one arrow. This is called the **final category**: it is also finite and there is not so very much more one can say about it either. The one arrow is actually forced to be the identity map on the one object.

A finite category, a category internal to finite sets, must have both a finite number of objects and a finite number of arrows. All finite categories are necessarily enriched over finite sets (i.e. have finite hom-sets).

A finite categories may be presented as a directed graph with a multiplication table for each object: each table will represent the arrows which juxtapose at that object.

We may view a finite category $\mathbb{F}$ as having an underlying directed graph such as the one below:



We may then arrange the composition tables by the objects at which the arrows meet. For each such table we may arrange the arrows coming in to the object (along the vertical axis) according to the object from which they come. Similarly, along the horizontal axis we may arrange the maps according to the object at which they end. The result of the composition (minimally) must then be an arrow with the correct domain and codomain. One then must check that the composition satisfies the identity and associative laws. The latter law is quite arduous to check as one must consider triples of composable maps and check that the two possible compositions are in fact equal (for good computational shortcuts see Light's generalized associativity test).

Finite categories are a great source of counter-examples for simple categorical non-facts. Notice also that there are two simple ways of constructing new finite categories from old ones. Given two categories $\mathbb{C}$ and $\mathbb{D}$ one can form the disjoint union of their arrows and objects to form a new category $\mathbb{C} + \mathbb{D}$ where the composition(s) are unchanged: this is the coproduct. Alternately one can put the arrows in parallel to form $\mathbb{C} \times \mathbb{D}$ the **product** category. Here one takes the cartesian product of both the objects and the arrows and define the composition pointwise, that is if $f : C_1 \longrightarrow C_2$ in $\mathbb{C}$ and $g : D_1 \longrightarrow D_2$ in $\mathbb{D}$ then in $\mathbb{C} \times \mathbb{D}$ there is the maps $(f, g) : (C_1, D_1) \longrightarrow (C_2, D_2)$ and $(f, g)(f', g') = (ff', gg')$.

2.2.3 Categories enriched in finite sets

A category all of whose homsets are finite is a category enriched in finite sets. While every finite category (i.e internal category in finite sets) is also a finite set enriched category, there are plenty of

		A	B	C
	A	1_A	x_1	x_2
A	1_A	1_A	x_1	x_2
B	y_1	y_1	e_1	y_3
C	z_1	z_1	z_3	f_2

		A	B		C	
	B	y_1	1_B	e_1	y_2	y_3
A	x_1	1_A	x_1	x_1	x_2	x_2
B	1_B	y_1	1_B	e_1	y_2	y_3
	e_1	y_1	e_1	e_1	y_3	y_3
C	z_2	z_1	z_2	z_3	f_1	f_2
	z_3	z_1	z_3	z_3	f_2	f_2

		A	B		C		
	C	z_1	z_2	z_3	1_C	f_1	f_2
A	x_2	1_A	x_1	x_1	x_2	x_2	x_2
B	y_2	y_1	1_B	e_1	y_2	y_2	y_3
	y_3	y_1	e_1	e_1	y_3	y_3	y_3
C	1_C	z_1	z_2	z_3	1_C	f_1	f_2
	f_1	z_1	z_2	z_3	f_1	f_1	f_2
	f_2	z_1	z_3	z_3	f_2	f_2	f_2

Figure 1: Composition tables for $\mathbb{F}$

finite set enriched categories which are not finite. The category of finite sets, Set_f, itself is certainly finite set enriched but not finite.

Another example of a finite set enriched category is the category of finite interference graphs, Intf_f. This category has objects finite sets equipped with a symmetric, anti-reflexive relation $(S, \bowtie)$. Explicitly such a relation satisfies:

- $x \bowtie y \Leftrightarrow y \bowtie x$,
- $x \bowtie y \Rightarrow x \neq y$.

A map between two objects in this category

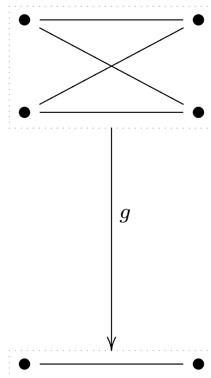
$$f : (X, \bowtie_X) \longrightarrow (Y, \bowtie_Y)$$

is a map between the underlying finite sets such that

$$x \bowtie x' \Rightarrow f(x) \bowtie f(x').$$

Here is an example of a map which collapses the four cycle C_4 onto the completely separated graph

S_2:



This category already has lots of interesting structure!

2.2.4 Monoids

A category which has one object and a set of maps is a **monoid** in Set. Monoids have been extensively studied in their own right under the more general title of semigroups. For example group theory is the study of monoids all of whose maps are isomorphisms (see section 3.1) and these besides being monoids are an important source of examples of categories.

There are large numbers of finite groups, larger numbers of finite monoids and, thus, by implication there are even larger numbers of finite categories!

2.2.5 Path categories

Given a directed graph $\mathbb{G}$ we may form a category from it which we call the **path** category of $\mathbb{G}$, denoted $\mathsf{Path}(\mathbb{G})$. The objects are the same as those of $\mathbb{G}$ but the arrows are sequences of arrows in $\mathbb{G}$ which juxtapose. The composition is, as might be expected, given by concatenation.

More formally, the arrows in $\mathsf{Path}(\mathbb{G})$ are given by triples $(A, [g_0, .., g_n], B)$ where, when the list of maps is non-empty, $A = D_0(g_0)$, $B = D_1(g_n)$, and $D_1(g_i) = D_0(g_{i+1})$ for $i = 0, ..., n-1$. When the list is empty we insist that $A = B$ which gives us maps of the form $(A, [], A)$ for each object A: these will serve as the identity maps. We define $D_0(A, l, B) = A$ and $D_1(A, l, B) = B$ and set $(A, l_1, B)(B, l_2, C) = (A, l_1@l_2, C)$, where $\_@\_$ is the usual concatenation operation.

2.2.6 Matrices over a rig

Let R be any **rig** (this a ring without the "n" for negatives), that is a set with an addition (a commutative associate operation $x + y$ with an identity 0) and a multiplication (that is an associative not necessarily commutative operation $x \cdot y$ with a unit 1) such that

$$x \cdot (y + z) = x \cdot y + x \cdot z \quad \text{and} \quad (y + z) \cdot x = y \cdot x + z \cdot x,$$

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z \quad \text{and} \quad x \cdot 1 = x = 1 \cdot x$$

$$x \cdot 0 = 0 = 0 \cdot y$$

then we may form $\mathsf{Mat}(R)$ the category of R–matrices. This category has objects the natural numbers and maps $n \times m$–matrices with the usual multiplication.

Somewhat unusually we also allow $0 \times n$ and $n \times 0$ matrices. The composites with these "empty" matrices are themselves empty.

Notice that $\mathsf{Mat}(R)$ has a converse involution given by transposition.

A special example of this category which is very well-studied is $\mathsf{Mat}(K)$ where K is a field (such as $\mathbb{R}$).

2.2.7 Kleene categories

Let us briefly consider a variant of both the above and previous example which has traditional computer science importance. Let $\mathbb{G}$ be a directed graph, we shall form a $\mathsf{Path}^*(\mathbb{G})$ by taking an arrows from A to B to be (regular) subset of the set of all paths from A to B.

Regular subsets are generated by first including the singleton path sets and closing these with respect to union, Kleene star, and composition. This means given a finite number of path sets which are parallel (i.e. have the same domain and codomian) their union is also a path set (recall finite sets include the empty set so the empty set is always a path set). The Kleene star of a set of endopaths is:

$$P^* = \{p_1 @ p_2 @ ... @ p_n | p_1, p_2, ..., p_n \in P\}$$

Finally the composition of one set of paths with another is given by taking the composite of all possible pairs of arrows from the sets.

Notice that in this category $f(g \cup h) = fg \cup fh$ and $(f \cup g)h = fh \cup gh$. This allows us to do a matrix construction and to form $\mathsf{Reg}(\mathbb{G})$:

Objects: Lists of objects of $\mathbb{G}$, e.g. $[G_1, .., G_n]$

Maps: Matrices of regular subsets

$$[a_{i,j}] : [G_1, .., G_n] \longrightarrow [G'_1, .., G'_m]$$

where $a_{i,j} : G_i \longrightarrow G'_j$.

Composition: Matrix multiplication where multiplication is composition and addition is union:

$$[a_{i,j}][b_{j,k}] = [\bigcup_j a_{i,j} b_{j,k}]$$

Identities: Identity matrices with the singleton set of identity path on the diagonal and empty sets off diagonal.

There is an important "trace" operation (which is a categorical trace) on these matrices which allows one to reduce the dimension across a common input and output state: this operation introduces feedback and makes the state internal.

For a 2-dimensional matrix this "trace" operation works as follows:

$$\mathsf{trace}_X \left(\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} : [A_1, A_2, X] \longrightarrow [B_1, B_2, X] \right)$$

$$= \begin{bmatrix} a_{11} \cup a_{13} a_{33}^* a_{31} & a_{12} \cup a_{13} a_{33}^* a_{32} \\ a_{21} \cup a_{23} a_{33}^* a_{31} & a_{22} \cup a_{23} a_{33}^* a_{32} \end{bmatrix} : [A_1, A_2] \longrightarrow [B_1, B_2]$$

where $(\_)^*$ is the Kleene star operation. This trace operation is the basis of the translation of finite state machines into regular languages. The idea is to view a finite state machine (with ϵ-transitions) as a matrix in this category: the domain is arranged to consist of a distinguished start state and the internal states while the codomain is a distinguished final state and the internal states. The trace operation can then be used to eliminate rows and columns until one obtains a regular expression from start state to final state corresponding to the automaton.

2.2.8 Sets

There are various categories we may form from the category of sets. The primary category, denoted by Set, is the category of sets and functions itself. However, there are two other variants which we shall want to use as examples: the category of relations, Rel, and the category of partial maps, Par.

The category of relations Rel is given by the following data:

Objects: Sets;

Maps: Relations $R : X \longrightarrow Y$;

Composition: $RS = \{(x,z)|\exists y.(x,y) \in R \wedge (y,z) \in S\}$;

Identities: $1_X = \{(x,x)|x \in X\}$.

The category of sets and partial maps, Par, and the category of sets and functions, Sets, itself, can be seen as subcategories of this category. The category of partial maps is the subcategory with the same objects but only those relations which are **deterministic**, that is (x,y) and (x,y') imply $y = y'$. The category of functions is a further subcategory of the category of partial maps Par with the same objects but with the maps restricted to those deterministic relations which are total. A relation $R : X \longrightarrow Y$ is **total** in case for every $x \in X$ there is a $y \in Y$ such that $(x,y) \in R$.

Notice that neither of these subcategories are full-subcategories as we are strictly reducing the set of maps between objects (in most cases!).

The category of relations has a converse $(\_)^o : \mathsf{Rel}^{\mathrm{op}} \longrightarrow \mathsf{Rel}$ obtained by reversing the ordered pairs. This converse operation, however, is not inherited by either the category Par or Sets. The relations in Rel may also be ordered by inclusion: this makes the category poset enriched. This enrichment *is* inherited by Par: for Sets this partial order is discrete (in the sense that two things are related only if they are the same).

2.2.9 Programming languages

We may think of a programming language as being a given by a collection of types which have programs defined between them. The ability to compose programs and the presence of the "do nothing" program together with the expectation that these will satisfy the basic axioms of a category means that programming languages may be modeled by categories.

Given a programming language there then is the question of which category precisely it describes. Given that many programming languages have grown in an ad hoc way this is, in general, a messy question. However, one might expect that for simple programming languages this would be an easy question to answer. There is always the "term model" to fall back on, that is the programming language itself with the equalities that should hold. However, this is usually regarded as a less than satisfactory answer: instead one usually seeks an answer which allows the language to be interpreted an independently well-understood mathematical structure . These are are generally known as **semantic** issues: formally one is seeking a functor with certain properties from the programming language into the "semantics" which is a category of independent origin. One usually hopes that the functor is full or faithful: often as program equality itself is not well-understood one takes equality to be "semantic equality": that is the equality relation given by the semantics so that fullness is the only real isssue.

The pursuit of semantics is a major theoretical direction in programming languages. However, there is another direction which is considerably less traditional but, nonetheless, of greater practical importance. One can ask which constructs should be present in a good programming language. This is a seemingly rather vague question, however, one can give a reasonable answer to such a question if one understands the theory of programming constructs. Category theory provides a powerful mathematical tool for addressing this sort of question: it gives a landscape view of the constructs involved in programming and how they can be fit together to make reasonable programming environments.

2.2.10 Products and sums of categories

Once one has a stock of categories one can begin to build new categories. There are two rather obvious constructions: the product and the disjoint union or sum of categories.

Let us discuss the sum: given a family, $(\mathbb{X}_i)_{i \in I}$, of categories one can just consider the disjoint union of the categories $\bigsqcup_{i \in I} \mathbb{X}_i$. This is a category whose objects (respectively maps) are the disjoint union of the objects (respectively maps) of each $\mathbb{X}_i$. Clearly objects from different components will not be connected by any maps. Indeed given any category there is always a (unique) decomposition of it into connected components which allows us to view it as a sum of connected categories.

It is always worth considering the exteremal examples: what happens when we sum the empty family of categories? Well, of course, we get a perfectly good category: the empty category, 0, the one which has no arrows and no objects!

If we think of this construction as putting categories side-by-side, the product a family, $(\mathbb{X}_i)_{i \in I}$, of categories, $\prod_{i \in I} \mathbb{X}_i$ is obtained by putting the categories in parallel. Thus the objects are tuples of objects $(X_i)_{i \in I}$ with $X_i \in \mathbb{X}_i$, the maps are tuples of maps $(f_i)_{i \in I}$ with $f_i \in \mathbb{X}_i$, domain $\partial_0((f_i)_{i \in I}) = (\partial_0(f_i))_{i \in I}$, and codomain $\partial_0((f_i)_{i \in I}) = (\partial_0(f_i))_{i \in I}$. Composition is then performed in parallel: $(f_i)_{i \in I}(g_i)_{i \in I} = (f_i g_i)_{i \in I}$.

Again the empty product of categories is also perfectly well defined: it has one object, the empty tuple (), and one arrow, the empty tuple (): so it is the category 1.

2.2.11 Slice categories

If $\mathbb{C}$ is any category and $X \in \mathbb{C}$ we may form the **slice category** $\mathbb{C}/X$. This has the following structure:

Objects: Maps of $\mathbb{C}$ to X, $f : C \to X$;

Maps: Triples $(f_1, g, f_2) : f_1 \longrightarrow f_2$ which are commutative triangles:

$$\begin{tikzcd} C_1 \arrow[rr, "f"] \arrow[dr, "f_1"'] & & C_2 \arrow[dl, "f_2"] \\ & X & \end{tikzcd}$$

Identities: $(f, 1_C, f) : f \longrightarrow f$.

Composition: $(f_1, g, f_2)(f_2, h, f_3) = (f_1, gh, f_3)$ which is well-defined as $ghf_3 = gf_2 = f_1$.

Now one must, in fact, check that this is a category which amounts to checking that we have a composition which satisfies the required axioms. However, this structure is directly inherited from $\mathbb{C}$ so the proof is straightforward.

It is worth noting a peculiar things about the slice categories over Set:

$$\begin{array}{rcl} \mathsf{Set}/0 & \cong & \mathbf{1} \\ \mathsf{Set}/1 & \cong & \mathsf{Set} \\ \mathsf{Set}/2 & \cong & \mathsf{Set} \times \mathsf{Set} \\ \mathsf{Set}/3 & \cong & \mathsf{Set} \times \mathsf{Set} \times \mathsf{Set} \\ & \ldots & \end{array}$$

In fact, we may regard Set/I as the I-indexed product of the category of sets with itself. This is because a map $C \longrightarrow I$ in Sets is "the same thing" as the I-indexed collection of sets $(C_i = \{c \in C | f(c) = i\})_{i \in I}$. A map in the slice category $k : f \longrightarrow f'$ can then be viewed as an indexed collection of maps $(k_i : C_i \longrightarrow C'_i)$.

This view of a slice category is rather special to sets. However, the properties of these slice categories of Set have been the inspiration for trying to codify abstractly the important type theoretic properties of a "set theory".

3 Elements of category theory

It is natural to consider maps between categories, otherwise known as functors, and it is no surprise to discover that these organize themselves into a category Cat. What is a little more surprising is that there are also maps between functors. This means that Cat is in fact a category which is enriched in categories.

It is this 2-dimensional structure which gives rise the notion of an adjunction. These codify algebraically and precisely what a "universal property" is. As these occur all over mathematics – and, indeed, in computer science – providing an abstract understanding of them was a significant contribution of the development of categorical thought.

Before discussing all this it is useful to talk about maps in categories ...

3.1 Epics, monics, retractions, and sections

A map $f : A \longrightarrow B$ in a category $\mathbb{C}$ is **monic** (or sometimes a monomorphism) in case whenever $k_1 f = k_2 f$ then $k_1 = k_2$. Dual to the notion of a monic map is that of an epic map: a map $f : A \longrightarrow B$ in a category $\mathbb{C}$ is **epic** (or sometimes an epimorphism) in case whenever $fh_1 = fh_2$ then $h_1 = h_2$. The fact that a map is monic does not stop it from being epic as well: a map that is both epic and monic we shall refer to as being **bijic** (sometimes bijection).

A map $f : A \longrightarrow B$ is a **section** in case there is a map $f' : B \longrightarrow A$ such that $ff' = 1_A$. Dual to a section is a retraction: a map $g : A \longrightarrow B$ is a **retraction** in case there is a map $g' : B \longrightarrow A$ such that $g'g = 1_B$. It is quite possible for a map to be both a section and a retraction: such a map is called an **isomorphism**. Clearly identity maps are always isomorphisms.

We have the following result linking these properties:

Lemma 3.1 *In any category $\mathbb{C}$ it is the case that:*

(i) *The composition of monics is monic;*
(ii) *The composition of epics is epic;*
(iii) *If fg is monic then f is monic;*
(iv) *If fg is epic then g is epic;*
(v) *All sections are monic;*
(vi) *All retractions are epic;*
(vii) *The compositions of sections is a section;*
(viii) *The composition of retractions is a retraction;*
(ix) *If fg is a section then f is a section;*
(x) *If fg is a retraction then g is a retraction.*

Isomorphisms are rather special: if $f : A \longrightarrow B$ is a map we shall refer to a map $g : B \longrightarrow A$ such that $fg = 1_A$ as the **right inverse** of f. Similarly a map $h : B \longrightarrow A$ such that $hf = 1_B$ will be referred to as the **left inverse** of f. A section has a right inverse and is itself a left inverse, while a retraction is a right inverse and has a left inverse. An isomorphism has both a left inverse and a right inverse:

Lemma 3.2 *If $f : A \longrightarrow B$ has a left inverse h and a right inverse g then $h = g$.*

PROOF: Observe f is both epic and monic as it is both a section and a retraction. Thus, $fh = fh1_A = fhfg = f1_Bg = fg = 1_A$ so that h is also a right inverse of f. But then $fh = fg$ and as f is epic $h = g$. $\Box$

As the inverse of an isomorphism f is unique we shall denote it f^{-1}. We have the following alternative characterizations of isomorphisms:

Lemma 3.3 *The following are equivalent:*

(i) f *is a section and a retraction;*	(iii) f *is an epic section;*
(ii) f *is a monic retraction*	(iv) f *is an isomorphism;*

When a category has *every* map an isomorphism it is called a **groupoid**. All groupoids have a converse:

$$(\_)^{-1} : \mathbb{G} \longrightarrow \mathbb{G}; f \mapsto f^{-1}$$

An endomap $e : A \longrightarrow A$ is an **idempotent** if $ee = e$. Notice that if $h : A \longrightarrow B$ is a retraction with left inverse $h' : B \longrightarrow A$ then hh' is an idempotent as $hh'hh' = h1_Bh' = hh'$. We shall say that an idempotent e is **split** if there is a retraction h with left inverse h' such that $e = hh'$. While an idempotent may be split in many different ways there is an important a sense in which there is essentially just one splitting:

Lemma 3.4 *Suppose $e : A \longrightarrow A$ is an idempotent and $h_1 : A \longrightarrow B_1$ has left inverse h'_1 and $h_2 : A \longrightarrow B_2$ has left inverse h'_2, with $e = h_1h'_1 = h_2h'_2$ then there is a unique isomorphism $k : B_1 \longrightarrow B_2$ such that $h_1k = h_2$ and $kh'_2 = h'_1$.*

Proof: Set $k = h'_1h_2$ then the required identities hold. Furthermore k is an isomorphism as $k^{-1} = h'_2h_1$ (as $h'_1h_2h'_2h_1 = h'_1eh_1 = h'_1h_1h'_1h_1 = 1_{B_1}$ and similarly for the other composite $h'_2h_1h'_1h_2 = 1_{B_2}$). Suppose k' also satisfies $h_1k' = h_2$ then $h_1k' = h_1k$ and as h_1 is a retraction and therefore epic it follows that $k = k'$. □

We shall say that the splitting of an idempotent is *unique up to unique isomorphism*. It is certainly not the case that idempotents will generally split in a category, however, there is an important construction which allows one to freely split idempotents.

Let $\mathbb{C}$ be any category. Let $\mathsf{Split}(\mathbb{C})$ be the following category:

Objects: Idempotents e of $\mathbb{C}$;

Maps: $(e_1, f, e_2) : e_1 \longrightarrow e_2$ where $e_1 : A \longrightarrow A$ and $e_2 : B \longrightarrow B$ is a map $f : A \longrightarrow B$ in $\mathbb{C}$ such that $e_1fe_2 = f$;

Compositions As in $\mathbb{C}$ on the middle coordinate: $(e_1, f, e_2)(e_2, g; e_3) = (e_1, fg, e_3)$.

Identities: the identity for an idempotent is that idempotent $(e, e, e) : e \longrightarrow e$.

Now it is not hard to show that this is a category. What is interesting about this category is that all the idempotents in it split:

Proposition 3.5 *Let $\mathbb{C}$ be any category then $\mathbb{C}$ is a full subcategory of $\mathsf{Split}(\mathbb{C})$ and all idempotents split in $\mathsf{Split}(\mathbb{C})$.*

Proof: We may regard $\mathbb{C}$ as a full subcategory of Split by letting the identity maps (which are certainly idempotent) represent the objects of $\mathbb{C}$ in $\mathsf{Split}(\mathbb{C})$.

Suppose $(e, k, e) : e \longrightarrow e$ is an idempotent in $\mathsf{Split}(\mathbb{C})$ then k is an idempotent in $\mathbb{C}$. But then we have maps $(e, k, k) : e \longrightarrow k$ and $(k, k, e) : k \longrightarrow e$ in $\mathsf{Split}(\mathbb{C})$ and it is easy to check that these provide a splitting for (e, k, e). □

To motivate why this construction may be of more than passing interest consider the category of partial recursive functions on the natural numbers, Rec. Each enumerable set may be characterized by an idempotent which is the computation which returns the element unchanged when it is in the recursively enumerable set but simply does not terminate on elements outside. In $\mathsf{Split}(\mathsf{Rec})$ there is an object or type for each enumerable set. Thus, this gives an example of a unityped system from which can be constructed a very rich type system.

3.2 Functors

A **functor** is a map of categories $F : \mathbb{C}_1 \longrightarrow \mathbb{C}_2$ which consists of a map F_0 of the objects and a map F_1 of the maps (we shall consistently drop these subscripts when the intended domain is clear) such that

- $\partial_0(F_1(f)) = F_0(\partial_0(f))$ and $\partial_1(F_1(f)) = F_0(\partial_1(f))$;
- $F_1(1_A) = 1_{F_0(A)}$;
- $F_1(fg) = F_1(f)F_1(g)$;

Clearly every category has an identity functor and the composition of functors is associative so that the following is immediate.

Lemma 3.6 *Categories and functors form a category* Cat*.*

If $F : \mathbb{C} \longrightarrow \mathbb{D}$ is a functor then $F^{\mathrm{op}} : \mathbb{C}^{\mathrm{op}} \longrightarrow \mathbb{D}^{\mathrm{op}}$ is a functor. A functor from the opposite of a category, $\mathbb{C}^{\mathrm{op}}$, is often called a **contravariant** functor from $\mathbb{C}$ as the maps get reversed (the *contra-* prefix). An ordinary functor (with domain $\mathbb{C}$) is sometimes called a **covariant** functor to emphasize that there is no twisting of maps involved.

A diagram in a category is a collection of arrows and objects satisfying certain composability relations. Suppose that P is a property involving a relationship between the maps and objects of a diagram. Examples of such properties, for the diagram consisting of a single map, that we have met so far are

- $P(f : A \longrightarrow B) = \exists g : B \longrightarrow A. fg = 1_A$, which says f is a section,
- $P(f : A \longrightarrow B) = \forall h, h' : X \longrightarrow A. hf = h'f \Rightarrow h = h'$ which says f is monic.

We shall say a functor F **preserves** a property P if whenever P holds of the arrows $f_1, .., f_n$ and objects $A_1, .., A_n$ then P holds of the arrows $F(f_1), .., F(f_n)$ and objects $F(A_1), .., F(A_n)$ in the codomain category.

Thus, a functor always preserves sections (and isomorphisms) but does not, in general, preserve monics (or epics). While functors may not preserve a property we are often interested in restriction attention to those functors which do preserve that property. Thus, while funcors do not in general preserve monics, we may well be interested in functors which do preserve monics.

We say that a functor **reflects** a property P of a diagram if whenever the property holds of the image under F of the diagram then the property must have held for the original arrows. (Notice that as functors do not in general even reflect composability, the requirement that the composability relations already hold in the domain is now important).

Thus, for example while functors always preserves isomorphisms they do not in general reflect isomorphisms.

The enriched view of a category gives another important view of a functor $F : \mathbb{C} \longrightarrow \mathbb{D}$ as being provided by a family of maps

$$F_{AB} : \mathbb{C}(A, B) \longrightarrow \mathbb{D}(F_0(A), F_0(B))$$

which must satisfy the following two diagrams:

$$\begin{tikzcd}
\mathbb{C}(A,B) \times \mathbb{C}(B,C) \arrow[r, "m_{ABC}"] \arrow[d, "F_{AB} \times F_{BC}"'] & \mathbb{C}(A,C) \arrow[d, "F_{AC}"] \\
\mathbb{D}(F_0(A), F0(B)) \times \mathbb{D}(F_0(B), F_0(C)) \arrow[r, "m_{F_0(A)F_0(B)F_0(C)}"'] & \mathbb{D}(F_0(A), f_0(C))
\end{tikzcd}$$

$$\begin{tikzcd}
1 \arrow[r, "1_A"] \arrow[dr, "1_{F_0(A)}"'] & \mathbb{C}(A,A) \arrow[d, "F_{AA}"] \\
& \mathbb{D}(F_0(A), F_0(A))
\end{tikzcd}$$

Using this point of view the idea of a **faithful** functor can be easily explained: it is a functor all of whose maps F_{AB} are injective. Similarly a **full** functor is one all of whose maps F_{AB} are surjective.

Notice that a category has a faithful functor to the final category (one object one map) if and only if it is a preorder! As we shall see in a moment, every (small) category has a full functor to a preorder.

Given a category $\mathbb{C}$ a **congruence** is a given by an equivalence relation on each hom-set satisfying $f \sim g$ and then $hfk \sim hgk$. Given a congruence we may form a new category with the same objects $\mathbb{C}/\sim$ whose hom-sets are the $\sim$–equivalence classes: $\mathbb{C}(A, B)/\sim_{AB}$. This is called the **quotient category** for the congruence $\sim$:

Objects: Those of $\mathbb{C}$;

Maps: Equivalence classes of maps under $\sim$ we may write these as $[f] : A \longrightarrow B$ where $f : A \longrightarrow B$ is a representative member of the equivalence class.

Identities: $[1_A] : A \longrightarrow A$;

Composition: $[f][g] = [fg]$.

Now it is not immediate that this is indeed a category. It is clear that the identities have the correct properties but what is not clear is that composition so defined is even a function. Specifically it is not clear that if we have $[f_1] = [f_2]$ and $[g_1] = [g_2]$ (in other words $f_1 \sim f_2$ and $g_1 \sim g_2$) that necessarily f_1g_1 and f_2g_2 are even related! Of course this is exactly where we must use the special property of a congruence, here is the argument:

$$\dfrac{\dfrac{f_1 \sim f_2}{f_1g_1 \sim f_2g_1}\ \text{Compose } 1\_g_1 \qquad \dfrac{g_1 \sim g_2}{f_2g_1 \sim f_2g_2}\ \text{Compose } f_2\_1}{f_1g_1 \sim f_2g_2}\ \text{Transitive}$$

There is an obvious functor $Q_\sim : \mathbb{C} \longrightarrow \mathbb{C}/\sim$ which is the identity on objects and carries a map f to its $\sim$–equivalence class $[f]$. This is clearly always a full functor.

There is an important way in which conguences arise:

Lemma 3.7 *If $F : \mathbb{X} \longrightarrow \mathbb{Y}$ is a functor then the relation on parallel arrows $f \sim_F g \Leftrightarrow F(f) = F(g)$ is a congruence. Furthermore F can be factorized as $Q_{\sim_F} F'$ where $Q_{\sim_F}$ full and the identity on objects and F' is faithful.*

Proof: We must show that $hfk \sim_F hgk$ but $F(hfk) = F(h)f(f)F(k) = F(h)F(g)F(k) = F(hgk)$.

We may define a functor $F' : \mathbb{X}/\sim_F \longrightarrow \mathbb{Y}$ by $F'([f]) = F(f)$. This is clearly well-defined and, furthermore, a faithful functor as if $F'([f]) = F'([g])$ then $F(f) = F(g)$. $\square$

Note that faithful functors need not, as functors, be monic by any means.

An important construction is as follow: let $F : \mathbb{A} \longrightarrow \mathbb{C}$ and $G : \mathbb{B} \longrightarrow \mathbb{C}$ be functors then we may form the **comma category**[4] of F over G which is denoted F/G as follows:

Objects: Triples $(A, F(A) \xrightarrow{f} G(B), B)$;

Maps: Quadruples $((A, f, B), a : A \longrightarrow A', b : B \longrightarrow B', (A', f', B'))$ where (A, f, B) and (A', f', B') are objects as above and a and b are maps which render

$$\begin{tikzcd}
F(A) \arrow[r, "f"] \arrow[d, "F(a)"'] & G(B) \arrow[d, "G(b)"] \\
F(A') \arrow[r, "f'"'] & G(B')
\end{tikzcd}$$

commutative.

Identities: $((A, f, B), 1_A, 1_B, (A, f, B))$;

Composition: With $((A_1, f_1, B_1), a, b, (A_2, f_2, B_2))((A_2, f_2, B_2), a', b', (A_3, f_3, B_3))$ defined to be $((A_1, f_1, B_1), aa', bb', (A_3, f_3, B_3))$ where the required commutativity is provided by:

$$\begin{tikzcd}
F(A_1) \arrow[r, "f_1"] \arrow[d, "F(a)"'] & G(B_1) \arrow[d, "G(b)"] \\
F(A_2) \arrow[r, "f_2"'] \arrow[d, "F(a')"'] & G(B_2) \arrow[d, "G(b')"] \\
F(A_3) \arrow[r, "f_3"'] & G(B_3)
\end{tikzcd}$$

Again the fact that this is a category must be checked although it follows easily from the fact that F and G are functors and that $\mathbb{A}$ and $\mathbb{B}$ are categories.

[4]This is a bit of a misnomer considering the notation chosen here! The original notation for this construction involved a comma.

Let us first observe that, indeed, the slice category construction is a special case of this construction. First notice that an object in a category corresponds precisely to a functor from the final category $\mathbf{1}$. Thus $\mathbb{C}/A$ can be read as the comma category using the identity functor on $\mathbb{C}$, $\mathbb{C} = 1_{\mathbb{C}} : \mathbb{C} \to \mathbb{C}$ and the functor from the final category $A : \mathbf{1} \to \mathbb{C}$. It is easy now to see that this comma category is just the slice category.

The comma category F/G has some obvious associated functors

$$\Pi_0 : F/G \longrightarrow \mathbb{A}; ((A, f, B), a, b, (A', f', B')) \mapsto a$$

$$\Pi_1 : F/G \longrightarrow \mathbb{B}; ((A, f, B), a, b, (A', f', B')) \mapsto b$$

There is also a canonical natural transformation

$$\alpha : \Pi_0; F \longrightarrow \Pi_1; G$$

where $\alpha_{(A,f,B)} = f : F(A) \longrightarrow G(B)$. Thus, the comma category gives a pasting square:

$$\begin{tikzcd}
F/G \arrow[r, "\Pi_0"] \arrow[d, "\Pi_1"'] & \mathbb{A} \arrow[d, "F"] \arrow[dl, Rightarrow, "\alpha"] \\
\mathbb{B} \arrow[r, "G"'] & \mathbb{C}
\end{tikzcd}$$

3.3 Natural transformations

Given two functors $F, G : \mathbb{C} \longrightarrow \mathbb{D}$ a **natural transformation** (or just a transformation) $\alpha : F \Rightarrow G$ is a family of maps $\alpha_C : F(C) \longrightarrow G(C)$, indexed by the objects of $\mathbb{C}$, in $\mathbb{D}$ such that for every map $f : C_1 \longrightarrow C_2$ in $\mathbb{D}$ the following diagram commutes:

$$\begin{tikzcd}
F(C_1) \arrow[r, "F(f)"] \arrow[d, "\alpha_{C_1}"'] & F(C_2) \arrow[d, "\alpha_{C_2}"] \\
G(C_1) \arrow[r, "G(f)"'] & G(C_2)
\end{tikzcd}$$

Our first observation is that this means that $\mathsf{Cat}(\mathbb{C}, \mathbb{D})$, which often written $\mathbb{D}^{\mathbb{C}}$, can be given the structure of a category. Ultimately this means that Cat is a Cat-enriched category – these are also known as **2-categories**.

Proposition 3.8 $\mathsf{Cat}(\mathbb{C}, \mathbb{D})$ *is a category with objects functors and maps natural transformations.*

Proof: First notice that every functor has an identity transformation given by $1_{F(A)} : F(A) \longrightarrow F(A)$. To compose natural transformations we simply define $(\alpha\beta)_A = \alpha_A \beta_A$: if this composition works then it is associative. It remains to check that the above requirement on the composite transformation holds: this can be seen by pasting the transformation squares together.

$$
\begin{tikzcd}
F(C_1) \arrow[r, "F(f)"] \arrow[d, "\alpha_{C_1}"'] & F(C_2) \arrow[d, "\alpha_{C_2}"] \\
G(C_1) \arrow[r, "G(f)"'] \arrow[d, "\beta_{C_1}"'] & G(C_2) \arrow[d, "\beta_{C_1}"] \\
H(C_1) \arrow[r, "H(f)"'] & H(C_2)
\end{tikzcd}
$$

□

In order to conveniently manipulate functors and natural transformations it is useful to develop the 2-categorical view of them. As there are many levels of activity in a general 2-category it is useful to introduce a special notation: the objects of a 2-category (in Cat these are categories) are often called **0-cells** while the maps (or functors) are called **1-cells** and the transformations between the 1-cells are called **2-cells**.

In this enriched view of Cat there are two sorts of composition: the composition of natural transformations, $\alpha\beta$, written by juxtaposition and the composition of functors which we shall write with a semicolon, $F;G$.

Proposition 3.9 Cat *is a* Cat*–enriched category.*

Th proof is left as an exercise. The main difficulty of the proof of this is to show that functor composition

$$\_;\_ : \mathsf{Cat}(\mathbb{A},\mathbb{B}) \times \mathsf{Cat}(\mathbb{B},\mathbb{C}) \longrightarrow \mathsf{Cat}(\mathbb{A},\mathbb{C})$$

is a functor of two arguments.

As the semicolon is an enriched composition we may apply it to the natural transformations as well as to the functors so that if $\alpha : F \longrightarrow G$ and $\alpha' : I \longrightarrow J$ then

$$\alpha;\alpha' : F;I \longrightarrow G;J$$

As a consequence of the enrichment one will then have the equation, which is called the **interchange law** which is a direct consequence of the fact that the composition is a functor of two arguments.

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

In order to represent these compositions we shall use pasting diagrams: these diagrams are very useful as interchange and associativity laws become graphical equalities. This means that one can immediately "see" that two composites are equal with a minimal amount of manipulation. Below is the representation of the interchange law:

$$
\begin{tikzcd}
 & F \arrow[dr, bend left] \arrow[d, phantom, "\Downarrow\alpha"] & & I \arrow[dr, bend left] \arrow[d, phantom, "\Downarrow\alpha'"] & \\
\mathbb{A} \arrow[ur, bend left, no head] \arrow[r, no head] \arrow[dr, bend right, no head] & G \arrow[r] \arrow[d, phantom, "\Downarrow\beta"] & \mathbb{B} \arrow[ur, bend left, no head] \arrow[r, no head] \arrow[dr, bend right, no head] & J \arrow[r] \arrow[d, phantom, "\Downarrow\beta'"] & \mathbb{C} \\
 & H \arrow[ur, bend right] & & K \arrow[ur, bend right] &
\end{tikzcd}
$$

Pasting diagrams may be translated back into 2-categorical notation but not in an unambiguous way as there are many ways of representing a given pasting diagram as in the 2-categorical "combinator" notation we have introduced. For example the above interchange can also be written:

$$
\begin{array}{l}
(\alpha; J)(\beta; \alpha')(H; \beta') \\
(\alpha; (\alpha'\beta'))(\beta; K) \\
(F; \alpha')(\alpha; \beta')(\beta; K) \\
....
\end{array}
$$

It is important to realize that not every pasting diagram is legal. Here are a description of the conditions to ensure that it is legal:

(a) The diagram must be planar with nodes labeled by objects (0-cells) arrows by functors (1-cells) and regions by (2-cells);

(b) There must be a start node (object or 0-cell) and an end node (object or 0-cell);

(c) The maps (functors or 1-cell) must form an acyclic directed graph;

(d) Every map (functor or 1-cell) must be on a path from the start node to the end node, which we will call a trip.

(e) For the transformations (2-cells) there must be a starting trip (path from start node to end node) and an ending trip.

(f) Every arrow must be either on the end trip or in the domain of exactly one transformation (but not both);

(g) Every arrow must be either on the start trip or in the codomain of exactly one transformation (but not both).

Notice that the conditions (b), (c), and (d) must already be true of commuting diagrams.

These systems are actually familiar in computer science: they occur as in formal languages. The rules of a context free language may be seen as 2-cells. The system above allows us a notation for describing the derivations in the language.

3.4 Adjoints

In this section we introduce one of the most important concepts of basic category theory: an adjunction. This notion occurs all over mathematics and plays an absolutely fundamental role in understanding how abstract settings, such as a programming languages, are constructed.

3.4.1 The universal property

Let $G : \mathbb{Y} \longrightarrow \mathbb{X}$ be a functor and X an object of $\mathbb{X}$, then an object $U \in \mathbb{Y}$ together with a map $\eta_X : X \longrightarrow G(U)$ is a **universal pair** for the functor G at the object X if for any $f : X \longrightarrow G(Y)$

there is a unique $f^\sharp : U \longrightarrow Y$ such that

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "f"'] & G(U) \arrow[d, dotted, "G(f^\sharp)"] \\ & G(Y) \end{tikzcd}$$

commutes.

It is useful to have in mind a particular instance of this universal property. A nice example is as follows: let $\mathbb{X}$ be the category of directed graphs and $\mathbb{Y}$ the category of categories, let the functor G be the "underlying functor" which forgets the compositional structure of a category, that is regards a category as no more than the "underlying" directed graph. The map which takes a directed graph and embeds it into the graph underlying the path category as the singleton paths (paths of length one) has the universal property for this "underlying" functor.

Consider a map of directed graphs into the graph underlying a category, $h : G \longrightarrow U(\mathbb{C})$, we can extend it uniquely to a functor from the path category to the category as follows. Let $h^\sharp : \mathsf{Path}(G) \longrightarrow \mathbb{C}$ be defined on arrows by

$$h^\sharp(A, [a_1, .., a_n], B) = h(a_1)..h(a_n) : h(A) \longrightarrow h(B)$$

then it is easy to check that this is a functor and that is uniquely determined by h. Before proceeding, it is well worth doing the exercise to check that this does really work.

In the next section on limits and colimits we shall be discussing another important situation in which these universal properties hold. However, we shall continue here to develop the theory of these universal properties first. The reader who needs more concrete motivation may like to start reading the section on limits and colimits in parallel.

We first make the simple but crucial observation:

Lemma 3.10 *If (η_X, U) and (η'_X, U') are universal at X for $G : \mathbb{X} \longrightarrow \mathbb{Y}$ then there is a unique isomorphism $\alpha : U \longrightarrow U'$ such that $\eta_X G(\alpha) = \eta'_X$.*

PROOF: We may define α as the unique map $(\eta'_X)^\sharp$:

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "\eta'_X"'] & G(U) \arrow[d, dotted, "G(\alpha)"] \\ & G(U') \end{tikzcd}$$

and by swapping the role of U and U' we obtain a $\beta : U' \longrightarrow U$ using the universal property of (U', η'_X). However, now

$$\eta_X G(\alpha\beta) = \eta_X G(\alpha) G(\beta) = \eta'_X G(\beta) = \eta_X$$

so $\alpha\beta = \eta_X^\sharp$ and $\eta_X^\sharp = 1_U$ so $\alpha\beta = 1_U$ and by a similar argument $\beta\alpha = 1_{U'}$. $\square$

Thus a universal pair (U, η) for a functor G at an object X is determined upto a unique isomorphism.

We may also use the Yoneda lemma to re-express this property: if $F : \mathbb{Y} \longrightarrow \mathsf{Sets}$ and there is a natural isomorphism $\alpha : \mathbb{Y}(U, \_) \to F$ then F is said to be **representable** with **universal element** $\alpha_U(1_U) \in F(U)$. In this case we have:

$$\alpha : \mathbb{Y}(U, \_) \Rightarrow \mathbb{X}(X, G(\_)) : \mathbb{Y} \longrightarrow \mathsf{Sets}$$

where $\alpha_Y(h) = \eta_X G(h)$ and the inverse maps sends f to $f^\sharp$. Thus the universal property above can be re-expressed by saying that $\mathbb{X}(X, G(\_))$ is a representable functor with universal element $\eta_X \in \mathbb{X}(X, G(U))$. This reminds us that specifying the map η_X actually suffices to determine the situation.

Now it is certainly not always the case that, for a functor G there will be universal pairs $(F(X), \eta_X)$ at each object X, however, if this is the case we have:

Proposition 3.11 *Let $G : \mathbb{X} \longrightarrow \mathbb{Y}$ be a functor such that for each $X \in \mathbb{X}$ there is a universal pair $(F(X), \eta_X)$ then:*

- *F is a functor with $F(g) = (g\eta)^\sharp$;*
- *$\eta_X : X \longrightarrow G(F(X))$ is a natural transformation;*
- *$\epsilon_Y = 1^\sharp_{G(Y)} : F(G(Y)) \longrightarrow Y$ is a natural transformation;*
- *The* **triangle equalities** *$\eta_{G(Y)} G(\epsilon_Y) = 1_{G(Y)}$ and $F(\eta_X)\epsilon_{F(X)} = 1_{F(X)}$ hold.*

Conversely given functors F and G with transformations η and ϵ, as above, which satisfy the triangle identities, then each $(F(X), \eta_X)$ is universal for G at X.

Proof: We start by verifying that F so defined is a functor it will then be immediate that η is a natural transformation. We must verify that F preserves identities, which is the observation that $(\eta_X)^\sharp = 1$, and that $F(f)F(g) = F(fg)$. For the latter we have that, as

$$\eta_X G((f\eta_Y)^\sharp (g\eta_Z)^\sharp) = \eta_X G((f\eta_Y)^\sharp) G((g\eta_Z)^\sharp) = f\eta_Y G((g\eta_Z)^\sharp) = fg\eta_Z$$

that $(f\eta_Y)^\sharp (g\eta_Z)^\sharp = (fg\eta_Z)^\sharp$.

It remains to prove that $\epsilon_Y = 1^\sharp_{G(Y)}$ is a natural transformation and that the second triangle equality holds – the first is immediate from the definition of ϵ. For the naturality of ϵ we have for a map $f : Y \longrightarrow Y'$ that

$$G(f) = \eta_{G(Y)} G((1_{G(Y)})^\sharp) G(f) = \eta_{G(Y)} G(\epsilon_Y) G(f)$$

so that $G(f)^\sharp = \epsilon_Y f$. Similarly,

$$G(f) = G(f)\eta_{G(Y')} G((1_{G(Y')})^\sharp) = \eta_{G(Y)} G(F(G(f)) G((1_{G(Y')})^\sharp) = \eta_{G(Y)} G(F(G(f)) G(\epsilon_{Y'}))$$

so that $G(f)^\sharp = G(F(G(f)))\epsilon_{Y'}$. Thus, ϵ is natural.

Finally, for the second triangle equality we have:

$$\eta_X G(F(\eta_X)\epsilon_{F(X)}) = \eta_X \eta_{F(X)} \epsilon_{F(X)}) = \eta_X$$

so that $F(\eta_X)\epsilon_{F(X)} = 1_{F(X)} = \eta_X^\sharp$.

For the converse, suppose we have (F, G, η, ϵ) and the triangle equalities then, given $f : X \longrightarrow G(Y)$ we can set $f^\sharp = F(f)\epsilon_Y$ then, using the naturality of η and the first triangle equality we have:

$$\eta_X G(f^\sharp) = \eta_X G(F(f)\epsilon_Y) = f\eta_{G(X)}\epsilon_Y = f.$$

To show uniqueness we suppose $\eta_X G(h) = f$ then

$$h = F(\eta_X)\epsilon_{F(X)}h = F(\eta_X)F(G(h))\epsilon_Y = F(\eta_X G(h))\epsilon_Y = F(f)\epsilon_Y = f^\sharp.$$

□

We say that F is **left adjoint** to G (equally G is **right adjoint** to F: notice that this is also the dual statement) in case, as in the proposition there are natural transformations η (called the **unit** of the adjunction) and ϵ (called the **counit** of the adjunction) satisfying the triangle equalities. We write this situation as:

$$(\eta, \epsilon) : F \vdash G : \mathbb{X} \longrightarrow \mathbb{Y}.$$

Recall there is a dual concept: let $F : \mathbb{X} \longrightarrow \mathbb{Y}$ be a functor and Y an object of $\mathbb{Y}$, then an object $V \in \mathbb{X}$ together with a map $\epsilon_Y : F(V) \longrightarrow Y$ is a **couniversal pair** for the functor F at the object Y if for any $g : F(X) \longrightarrow Y$ there is a unique $g^\flat : X \longrightarrow V$ such that

$$\begin{tikzcd} F(X) \arrow[r, "g"] \arrow[d, dotted, "F(g^\flat)"'] & Y \\ F(V) \arrow[ur, "\epsilon_Y"'] & \end{tikzcd}$$

commutes.

When F has for each $Y \in \mathbb{Y}$ a couniversal pair $(G(Y), \epsilon_Y)$ then this also gives rise to an adjunction but this time the constructed functor G is right adjoint to F. In terms of universal elements we now have a natural isomorphism:

$$\alpha : \mathbb{X}(\_, V) \Rightarrow \mathbb{Y}(F(\_), Y) : \mathbb{X}^{\mathrm{op}} \longrightarrow \mathsf{Sets}.$$

which is determined by the **couniversal element** ϵ_Y. The fact, that this is a dual concept is apparent in this formulation as we have replaced $\mathbb{X}$ by $\mathbb{X}^{\mathrm{op}}$.

3.5 Basic properties of adjoints

There is the following characterization of an adjoints which here we state for Sets–enriched categories. The result is true more generally for categories enriched elsewhere (e.g. Cat-enriched categories for example).

Theorem 3.12 *The following are equivalent for* Sets*–enriched categories and functors* $F : \mathbb{X} \longrightarrow \mathbb{Y}$ *and* $G : \mathbb{Y} \longrightarrow \mathbb{X}$*:*

(i) *An adjoint* $(\eta, \epsilon) : F \dashv G : \mathbb{X} \longrightarrow \mathbb{Y}$*;*

(ii) *Two combinators* $(\_)^\flat$ *and* $(\_)^\sharp$ *where*

$$(g : F(A) \longrightarrow B)^\flat : A \longrightarrow G(B) \quad \textit{and} \quad (f : A \longrightarrow G(B))^\sharp : F(A) \longrightarrow B$$

such that

- $(f^\sharp)^\flat = f$ *and* $(g^\flat)^\sharp = g$*,*
- $(F(h)fk)^\flat = hf^\flat G(k)$ *and* $(k' g G(h'))^\sharp = F(k')g^\sharp h'$*;*

(iii) *A natural isomorphism* $(\_)^\flat : \mathbb{Y}(F(\_), \_) \Rightarrow \mathbb{X}(\_, G(\_)) : \mathbb{X}^{\mathrm{op}} \times \mathbb{Y} \longrightarrow \mathsf{Sets}$*;*

(iv) *An isomorphism of categories* $(\_)^\sharp : \mathbb{X}/G \longrightarrow \mathbb{F}/\mathbb{Y}$ *such that* $(\_)^\sharp; \Pi_0 = \Pi_0$ *and* $(\_)^\sharp; \Pi_1 = \Pi_1$*;*

(v) *For each* $X \in \mathbb{X}$ *there is a pair* $(F(X), \eta_X)$ *which is universal for* G *at* X *such that* $F(h) = (h\eta)^\sharp$*;*

(vi) *For each* $Y \in \mathbb{Y}$ *there is a pair* $(G(Y), \epsilon_Y)$ *which is couniversal for* F *at* Y *such that* $G(k) = (\epsilon k)^\flat$*.*

Proof: We already know that the universal properties (and couniversal properties for that matter) are equivalent to giving adjoints and it is clear that *(iv)* and *(v)* are dual. Thus it remains to prove the following:

(i) $\Rightarrow$ ***(ii)*** : Given the above adjunction we may define $h^\flat = \eta G(h)$ and $g^\sharp = F(g)\epsilon$. Now these are inverse as

$$(h^\flat)^\sharp = (\eta G(h))^\sharp = F(\eta G(h))\epsilon = F(\eta)\epsilon h = h$$

and the argument for $(f^\sharp)^\flat = f$ is dual. Now notice that

$$(F(h)fk)^\flat = \eta G(F(h)fk) = h\eta G(f)G(k) = hf^\flat G(k)$$

and the other identity follows by the dual argument.

(ii) $\Leftrightarrow$ ***(iii)*** : The "sharp" and "flat" now are going to be viewed as natural isomorphisms; the announced properties of sharp and flat are exactly saying that regarded as transformations they are natural!

(ii) $\Leftrightarrow$ ***(iii)*** : The sharp and flat are now going to be regarded as functors. The commuting square:

$$\begin{tikzcd} X \arrow[r, "h"] \arrow[d, "f"'] & G(Y) \arrow[d, "G(g)"] \\ X' \arrow[r, "h'"'] & G(Y') \end{tikzcd}$$

is a map $(f, g) : h \longrightarrow h'$ in $\mathbb{X}/G$ under the "sharp" combinator it is carried to the square:

$$\begin{tikzcd} F(X) \arrow[r, "h^\sharp"] \arrow[d, "F(f)"'] & Y \arrow[d, "g"] \\ F(X') \arrow[r, "(h')^\sharp"'] & Y' \end{tikzcd}$$

which commutes as $F(f)(h')^\sharp = (fh')^\sharp = (hG(g))^\sharp = h^\sharp g$. Thus, "sharp" does indeed give a functor which clearly has an inverse and, furthermore, this functor commutes with the projections from the two slice categories as required.

Conversely if we are given such an isomorphism of categories then by "sharping" the following two diagrams:

$$\begin{tikzcd} X \arrow[r, "fh"] \arrow[d, "f"'] & G(Y) \arrow[d, equal] \\ X' \arrow[r, "h"'] & G(Y) \end{tikzcd} \qquad \begin{tikzcd} X' \arrow[r, "h"] \arrow[d, equal] & G(Y) \arrow[d, "G(k)"] \\ X' \arrow[r, "hG(k)"'] & G(Y') \end{tikzcd}$$

we obtain:

$$\begin{tikzcd} F(X) \arrow[r, "(fh)^\sharp"] \arrow[d, "F(f)"'] & Y \arrow[d, equal] \\ F(X') \arrow[r, "h^\sharp"'] & Y \end{tikzcd} \qquad \begin{tikzcd} F(X') \arrow[r, "h^\sharp"] \arrow[d, equal] & Y \arrow[d, "k"] \\ F(X') \arrow[r, "(hG(k))^\sharp"'] & Y' \end{tikzcd}$$

Which gives the combinator identity

$$(k'gG(h'))^\sharp = F(k)(gG(h))^\sharp = F(k')g^\sharp h'$$

in two steps. The other required identity for the "flat" functor follows dually.

(ii) $\Rightarrow$ ***(iv)*** : We must now recover the universal property from the "sharp" and "flat" combinators: set $\eta_X = (1_{F(X)})^\flat$ then given $f : X \longrightarrow G(Y)$ we have show

$$(1_{F(X)})^\flat G(f^\sharp) = (1_{F(X)} f^\sharp)^\flat = (f^\sharp)^\flat = f$$

and it remains to show that $f^\sharp$ is unique. So suppose $f = (1_{F(X)})^\flat G(g)$ then

$$f^\sharp = ((1_{F(X)})^\flat G(g))^\sharp = ((1_{F(X)} g)^\flat)^\sharp = (g^\flat)^\sharp = g.$$

Finally we note

$$(f\eta)^\sharp = (f(1_{F(X')})^\flat)^\sharp = F(f)((1_{F(X')})^\flat)^\sharp = F(f)1_{F(X')} = F(f)$$

So that the definition of F agrees with that given by the universal property.

$\square$

We shall often write the two-way transformation as a two-way logical inference:

$$\dfrac{X \xrightarrow{\;f = g^\flat\;} G(Y)}{F(X) \xrightarrow[\;g = f^\sharp\;]{} Y} \; F \dashv G$$

This allows us to move from maps $h : F(X) \longrightarrow Y$ to $h^\flat : X \longrightarrow G(Y)$ and back $f : X \longrightarrow G(Y)$ to $f^\sharp : F(X) \longrightarrow Y$ where these moves are mutually inverse and natural. This style of handling adjoints as we shall see shortly allows a convenient way of reasoning about the effect of adjoints.

Left adjoints preserve universal pairs and dually, although we do not formally state it, right adjoints preserve couniversal pairs:

Proposition 3.13 *If $G : \mathbb{Y} \longrightarrow \mathbb{X}$ and $(\eta', \epsilon') : H \dashv K : \mathbb{Y} \longrightarrow \mathbb{Z}$ and (U, η') is a universal pair for then $(H(U), \eta_U \eta')$ is a universal pair for $K; G$.*

Proof: Suppose $f : X \longrightarrow G(K(Z))$ then we obtain a unique $f^{\sharp'} : U \longrightarrow K(Z)$ now we may use the universal property of the adjunction to give a unique $(f^{\sharp'})^\sharp : H(U) \longrightarrow Z$ such that $\eta' G(f^{\sharp'})^\sharp) = f$.

$$\cfrac{\cfrac{X \xrightarrow{f} G(K(Z))}{U \xrightarrow{f^{\sharp'}} K(Z)}\ \text{Universal}}{H(U) \xrightarrow[(f^{\sharp'})^\sharp]{} Z}\ H \dashv K$$

$\square$

This, in particular, means that if G has a left adjoint then $K; G$ will have a left adjoint whose unit is $\eta_X \eta'_{F(X)}$. This gives:

Corollary 3.14 *If F and G are left (right) adjoints then $F; G$ is a left (right) adjoint.*

Proof: While the above remarks establish that this observation is correct we may also see this very directly by using the inferences:

$$\cfrac{\cfrac{X \xrightarrow{f} G(K(Z))}{F(X) \xrightarrow{f^{\sharp'}} K(Z)}\ F \dashv G}{H(F(X)) \xrightarrow[(f^{\sharp'})^\sharp]{} Z}\ H \dashv K$$

$\square$

3.6 Basic limits and colimits

Perhaps the most basic and most pervasive examples of universal and couniversal pairs arise through limits and colimits.

3.6.1 Initial and final objects

An **initial object** in a category $\mathbb{C}$ is an object which has exactly one map to every object (including itself) in the category. We shall often denote an initial object as the numeral 0 to remind us that it is a starting point and denote the unique map as $?_A : 0 \longrightarrow A$.

In Sets the initial object is the empty set, in vector spaces it is the 0-dimensional vector space, and in Cat it is the empty category.

Dual to an initial object is a **final object**: a final object in a category $\mathbb{C}$ is an object to which every object has exactly one map. We shall often denote the final object by the numeral 1 and the unique map by $!_A : A \longrightarrow 1$.

In Sets the final object is the one element set, in vector spaces the final object is the same as the initial object (that is the 0-dimensional vector space) and in Cat it is the category with one object and one arrow.

A simple observation is:

Lemma 3.15 *If K and K' are initial in $\mathbb{C}$ then there is a unique isomorphism $\alpha : K \longrightarrow K'$.*

Proof: As K is initial there is exactly one map $\alpha : K \longrightarrow K'$. Conversely, as K' is initial there is a unique map $\alpha' : K' \longrightarrow K$. This map is the inverse of α as $\alpha\alpha' : K \longrightarrow K$ is the unique endo-map on K namely the identity and similarly we obtain $\alpha'\alpha = 1'_K$. ☐

Thus initial objects (and by duality final objects) are unique up to unique isomorphism.

While there can only be one map to a final object there can be many maps from a final object to a given object (consider Sets for example). These maps are often called **elements** and we make the following observation:

Lemma 3.16 *Elements in any category are sections.*

Proof: An element is a map $a : 1 \longrightarrow A$ and has a right inverse $! : A \longrightarrow 1$. ☐

One way to view an object in a category is as a functor from the final category.

Proposition 3.17 *An object K in a category $\mathbb{C}$ is an initial object if and only if $K \dashv ! : 1 \longrightarrow \mathbb{C}$.*

Proof: Consider the universal diagram

$$\begin{tikzcd} \star \arrow[r, "1_\star"] \arrow[dr, "1_\star"'] & !(K) \arrow[d, dotted, "!(1_\star^\sharp)"] \\ & !(X) \end{tikzcd}$$

As the final category only has one map each object must have a unique map from the initial object.

It is also clear that the counit is an isomorphism as it is the only map of the final category. This means that K, the functor, is full and faithful. ☐

3.6.2 Binary products and coproducts

Perhaps one of the most fundamental structures a category can have is a product. In fact, it is so fundamental we have already assumed several times that the reader knew what a product was in order to facilitate the development. Let A and B be any two objects in a category then a product of A and B is an object, often written $A \times B$ equipped with two maps $\pi_0 : A \times B \longrightarrow A$ and $\pi_1 : A \times B \longrightarrow B$ such that given any object W with two maps $f : W \longrightarrow A$ and $g : W \longrightarrow B$ there is a unique map, often written $\langle f, g\rangle : W \longrightarrow A \times B$, such that $\langle f, g\rangle\pi_0 = f$ and $\langle f, g\rangle\pi_1 = g$.

The maps π_0 and π_1 are called **projections**.

This can be depicted graphically as

$$
\begin{tikzcd}
 & & A \\
W \arrow[rru, "f"] \arrow[r, dotted, "{\langle f,g \rangle}"] \arrow[rrd, "g"'] & A \times B \arrow[ru, "\pi_0"] \arrow[rd, "\pi_1"'] & \\
 & & B
\end{tikzcd}
$$

The product in Sets is the cartesian product, in vector spaces is what is often called the "direct sum" (the dimensions of the vector spaces are added), and in Cat is the product of categories.

This time we shall start by describing the couniversal property which characterizes the product from this we can then interpret the sense in which products are unique. To state the universal property we need to consider the diagonal functor $\Delta : \mathbb{C} \longrightarrow \mathbb{C} \times \mathbb{C} : f \longrightarrow (f, f)$ and the meaning of a couniversal object at (A, B) for Δ:

$$
\begin{tikzcd}
\Delta(C) \arrow[d, dotted, "\Delta(\langle f,g \rangle)"'] \arrow[dr, "{(f,g)}"] & \\
\Delta(A \times B) \arrow[r, "{(\pi_0,\pi_1)}"'] & (A, B).
\end{tikzcd}
$$

This gives us immediately that:

Lemma 3.18 *The product of A and B is the couniversal pair at (A, B) for the diagonal functor.*

The uniqueness of couniversal pairs tells us the sense in which a product is unique up to a unique isomorphism. It is worth unwinding this statement more explicitly to give a direct proof. The result very closely follows the proof style given for the initial object:

Suppose $(K, (a_0, a_1))$ and $(K', (a'_0, a'_1))$ are both products of A and B then there is a unique map $\alpha : K \longrightarrow K'$ such that $\alpha a'_0 = a_0$ and $\alpha a'_1 = a_1$ as K' is a product. But similarly there is a unique map $\alpha' : K' \longrightarrow K$ such that $\alpha' a_0 = a'_0$ and $\alpha' a_1 = a'_1$. The composite $\alpha\alpha' : K \longrightarrow K$ has $\alpha\alpha' a_0 = \alpha a'_0 = a_0$ and $\alpha\alpha' a_1 = \alpha a'_1 = a_1$. However $1_K : K \longrightarrow K$ also has this property so we must conclude $\alpha\alpha' = 1_K$ and similarly $\alpha'\alpha = 1_{K'}$.

A product has a number maps associated with it. The first map we consider is the diagonal map $\Delta = \langle 1_A, 1_A \rangle : A \longrightarrow A \times A$ this is the unit of the adjunction implied by the existence of all couniversal pairs.

Observe that $\langle 1_A, 1_A \rangle \pi_0 = 1_A$ this gives:

Lemma 3.19 *The diagonal map if it exists for an object in any category is a section and therefore monic.*

One might think that the projections must be epimorphisms: this is clearly so for cartesian powers ($A \times A$, $A \times A \times A$, ..) but is definitely not the case in general. Even in Sets there is a counterexample to this: $A \times 0 \xrightarrow[\pi_0]{} A$ for nonempty A is not surjective and thus not epic.

Of course $\_ \times \_$ is a functor $f \times g$ is define to be $\langle \pi_0 f, \pi_1 g \rangle$ as illustrated by:

$$\begin{tikzcd}
 & A \arrow[r, "f"] & A' \\
A \times B \arrow[ur, "\pi_0"] \arrow[dr, "\pi_1"'] \arrow[r, dotted, "f \times g"] & A' \times B' \arrow[ur, "\pi_0"] \arrow[dr, "\pi_1"'] & \\
 & A' \arrow[r, "g"'] & B'
\end{tikzcd}$$

Next for any binary product there is a symmetry map:

$$\begin{tikzcd}
 & & A \\
A \times B \arrow[urr, "\pi_0"] \arrow[drr, "\pi_1"'] \arrow[r, dotted, "c_{AB}"] & B \times A \arrow[ur, "\pi_1"'] \arrow[dr, "\pi_0"] & \\
 & & B
\end{tikzcd}$$

It is not hard to check that $c_{AB} c_{BA} = 1_{A \times B}$ and therefore we have:

Lemma 3.20 *The symmetry map for any product is an isomorphism.*

and $\Delta_A c_{AA} = \Delta_A$: the trick is to break the maps into their components by post-composing them with the projections.

Lastly there is an important map which allows one to re-associate products:

$$a_{ABC} : (A \times B) \times C \longrightarrow A \times (B \times C)$$

this map is the unique map determined by the equations

$$a_{ABC}\pi_0 = \pi_0\pi_0 \quad a_{ABC}\pi_1\pi_0 = \pi_0\pi_1 \quad a_{ABC}\pi_1\pi_1 = \pi_1.$$

This map has an obvious inverse:

$$a_{ABC}^{-1} : A \times (B \times C) \longrightarrow (A \times B) \times C$$

determined by the equations

$$a_{ABC}^{-1}\pi_0\pi_0 = \pi_0 \quad a_{ABC}^{-1}\pi_0\pi_1 = \pi_0\pi_0 \quad a_{ABC}^{-1}\pi_1 = \pi_1\pi_1.$$

Lemma 3.21 *The associativity map for any product is an isomorphism.*

In many structures which are "product like" these isomorphisms are present even though the projections or diagonals are absent. The "coherence" diagrams satisfied by these isomorphisms then becomes significant.

We shall say that a category $\mathbb{C}$ **has binary products** if every pair of objects has a product:

Proposition 3.22 *The following are equivalent:*

(i) *A category $\mathbb{C}$ has binary products;*

(ii) *The diagonal functor is a left adjoint:*

$$(\Delta, (\pi_0, \pi_1)) : \Delta \dashv \_ \times \_ : \mathbb{C} \longrightarrow \mathbb{C} \times \mathbb{C};$$

(iii) *There is and object operation $(A, B) \mapsto A \times B$ on $\mathbb{C}$ with two families of maps $\pi_0^{A,B} : A \times B \longrightarrow A$ and $\pi_1^{A,B} : A \times B \longrightarrow B$ together with a pairing combinator:*

$$\frac{f : X \longrightarrow A \qquad g : X \longrightarrow B}{\langle f, g\rangle : X \longrightarrow A \times B}$$

such that

$$\langle f, g\rangle\pi_0 = f \qquad \langle f, g\rangle\pi_1 = g \qquad \langle \pi_0, \pi_1\rangle = 1_{A\times B} \qquad h\langle f, g\rangle = \langle hf, hg\rangle.$$

Proof: The first two formulations given the view of products as a couniversal pair are clearly equivalent. For the final formulation it is straightforward to check that a product satisfies these identities but less obvious that these define a product. To establish this we show that the pairing map is unique: suppose that k has $k\pi_0 = f$ and $k\pi_1 = g$ then

$$k = k1_{A\times B} = k\langle \pi_0, \pi_1\rangle = \langle k\pi_0, k\pi_1\rangle = \langle f, g\rangle.$$

$\square$

The equality $\langle \pi_0, \pi_1\rangle = 1_{A\times B}$ is called to **surjective pairing** requirement. It is this identity which often is the most difficult to secure. For example, while it is possible in the lambda calculus to define many pairing combinators (satisfying everything but this last identity) it is provably impossible to produce a surjective pairing.

We should also note here that the formation of pairs can be reversed (which is the content of the surjective pairing condition, thus the inference rule may be written as a two-way inference:

$$\frac{h\pi_0 = f : X \longrightarrow A \qquad h\pi_1 = g : X \longrightarrow B}{\overline{h = \langle f, g\rangle : X \longrightarrow A \times B}}$$

Dual to the notion of a product is the notion of a coproduct. Let A and B be any two objects in a category then a **coproduct** of A and B is an object, often written $A + B$ equipped with two maps $\iota_0 : A \longrightarrow A + B$ and $\iota_1 : B \longrightarrow A + B$ such that given any object V with two maps $h : A \longrightarrow V$ and $k : W \longrightarrow B$ there is a unique map, often written $\langle h|k\rangle : W \longrightarrow A \times B$, such that $\iota_0\langle h|k\rangle = h$ and $\iota_1\langle h|k\rangle = k$.

The maps ι_0 and ι_1 are called the **coprojections**.

We express this universal property diagrammatically as follows:

$$\begin{tikzcd}
A \arrow[dr, "\iota_0"'] \arrow[drr, "h"] & & \\
& A + B \arrow[r, dotted, "\langle h|k\rangle"] & V \\
B \arrow[ur, "\iota_1"] \arrow[urr, "h"'] & &
\end{tikzcd}$$

Now it is interesting to note that the coproduct in Sets is entirely different from the product. It is the disjoint union of the two sets the coprojections are the embeddings of the components into the disjoint union. In vector spaces the product and coproduct coincide — when this happens, in a nice way, we shall say that we have biproducts (see later). In Cat the coproduct is the disjoint union of the categories much as for sets.

The analogue to the diagonal maps is called the **codiagonal** and is the map

$$\nabla = \langle 1_A | 1_A \rangle : A + A \longrightarrow A.$$

We say a category **has coproducts** if every pair of objects has a coproduct.

3.6.3 Limits and colimits of diagrams

There is a general notion of the limit and colimit of a "diagram" which we now introduce. This leads to the notion of a complete category, that is a category in which the limits of all (small) diagrams exist. Of course, certain limits (e.g. equalizers and products) imply the presence of all other limits so that completeness can be reduced to having all equalizers and all (small) products.

A diagram is, concretely a morphism of a directed graph into a category. Ocassionally this notion of a diagram is too restrictive as we may also want to say that certain composites in the diagram must be equal. A directed graph together with the specification that certain composites are equal is just a **presentation** of a category. We may construct an actual category from such a presentation by moving to the path category generated by the graph and "forcing" the equalities we wish to hold to be true. This we may do by generating the smallest congruence on the path category which includes the desired equalities. The category we want is then the quotient category with respect to this congruence and the diagrams are then functors from this category.

In this section we shall be ambivalent about whether we are working with functors from categories, presentations of categories, or just diagrams. The results we prove rely only on the directed graph structure and not on any overlying commuting constraints.

Thus, a **diagram** $D : \mathbb{G} \longrightarrow U(\mathbb{C})$ is a map of directed graphs into the underlying directed graph of $\mathbb{C}$ (we know this corresponds to a functor $D^\sharp : \mathsf{Path}(\mathbb{G}) \longrightarrow \mathbb{C}$) then a D–**cone** over this diagram consists of an object A, called the **apex** of the cone together with for each node N of $\mathbb{G}$ a map $\alpha_N : A \longrightarrow D(N)$ such that for each arrow of $\mathbb{G}$, $a : N_1 \longrightarrow N_2$, we have $\alpha_{N_1} G(a) = \alpha_{N_2}$.

A morphism of cones $(\alpha, h, \beta) : \alpha \longrightarrow \beta$ is given by a map in $\mathbb{C}$, $h : A \longrightarrow B$ between the apexes of the cones such that $\alpha_N = h\beta_N$ for all the nodes of the diagram. We observe:

Lemma 3.23 *The cones over $D : \mathbb{G} \longrightarrow \mathbb{C}$ form a category, $\mathsf{Cone}_D(\mathbb{C})$, with objects the cones and maps the morphisms of cones.*

This is straightforward to check and we leave it to the reader.

A **limit** of a diagram is a final object in the category $\mathsf{Cone}_D(\mathbb{C})$. We often write the apex of this cone as $\mathsf{Lim}(D)$ with **projections** $\pi_N : \mathsf{Lim}(D) \longrightarrow G(N)$.

We may display diagrammatically the concept of a limit of a diagram as follows:

$$\begin{tikzcd}
A \arrow[rrrr, dotted] \arrow[dr, "\alpha_{N_1}"'] \arrow[ddrr, "\alpha_{N_3}"] \arrow[drrr, "\alpha_{N_2}"] & & & & \mathsf{Lim}(D) \arrow[dlll, "\pi_{N_1}"'] \arrow[dl, "\pi_{N_2}"] \arrow[ddll, "\pi_{N_3}"] \\
& D(N_1) \arrow[rr, "D(a)"] & & D(N_2) & \\
& & D(N_3) \arrow[ur, "D(b)"'] & &
\end{tikzcd}$$

First we note that this notion of limit subsumes both the definition of a final object (take $\mathbb{G}$ to be the empty graph) and the definition of the product (take $\mathbb{G}$ to be the discrete graph – with no arrows – with two nodes). Furthermore it suggests what should be the definition of an n-ary product: namely, the limit of a discrete diagram with n nodes. However, as illustrated above there are many more shapes diagrams may take and, thus, a whole variety of limits of which we should develop some understanding.

Before doing so, however, it is useful to relate this notion of limit to that of a universal pair. To do this, as is often the case, it is a matter of choosing ones category and functors carefully.

Let $\mathsf{Dgrm}(\mathbb{G}, \mathbb{C})$ be the category of **$\mathbb{G}$-shaped diagrams** in $\mathbb{C}$ having objects morphisms of directed graphs $D : \mathbb{G} \longrightarrow U(\mathbb{C})$ and maps natural transformations (equivalently natural transformations between functors $D^\sharp : \mathsf{Path}(\mathbb{G}) \longrightarrow \mathbb{C}$). From general principles we know that this is a category.

There is also a diagonal functor $\Delta : \mathbb{C} \longrightarrow \mathsf{Dgrm}(\mathbb{G}, \mathbb{C})$ which takes an object C to the degenerate diagram on that object where all the nodes of $\mathbb{G}$ become the object C itself and the arrows of $\mathbb{G}$ become the identity map 1_C. We now have:

Proposition 3.24 $\mathsf{Dgrm}(\mathbb{G}, \mathbb{C})$ *is a category with a diagonal functor* $\Delta : \mathbb{C} \longrightarrow \mathsf{Dgrm}(\mathbb{G}, \mathbb{C})$*. Furthermore a couniversal pair at diagram D for Δ is exactly a limit of D,* $(\mathsf{Lim} D, \pi)$*.*

Proof: This is more a matter of translation than of proof! The first thing to realize is that a natural transformation $h : \Delta(A) \longrightarrow D$ is exactly a cone. Thus, the couniversal diagram:

$$\begin{tikzcd}
\Delta(A) \arrow[d, dotted, "\Delta(\langle f \rangle)"'] \arrow[dr, "f"] & \\
\Delta(\mathsf{Lim}(D)) \arrow[r, "\pi"'] & D.
\end{tikzcd}$$

also ensures that $(\mathsf{Lim}(D), \pi)$ is final in the category of cones. $\square$

We shall say that a category is **complete** in case $\Delta : \mathbb{C} \longrightarrow \mathsf{Dgrm}(\mathbb{G}, \mathbb{C})$ has a right adjoint for each graph $\mathbb{G}$. Here the size of the directed graphs is important: we are tacitly assuming that the directed graphs we are talking about are "small" in the sense that both the nodes and arrows form sets. However, we could assume that both the node and arrow sets must be finite: we will say the category is **finitely complete** if for any finite $\mathbb{G}$ the functor Δ has a right adjoint.

Dual to the notion of limit is that of colimit. A colimit of a diagram D is a universal pair, often written $(\mathsf{coLim}D, \iota)$, at D for the functor Δ.

We may display a colimit diagrammatically as:

$$\begin{tikzcd}
& D(N_1) \arrow[rr, "D(a)"] \arrow[dr, "D(b)"] \arrow[ddl, "\iota_{N_1}"'] \arrow[ddrrr, "\beta_{N_1}"] & & D(N_2) \arrow[ddlll, "\iota_{N_2}"] \arrow[ddr, "\beta_{N_2}"] & \\
& & D(N_3) \arrow[dll, "\iota_{N_3}"] \arrow[drr, "\beta_{N_3}"] & & \\
\mathsf{coLim}D \arrow[rrrr, dotted] & & & & A
\end{tikzcd}$$

We shall say that a category is **cocomplete** in case $\Delta : \mathbb{C} \longrightarrow \mathsf{Dgrm}(\mathbb{G}, \mathbb{C})$ has a left adjoint for each small graph $\mathbb{G}$. We will say the category is **finitely cocomplete** if for any finite $\mathbb{G}$ the functor Δ has a left adjoint.

Because a colimit is a universal pair and universal pairs are preserved by left adjoints, useful fact is:

Lemma 3.25 *Left adjoints preserve colimits and right adjoints preserve limits.*

3.7 Special limits: equalizers and pullbacks

There are many shapes for diagrams and in this section we start by focusing on two special shapes: equalizer diagrams and pullback diagrams. We show how the existence of limits for various shapes of diagrams imply the existence of limits for others, culminating in proving that all limits exist when products and equalizers are present.

An **equalizer diagram** is a parallel pair of arrows:

$$A \underset{g}{\overset{f}{\rightrightarrows}} B$$

a cone for the above equalizer diagram is called **an equalizer** of f and g and is given by an object Q together with a map $q : Q \longrightarrow A$ such that $qf = qg$ (this map is the map $Q \longrightarrow B$ required to make a cone). A limit (E, e) is called **the equalizer** even though it is not unique and satisfies the following couniversal property

$$\begin{tikzcd}
Q \arrow[drr, "q"] \arrow[dr, dotted, "k"'] & & & \\
& E \arrow[r, "e"'] & A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & B
\end{tikzcd}$$

that there is a unique k such that $ke = q$.

We shall often write the equalizer of f and g as the object couniversal pair $\pi_{f=g} : f = g \longrightarrow A$. We observe:

Lemma 3.26 *Suppose (E, e) is the equalizer of $A \overset{f}{\underset{g}{\rightrightarrows}} B$ then e is monic.*

PROOF: Suppose $xe = xy = z : X \longrightarrow A$ then $zf = zg$ so there is a unique $k : X \longrightarrow E$ such that $ke = z$. Therefore $k = x = y$ and so certainly $x = y$. □

A **pullback diagram** is a binary fan of arrows:

$$\begin{tikzcd}
& A \arrow[d, "f"] \\
B \arrow[r, "g"'] & C
\end{tikzcd}$$

a cone for the above pullback diagram is called **a pullback** of f and g and is given by an object Q together with two maps $q_A : Q \longrightarrow A$ and $q_B : Q \longrightarrow B$ such that $q_A f = q_B g$ (this map is the map $Q \longrightarrow C$ required to make a cone). A limit (E, e_A, e_B) is called **the pullback** even though it is not unique and satisfies the following couniversal property

$$\begin{tikzcd}
Q \arrow[drr, "q_A"] \arrow[dr, dotted, "k"] \arrow[ddr, "q_B"'] & & \\
& E \arrow[r, "e_A"'] \arrow[d, "e_B"] & A \arrow[d, "f"] \\
& B \arrow[r, "g"'] & C
\end{tikzcd}$$

that there is a unique k such that $ke_A = q_A$ and $ke_B = q_B$.

We shall occasionally write the pullback of f and g as the couniversal pair $(f \wedge g, \pi)$. Sometimes we shall want to view the pullback diagram in an asymmetric way and so we shall say (in the above) e_A is **the pullback along** f of g.

Lemma 3.27 *In any category*

(i) *the pullback of a monic along any map is a monic;*

(ii) *the pullback of a retraction along any map is a retraction;*

(iii) *the pullback of an isomorphism along any map is an isomorphism.*

Proof:

(i) Suppose g is monic and $k_1e_A = k_2e_A$ then

$$k_1e_Bg = k_1e_Af = k_2e_Af = k_2e_Bg$$

so as $g$is monic $k_1e_B = k_2e_B$.

$$\begin{tikzcd}
Q \arrow[drr, "k_1e_A=k_2e_A"] \arrow[ddr, "k_1e_B=k_2e_b"'] \arrow[dr, dotted] & & \\
& E \arrow[r, "e_A"'] \arrow[d, "e_B"] & A \arrow[d, "f"] \\
& B \arrow[r, tail, "g"'] & C
\end{tikzcd}$$

However, this makes k_1 and k_2 comparison maps from the outer square to the pullback.

(ii) Suppose now g is a retraction so there is a g' with $g'g = 1_C$. Now consider the comparison map given by:

$$\begin{tikzcd}
A \arrow[drr, equal] \arrow[ddr, "fg'"'] \arrow[dr, dotted, "k"] & & \\
& E \arrow[r, "e_A"'] \arrow[d, "e_B"] & A \arrow[d, "f"] \\
& B \arrow[r, two heads, "g"'] & C
\end{tikzcd}$$

Clearly it is a left inverse for e_A.

(iii) As an isomorphism is a monic retraction we may combine the two previous parts to obtain this one.

□

A helpful observation is the following:

Lemma 3.28 *In any category $f : A \longrightarrow B$ is monic if and only if the following square is a pullback:*

$$\begin{tikzcd}
A \arrow[r, equal] \arrow[d, equal] & A \arrow[d, "f"] \\
A \arrow[r, "f"'] & B.
\end{tikzcd}$$

Proof: If this square is a pullback then whenever $xf = yf$ there is a unique comparison map

$$\begin{tikzcd}
X \arrow[drr, "x"] \arrow[ddr, "y"'] \arrow[dr, dotted] & & \\
& A \arrow[r, equal] \arrow[d, equal] & A \arrow[d, "f"] \\
& A \arrow[r, "f"'] & B.
\end{tikzcd}$$

which shows $x = y$. Conversely if f is monic then whenever we form the outer square so that it commutes $x = y$ so that this also gives a comparison map, whose uniqueness is forced by the fact that f is monic. □

This then allows us to link the property of being monic to a couniversal property whence we can conclude:

Corollary 3.29 *Right adjoints preserve monics and (dually) left adjoints preserve epics.*

Pullback squares can be pasted together and the following observation is key to the behavior of squares under pasting with respect to pullbacks:

Proposition 3.30 *In the following (commuting) diagram:*

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "a"'] & B \arrow[r, "g"] \arrow[d, "b"'] & C \arrow[d, "c"] \\ A' \arrow[r, "f'"'] & B' \arrow[r, "g'"'] & C' \end{tikzcd}$$

(i) *if the two inner squares are pullbacks the outer square is a pullback;*

(ii) *if the rightmost square and outer square is a pullback the leftmost square is a pullback.*

Proof:

(i) Suppose $xc = yf'g' : X \longrightarrow C$ then there is a unique map $w : X \longrightarrow B$ such that $wg = x$ and $wb = yf'$. The latter equality gives a unique $v : X \longrightarrow A$ such that $va = x$ and $vf = w$. But this means $vfg = wg = x$. Thus, v is the desired comparison map. It remains to check uniqueness which is straightforward.

(ii) Suppose $xb = yf'$ Then $xgc = yf'g'$ so there is a unique k such that $ka = y$ and $kfg = xg$. But now both kf and x have the property that $xg = kfg$ and $xb = yf' = kaf' = kfb$. Thus, they both give the unique comparison map to the rightmost square, so $kf = x$. This means that k is a comparison map to the leftmost square: again it is straightforward to check that k is unique.

□

3.8 Completeness and cocompleteness

We shall say a category **has equalizers** or **has pullbacks** in case the appropriate limits always exist. The category of Sets has both equalizers and pullbacks. Equalizers can be described very simply in Sets, the limit cone for parallel set maps f and g is:

$$\{a \in A | f(a) = g(a)\} \subseteq A \overset{f}{\underset{g}{\rightrightarrows}} B.$$

Finite dimensional vector spaces over a field K (take, for example, $K = \mathbb{R}$) also have equalizers. These, however, this time are harder to describe but is basically the solution space for a set of linear equations. The following result shows that these examples also have pullbacks:

Proposition 3.31

(i) *If a category has binary products and equalizers then it has pullbacks;*

(ii) *If a category has pullbacks and binary products then it has equalizers;*

(iii) *If a category has a final object and pullbacks then it has binary products;*

(iv) *If a category has binary products then it has n-ary products for all $n > 0$.*

PROOF:

(i) We may construct the pullback as the equalizer:

$$\begin{tikzcd} f \wedge g \arrow[r, "\langle \pi_A, \pi_B \rangle"] & A \times B \arrow[r, shift left, "\pi_0 f"] \arrow[r, shift right, "\pi_1 g"'] & C \end{tikzcd}$$

Where we notice that $\langle x, y\rangle \pi_0 f = \langle x, y\rangle \pi_1 g : X \longrightarrow C$ if and only if $xf = yg$ and the unique $k : X \longrightarrow f \wedge g$ has $k\langle \pi_A, \pi_B\rangle = \langle k\pi_A, k\pi_B\rangle = \langle f, g\rangle$ which is the case if and only if $k\pi_A = f$ and $k\pi_B = g$ as required.

(ii) The equalizer of two maps may be constructed using the following pullback:

$$\begin{tikzcd} f = g \arrow[r, "\pi"] \arrow[d, "\pi f"'] & A \arrow[d, "\langle f,g \rangle"] \\ B \arrow[r, "\Delta"'] & B \times B. \end{tikzcd}$$

Suppose $xf = xg : X \longrightarrow B$ then

$$xf\Delta = xf\langle 1, 1\rangle = \langle xf, xf\rangle = \langle xf, xg\rangle = x\langle f, g\rangle$$

so that there is a map $k : X \longrightarrow f = g$ with $k\pi = x$ which is forced to be unique as π is monic being a pullback of the section Δ.

(iii) The binary fan

$$\begin{tikzcd} & A \arrow[d, "!"] \\ B \arrow[r, "!"'] & 1 \end{tikzcd}$$

has the pullback the product of A and B as any cone of the one is a cone of the other.

(iv) The n-ary product may be built from a composite of binary products.

$\square$

The main observation of this subsection is as follows:

Proposition 3.32 *A category is (finitely) complete if and only if it has (finite) products and equalizers.*

PROOF: If the category is (finitely) complete then it will have all (finite) products and equalizers so the content of the result is in the converse. Let $D : \mathbb{G} \longrightarrow U(\mathbb{C})$ be a diagram then we have the arrows of the graph G_1 and the nodes of the graph G_0 and we can form the following equalizer:

$$V \xrightarrow{\langle v_a \rangle_{a \in G_0}} \prod_{a \in G_0} D(a) \underset{\langle \pi_{\partial_1(f)} \rangle_{f \in G_1}}{\overset{\langle \pi_{\partial_0(f)} D(f) \rangle_{f \in G_1}}{\rightrightarrows}} \prod_{f \in G_1} D(\partial_1(f))$$

Notice that any equalizer of this equalizer is also a cone for D as

$$\begin{aligned} v_a D(f) &= \langle v_a \rangle_{a \in G_0} \pi_{\partial_0(f)} D(f) f \\ &= \langle v_a \rangle_{a \in G_0} \langle \pi_{\partial_0(g)} D(g) \rangle_{g \in G_1} \pi_f \\ &= \langle v_a \rangle_{a \in G_0} \langle \pi_{\partial_1(g)} \rangle_{g \in G_1} \pi_f \\ &= \langle v_a \rangle_{a \in G_0} \pi_{\partial_1(f)} \\ &= v_b. \end{aligned}$$

and conversely very cone for D gives rise to an equalizer of this diagram as

$$\begin{aligned} &\langle v_a \rangle_{a \in G_0} \langle \pi_{\partial_0(f)} D(f) \rangle_{f \in G_1} \\ &\quad = \langle \langle v_a \rangle_{a \in G_0} \pi_{\partial_0(f)} D(f) \rangle_{f \in G_1} \\ &\quad = \langle \langle v_{\partial_0(f)} D(f) \rangle_{f \in G_1} \\ &\quad = \langle \langle v_{\partial_1(f)} \rangle_{f \in G_1} \\ &\quad = \langle v_a \rangle_{a \in G_0} \langle \pi_{\partial_1(f)} \rangle_{f \in G_1}. \end{aligned}$$

Finally a morphism of cones over D becomes a morphism of equalizers as a map to a product is determined by its components. This means that the two limits have isomorphic cone categories. Thus if either has a limit it is also a limit of the other. □

4 Exercises for part I

(1) Prove that the following data forms a category:

Objects: Sets

Maps: Relations

Identities: $1_A = \{(x,y) \in A \times A | x \neq y\}$

Composition: $R;S = \{(x,z) \in A \times C | \forall y \in B \cdot (x,y) \in R \vee (y,z) \in S\}$

(2) A **rig** R is a set together with an operation $\_ + \_$ which is commutative and associative with unit 0, and an operation $\_ \cdot \_$ which is associative with unit 1 which distributes over the $\_ + \_$. Thus, these operations satisfying:

- $(x+y)+z = x+(y+z)$
- $x+y = y+z$
- $0+x = x = x+0$
- $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
- $x \cdot 1 = x = 1 \cdot x$
- $x \cdot (y+z) = (x \cdot y) + (x \cdot z)$
- $(y+z) \cdot x = (y \cdot x) + (z \cdot x)$
- $0 \cdot z = 0 = z \cdot 0$

Show that the category of matrices, $\mathbf{Mat}_R$, over a rig R form a category:

Objects: Natural numbers,

Maps: $n \times m$-Matrices for n and m natural numbers,

Composition: Matrix composition,

Identities: Diagonal matrices.

(3) Provide a functor from boolean matrices to finite sets and relations:

$$H : \mathbf{Mat}_{\mathrm{bool}} \to \mathbf{Rel}_f$$

Now provide a functor in the opposite direction (the difficulty is that there are many finite sets of the same cardinallity ...)

$$G : \mathbf{Rel}_f \to \mathbf{Mat}_{\mathrm{bool}}$$

and show that there is a natural isomorphism

$$\alpha : \mathrm{Id}_{\mathbf{Rel}_f} \to G;;H$$

and that $H;;G$ is the identity functor. Conclude that the categories are equivalent!

A question to ponder: given this relationship, what is $\mathbf{Mat}_L$ for some distributive lattice L?

(4) In a partially ordered set (viewed as being a category) what is an initial object, final object, product, and coproduct?

(5) A zero object is an object which is both initial and final. Prove that in a category with a zero object all projections from products are epic.

(6) Prove that in the category of monoids not every bijic is an isomorphism.

(7) A preinitial object is an object with *at most* one map from any object. Prove that P is preinitial if and only if

$$\begin{tikzcd} P \arrow[dr, "1_P"] & \\ & P \\ P \arrow[ur, "1_P"'] & \end{tikzcd}$$

is a coproduct.

A question to ponder: what are the preinitial objects in the category of (commutative) rings?

(8) Prove carefully that the symmetry and associativity maps for binary products are isomorphisms.

(9) If a category has products and coproducts prove that

$$\langle\langle f|g\rangle, \langle h|k\rangle\rangle = \langle\langle f,h\rangle|\langle g,k\rangle\rangle : A + B \longrightarrow C \times D.$$

(10) If a category $\mathbb{C}$ has products (and therefore a final object 1) prove that:

$$\begin{tikzcd} & A \arrow[dl, equal] \arrow[dr, "!"] & \\ A & & 1 \end{tikzcd}$$

is a limit cone. Show, therefore that $A \times 1$ and $1 \times A$ are canonically (in the sense that these maps always exist) isomorphic to A.

(11) If a category $\mathbb{C}$ has products prove that the following diagrams are always pullbacks:

$$\begin{tikzcd} A \times B \arrow[r, "f \times 1_B"] \arrow[d, "\pi_0"'] & A' \times B \arrow[d, "\pi_0"] \\ A \arrow[r, "f"'] & A' \end{tikzcd} \qquad \begin{tikzcd} A \arrow[r, "\langle 1_A, f\rangle"] \arrow[d, "f"'] & A \times B \arrow[d, "f \times 1_B"] \\ B \arrow[r, "\Delta"'] & B \times B. \end{tikzcd}$$

(12) If a category has products prove that

$$A \xrightarrow{\Delta} A \times A \overset{\pi_0}{\underset{\pi_1}{\rightrightarrows}} A$$

gives the equalizer of π_0 and π_1.

(13) Prove that in any category the pushout along any map of a section is itself a section. Show that the pullback of a section is not necessarily a section: give a counter example in finite sets.

(14) Prove that if $mr = 1_Y$ and $e = rm$ then, in the following diagram

$$Y \xrightarrow{m} X \underset{1_X}{\overset{e}{\rightrightarrows}} X \xrightarrow{r} Y$$

m is the equalizer of e and 1_X and r the coequalizer.

(15) Prove that if a functor $F : \mathbb{C} \longrightarrow \mathbb{D}$ preserves pullbacks and $\mathbb{C}$ has products that F preserves equalizers (Hint: show that $\langle F(p_0), F(p_1)\rangle$ is monic).

(16) Describe coequalizers in Sets. Prove that Sets is complete and cocomplete.

(17) Suppose that $(\eta, \epsilon) : F \dashv G : \mathbb{X} \longrightarrow \mathbb{Y}$ is a reflection (ϵ is a natural isomorphism) and that $\mathbb{X}$ has colimits then $\mathbb{Y}$ has colimits.

(18) Certain limits can be obtained from other limits:

(a) Prove that any category has limits for acyclic graphs with a source. These are graphs with an object from which any other object can be reached (in a unique way) following the direction of the arrows (hint: a tree in fact!).

(b) (Harder) Prove that any category with equalizers has limits for finite graphs with a source (any other object can be reached from the source in a not necessarily unique way following the direction of the arrow).

(c) Prove that if a category has pullbacks it has limits for all finite connected acyclic graphs. These are the diagrams which regarding each (directional) arrow as a two-way arrow leaves every object reachable from every other in a unique way.

(d) (Harder) Prove that if a category has pullbacks and equalizers that every finite connected diagram has a limit. These are the diagrams which regarding each (directional) arrow as a two way arrow leaves every object reachable from every other (possibly in many ways).

(19) Show that a small category (both objects and arrows are sets) which is complete is a preorder. Give a concrete proof that any meet preserving map from a complete poset has left adjoint.

(20) A complete partial order is a partial order with all limits or, equivalently all meets. Prove that the inclusion of complete partial orders into all partial orders is a reflection (you may need to look up some lattice theory here!).

(21) Show that $\mathsf{Sets}^{\mathbb{C}^{\mathrm{op}}}$ for any small category $\mathbb{C}$ is complete and cocomplete.

(22) Show that a congruence on a 2-category is an equivalence relation on the 2-cells which only relates like typed 2-cells and satisfies:

(a) if $\alpha \sim \beta$ then $\gamma\alpha\gamma' \sim \gamma\beta\gamma'$ whenever these vertical composites are defined,

(b) if $\alpha \sim \beta$ then $\gamma; \alpha; \gamma' \sim \gamma; \beta; \gamma'$ whenever these horizontal composites are defined.

Show that every 2-functor can be factorized into a 2-functor which is surjective on 2-cells and bijective on 1-cells and 0-cells followed by a 2-faithful 2-functor (i.e one which is injective on the 2-cells hom-sets).

Part II

Restriction Categories

Partial functions arise everywhere in mathematics. One area where partiality arises rather brutally is in computability [11]. A standard way to approach computability theory is through primitive recursive functions and minimalization. The latter construction fundamentally introduces partiality. The aim of this section is to introduce an abstract and yet very convenient categorical formulation of partiality.

One way to define partiality categorically is to use "partial map categories" – a. presentation of partial map categories is given in chapter 2 of [2]. The maps of a partial map category embody the intuition of what a partial map should be: they are spans $(m, f) : X \to Y$, that is pair of maps $X \xleftarrow{m} X' \xrightarrow{f} Y$, whose left legs are monic. These spans give a direct representation of partiality: X' is the domain of definition for f, and m is the monic inclusion of the domain of definition into the source of the span. The drawback to partial map categories so constructed is that composition relies on forming pullbacks and taking spans up to an equivalence relation. This means that reasoning based on this formulation becomes rather cumbersome.

Restriction categories, [8], provide a much simpler approach to handling partiality. In a restriction category, the partiality of a map, f, is determined through a *restriction idempotent*, $\overline{f}$, associated with f. This association of idempotent with map capture partiality in a completely algebraic manner: the restriction satisfies just four simple equations. This simplifies the reasoning required for partiality.

The first objective of these notes is to show that restriction categories completely capture the notion of partiality. To achieve this we shall establish an equivalence between partial map categories and restriction categories in which idempotents split.

The next section follows [7] closely: there is more material in that reference.

The notion of restriction actually has quite a long and interesting history: an account of this is provided in [9]. Although we shall use restriction categories here to provide an abstract account of computability, they have implication in other areas of mathematics: Manes highlights some of these applications to semigroup theory and topology in [23].

5 Introduction to restriction

Definition 5.1 *A* **restriction category** *is a category equipped with a* **restriction** *operator*

$$\frac{A \xrightarrow{f} B}{A \xrightarrow[\overline{f}]{} A}$$

which given a map returns an endomorphism on the domain of that map satisfying: he definition of a restriction category:

[R1]	$\overline{f}\, f = f$	**[R3]**	*If* $dom(f) = dom(g)$ *then* $\overline{\overline{f}\, g} = \overline{g}\, \overline{f}$
[R2]	*If* $dom(f) = dom(g)$ *then* $\overline{g}\, \overline{f} = \overline{f}\, \overline{g}$	**[R4]**	*If* $dom(g) = cod(f)$ *then* $\overline{g}\, f = f \overline{gf}$

Here are some examples of restriction categories:

Example 5.2 *The trivial restriction.*

The restriction defined by $\overline{f} = 1_{dom(f)}$ is, on any category, always a restriction operator, called the trivial restriction. A category with this trivial restriction is called a trivial restriction category.

Example 5.3 *Sets and partial functions.*

Take the category of sets and partial functions. One can define a restriction for $A \xrightarrow[f]{} B$ here by
$\overline{f}\,(x) = \begin{cases} x & f(x) \text{ defined} \\ \uparrow & \text{otherwise} \end{cases}$

As this is a key example let us verify the axioms:

[R1] $f(\overline{f}\,(x)) = f(x)$ whether f(x) is defined or not.

[R2] If both g and f are defined at x, then $\overline{g}\,(\overline{f}\,(x) = x = \overline{f}\,(\overline{g}\,(x))$. If both g and f are undefined at x, then $\overline{g}\,(\overline{f}\,(x)) = \uparrow = \overline{f}\,(\overline{g}\,(x))$. Finally, only one of the functions is defined at x, then without loss we may assume f is defined and g is not. Then $\overline{f}\,(\overline{g}\,(x)) = \overline{f}\,(\uparrow) = \uparrow = \overline{g}\,(x) = \overline{g}\,(\overline{f}\,(x))$.

[R3] First if f is defined at x, then $g(\overline{f}\,(x)) = g(x)$, so that if g is defined at x, $\overline{\overline{f}\, g}\,(x) = x = \overline{g}\,(\overline{f}\,(x))$, and if g is undefined at x, then $\overline{\overline{f}\, g}\,(x) = \uparrow = \overline{g}\,(\overline{f}\,(x))$. On the other hand, if f is undefined at x, then $\overline{f}\,(g(x))$ is undefined at x, so $\overline{\overline{f}\, g}\,(x) = \uparrow = \overline{g}\, \uparrow = \overline{f}\,(\overline{g}\,(x))$.

[R4] In the case that f is defined at x, and g is defined at, $f(x)$, then $\overline{g}\,(f(x)) = f(x)$. On the other hand, $g(f(x))$ is defined so $f(\overline{fg}\,(x)) = f(x)$. If g were not defined at $f(x)$, then $\overline{g}\,(f(x)) = \uparrow = f(\uparrow) = f(\overline{fg}\,(x))$. In the case that f is undefined at x, then g is necessarily undefined at $f(x)$, and $\overline{g}\,(f(x)) = \uparrow = f \uparrow = f(\overline{gf}\,(x))$.

Here the restriction $\overline{f} : A \to A$ for any partial map f is precisely the domain of definition of f.

Example 5.4 *The free restriction category on a graph*

Another restriction category, the free restriction category on a graph can be given explicitly, and is a nice example of a restriction category which gives an unexpected presentation of partiality. First, we will give the construction then show the restriction axioms hold.

Take a directed graph, G, then form a category, the free restriction category [6], where

Objects: Nodes of G

Maps: $A \xrightarrow{(s,S)} B$ where S is a finite prefix-closed set of paths out of A, and $s \in S$ is a path from $A \longrightarrow B$ called the **trunk**. Being prefix closed requires that if rt is a path in S, then r is a path in S.

Composition: Given, $A \xrightarrow{(s,S)} B$ and $B \xrightarrow{(t,T)} C$ take the composite to be:

$$(s,S)(t,T) : A \longrightarrow C = (st, S \cup sT) : A \longrightarrow C.$$

Identities: $((A,[],A),\{(A,[],A)\}) : A \longrightarrow A$ (we will write $1_A := (A,[],A)$).

The restriction on this category is given by $\overline{(s,S)} = ((A,[],A),S)$. The restriction axioms are particularly easy to check.

[R1] $(1_A,S)(s,S) = (s, S \cup S) = (s,S)$

[R2] $(1_A,S)(1_A,T) = (1_A, T \cup S) = (1_A, S \cup T) = (1_A,T)(1_A,S)$.

[R3] $\overline{\overline{(t,T)}\,(s,S)} = \overline{(1_A,T)(s,S)} = \overline{(s, T \cup S)} = (1_A, T \cup S) = (1_A,T)(1_A,S)$

[R4] $(t,T)\overline{(s,S)} = (t,T)(1_A,S) = (t, T \cup tS) = (1_A, T \cup tS)(t,T) = \overline{(ts, T \cup tS)}\,(t,T) = \overline{(t,T)(s,S)}\,(t,T)$.

The trunk, $A \xrightarrow{t} B$, may be thought of as a particular choice of path out of A. The restriction which reduces a map to having an empty trunk can be thought of as leaving the choice of path out of A open. This partiality then is a sort of indeterminacy on the paths out of A.

5.1 Restriction basics

The following lemma shows that restricted maps are idempotents, and that the restriction operator itself is idempotent.

Lemma 5.5 *Let, $\mathbb{C}$, be any restriction category, and let $f, g \in \mathbf{C}$ with dom$(g) =$ cod(f).*

$$(i)\ \overline{f}\,\overline{f} = \overline{f} \quad (ii)\ \overline{fg}\,\overline{f} = \overline{fg} \quad (iii)\ \overline{f\overline{g}} = \overline{fg} \quad (iv)\ \overline{\overline{f}} = \overline{f}$$

Proof:

(i) $\overline{f}\,\overline{f} = \overline{\overline{f}f} = \overline{f}$.

(ii) $\overline{fg}\,\overline{f} = \overline{f}\,\overline{fg} = \overline{\overline{f}fg} = \overline{fg}$.

(iii) $\overline{f\overline{g}} = \overline{\overline{gf}\, f} = \overline{fg}\;\overline{f} = \overline{fg}$.

(iv) $\overline{f} = \overline{f1} = \overline{1\overline{f}} = \overline{\overline{f}}$.

□

Since restriction maps are idempotent and commute the restriction idempotents associated to an object A form a semilattice, $\mathcal{O}(A)$. We think of this semilattice topologically as specifying the open sets of A. Each map $f : A \longrightarrow B$ then, acting like a continuous map, carries open sets backward

$$f^* : \mathcal{O}(B) \longrightarrow \mathcal{O}(A); e \mapsto \overline{fe}$$

this map preserves the meet operation but does not preserve the top element: it is, therefore, a **stable** map of semilattices.

Every restriction category is order enriched: given two parallel maps $f, g : X \longrightarrow Y$ in a restriction category we shall write

$$f \leq g \Leftrightarrow \overline{f}\, g = f.$$

Lemma 5.6 *In any restriction category:*

(i) The relation $f \leq g$ is a partial order on each hom-set;

(ii) If $f \leq g$ then, for every h and k for which the composites are defined, $hfk \leq hgk$.

PROOF:

(i) Clearly it is reflexive. For transitivity suppose $f \leq g$ and $g \leq h$ then

$$f = \overline{f}\, g = \overline{f}\;\overline{g}\, h = \overline{\overline{f}\, g}\, h = \overline{f}\, h.$$

For anti-symmetry suppose $f \leq g$ and $g \leq f$ then $f = \overline{f}\, g$ and $g = \overline{g}\, f$ so

$$f = \overline{f}\, g = \overline{f}\;\overline{g}\, g = \overline{\overline{g}\, f}\, g = \overline{g}\, g = g.$$

(ii) If $f \leq g$ then $\overline{hfk}\, hgk = h\overline{fk}\, gk = h\overline{fk}\;\overline{f}\, gk = h\overline{fk}\, fk = hfk$.

□

Two parallel maps $f, g : X \longrightarrow Y$ in a restriction category are **compatible**, written $f \smile g$, in case $\overline{f}\, g = \overline{g}\, f$. Intuitively this means that they agree where they are both defined:

Lemma 5.7 *In any restriction category:*

(i) $f \smile g$ if and only if $\overline{g}\, f \leq g$ or $\overline{f}\, g \leq f$;

(ii) If $f \smile g$ then, for every h and k for which the composites are defined, $hfk \smile hgk$.

PROOF:

(i) If $\overline{f}\, g \leq f$ if and only if $\overline{\overline{f}\, g}\, f = \overline{f}\, g$, but then $\overline{g}\, f = \overline{g}\;\overline{f}\, f = \overline{\overline{f}\, g}\, f$.

(ii) We must show $\overline{hfk}\,hgk \leq hfk$ when $\overline{f}\,g \leq f$ but $\overline{hfk}\,hgk = h\overline{fk}\,gk = h\overline{fk}\,\overline{f}\,gk \leq h\overline{fk}\,fk = hfk$.

□

Definition 5.8 *A* **total map** *, $A \xrightarrow{f} B$, in a restriction category is a map for which $\overline{f} = 1_A$.*

Recalling the example of sets and partial functions above, a map, f, is total - that is the restriction is the identity – when $\forall x.\overline{f}\,(x) = x$ which means precisely that f is defined at all x, giving the usual meaning of totality. By way of contrast, a map in the free restriction category is total when $\overline{(s,S)} = (1_A, \{1_A\})$. But this means $S = \{frm[o]--_A\}$ to begin with; thus, the only total maps in this category are the identity maps.

Lemma 5.9 *In any restriction category,*

(i) *All monics are total*

(ii) *If f and g are total and composable, then fg is total*

(iii) *If fg is total, then f is total*

PROOF:

(i) We have $\overline{f}\,f = f = 1f$ which implies, as f is monic, that $\overline{f} = 1$

(ii) Assuming f and g are total and composable, then $\overline{fg} = \overline{f\overline{g}} = \overline{f1} = 1$

(iii) Assuming $\overline{fg}$ is total then $\overline{f} = \overline{f}\;\overline{fg} = \overline{fg} = 1$.

□

As identities are monic, there is an immediate corollary:

Corollary 5.10 *The total maps of a restriction category, $\mathbb{C}$, form a subcategory denoted* $\mathsf{Total}(\mathbb{C})$*.*

In restriction categories there is a weaker notion of "isomorphism": a map $f : A \longrightarrow B$ is a **partial isomorphism** if there is a map $f^{(-1)} : B \longrightarrow A$ such that $ff^{(-1)} = \overline{f}$ and $f^{(-1)}f = \overline{f^{(-1)}}$. The notation suggests that the partial inverse is unique and this is the case:

Lemma 5.11 *In any restriction category:*

(i) *If f is a partial isomorphism then $f^{(-1)}$ is unique;*

(ii) *Partial isomorphisms are closed to composition.*

PROOF:

(i) Suppose g an h are partial inverse to f then

$$g = \overline{g}\,g = gfg = g\overline{f} = gfh = \overline{g}\,h = \overline{h}\,\overline{g}\,h = hfgfh = h\overline{f}\,fh = hfh = \overline{h}\,h = h.$$

(ii) If f and g are partial isomorphisms then fg has partial inverse $g^{(-1)}f^{(-1)}$ as

$$fgg^{(-1)}f^{(-1)} = f\overline{g}\,f^{(-1)} = \overline{fg}\,ff^{(-1)} = \overline{fg}\;\overline{f}\; = \overline{fg}\,.$$

$\square$

A restriction category in which all maps are partial isomorphisms is called an **inverse category**: a one object inverse category is an inverse semigroup (with a unit). Thus, there is another ready source of restriction categories!

SLOGAN
Inverse categories are to restriction categories what groupoids are to categories.

One can also ask whether restriction idempotents split:

Definition 5.12 *A restriction idempotent, e, is a* split restriction *if there exists an m and r such that $e = rm$, and $mr = 1$. We shall say m is a* restriction monic *whenever it is part of a splitting of an idempotent, that is there is an r such that $\overline{rm}\; = rm$, and $mr = 1$.*

Notice that a restriction monic is precisely a partial isomorphism which is monic, Therefore its retraction is unique and, in fact, the restriction idempotent which it splits is unique.

Lemma 5.13 *Let $\mathbb{C}$ be a restriction category. Then*

(i) *If $ms = 1$ then $\overline{sm}\; = \overline{s}$*

(ii) *Suppose $mr = 1 = ms$, and both $rm = \overline{rm}$ and $sm = \overline{sm}$ then $r = s$;*

(iii) *Suppose $mr = 1 = nr$, and either $rm = \overline{rm}$ or $rn = \overline{rn}$ then $m = n$.*

PROOF:

(i) $\overline{s}\; = \overline{sms}\; = \overline{sm}\,\overline{sms}\; = \overline{sm}\,\overline{s}\; = \overline{sm}\,.$

(ii) Here we can either use the fact that it is a partial isomorphism or calculate directly using $rm = \overline{rm}\; = \overline{r}:\; r = r(ms) = \overline{rm}\,s = \overline{rm}\,\overline{sm}\,s = \overline{\overline{sm}\,rm}\,s\overline{smrm}\,s = \overline{sm}\,s = \overline{s}\,s = s.$

(iii) Suppose without loss $rm = \overline{rm}\; = \overline{r}$ then $rm = r(nr)m = (rn)(rm) = rn\overline{r}\; = \overline{rn\overline{r}}\,rn = \overline{r}\,rn = rn$ and since, r is epic, $m = n$.

$\square$

Definition 5.14 *A* **split restriction category** *is a restriction category in which all restriction idempotents split.*

Split restriction categories play an important role in what follows because in a split restriction category from any map we can extract a monic, to represent its domain, and a total map on that domain to obtain the data of a partial map:

$$A \xrightarrow{f} B := A \xleftarrow{f_0} A' \xrightarrow{f_1} B$$

where $\overline{f}$ splits as $rm = \overline{f}$ and $mr = 1$ then $f_0 = m$ and $f_1 = mf$, where mf is total as $\overline{mf} = \overline{m\overline{f}} = \overline{mrm} = \overline{m} = 1$. Thus, in split restriction categories one has a direct representation of the domain of definition of the maps.

A **restriction functor** is just a functor with the additional property that the restriction is preserved $F(\overline{x}) = \overline{F(x)}$. Natural transformations between restriction functors come in two main varities: **strict transformations** and **lax trannsformations**. In both cases they are given by families of maps so that respectively

$$\begin{tikzcd} F(A) \arrow[r, "F(f)"] \arrow[d, "\alpha_A"'] & F(B) \arrow[d, "\alpha_B"] \\ G(A) \arrow[r, "G(f)"'] & G(B) \end{tikzcd} \qquad \begin{tikzcd} F(A) \arrow[r, "F(f)"] \arrow[d, "\alpha_A"'] \arrow[dr, phantom, "\geq"] & F(B) \arrow[d, "\alpha_B"] \\ G(A) \arrow[r, "G(f)"'] & G(B) \end{tikzcd}$$

in the latter case the naturallity diagrams are required to commute up to inequality.

Given any restriction category, $\mathbb{X}$, and any collection E of idempotents which includes the identities then we may split these idempotents and form the category $\mathsf{Split}_E(\mathbb{X})$. The objects as usual are the idempotents and the maps $f : e_1 \longrightarrow e_2$ are those with $e_1 f e_2 = f$. We may also redefine the restriction to be:

$$\frac{e \xrightarrow{f} e}{e \xrightarrow{e\overline{f}} e}$$

Clearly $e\overline{f}\,f = f$ so **[R1]** holds, $e\overline{f}\,e\overline{g} = \overline{e\overline{f}}\,\overline{e\overline{g}}\,e = \overline{e\overline{g}}\,\overline{e\overline{f}}\,e = e\overline{g}\,e\overline{f}$ so **[R2]** holds. But also $e\overline{f}\,e\overline{g} = \overline{e\overline{f}}\,\overline{e\overline{g}}\,e = \overline{e\overline{f}\,\overline{e\overline{g}}}\,e = e\overline{f\overline{g}}$ so **[R3]** holds. Finally $fe'\overline{g} == f\overline{g} = ef\overline{g} = e\overline{fg}\,f$ so **[R3]** holds. This means we have:

Proposition 5.15 *For any restriction category $\mathbb{X}$ and set of idempotents which include the identities $\mathsf{Split}_E(\mathbb{X})$ is a restriction category and the embedding*

$$K_E : \mathbb{X} \longrightarrow \mathsf{Split}_E(\mathbb{X}) : \quad \begin{tikzcd} X \arrow[d, "f"] \\ Y \end{tikzcd} \quad \mapsto \quad \begin{tikzcd} 1_X \arrow[d, "f"] \\ 1_Y \end{tikzcd}$$

is a restriction functor such that $K_E(e)$ for every $e \in E$ is split.

The case which is particular interest to is when $E = \mathcal{O}$, that is all the restriction idempotents. In that case, in $\mathsf{Split}_{\mathcal{O}}(\mathbb{X})$ it is easy to see that all the restriction idempotents split.

5.2 $\mathcal{M}$-categories

To understand why restriction categories give a clean way to discuss partial maps it is useful to have a more explicit formulation of a partial map category so that the two notions can be compared.

Definition 5.16 *Let $\mathbb{C}$ be a category, and $\mathcal{M}$ a class of monics in $\mathbb{C}$. $\mathcal{M}$ is a* stable system of monics *in case*

[SSM1] *All isomorphisms are in $\mathcal{M}$*

[SSM2] *$\mathcal{M}$ is closed to composition*

[SSM3] *For any $m : B' \to B \in \mathcal{M}$, $f : A \to B \in \mathbf{C}$ the following pullback, called an $\mathcal{M}$-pullback, exists and $m' \in \mathcal{M}$:*

$$\begin{tikzcd} A' \arrow[r, "f"] \arrow[d, "m'"'] & B' \arrow[d, "m"] \\ A \arrow[r, "f"'] & B \end{tikzcd}$$

An $\mathcal{M}$-Category *is a pair $(\mathbf{C}, \mathcal{M})$ where $\mathbf{C}$ is a category with a specified system of stable monics $\mathcal{M}$.*

Example 5.17 *Trivial system of monics*

Give any category, take $\mathcal{M}$ to be the collection of isos. This is called the trivial system of monics.

Example 5.18 *Categories with pullbacks and all monics*

Given a category with pullbacks, the set of all monics form a stable system of monics. Isomorphisms are monic, composite of monics are monic. Further, pullbacks of monics along any map are monic:

$$\begin{tikzcd} A' \arrow[r, "f'"] \arrow[d, "d"'] & B' \arrow[d, "m"] \\ A \arrow[r, "f"'] & B \end{tikzcd}$$

In particular, (**Sets**,Monics) is an $\mathcal{M}$-category.

Proposition 5.19 *Given any split restriction category, $\mathbb{C}$, the collection of restriction monics $\mathcal{M}_{\mathbb{C}}$ forms a stable system of monics in the subcategory* Total$(\mathbb{C})$*.*

Proof:

[SSM1] $1 = \overline{1}$ is a restriction idempotent, so any isomorphism is the monic part of the splitting of a restriction idempotent.

[SSM2] If m and m' split restriction idempotents with retractions r and r' respectively then $rr'm'm = r\overline{r'}m = \overline{rr'}\,rm = \overline{rr'}\,\overline{r} = \overline{rr'}$ so the composite of restriction monics is a restriction monic.

[SSM3] The proof that M_C has $\mathcal{M}$-pullbacks is a bit more involved. It involves constructing pullbacks in the total maps of a split restriction category using only the splittings and composition. Suppose that $m_e : B' \longrightarrow B \in \mathcal{M}_{\mathbb{C}}$ and $f : A \longrightarrow B \in \mathsf{Total}(\mathbb{C})$, the fact that we are in a split restriction category immediately gives the following data:

$$\begin{tikzcd}
 & B' \arrow[d, "m_e"'] \\
A \arrow[r, "f"'] \arrow[loop below, "\overline{fe}"] & B \arrow[u, dotted, bend right, "r_e"'] \arrow[loop below, "\overline{e}=e"]
\end{tikzcd}$$

Next, we can form a square by splitting $\overline{fe} = mr$.

$$\begin{tikzcd}
A' \arrow[r, "mfr_e"] \arrow[d, "m"] & B' \arrow[d, "m_e"'] \\
A \arrow[r, "f"'] \arrow[u, dotted, bend left, "r"] \arrow[loop below, "\overline{fe}"] & B \arrow[u, dotted, bend right, "r_e"'] \arrow[loop below, "\overline{e}"]
\end{tikzcd}$$

To show that the square commutes, i.e. $mf = mfr_em_e$, note that

$$mfr_em_e = mf\overline{e}\,fm = f\overline{ef}\,m = fmrm = fm$$

Furthermore, this all must happen within $\mathbf{Total}(\mathbf{C})$, so we must show that mfr_e is a total map. But, f is assumed to be total and m, m_e are total since they are monic and so mf is total. Now since $mf = mfr_em_e$, r_efm is total. The splitting of restrictions guaranteed that m is monic, so it remains to show that we indeed have in this square a pullback. Suppose we have an object Q with maps $k : Q \longrightarrow B'$ and $h :\longrightarrow A$ in $\mathsf{Total}(\mathbb{C})$ that form a commuting square:

$$\begin{tikzcd}
Q \arrow[dr, dotted, "q"] \arrow[drr, bend left, "k"] \arrow[ddr, bend right, "h"'] & & \\
 & A' \arrow[r, "mfr_e"'] \arrow[d, "m"] & B' \arrow[d, "m_e"] \\
 & A \arrow[r, "f"'] & B
\end{tikzcd}$$

with regions labelled (a) and (b).

Now, we know there is a map from $q : Q \longrightarrow A'$, namely hr. To see that this map makes (a) commutes we have

$$h = \overline{km_e}\,h = \overline{km_er_em_e}\,h = \overline{km_ee}\,h = h\overline{fe} = hrm = qm$$

the fact that this commutes also shows that q is total. (b) commutes as $qmfr_em_e = qmf = hf = km_e$ and m_e is monic so $qmfr_e = k$. Finally, the fact that m is monic ensures that q is unique.

$\square$

Thus any split restriction category has its the total map category an $\mathcal{M}$-category. Given any restriction category we can always split the idempotents and whence associate with it an $\mathcal{M}$-category.

The importance of an $\mathcal{M}$-category is that one may build a partial map category from it by the following well-known construction. The construction is designed to catch the intuition behind partial maps:

Definition 5.20 *Let $(\mathbb{C}, \mathcal{M})$ be an $\mathcal{M}$-Category. Define $Par(\mathbb{C}, \mathcal{M})$ to be the category where*

Objects: *The objects of $\mathbb{C}$*

Maps: $A \xrightarrow{(m,f)} B$ *are classes of spans (m, f),*

$$\begin{tikzcd} & A' \arrow[dl, "m"'] \arrow[dr, "f"] & \\ A & & B \end{tikzcd}$$

where $m \in \mathcal{M}$. The classes of spans are formed under the equivalence relation $\sim$ which is the relation, $(m, f) \sim (m', f')$ if there is an isomorphism, ϕ, such that both triangles in the following diagram commute.

$$\begin{tikzcd} & A' \arrow[r, dotted, "\alpha"] \arrow[dl, bend right, "m"'] \arrow[drr, "f"] & A'' \arrow[dll, "m'"] \arrow[dr, bend left, "f'"] & \\ A & & & B \end{tikzcd}$$

Identities: $A \overset{(1_A,1_A)}{\longrightarrow} A$

Composition: *By pullback; i.e. given $A \longrightarrow (m, f)B$ $B \longrightarrow (m', f')C$, the pullback*

$$\begin{tikzcd} & & A'' \arrow[dl, "m''"'] \arrow[dr, "f''"] & & \\ & A' \arrow[dl, "m"'] \arrow[dr, "f"] & & B' \arrow[dl, "m'"'] \arrow[dr, "f'"] & \\ A & & B & & C \end{tikzcd}$$

gives a composite $(m''m, f''f') : A \longrightarrow C$. Note for associativity the equivalence relation $\sim$ is important.

Lemma 5.21 *$Par(\mathbb{C}, \mathcal{M})$ is a restriction category with the restriction structure given by $\overline{(m, f)} = (m, m)$.*

Proof:

[R1] Consider $(m, m)(m, f)$, noting $m : A' \to A$ is monic iff

$$\begin{tikzcd}
A' \arrow[r, "1_{A'}"] \arrow[d, "1_{A'}"'] & A' \arrow[d, "m"] \\
A' \arrow[r, "m"'] & A
\end{tikzcd}$$

is a pullback, so the composite is

$$\begin{tikzcd}
& & A' \arrow[dl, "1_{A'}"'] \arrow[dr, "1_{A'}"] & & \\
& A' \arrow[dl, "m"'] \arrow[dr, "m"] & & A' \arrow[dl, "m"'] \arrow[dr, "f"] & \\
A & & A & & B
\end{tikzcd}$$

Thus, $(m, m)(m, f) = (m1_{A'}, f1_{A'}) = (m, f)$, giving R1.

[R2] Consider $\overline{(m, f)} = (m, m)$, and $\overline{(m', g)} = (m', m')$

$$\begin{tikzcd}
& & A' \arrow[dl, "1_{A'}"'] \arrow[dr, "1_{A'}"] & & \\
& A' \arrow[dl, "m"'] \arrow[dr, "m"] & & A' \arrow[dl, "m"'] \arrow[dr, "f"] & \\
A & & A & & B
\end{tikzcd}
\qquad
\begin{tikzcd}
& & A' \arrow[dl, "1_{A'}"'] \arrow[dr, "1_{A'}"] & & \\
& A' \arrow[dl, "m"'] \arrow[dr, "m"] & & A' \arrow[dl, "m"'] \arrow[dr, "f"] & \\
A & & A & & B
\end{tikzcd}$$

To make diagrams equal we need $\alpha = 1_{A''}$ and $x'm' = xm$, but this is given by the commutativity of the pullback.

[R3] Take $A \xrightarrow{(m, f)} B$, $A \to (m', f')C$, then we have,

$$\begin{tikzcd}
& A \arrow[dl, bend right, "m"'] \arrow[dr, bend left, "f"] & & A'' \arrow[dlll, "m'"] \arrow[dr, bend left, "f'"] & \\
A & & B & & C
\end{tikzcd}$$

First, consider $(m, m)(m', f')$,

$$\begin{tikzcd}
& & A'' \arrow[dl, "m''"'] \arrow[dr, "f''"] & & \\
& A' \arrow[dl, "m"'] \arrow[dr, "m"] & & B' \arrow[dl, "m'"'] \arrow[dr, "f'"] & \\
A & & A & & B
\end{tikzcd}$$

Thus, $(m,m)\overline{(m'f')} = \overline{(m''m, f''f)} = (m''m, m''m)$

Next, consider $\overline{(m,f)}\;\overline{(m',f')} = (m,m)(m',m')$,

$$\begin{tikzcd}
 & & A'' \arrow[dl, "m''"'] \arrow[dr, "f''"] & & \\
 & A' \arrow[dl, "m"'] \arrow[dr, "m"] & & B' \arrow[dl, "m"'] \arrow[dr, "m"] & \\
A & & A & & A
\end{tikzcd}$$

Next, note that $f''m' = m''m$ by the commutativity of the pullback; hence, $\overline{(m,f)}\;\overline{(m',f')} = (m''m, f''m') = (m''m, m''m)$. Therefore, [R3] holds.

[R4] Take $A \xrightarrow{(m,f)} B, B \xrightarrow{(m',g)} C$ then the composite gives $(m,f)(m',g) = (m''m, f'g)$, as

$$\begin{tikzcd}
 & & A'' \arrow[dl, "m''"'] \arrow[dr, "f'"] & & \\
 & A' \arrow[dl, "m"'] \arrow[dr, "f"] & & B' \arrow[dl, "m'"'] \arrow[dr, "g"] & \\
A & & B & & C
\end{tikzcd}$$

Next consider “$f\overline{g}$,” i.e. $(m,f)(m',m')$,

$$\begin{tikzcd}
 & & A'' \arrow[dl, "m''"'] \arrow[dr, "f'"] & & \\
 & A' \arrow[dl, "m"'] \arrow[dr, "f"] & & B' \arrow[dl, "m'"'] \arrow[dr, "m'"] & \\
A & & B & & B
\end{tikzcd}$$

Thus using the commutativity of the pullback to give $f'm' = m''f$, conclude $(m,f)(m',m') = (m''m, f'm') = (m''m, m''f)$

Next consider “$\overline{f\overline{g}}\,f$, i.e. $(m''m, m''m)(m,f)$,

$$\begin{tikzcd}
 & & A'' \arrow[dl, "1_{A''}"'] \arrow[dr, "m''"] & & \\
 & A'' \arrow[dl, "m''m"'] \arrow[dr, "m''m"] & & A' \arrow[dl, "m"'] \arrow[dr, "f"] & \\
A & & A & & B
\end{tikzcd}$$

Thus, $(m''m, m''m)(m,f) = (m''m, m''f)$, and so [R4] holds.

Therefore, $\mathsf{Par}(\mathbb{C}, \mathcal{M})$ with $\overline{(m, f)} = (m, m)$ is a restriction category. $\square$

Lemma 5.22 *The restriction structure on* $\mathsf{Par}(\mathbb{C}, \mathcal{M})$ *is split.*

Proof: Note that $(1, 1) = (1, m)(m, 1)$, and $(m, m) = (m, 1)(1, m)$ since,

$$\begin{tikzcd}
& & A \arrow[dl, "1_A"'] \arrow[dr, "1_A"] & & \\
& A \arrow[dl, "m"'] \arrow[dr, "1_A"] & & A \arrow[dl, "1_A"'] \arrow[dr, "m"] & \\
A & & A & & A
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
& A \arrow[dl, "m"'] \arrow[dr, "m"] & \\
A & & A
\end{tikzcd}$$

and

$$\begin{tikzcd}
& & A \arrow[dl, "1_A"'] \arrow[dr, "1_A"] & & \\
& A \arrow[dl, "1_A"'] \arrow[dr, "m"] & & A \arrow[dl, "m"'] \arrow[dr, "1_A"] & \\
A & & A & & A
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
& A \arrow[dl, "1_A"'] \arrow[dr, "1_A"] & \\
A & & A
\end{tikzcd}$$

$\square$

This means that given a restriction category we can split the idempotents, take the $\mathcal{M}$-category of total maps, and finally take the partial map category of this. A key observation is:

Lemma 5.23 *There is a full and faithful embedding for any restriction category*

$$E : \mathbb{C} \longrightarrow \mathsf{Par}(\mathsf{Total}(\mathsf{Split}(\mathbb{C})), \mathcal{M}_{\mathsf{Split}(\mathbb{C})}).$$

Proof: If $\mathbb{C}$ is a split restriction category then we shall prove the stronger property that the embedding $E : \mathbb{C} \longrightarrow \mathsf{Par}(\mathsf{Total}(\mathbb{C}), \mathcal{M}_{\mathbb{C}})$ is an isomorphism of categories. This will prove the lemma.

Define $E(C) = C$ and $E(f) = (m_{\overline{f}}, m_{\overline{f}} f)$. Set $E^{-1}(m, g) = rg$ where r is the unique retraction of m in $\mathbb{C}$. Then $E(E^{-1}(m, g)) = E(rg)$ but $\overline{rg} = \overline{r}$ as g is total and $\overline{r} = rm$ so (up to $\sim$ equality) $E(rg) = (m, g)$. Conversely, $E^{-1}(E(f)) = E^{-1}(m_{\overline{f}}, m_{\overline{f}} f) = r_{\overline{f}} m_{\overline{f}} f = f$. So the maps are certainly inverse and it suffices to prove that one is functorial.

Consider E: clearly it preserves identities so the only difficulty concerns composition. But $E(fg) = (m_{\overline{fg}}, m_{\overline{fg}} fg)$ and $E(f)E(g) = (m_{\overline{f}}, m_{\overline{f}} f)(m_{\overline{g}}, m_{\overline{g}} g)$. Calculating the pullback we must split $\overline{m_{\overline{f}} f \overline{g}} = \overline{m_{\overline{f}} fg}$ to obtain $(m_{\overline{m_{\overline{f}} fg}} m_{\overline{f}}, m_{\overline{m_{\overline{f}} fg}} m_{\overline{f}} f)$. But $m_{\overline{m_{\overline{f}} fg}} m_{\overline{f}}$ splits $\overline{fg}$ as:

$$r_{\overline{f}} r_{\overline{m_{\overline{f}} fg}} m_{\overline{m_{\overline{f}} fg}} m_{\overline{f}} = r_{\overline{f}} \overline{m_{\overline{f}} fg}\, m_{\overline{f}} = r_{\overline{f}} m_{\overline{f}} \overline{fg} = \overline{f}\, \overline{fg} = \overline{fg}.$$

$\square$

This allows us to state the completeness result for restriction categories in the following form:

Theorem 5.24 *Every restriction category occurs as a full subcategory of a partial map category.*

5.3 Joins and meets

A restriction category is said to have **(finite) joins** if:

[J1] For each pair of objects A and B there is a zero map $0_{AB} : A \longrightarrow B$ which is stable so that for every $A \xrightarrow{f} B$, $0_{AB} \leq F$ and $f 0_{BC} g = 0_{AC}$.

[J2] For all compatible arrows f and g there is a join $f \vee g$ (in the sense that $f \leq f \vee g$, $g \leq f \vee g$ and whenever $f \leq h$ and $gleqh$ the $f \vee g \leq h$) which is stable so that $h(f \vee g) = (hf) \vee (hg)$.

A **(finite) join restriction category** is restriction category with finite joins. We shall drop the "finite" prefix in these notes, however, before we do we should remark that a restriction category with arbitrary joins is an interesting beast. The open set lattices $\mathcal{O}(A)$ are locales and this makes them especially interesting from the topological perspective. Furthermore each finite join restriction category can be completed to one with arbitrary joins.

The definition was given in a minimalist way we note:

Lemma 5.25 *In any join restriction category:*

(i) $\overline{0} = 0$*;*

(ii) *For every* g*,* $0g = 0$*;*

(iii) $\overline{f \vee g} = \overline{f} \vee \overline{g}$*;*

(iv) $(f \vee g)h = (fh) \vee (gh)$*.*

Proof:

(i) $0 \leq \overline{0}$ so that $0 = \overline{0}$.

(ii) $0g = \overline{g0}\, g = g\overline{0} = g0 = 0$;

(iii) It is clear that $\overline{f} \vee \overline{g} \leq \overline{f \vee g}$, and so $\overline{f} \vee \overline{g} = \overline{\overline{f} \vee \overline{g}}$. We must show the reverse inequality

$$\begin{aligned}
\overline{f \vee g}\,(\overline{f} \vee \overline{g}) &= \overline{f \vee g}\,(\overline{f} \vee \overline{g}) \\
&= \overline{(\overline{f} \vee \overline{g})(f \vee g)} = \overline{(\overline{f} \vee \overline{g})f \vee (\overline{f} \vee \overline{g})g} \\
&= \overline{\overline{f}\,(\overline{f} \vee \overline{g})f \vee \overline{g}\,(\overline{f} \vee \overline{g})g} = \overline{\overline{f}\,f \vee \overline{g}\,g}\,\overline{f \vee g}
\end{aligned}$$

(iv) Again it is clear that $(fh) \vee (gh) \leq (f \vee g)h$, we must show that the reverse inequality holds. To do this we shall first establish that $(f \vee g)\overline{h} = (f\overline{h}) \vee (g\overline{h})$ as

$$\begin{aligned}
(f \vee g)\overline{h} &= \overline{(f \vee g)h}\,(f \vee g) = (\overline{(f \vee g)h}\,f \vee \overline{(f \vee g)h}\,g) \\
&= (\overline{(f \vee g)h}\;\overline{f}\,f \vee \overline{(f \vee g)h}\;\overline{g}\,g) = (\overline{f}\;\overline{(f \vee g)h}\,f \vee \overline{g}\;\overline{(f \vee g)h}\,g) \\
&= (\overline{\overline{f}\,(f \vee g)h}\,f \vee \overline{\overline{g}\,(f \vee g)h}\,g) = (\overline{fh}\,f \vee \overline{gh}\,g) = (f\overline{h} \vee g\overline{h})
\end{aligned}$$

It remains to show that $(f \vee g)h \leq (fh) \vee (gh)$ and to show this it suffices to show $\overline{(f \vee g)h} = \overline{(fh) \vee (gh)}$ but for this we have:

$$\begin{aligned}
\overline{(fh) \vee (gh)} &= \overline{fh} \vee \overline{gh} = \overline{f\overline{h}} \vee \overline{g\overline{h}} = \overline{f\overline{h} \vee g\overline{h}} \\
&= \overline{(f \vee g)\overline{h}} = \overline{(f \vee g)h}
\end{aligned}$$

$\square$

If a join restriction category has coproducts we now show that the coproducts are absolute in the sense that any restriction functor which preserves the joins must necessarily preserve the coproducts. To see why this is so consider:

Lemma 5.26 *In any restriction category with joins and coproducts:*

(i) The coproduct embeddings $\sigma_0 : A \longrightarrow A + B$ and $\sigma_1 : B \longrightarrow A + B$ are not only total but sections;

(ii) The coproduct functor is a restriction functor, that is $\overline{f+g} = \overline{f} + \overline{g}$;

(iii) The coproduct embeddings are restriction monics;

(iv) The copairing map $\langle f|g\rangle = \langle f|0\rangle \vee \langle 0|g\rangle$;

(v) $\langle f \vee f'|g \vee g'\rangle = \langle f|g\rangle \vee \langle f'|g'\rangle$.

Proof:

(i) Consider the copoduct embedding $\sigma_0 : A \longrightarrow A + B$: there is a map $\langle 1|0_{B,A}\rangle : A + B \longrightarrow A$ and $\sigma_0\langle 1_A|0_{B,A}\rangle = 1_A$. So σ_0 is a section.

(ii) We have:

$$\begin{aligned} \overline{f+g} &= \langle \sigma_0\overline{f+g}\,|\sigma_1\overline{f+g}\,\rangle \\ &= \langle \overline{\sigma_0(f+g)}\,\sigma_0|\overline{\sigma_1(f+g)}\,\sigma_1\rangle \\ &= \langle \overline{f}\,\sigma_0|\overline{g}\,\sigma_1\rangle = \overline{f} + \overline{g}\,. \end{aligned}$$

(iii) Consider $\langle 1|0_{B,A}\rangle\sigma_0 = \langle \sigma_0|0_{B,A}\sigma_0\rangle = \langle \sigma_0|0_{B,A+B}\rangle = 1+0$: as $1+0$ is a restriction idempotent this shows that $\langle 1|0_{B,A}\rangle$ is the partial inverse to σ_0.

(iv) Note that $\sigma_0(\langle f|0\rangle \vee \langle 0|g\rangle) = .(\sigma_0\langle f|0\rangle) \vee (\sigma_0\langle 0|g\rangle) = f \vee 0 = f$ so that $\langle f|0\rangle \vee \langle 0|g\rangle = \langle f|g\rangle$.

(v) $\sigma_0(\langle f|g\rangle \vee \langle f'|g'\rangle) = (\sigma_0\langle f|g\rangle) \vee (\sigma_0\langle f'|g'\rangle) = f \vee f'$.

$\square$

The absolute nature of coproducts follows from:

Proposition 5.27 *In any restriction category with joins $A \xrightarrow{a} C \xleftarrow{b} B$ is a coproduct if and only if a and b are restriction monics such that $\overline{a^{(-1)}}\;\overline{b^{(-1)}} = 0$ and $\overline{a^{(-1)}} \vee \overline{b^{(-1)}} = 1_C$.*

Proof: If this were a coproduct the above establishes that the coproduct embeddings must be partial isomorphisms also $1_{A+B} = (\sigma_0 + 0) \vee (\sigma_1 + 0)$ and $(\sigma_0 + 0)(\sigma_1 + 0) = 0 + 0 = 0$.

Conversely define $\langle f|g\rangle := (a^{(-1)}f) \vee (b^{(-1)}g)$ where $a^{(-1)}f \smile b^{(-1)}g$ as $\overline{a^{(-1)}}\;\overline{b^{(-1)}} = 0$. Then

$$a((a^{(-1)}f) \vee (b^{(-1)}g)) = (aa^{(-1)}f) \vee (ab^{(-1)}g) = f \vee (a\overline{a^{(-1)}}\,b^{(-1)}g) = f \vee 0 = f.$$

Thus, it remains only to show this map is unique. For this suppose:

$$\begin{tikzcd}
A \arrow[r, "a"] \arrow[dr, "f"'] & C \arrow[d, "h"] & B \arrow[l, "b"'] \arrow[dl, "g"] \\
& D &
\end{tikzcd}$$

then $a^{(-1)}f = a^{(-1)}ah = \overline{a^{(-1)}}\,h$ and

$$h = 1h = (\overline{a^{(-1)}} \vee \overline{b^{(-1)}}\,)h = (\overline{a^{(-1)}}\,h) \vee (\overline{b^{(-1)}}\,h) = (a^{(-1)}f) \vee (b^{(-1)}g).$$

$\square$

Corollary 5.28 *If F is a join preserving restriction functor between join restriction categories then F preserves coproducts.*

A restriction category can also have the **meets** of parallel maps, written $f \cap g$. This operation is defined for all parallel maps and satisfies:

[Mt1] $f \cap f = f$

[Mt2] $f \cap g \leq g$ and $f \cap g \leq f$

[Mt3] $h(f \cap g) = hf) \cap (hg)$

A restriction category with meets is called a **meet restriction category**.

The meet of two partial maps in sets and partial maps is precisely their intersection (viewed as relations), that it is the part on which they agree. So, for example, we would expect the meet of two compatible maps to be the restriction of either to their common domain:

Lemma 5.29 *In any meet restriction category:*

(i) *If $h \leq f$ and $h \leq g$ then $h \leq f \cap g$ (so it really is the meet!);*

(ii) $\overline{h}\,(f \cap g) = (\overline{h}\,f \cap g) = (f \cap \overline{h}\,g)$*;*

(iii) *If $f \leq f'$ then $f \cap g \leq f' \cap g$;*

(iv) *If $f \smile g$ then $f \cap g = \overline{f}\,g$.*

(v) $(f \cap g)\overline{h} = (f\overline{h} \cap g\overline{h}\,)$*;*

(vi) $\overline{f \cap g}\,f = \overline{f \cap g}\,g$*.*

Proof:

(i) $\overline{h}\,(f \cap g) = \overline{h}\,f \cap \overline{h}\,g = h \cap h = h$.

(ii) $(\overline{h}\,f \cap g) \leq \overline{h}\,f$ and $(\overline{h}\,f \cap g) \leq g$ so that certainly $(\overline{h}\,f \cap g) \leq \overline{h}\,g$, thus, $(\overline{h}\,f \cap g) \leq (\overline{h}\,f \cap \overline{h}\,g) = \overline{h}\,(f \cap g)$.

(iii) If $f \leq f'$ then $f = \overline{f}\, f'$ so $\overline{f}\,(f' \cap g) = (\overline{f}\, f' \cap g) = (f \cap g)$ so that $f \cap g \leq f' \cap g$.

(iv) If $f \smile g$ then $\overline{f}\, g = \overline{g}\, f$ so that $f \cap g = \overline{f}\ \overline{g}\,(f \cap g) = \overline{g}\, f \cap \overline{f}\, g = \overline{f}\, g$.

(v) Clearly $(f\overline{h} \cap g\overline{h}\,) \leq (f \cap g)\overline{h}$ for the converse we have:

$$\begin{aligned}(f \cap g)\overline{h} \quad &= \quad \overline{(f \cap g)h}\,(f \cap g) = ((\overline{(f \cap g)h}\, f) \cap \overline{(f \cap g)h}\, g) \\ &\leq \quad (\overline{fh}\, f) \cap \overline{gh}\, g) = (f\overline{h}\,) \cap (g\overline{h}\,).\end{aligned}$$

(vi) As $f \cap g \leq f$ and $f \cap g \leq g$ then $\overline{f \cap g}\, f = f \cap g = \overline{f \cap g}\, g$.

$\square$

Notice that we do not ask for this meet to have a top element as this will not be the case in general. Similarly, we do not ask that $(f \cap g)h = (fh) \cap (gh)$ because this will not be the case in general.

One might expect to have to demand that if a meet restriction category has joins that the meet distribute over the join. However, it is a remarkable coincidence of structure that this is already implicit in the situation:

Lemma 5.30 *In any meet restriction category with joins the meet distributes over the join:*

$$h \cap (f \vee g) = (h \cap f) \vee (h \cap g).$$

PROOF:

$$\begin{aligned}h \cap (f \vee g) \quad &= \quad \overline{(f \vee g)}\, h \cap (f \vee g) = (\overline{f}\ \vee \overline{g}\,)h \cap (f \vee g) \\ &= \quad (\overline{f}\,(h \cap (f \vee g))) \vee (\overline{g}\,(h \cap (f \vee g))) = (h \cap \overline{f}\,(f \vee g)) \vee (h \cap \overline{g}\,(f \vee g)) \\ &= \quad (h \cap \overline{f}\,(f \vee g)) \vee (h \cap \overline{g}\,(f \vee g)) = (h \cap f) \vee (h \cap g)\end{aligned}$$

$\square$

In a restriction category, f is a **partial monic** in case whenever $g_1 f = g_2 f$ then $g_1\overline{f}\ = g_2\overline{f}\,$. Clearly monics are partial monics and so, therefore, are restriction monics.

Lemma 5.31

(i) *In any restriction category: if f and g are partial monics then fg is a partial monic and, furthermore, if fg is a partial monic then $f\overline{gf}$ is a partial monic.*

(ii) *In a meet restriction category f is a partial monic if and only if $fx \cap fy = f(x \cap y)$ for all x and y.*

PROOF:

(i) Suppose f and g are partial monics and $h_1 fg = h_2 fg$ then, using that g is partial monic yields $h_1\overline{fg}\, f = h_1 f\overline{g}\ = h_2 f\overline{g}\ = h_2\overline{gf}\, f$ which allows one to use the fact that f is partial monic to obtain $h_1\overline{fg}\ = h_2\overline{fg}\,$.

Now suppose that fg is a partial monic and $h_1\overline{fg}\, f = h_2\overline{fg}\, f$ so that $h_1\overline{fg}\, fg = h_2\overline{fg}\, fg$, if follows as fg is partial monic that $h_1\overline{fg}\ = h_2\overline{fg}\ $ which shows $\overline{fg}\, f$ is partial monic.

(ii) As $\overline{xf \cap yf}\, xf = \overline{xf \cap yf}\, yf = xf \cap yf$ and f is a partial monic it follows that

$$\overline{xf \cap yf}\, x = \overline{xf \cap yf}\, x\overline{f} = \overline{xf \cap yf}\, y\overline{f} = \overline{xf \cap yf}\, y$$

so that

$$\overline{xf \cap yf}\, (x \cap y)f = (\overline{xf \cap yf}\, x\overline{f}\,) \cap (\overline{xf \cap yf}\, y\overline{f}\,)f = \overline{xf \cap yf}\, xf = xf \cap yf$$

showing that $xf \cap yf \leq (x \cup y)f$. The inequality $(x \cup y)f \leq xf \cap yf$ is, however, immediate. Thus we have established that when f is partial monic $(x \cup y)f = xf \cap yf$.

Conversely, suppose $(x \cup y)f = xf \cap yf$ then when $xf = yf$ it follows $xf = (x \cup y)f = yf$ so that

$$x\overline{f} = \overline{xf}\, x = \overline{(x \cup y)f}\, x = \overline{(x \cup y)f}\, x = \overline{(x \cup y)f}\, y = \overline{yf}\, y = y\overline{f}\,.$$

$\square$

An important observation concerning meet restriction categories is that *all* idempotents split if and only if the restriction idempotents split. This is a consequence of:

Lemma 5.32 *If e is an idempotent and the restriction idempotent $1 \cap e$ splits as $mr = 1$ and $rm = 1 \cap e$ then e splits and $m(er) = 1$ and $(er)m = e$.*

PROOF: We us ethe fact that $\overline{(f \cap g)}\, f = \overline{(f \cap g)}\, g$ and that $(1 \cap e) = \overline{(1 \cap e)}$ (as $(1cape \leq 1)$ to conclude that $(1 \cap e)e = \overline{(1 \cap e)}\, e = \overline{(1 \cap e)}\, 1 = 1 \cap e$. This allow the calculations $m(er) = m(1 \cap e)er = m(1 \cap e)r = mr = 1$ and $erm = e(1 \cap e) = (e \cap e) = e$. $\square$

5.4 Cartesian restriction categories

A **cartesian restriction category**[5], $\mathbb{X}$, is a restriction category which has a (partial) final object and for which each pair of objects has a (partial) product.

To say that there is a **partial final object** means, explicitly, there is an object 1 which has from each object a total map $!_A : A \longrightarrow 1$ so that any map $f : A \longrightarrow 1$ has $f = \overline{f}\, !_A$ This, in turn, means the maps from $A \longrightarrow 1$ are in bijective correspondence to the restriction idempotents on A, $\mathcal{O}(A)$. Clearly a partial terminal object becomes a true terminal object in the total map subcategory.

To say that $\mathbb{X}$ has binary (partial) products is to say that for each $A, B \in \mathbb{X}$ there is an object $A \times B$ with total maps $\pi_0 : A \times B \longrightarrow A$ and $\pi_1 : A \times B \longrightarrow B$ such that for any pair of maps $f : Z \longrightarrow A$ and $g : Z \longrightarrow B$ there is a unique map $\langle f, g\rangle : Z \longrightarrow A \times B$ such that $\langle f, g\rangle\pi_0 = \overline{g}\, f$ and $\langle f, g\rangle\pi_1 = \overline{f}\, g$. This means that $\_ \times \_ : \times \times \mathbb{X} \longrightarrow \mathbb{X}$ is a restriction functor and that : $\langle 1_A, 1_A\rangle = \Delta_A : A \longrightarrow A \times A$ is total and a natural transformation.

Notice that the projections are lax natural rather than natural.in the sense that we have:

$$\begin{tikzcd} A \times B \arrow[r, "f \times g"] \arrow[d, "\pi_0"'] \arrow[dr, phantom, "\geq"] & A' \times B' \arrow[d, "\pi_0"] \\ A \arrow[r, "f"'] & A' \end{tikzcd}$$

[5]Di Paola and Heller called partial products “near products” and these where the basis for Rossolini and Robinson P-categories. A cartesian restriction category is a P-category with a one “element object” in their terminology. The terminology we use here appropriately emphasizes the parallel to ordinary categories.

Also note that on the total category this means we have real products. This ensures that the cartesian structure is unique.

Here are some useful manipulations with regard to the restriction structure:

Lemma 5.33 *In any cartesian restriction category:*

(i) $\overline{\langle f, g\rangle} = \overline{f}\,\overline{g}$

(ii) If $e = \overline{e}$ *then* $e\langle f, g\rangle = \langle ef, g\rangle = \langle f, eg\rangle$

A rather useful observation is as follows:

Proposition 5.34 *A split restriction category is a cartesian restriction category if and only it its total map category is cartesian category.*

PROOF: The difficulty is to prove that if a split restriction category has a total map category which is cartesian that it must e a cartesian restriction category. To achieve this we establish that there is a natural cartesian restriction structure on the whole category if there is one on the total map category.

For the partial terminal object suppose $A \xrightarrow{f} 1$ then this may be represented as a span $A \xleftarrow{m_{\overline{f}}} A' \xrightarrow{m_{\overline{f}} f} 1$ as $m_{\overline{f}} f$ is total it must be the unique $!; A' \longrightarrow 1$ but this immediately makes $f = \overline{f}\,!$.

A similar argument works for the pairing: $\langle f, g\rangle$ is obtained by first making f and g have the same domain by altering them to become $\overline{g}\,f$ and $\overline{f}\,g$ and splitting their now common idempotent. This gives two total maps $f' : \overline{f}\,\overline{g} \longrightarrow A$ and $g' : \overline{f}\,\overline{g} \longrightarrow B$ as we have products in the total category we may pair these giving a spans of a restriction monic and a total map which corresponds uniquely to a partial map in the category. □

The point is that any partial map category, $\mathsf{Par}(\mathbb{X}, \mathcal{M})$, of a category $\mathbb{X}$ which already has products immediately gives rise to a cartesian restriction category. This therefore provides many examples.

We now turn to the effect of adding joins and meets to a cartesian restriction category. Again it would seem that we should have to enforce that these structures work together nicely: remarkably everything, as above, is already implicit. In particular, we would like the partial product functor to preserve joins $(f \vee g) \times h = (f \times h) \vee (g \times h)$ but there is no need to add this as a requirement:

Lemma 5.35 *In any cartesian restriction category with joins* $(f \vee g) \times h = (f \times h) \vee (g \times h)$.

PROOF: We shall first prove $\langle f \vee g, h\rangle = \langle f, h\rangle \vee \langle g, h\rangle$:

$$\begin{aligned}
\langle f \vee g, h\rangle &= \overline{\langle f \vee g, h\rangle}\,\langle f \vee g, h\rangle = \overline{f \vee g}\;\overline{h}\,\langle f \vee g, h\rangle \\
&= \overline{f} \vee \overline{g}\,\langle f \vee g, h\rangle = (\overline{f}\,\langle f \vee g, h\rangle) \vee (\overline{g}\,\langle f \vee g, h\rangle) \\
&= \langle \overline{f}\,(f \vee g), h\rangle \vee \langle \overline{g}\,(f \vee g), h\rangle = \langle f, h\rangle \vee \langle g, h\rangle
\end{aligned}$$

Now

$$(f \vee g) \times h = \langle \pi_0(f \vee g), \pi_1 h\rangle = \langle (\pi_0 f) \vee (\pi_0 g), \pi_1 h\rangle = \langle \pi_0 f, \pi_1 h\rangle \vee \langle \pi_0 g, \pi_1 h\rangle = (f \times h) \vee (g \times h).$$

$\square$

Suppose now that we add coproducts to a join cartesian restriction category as coproducts are absolute and $A \times \_$ preserves joins it follows, in the light of 5.28, that $A \times (B + C) \cong (A \times B) + (A \times C)$ in other words the distributivity of products over coproducts holds. We shall therefore call a cartesian restriction category which has joins and coproducts a **distributive restriction category**.

We now consider the effect of adding meets to a cartesian restriction category.

In a cartesian restriction category an object A is said to be **discrete** in case its diagonal map $\Delta : A \longrightarrow A \times A$ is a restriction monic. Explicitly this means that Δ_A has a partial inverse $\Delta_A^{(-1)}$ and, to remind us of its relationship to equality, we shall sometimes write the restriction idempotent $\overline{\Delta_A^{(-1)}}$ as eq_A.

A cartesian restriction category in which all objects are discrete is called a **discrete restriction category** (note the cartesianness is assumed). The following observation links meets and discreteness:

Proposition 5.36 *If $\mathbb{X}$ is a cartesian restriction category then $\mathbb{X}$ is discrete if and only if $\mathbb{X}$ is a meet restriction category.*

PROOF: If $\mathbb{X}$ is discrete define $f \cap g = \langle f, g \rangle \Delta^{(-1)}$ then

[Mt1] $f \cap f = \langle f, f \rangle \Delta^{(-1)} = \Delta (f \times f) \Delta^{(-1)} = f \Delta \Delta^{(-1)} = f$

[Mt2] To prove $f \cap g \leq f$ we shall first establish that $\overline{(f \cap g)}\, f = \overline{(f \cap g)}\, g$ by:

$$\begin{aligned}
\overline{(f \cap g)}\, f &= \overline{\langle f, g > \Delta^{(-1)}}\, f = \langle f, g > \Delta^{(-1)}\ \overline{g}\, f \\
&= \overline{\langle f, g \rangle \Delta^{(-1)}}\ \langle f, g \rangle \pi_0 = \langle f, g \rangle \Delta^{(-1)} \Delta \pi_0 \\
&= \langle f, g \rangle \Delta^{(-1)} \Delta \pi_0 = \langle g, f \rangle \Delta^{(-1)} \Delta \pi_0 \\
&= \overline{\langle g, f \rangle \Delta^{(-1)}}\, \langle g, f \rangle \pi_0 = \overline{\langle f, g \rangle \Delta^{(-1)}}\, g \\
&= \overline{(f \cap g)}\, g
\end{aligned}$$

then we have

$$\begin{aligned}
(f \cap g) \Delta &= \langle f, g \rangle \Delta^{(-1)} \Delta = \overline{\langle f, g \rangle \Delta^{(-1)}}\, \langle f, g \rangle \\
&= \langle f, \overline{f \cap g}\, g \rangle = \langle f, \overline{f \cap g}\, f \rangle = \overline{f \cap g}\, \langle f, f \rangle = \overline{f \cap g}\, f \Delta
\end{aligned}$$

so that $f \cap g \leq f$ and similarly $f \cap g \leq g$.

[Mt3] $h (f \cap g) = h \langle f, g \rangle \Delta^{(-1)} = \langle hf, hg \rangle \Delta^{(-1)} = (hf \cap hg)$.

So that this implies that when $\mathbb{X}$ is discrete it is a meet restriction category.

Conversely suppose that $\mathbb{X}$ has partial products and is a meet restriction category then set $\Delta^{(-1)} = \pi_0 \cap \pi_1$. Clearly

$$\Delta (\pi_0 \cap \pi_1) = (\Delta \pi_0 \cap \Delta \pi_1) = 1 \cap 1 = 1$$

and

$$\begin{aligned}
(\pi_0 \cap \pi_1) \Delta &= \langle (\pi_0 \cap \pi_1), (\pi_0 \cap \pi_1) \rangle \\
&= \langle \overline{(\pi_0 \cap \pi_1)}\, \pi_0, \overline{(\pi_0 \cap \pi_1)}\, \pi_1 \rangle \\
&= \overline{(\pi_0 \cap \pi_1)}\, \langle \pi_0, \pi_1 \rangle = \overline{(\pi_0 \cap \pi_1)}
\end{aligned}$$

showing $\pi_0 \cap \pi_1$ is partial inverse to Δ. $\square$

6 Turing categories

Turing categories are cartesian restriction categories with a special kind of weakly universal object, called a Turing object . After the basic definition we show that one can normalize the weak universal structure associated with a Turing object to facilitate verifying that a category is a Turing category. Next some examples are presented mainly from recursion theory; Turing categories are supposed to embody an abstract notion of computability, and so they had better encompass the traditional, concrete notions. The Turing structure on a category is structure which is not canonical and it is interesting to see how different Turing structures on a category are related.

The central definitions of these notes is:

Definition 6.1 (Turing category) *Let $\mathbb{C}$ be a cartesian restriction category.*

(i) Given a morphism $\tau_{X,Y} : A \times X \longrightarrow Y$, a morphism $f : Z \times X \longrightarrow Y$ is said to admit a $\tau_{X,Y}$-index *when there exists a total map $h : Z \longrightarrow A$ for which the following diagram is commutative:*

$$\begin{tikzcd} A \times X \arrow[r, "\tau_{X,Y}"] & Y \\ Z \times X \arrow[u, dashed, "h\times X"] \arrow[ur, "f"'] & \end{tikzcd}$$

In this case, h is called a $\tau_{X,Y}$-index for f.

(ii) A morphism $\tau_{X,Y}$ is called a universal application *when every $f : Z \times X \longrightarrow Y$ admits a $\tau_{X,Y}$-index. We will also simply say that $\tau_{X,Y}$ is* universal.

(iii) A Turing object *in $\mathbb{C}$ is an object A such that for each $X, Y \in \mathbb{C}$, there is a universal application $\tau_{X,Y} : A \times X \longrightarrow Y$.*

(iv) The category $\mathbb{C}$ is called a **Turing category** *if it possesses a Turing object.*

One way to think of a Turing object A is as an object which plays, for each pair of objects X, Y, the role of a weak exponential of X and Y; in that light, the universal application morphism $\tau_{X,Y} : A \times X \longrightarrow Y$ then acts as an evaluation map. We do not insist that the "exponential transpose" h of f is unique; in the (rather special) case where transposes are unique we say that $\tau_{X,Y}$ is **extensional**.

In the case where $Z = 1$, the terminal object, this means that for each morphism $f : X \longrightarrow Y$, there is a total point $a : 1 \longrightarrow A$, such that $(a \times 1)\tau_{1,Y} = f$. We shall refer to a as a **code** for f.

It is important to realize that, given a particular Turing object, the universal morphisms $\tau_{X,Y}$ are not part of the data of a Turing object. Certainly, the existence of such data is required but generally, a Turing object may have many possible families $\tau_{X,Y}$ which realize it as a Turing object. Similarly, in a Turing category, we do not take a choice of Turing object to be part of the structure but rather require the existance of such an object; a Turing category may possess many non-isomorphic Turing objects.

Definition 6.2 (Turing structure) *Let $\mathbb{C}$ be a Turing category and A an object of $\mathbb{C}$.*

1. *A family of maps $\tau = \{\tau_{X,Y} : A \times X \longrightarrow Y | X, Y\}$ where X, Y range over all objects of $\mathbb{C}$ will be called an* **applicative family for** *A.*

2. *An applicative family τ is called* **universal for** *A when each $\tau_{X,Y}$ is universal, in which case we also say that τ is a* **Turing structure** *on A.*

3. *A pair (A, τ), where τ is universal for A is called a* **Turing structure** *on $\mathbb{C}$.*

The question of how different Turing objects in a given category are related and of how different Turing structures on a given Turing object are related will be discussed shortly.

We should give some examples at this point, but as one can imagine, verifying whether a given category is a Turing category using the above definition may not be straightforward. We therefore postpone this until we have gathered a few results which will allow for an equivalent formulation which is easier to check in practice.

As a first observation, we note that a Turing object is a **universal object**, in the strong sense that every object is a retract of it.

Lemma 6.3 *In a Turing category $\mathbb{C}$ with Turing object A, every object X is a retract of A.*

PROOF: Consider the diagram

$$\begin{tikzcd}
A \times 1 \arrow[r, "\tau_{1,X}"] & X \\
X \times 1 \arrow[u, "\widetilde{\pi_X} \times 1"] \arrow[ur, "\pi_X"'] &
\end{tikzcd}$$

where $\widetilde{\pi_X}$ is a $\tau_{1,X}$-index for π_X. Precomposing both ways around with the isomorphism $X \longrightarrow X \times 1$ gives the desired result. $\square$

In particular we have that all finite powers $1, A, A^2, A^3, \ldots$ are retracts of A. This example not only indicates something about the size of the object A (which, if we were working over the category of sets, would have to be either a singleton or infinite) but also tells us that there is an internal pairing operation on A.

We next turn to the problem of normalizing the Turing structure in the sense of showing how such structure can be generated from some key components. One of these key ingredients is a **Turing morphism**: this is a universal self-application map, which we shall write as $A \times A \xrightarrow{\bullet} A$. Clearly, given a universal applicative family $\tau = \{\tau_{X,Y}\}$ on A, the map $\tau_{A,A} : A \times A \longrightarrow A$ is such a Turing morphism. However, simply having such a Turing morphism is not sufficient to generate a Turing structure on A; one needs the additional requirement that A is a universal object.

Let X, Y be arbitrary objects, and suppose we seek to construct a universal application map $\tau_{X,Y} : A \times X \longrightarrow Y$. Let us assume that we have embedding-retraction pairs $(m_X, r_X) : X \triangleleft A$ and $(m_Y, r_Y) : Y \triangleleft A$. We define $\tau_{X,Y}$ to be the composite

$$A \times X \xrightarrow{A \times m_X} A \times A \xrightarrow{\bullet} A \xrightarrow{r_Y} Y.$$

To verify that this morphism is a universal application suppose we are given $f : Z \times X \longrightarrow Y$. Consider the diagram

$$\begin{tikzcd}
A \times X \arrow[r, "A \times m_X"] & A \times A \arrow[r, "\bullet"] & A \arrow[r, "r_Y"] & Y \\
Z \times X \arrow[u, dotted, "h \times X"] \arrow[r, "Z \times m_X"'] & Z \times A \arrow[u, dotted, "h \times A"] \arrow[r, "Z \times r_X"'] & Z \times X \arrow[u, "f m_Y"] \arrow[ur, "f"'] &
\end{tikzcd}$$

The triangle on the right commutes because of the equality $m_Y r_Y = 1$. The square in the middle uses the fact that $\bullet$ is a Turing morphism and hence that there is an index h for the composite $(Z \times r_X) f m_Y$. Obviously the left hand square commutes, and the bottom composite is simply the identity. Thus h is also an index for f relative to the top composite.

This way of constructing the morphisms $\tau_{X,Y}$ from the single instance $\bullet = \tau_{A,A}$ is not canonical. There may in general be many ways of extending $\tau_{A,A}$ to a Turing structure on A. Because we have constructed all Turing morphisms from $\bullet$ under the assumption that every object is a retract of A, we have proved the equivalence of *(i)* and *(ii)* below:

Theorem 6.4 (Recognition of Turing categories) *For a cartesian restriction category $\mathbb{C}$, the following are equivalent:*

(i) $\mathbb{C}$ *is Turing category;*

(ii) *There is an object A of which every object is a retract, and for which there exists a Turing morphism* $A \times A \xrightarrow{\bullet} A$ *,*

(iii) *There is an object A of which every object is a retract which has a coding morphism.*

It remains to explain *(iii)*: we shall say that $\circ : A \times A \longrightarrow A$ is a **coding morphism** in case for every map $f : A \longrightarrow A$ there is a not necessarily unique total point $c[f] : 1 \longrightarrow A$, which we call a **code**, such that

$$\begin{tikzcd}
A \times A \arrow[r, "\circ"] & A \\
A \arrow[u, dotted, "\langle !c[f],1\rangle"] \arrow[ur, "f"'] &
\end{tikzcd}$$

commutes.

We shall prove that, when every object is a retract of A, having a coding morphism implies that one has a Turing morphism and conversely having a Turing structure implies that one has a coding morphism.

Let us start by assuming we have a coding morphism and $A \times A \lhd_{(s,r)} A$ then given $g : A \times A \longrightarrow A$ we have:

$$\begin{tikzcd}
A \times A \arrow[r, "r\times 1"] & A \times A \times A \arrow[r, "1\times s"] & A \times A \arrow[rr, "\circ"] & & A \\
A \times A \times A \arrow[u, "s\times 1"] \arrow[ur, equal] & & & & \\
A \times A \arrow[u, dotted, "\langle !c[g],\pi_0,\pi_1\rangle"] \arrow[uur, dotted, "\langle !c[g],\pi_0,\pi_1\rangle"'] \arrow[rr, "s"'] \arrow[rrr, equal, bend right] & & A \arrow[r, "r"'] \arrow[uu, dotted, "\langle !c[rg],1\rangle"'] & A \times A \arrow[uur, "g"'] &
\end{tikzcd}$$

showing that defining $\bullet := (r \times 1)(1 \times s)\circ$ provides a Turing morphism: it has $\langle !c[g], 1\rangle s$ as an index for g. Notice that this index is total as s is total. Notice, also, that a coding morphism is not necessarily a Turing morphism.

Conversely, if one has Turing structure (which follows from having a Turing morphism) then there is a index p for

$$\begin{tikzcd}
A \times A \arrow[rr, "\tau_{A,A}=\bullet"] & & A \\
1 \times A \arrow[u, dotted, "p\times 1"] \arrow[r, "\pi_1"'] & A \arrow[ur, "g"'] &
\end{tikzcd}$$

showing that $\tau_{A,A}$ also serves as a coding morphism.

6.1 Examples

We now present some examples of Turing categories, ranging from the Turing category for classical recursion theory and models of the lambda calculus to less well-known examples.

6.1.1 The classical recursion category

We first recall some concepts and notation from elementary recursion theory. First of all, one may consider a suitable enumeration $\phi_0, \phi_1, \ldots$ of the partial recursive functions $f : \mathbb{N} \longrightarrow \mathbb{N}$ (of one variable), based on the fact that such a function can be described by finitely many instructions, and that these instructions can be coded into one single number. Alternatively, one may consider a coding for Turing machines, since the Turing machine-computable functions are exactly the partial recursive ones. Similarly, there exist enumerations of partial recursive functions of several variables; $\phi_0^{(n)}, \phi_1^{(n)}, \ldots$ will be an enumeration of the n-ary partial recursive functions. When $f = \phi_e$, one also says that the number e is a **code** for f.

The key properties one can prove about such a family $\phi_m^{(n)}$ (and this is where one needs the enumeration to be "suitable") are:

1. **The Universality Theorem.** This result says, that the functions (for each $n > 0$)

   $$\Phi^{(n)}(e, x_1, \ldots, x_n) = \phi_e(x_1, \ldots, x_n),$$

   called the **universal functions**, are partial recursive.

2. **The Parameter Theorem.** This says that there are primitive recursive functions S_m^n, for each $n, m > 0$, such that

   $$\Phi^{(n+m)}(e, x_1, \ldots, x_m, u_1, \ldots, u_n) = \Phi^{(m)}(S_m^n(e, x_1, \ldots, x_m), u_1, \ldots, u_n).$$

   Moreover, the S_m^n functions may be taken to be injective.

(In some texts, the Universality Theorem is taken to be part of the Enumeration Theorem. Moreover, some texts take the validity of the above two theorems to be the *definition* of a suitable enumeration.) We now fix such an enumeration and use it to define the so-called *Kleene-application* on the natural numbers; this is the function $\mathbb{N} \times \mathbb{N} \xrightarrow{\bullet} \mathbb{N}$, defined by

$$n \bullet x = \phi_n(x).$$

That is, $n \bullet m$ is the result of applying the n-th partial recursive function to input m (and is undefined whenever that computation diverges).

Having this at our disposal, consider the category whose objects are the finite powers of $\mathbb{N}$, and where a morphism $\mathbb{N}^k \longrightarrow \mathbb{N}^m$ is an m-tuple of partial recursive functions of k variables. In case $k = 0$, we simply get m-tuples of elements of $\mathbb{N}$. Denote this category by $\mathsf{Comp}(\mathbb{N})$.

The object $\mathbb{N}$ is a Turing object in this category. First, any (primitive) recursive pairing function $p : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$ together with unpairing functions $p_0, p_1 : \mathbb{N} \longrightarrow \mathbb{N} \times \mathbb{N}$ will exhibit $\mathbb{N} \times \mathbb{N}$ as a retract of $\mathbb{N}$ (in fact, these objects are isomorphic), and similarly for the other objects.

The application on $\mathbb{N}$ is Kleene-application; the Universality Theorem guarantees that this is a partial recursive map, and the weak universal property of $\bullet$ is now an application of the Parameter Theorem.

Note that all of the universal functions are in fact Turing morphisms in this category. In the light of this example, we thus arrive at a justification for the axioms of a Turing category; they are supposed to give a minimal setting in which the Universality Theorem and the Parameter Theorem hold. (The fact that classically the reparametrization functions are primitive recursive and not just merely total recursive does not have a direct analogue in a general Turing category; the fact that they may be taken to be injective does - see Section 6.2.)

6.1.2 Variation

A minor variation on the above category is obtained by considering as objects all recursively enumerable (r.e.) subsets of $\mathbb{N}^k, k \geq 0$, and (tuples of) partial recursive functions between those. Again, $\mathbb{N}$, equipped with the Kleene-application, is a Turing object in this category. (This category is in fact obtained from the first example by splitting the restriction idempotents.)

One could also consider a "smaller" version of the classical recursion category $\mathsf{Comp}(\mathbb{K}_1)$, namely by taking only the objects 1 and $\mathbb{N}$ (and all computable maps between them). Since $\mathbb{N} \cong \mathbb{N} \times \mathbb{N}$ via a computable isomorphism, this is actually a skeleton of $\mathsf{Comp}(\mathbb{K}_1)$. One recovers the old category by splitting the idempotents which correspond to the products.

6.1.3 Reflexive objects

A **reflexive object** in a cartesian closed category is an object A together with embedding-retraction pairs $1 \triangleleft A,\ A \times A \triangleleft A,\ A^A \triangleleft A$. It is well-known that such an object is a model of untyped lambda calculus (see, e.g. [18]). A category where the search for non-trivial reflexive objects often leads to success is that of (directed-)complete partial orders (CPOs) and Scott-continuous maps. Given a reflexive object A in that category, one may consider the smallest cartesian closed subcategory generated by A. This is a Turing category with Turing object A; the Turing morphism is defined by taking an embedding-retraction pair $(m, r) : A^A \triangleleft A$, and by defining

$$\bullet_A :\ A \times A \xrightarrow{r \times A} A^A \times A \xrightarrow{ev} A.$$

This category only has total maps; although one might initially expect a treatment of recursion theory to have partiality (as Di Paola and Heller insisted), this limiting case is not excluded and, indeed, we wish to make a point here that the study of categorical models of the untyped extebsional lambda calculus may be viewed as the study of cartesian closed (total) Turing categories.

Also, we note that, more generally, one could work in a *partial* cartesian closed Turing category (for more on this notion, see, e.g. [10], or [27]), and look for reflexive objects in there; those correspond to models of the untyped *partial* lambda calculus (see [24]). Hence in general cartesian closed Turing categories will correspond to partial lambda algebras.

6.1.4 Classifying categories

So far, our examples were set-based, in the sense that the Turing object was always a set equipped with some extra data (or in the more precise sense that the Turing categories were faithful over Par). Thus, they embody set-based notions of computation. We will now sketch an example which is quite different.

The theory **Partial Combinatory Logic** (PCL, for short) is a unityped partial algebraic theory which has two constants k and s, and a binary application symbol $\bullet$. From these (and variables of course), partial terms are formed; the term formation rules include the usual rules, but there is the added rule of **term restriction**: given terms t, r, one may form a new term $t_{|r}$, to be thought of as t restricted to (the domain of) r. The PCL axioms are $\mathsf{k}xy = x, \mathsf{s}xyz = xz(yz)$ and $x_{|sxy} = x$, where the application is written as an invisible, left associating, operation. The first two axioms are the usual axioms for the combinators k and s and the last axiom is to be read as "$\mathsf{s}xy$ is total". The models of this theory are the partial combinatory algebras which will be discussed in Section 6.4.

Now for ordinary equational theories one can always consider a **term model**, which is built from the syntax of the theory and which is an initial model. In the case of a partial theory such as PCL, this is replaced by a **classifying category**: this is a cartesian restriction category containing a generic model. It follows from the specific properties of PCL (combinatory completeness) that the classifying category for PCL is a Turing category. Because of its initiality, it may be regarded as a generic model of computation.

6.1.5 Turing categories vs. Kleene categories

There is a variation on the definition of Turing category, in which one demands that every morphism f now has an index h *up to inequality*, in the sense that $f \leq \bullet(h \times 1)$ instead of the usual $f = \bullet(h \times 1)$. Such an h is then called a realizer for f. One is tempted to call such structure a Kleene category, since they are more suited for purposes of realizability than for computability.

Many things which are true for Turing categories are already true for Kleene categories. For example, in a Kleene category each object is a retract of the Kleene object, and hence one can normalize the structure. However, there are some important differences: some recursion theoretic concepts, such as m-completeness (see Section 6.3), make sense in Kleene categories, but basic results about them break down.

6.2 Comparing Turing structures

We remarked earlier that a given Turing category may have many different Turing structures. We now investigate how different Turing structures on a Turing category are related, and give some criteria for verifying whether certain data induces Turing structure. Categorically, this may be viewed as a coherence issue; on the other hand it immediately leads to the subject of simulations, and is closely related to the classical concept of recursive invariance.

Let us fix a Turing category $\mathbb{C}$. All we know is that there exists a Turing object, say A. But already from A, we can obtain more Turing objects, namely the powers $A^2, A^3, \ldots$. This follows directly from the following Lemma, which gives a criterion for an object to be a Turing object.

Lemma 6.5 *Let A be a Turing object in $\mathbb{C}$. Then an object B is a Turing object if and only if A is a retract of B.*

PROOF: Clearly, by Lemma 6.3, the condition is necessary. Conversely, let us assume that A is a retract of B, and let us construct universal application morphisms for B. Define $\sigma_{X,Y} : B \times X \longrightarrow Y$ as the composite

$$\sigma_{X,Y} :\ B \times X \xrightarrow{r \times X} A \times X \xrightarrow{\tau_{X,Y}} Y$$

where the map r is a retraction of B onto A, and where $\tau_{X,Y}$ is a universal application for A. Given $f : Z \times X \longrightarrow Y$, we now can consider an index $h : Z \longrightarrow A$ for f relative to A, and compose this index with a splitting of r. $\Box$

Thus, all Turing objects in a Turing category are retracts of each other. From this, it does not, in general, follow that they are isomorphic - this requires (a computable version of) the Cantor-Schröder-Bernstein Theorem.

Even though the above Lemma is, as a coherence result, quite crude, in some cases it still gives us an easy way to determine which objects in a given Turing category are Turing objects. For example, consider the category where the objects are r.e. subsets of powers of $\mathbb{N}$, and where the morphisms are the partial recursive functions between those (example 6.1.2). Of course, $\mathbb{N}$ is a Turing object, as are all powers of $\mathbb{N}$. Moreover, it is known (see [25]) that every infinite r.e. set U can be viewed as the range of an injective total recursive function, and this function has a recursive section, exhibiting $\mathbb{N}$ as a retract of U. Thus, every such infinite U is a Turing object as well. Obviously a finite set cannot be a Turing object, which leads us to the conclusion that the Turing objects are exactly the infinite r.e. sets.

Now let us suppose that we have two Turing objects A and B, together with chosen structure $\tau = \{\tau_{X,Y} | X, Y\}$ on A and $\sigma = \{\sigma_{X,Y} | X, Y\}$ on B. How are τ and σ related? Well, we first pick an embedding-retraction pair $(m, r) : A \triangleleft B$. Then consider the diagram

$$\begin{tikzcd}
& B \times X \arrow[rr, "\sigma_{X,Y}"] & & Y \\
A \times X \arrow[r, "m \times 1_X"'] & B \times X \arrow[u, "h \times X"] \arrow[r, "r \times 1_X"'] & A \times X \arrow[r, "\tau_{X,Y}"'] & Y \arrow[u, "1"']
\end{tikzcd}$$

The arrow h is a σ-index for the composite $(r \times 1_X)\tau_{X,Y}$. Since $mr = 1$, this gives the equation

$$\tau_{X,Y} = (mh \times 1_X)\sigma_{X,Y}.$$

Thus, the map mh relates the two application morphisms; because m was just a choice of embedding and h was just any index, no coherence (varying X and Y) can be expected.

Of course, we may reverse the roles of A and B, and then we will get a comparison morphism in the other direction: given an embedding-retraction pair $(n, t) : B \triangleleft A$, we find j such that

$$\sigma_{X,Y} = (nj \times 1_X)\tau_{X,Y}.$$

We may turn the above observation into a criterion for when a family $\sigma_{X,Y} : B \times X \longrightarrow Y$ defines Turing structure on B.

Lemma 6.6 *Let A be a Turing object with universal maps $\tau_{X,Y}$. A family $\sigma_{X,Y} : B \times X \longrightarrow Y$ constitutes a Turing structure on B if and only if there exist total maps $p_{X,Y} : A \longrightarrow B$ such that $\tau_{X,Y} = (p_{X,Y} \times 1_X)\sigma_{X,Y}$.*

PROOF: One direction was proved in the preceding discussion. For the converse, suppose such a family $p_{X,Y}$ is given. Then, for a morphism $f : Z \times X \longrightarrow Y$, one finds a σ-index by first finding a τ-index $\tilde{f} : X \longrightarrow A$ and then composing with p. $\square$

We can also use this to give a criterion for when a morphism $B \times B \xrightarrow{\bullet} B$ is a Turing morphism.

Lemma 6.7 *Let A be a Turing object and let $\tau_{B,B}$ be a universal application map. Then a morphism $B \times B \xrightarrow{\bullet} B$ is a Turing morphism precisely when there is a total map $p : B \longrightarrow A$ making the following diagram commute:*

$$\begin{tikzcd}
A \times B \arrow[r, "\tau_{B,B}"] & B \\
B \times B \arrow[u, "p \times B"] \arrow[ur, "\bullet"'] &
\end{tikzcd}$$

6.3 Comparing Turing morphisms

We now know that different Turing objects must be retracts of each other, and that between different universal applicative families there must always be comparison morphisms. To complete the description of the coherence among different Turing structures, we would like to give characterizations in terms of the normalized structures. This is certainly possible, but in order to give a complete account one needs both the language and the results about simulations. We therefore present a few simple observations here which can be stated in elementary terms, and illustrate these by connecting them to recursion-theoretic phenomena. As a corollary, we show that the Halting set is m-complete in any Turing category.

Let us first illustrate how it can happen that a given Turing object A has different Turing morphisms. Suppose we have picked a Turing morphism $A \times A \xrightarrow{\bullet} A$ on A; then we can change this into another application $\bullet'$ by considering the following composite:

$$A \times A \xrightarrow{r \times A} (A \times A) \times A \xrightarrow{\pi_0 \times A} A \times A \xrightarrow{\bullet} A.$$

Here, we have chosen an embedding-projection pair $(m, r) : A \times A \lhd A$. Now $\bullet'$ is again a Turing morphism: given $f : X \times A \longrightarrow A$, consider the diagram

$$\begin{tikzcd}
A \times A \arrow[r, "r \times A"] & (A \times A) \times A \arrow[r, "\pi_0 \times A"] & A \times A \arrow[r, "\bullet"] & A \\
X \times A \arrow[u, "h \times A"] \arrow[r, "m_X \times A"'] & A \times A \arrow[u, "\langle g,1 \rangle \times A"] \arrow[ur, "g \times A"'] \arrow[r, "r_X \times A"'] & X \times A \arrow[ur, "f"'] &
\end{tikzcd}$$

where g is an index of $(r_X \times A)f$ relative to $\bullet$, and where we have chosen $(m_X, r_X) : X \lhd A$. Then h may be defined as the composite

$$h = m_X \langle g, 1 \rangle m,$$

since then

$$\begin{aligned}
(h\times A)(r\times A)(\pi_0\times A)\bullet &= ((m_X\langle g,1\rangle m)\times A)(r\times A)(\pi_0\times A)\bullet\\
&= (m_X\times A)(\langle g,1\rangle mr\times A)(\pi_0\times A)\bullet\\
&= (m_X\times A)(\langle g,1\rangle\times A)(\pi_0\times A)\bullet\\
&= (m_X\times A)(g\times A)\bullet\\
&= (m_X\times A)(r_X\times A)f\\
&= f
\end{aligned}$$

Observe that this new index h is in fact a section, and thus in particular a monomorphism. Thus we have in fact shown that every Turing category possesses a Turing structure for which every map has a monic index.

Remark. In classical recursion theory one often exploits the scholium to the Parameter Theorem that the reparametrization functions may be taken to be injective. The construction is also related to what is sometimes called "Padding": the fact that, given a particular code for a function, one can effectively generate infinitely many more codes (see [25]). Now in a general Turing category this result will not hold for every Turing structure (for example if the structure is extensional). However, the above observation shows that there always exists a Turing structure *on the same Turing object* for which padding does work (provided the category is non-trivial of course).

This example tells us more than just about the possibility of creating different Turing morphisms: it tells us that certain properties (in this case, having monic indices) are not properties of the Turing category, but are properties of specific choices of structure.

Before we look at the relation between different Turing morphisms, we introduce a bit of notation which will also prove useful later on. Let us be given a Turing morphism $\bullet$ on A. From this, we may define a family of iterated application morphisms $\bullet^{(n)}: A\times A^n \longrightarrow A$. These are defined inductively: $\bullet^{(1)}=\bullet$, and if we have defined $\bullet^{(n)}$, then let $\bullet^{(n+1)}$ be the composite

$$A\times A\times A^n \xrightarrow{\bullet\times 1} A\times A^n \xrightarrow{\bullet^{(n)}} A.$$

Thus, $\bullet^{(n)}$ is n-fold application, associated to the left. In case no confusion is possible we will be sloppy and omit the parameter n from the notation and simply write $\bullet$ for all of the members of this family.

Lemma 6.8 *Each of the derived morphisms $\bullet^{(n)}$ is a universal application.*

PROOF: This is proved by induction on n, where the case $n=1$ is true by assumption that $\bullet$ is a Turing morphism. If we have proved the statement for $n=k$, then consider $f: Z\times A^{k+1}\longrightarrow A$, and construct an index for f according to

$$\begin{tikzcd}
A\times A\times A^k \arrow[r, "\bullet\times 1"] & A\times A^k \arrow[r, "\bullet^k"] & A\\
X\times A\times A^k \arrow[u, "h\times A^k"] \arrow[r, "="] & X\times A\times A^k \arrow[u, "g\times A^k"] \arrow[ur, "f"'] &
\end{tikzcd}$$

where $g : X \times A \longrightarrow A$ is a $\bullet^{(k)}$-index for f (which exists by induction hypothesis), and $h : X \longrightarrow A$ is a $\bullet$-index for g.

$\square$

We have started the inductive definition of $\bullet^{(n)}$ at $n = 1$, and one may ask what $\bullet^{(0)}$ should be! Given that the morphisms $\bullet^{(n)}$ allow us to view elements of A as n-ary morphisms, a natural choice would be to let $\bullet^{(0)}$ be the identity, for then we regard an element of A as a 0-ary function, that is, a constant. However, there are compelling reasons for making another choice: we define $\bullet^{(0)}$ to be the composite

$$A \xrightarrow{\Delta} A \times A \xrightarrow{\bullet} A.$$

A first argument in favour of this choice is that many of the interesting properties of the family $\bullet^{(n)}$ are shared by the above map, but not by the identity. (More reasons will be provided in Section 6.4.) For example, it is also a universal application: given $f : X \longrightarrow A$, let $d : X \longrightarrow A$ be a $\bullet$-index for $f\pi_X : X \times A \longrightarrow A$. Then d is also a $\bullet^{(0)}$-index for f. See the following diagram.

$$\begin{tikzcd}
A \arrow[r, "\Delta"] & A \times A \arrow[r, "\bullet"] & A \\
X \arrow[u, "d"] \arrow[r, "\langle 1,d \rangle"'] & X \times A \arrow[u, "d\times A"] \arrow[r, "\pi_X"'] & X \arrow[u, "f"']
\end{tikzcd}$$

We digress for a moment to note that we have actually proved a result from recursion theory here, namely the completeness of the Halting set. Let us briefly explain this: if two domains $e \in \mathcal{O}(X), e' \in \mathcal{O}(Y)$ are given, we say that e (many-one) reduces to e', written $e \leq_m e'$, when there is a total map $f : X \longrightarrow Y$ such that $e = f^*(e')$, i.e. e is the pullback of e' along f. A domain is said to be complete if every other domain reduces to it[6].

Now for any Turing structure $\tau = \{\tau_{X,Y}\}$ on A, the domain of $\tau_{1,A}$ is indeed complete: given a domain $e \in \mathcal{O}(X)$, consider

$$\begin{tikzcd}
A \arrow[r, "\tau_{1,A}"] & A \\
X \arrow[u, "h"] \arrow[r, "e"] & X \arrow[u, "m"']
\end{tikzcd}$$

where $m : X \longrightarrow A$ is a choice of embedding and h is an index for em. Because $me = \tau_{1,A}h$, the domains of $h\tau_{1,A}$ and of em are equal. Thus, letting K (so $K = \overline{\bullet^{(0)}}$, we get

$$e = \overline{em} = \overline{h\tau_{1,A}} = h^*(K),$$

by totality of m, which shows that $e \leq_m K$.

Now we may apply this to the domain of the morphism $\bullet^{(0)}$; this gives:

Corollary 6.9 (Completeness of the halting set) *The domain K is complete.*

[6] A more familiar categorical formulation of a complete predicate would be: e is complete if and only if it is a weak generic object in the appropriate subobject fibration (over the category of total maps).

Similar reasoning shows the completeness of other domains. Indeed, if we take any total point $a : 1 \longrightarrow A$ of A, then we may consider the composite $A \xrightarrow{\langle 1,a\rangle} A \times A \xrightarrow{\bullet} A$. The domain of this map will then be complete, for if we are given a domain e on X, consider

$$\begin{tikzcd}
A \arrow[r, "\langle 1,a\rangle"] & A \times A \arrow[r, "\bullet"] & A \\
X \arrow[u, "h"'] \arrow[r, "\langle 1,a\rangle"'] & X \times A \arrow[u, "h\times A"] \arrow[r, "e\pi_X"'] & X \arrow[u, "m"']
\end{tikzcd}$$

where h is an index for $\pi_X em$. Since h is total and the bottom composite is e, this shows that e is a pullback of the domain of the top composite.

One can strengthen this to encompass so called 1-completeness. Here the map along which the reduction is performed is not only required to be total but also it must be monic. To achieve this one must simply modifying the Turing morphism so as to ensure that indices are monic.

We return to the problem of relating different Turing morphisms. Suppose we have two Turing objects A, B, with specified Turing morphisms $\bullet_A = \tau_{A,A}$ and $\bullet_B = \sigma_{B,B}$. Choose a retraction $(m, r) : A \lhd B$. First, consider the diagram

$$\begin{tikzcd}
 & B \times B \arrow[rr, "\bullet_B"] & & B \\
A \times A \arrow[r, "m\times m"'] & B \times B \arrow[u, "h\times B"] \arrow[r, "r\times r"'] & A \times A \arrow[r, "\bullet_A"'] & A \arrow[u, "m"']
\end{tikzcd}$$

This gives an equation

$$(mh \times m)\bullet_B = \bullet_A m$$

relating the two Turing morphisms. Written in the more palatable internal language of the category (using infix notation for the applications), this reads

$$h(m(x)) \bullet_B y = m(x \bullet_A y).$$

Thus, m preserves the application modulo the "twist" h.

Remark. In recursion theory, one considers two acceptable systems of indices (enumerations) $\phi_0, \phi_1, \ldots$ and $\psi_0, \psi_1, \ldots$, or the partial recursive functions, giving rise to Kleene applications $\bullet_\phi$ and $\bullet_\psi$, respectively. Then one asks how these are related. One answer is that there is a total recursive function h for which

$$h(x) \bullet_\psi y = x \bullet_\phi y.$$

(The embedding m which figures in our equation is the identity here.) In addition, one can show that h may be taken to be an isomorphism. It is the latter aspect which is specific to classical recursion theory and does not generalize to our setting. Also, it has been shown that there must exist a total recursive g such that $g(x) \bullet g(y) = g(x \bullet y)$ (see [25]); again, this exploits specific properties of the natural numbers and does not generalize.

We can reformulate the relation between $\bullet_A$ and $\bullet_B$ using the iterated application $\bullet_B^{(2)}$. Indeed, consider

$$\begin{tikzcd}
 & B \times B \times B \arrow[rr, "\bullet_B^{(2)}"] & & B \\
A \times A \arrow[r, "m\times m"'] & B \times B \arrow[u, "u\times B\times B"] \arrow[r, "r\times r"'] & A \times A \arrow[r, "\bullet_A"'] & A \arrow[u, "m"']
\end{tikzcd}$$

where $u : 1 \longrightarrow B$ is now a code. Written in the internal language again, this means that

$$(u \bullet_B m(x)) \bullet_B m(y) = m(x \bullet_A y).$$

We may summarize this in the following Lemma:

Lemma 6.10 *If $(A, \bullet_A)$ and $(B, \bullet_B)$ are both Turing objects, then there are codes $u : 1 \longrightarrow A$ and $v : 1 \longrightarrow B$, and morphisms $m : A \longrightarrow B$ and $n : B \longrightarrow A$ such that the following diagrams commute:*

$$\begin{tikzcd}
A \times A \arrow[r, "\bullet_A"] \arrow[d, "\langle u,m,m\rangle"'] & A \arrow[d, "m"] \\
B \times B \times B \arrow[r, "\bullet_B"'] & B
\end{tikzcd}
\qquad
\begin{tikzcd}
B \times B \arrow[r, "\bullet_B"] \arrow[d, "\langle v,n,n\rangle"'] & B \arrow[d, "n"] \\
A \times A \times A \arrow[r, "\bullet_A"'] & A
\end{tikzcd}$$

6.4 Partial Combinatory Algebras

From the few examples in the previous section, as well as the recognition theorem, we saw that Turing categories are, in a certain sense, generated from a single object which comes equipped with a binary application map. This binary application makes a Turing object an internal partial combinatory algebra in its ambient Turing category. This section introduce partial combinatory algebras and describes how they can be used to generate Turing categories.

Throughout this section, we will work in an arbitrary cartesian restriction category $\mathbb{C}$.

Definition 6.11 (Applicative system) *An* **applicative system** $\mathbb{A} = (A, \bullet)$ *in* $\mathbb{C}$ *consists of an object A together with a morphism* $A \times A \xrightarrow{\bullet} A$ *, called* **application**.

There are no axioms to be satisfied, and therefore this notion is rather wide. In order to select a class of interesting applicative systems, we focus on the morphisms which can be "computed" by an applicative system. Before we can define what we mean by this, we recall from 6.3 the family of iterated application morphisms $\bullet^n : A \times A^n \longrightarrow A$, derived from $\bullet$, including the special case for $n = 0$, where $\bullet^{(0)}$ is the composite $\bullet\Delta$.

We now define:

Definition 6.12 (Computable morphism, absolute version) *Let $\mathbb{A} = (A, \bullet)$ be an applicative system.*

(i) *For $n \geq 0$, a morphism $f : A^n \longrightarrow A$ is said to be* $\mathbb{A}$-computable *(or* computable *whenever $\mathbb{A}$ is clear from the context) when there exists a total point $p : 1 \longrightarrow A$ such that*

$$\begin{tikzcd}
A \times A^n \arrow[r, "\bullet"] & A \\
A^n \arrow[u, "p\times 1"] \arrow[ur, "f"']
\end{tikzcd}$$

is commutative. (We suppress the isomorphism $A^n \cong 1 \times A^n$). Moreover, if $n > 1$, it is required that the morphism

$$1 \times A^{n-1} \xrightarrow{p\times 1} A \times A^{n-1} \xrightarrow{\bullet^{n-1}} A$$

is total.

In this situation, we call p a **code** *for the morphism f.*

(ii) *More generally, a morphism $A^n \longrightarrow A^m$ is computable when all components are; finally, we say that a morphism $A^n \longrightarrow 1$ is computable if its domain is (as a map $A^n \longrightarrow A^n$).*

The second clause in the definition which requires a to be total in its first $n-1$ arguments can be justified in a few different ways; see the remarks at the end of the section.

Remark. One can, in the definition of a computable morphism, replace the condition that the relevant diagram is commutative on the nose, by the relaxed condition that it commutes up to inequality (i.e. that $f \leq \bullet(p \times 1)$). This is typically referred to as the *realizability* of a morphism f. Thus, computability is a stricter notion which requires the domain of f to be captured by the code p.

In order to single out interesting applicative structures we may ask for certain useful morphisms to be computable. For example, we may want the identity morphism on A, or more generally the projection morphisms to be computable; one could additionally ask that the computable maps are closed under composition. Ultimately, one wants to extract a category from $\mathbb{C}$, where the objects are the finite powers of A and the morphisms are the computable maps. This category, which will be denoted by $\mathsf{comp}(\mathbb{A})$, can be constructed as follows:

Objects: Formal finite powers of A. We may identify these with the natural numbers $\underline{0}, \underline{1}, \ldots$.

Morphisms: A morphism $\underline{n} \longrightarrow \underline{m}$ is an $\mathbb{A}$-computable morphism $A^n \longrightarrow A^m$ in the base category $\mathbb{C}$.

One could directly impose the requirement on $\mathbb{A}$ that this forms a category; however, we will take the more common approach of discussing combinatory completeness first and then showing that these two things are equivalent.

From $\mathbb{A} = (A, \bullet)$, we can build a subcategory of $\mathbb{C}$, called the $\mathbb{A}$-*polynomial category*, or simply the *polynomial category*; this is defined to be the smallest cartesian restriction subcategory of $\mathbb{C}$ on the objects $1, A, A^2, \ldots$ which contains all total points of A as well as the application map. Morphisms in this subcategory are called $\mathbb{A}$-*polynomial morphisms*; think of such a morphism $f : A^n \longrightarrow A$ as a polynomial in n variables with coefficients in A.) The key definition is now:

Definition 6.13 (Combinatory Completeness) *An applicative system is* **combinatory complete** *when every polynomial morphism is computable.*

It is a classic result that combinatory completeness already follows from two instances (see, for example, [4]). Define the map k_0 to be the first projection $\pi_0 : A \times A \longrightarrow A$, and define the map s_0 to be the composite

$$A \times A \times A \xrightarrow{\langle \pi_{13}, \pi_{23} \rangle} (A \times A) \times (A \times A) \xrightarrow{\bullet \times \bullet} A \times A \xrightarrow{\bullet} A.$$

Theorem 6.14 *For an applicative system $\mathbb{A} = (A, \bullet)$, the following are equivalent:*

1. *$\mathbb{A}$ is combinatory complete*
2. *the polynomial morphisms k_0, s_0 are $\mathbb{A}$-computable.*

A choice of codes for k_0, s_0 is usually denoted by k and s, respectively. It is important to note that these codes of combinators (as they are usually called) are by no means unique.

We now define a**Partial Combinatory Algebra** (PCA for short) to be a combinatory complete applicative system. Often, a PCA is taken to be an applicative system together with chosen combinators k and s; here we shall try to avoid making these choices as the constructions and morphisms we shall be considering will not rely on having made such a choice.

Before we turn to examples of PCAs, we give the promised result (proved in the total case by Longo and Moggi) that combinatory completeness is precisely what is needed to guarantee the computable maps do form a category.

Theorem 6.15 (Categorical characterization of completeness) *For an applicative system $\mathbb{A} = (A, \bullet)$ in a category $\mathbb{C}$, the following are equivalent:*

(i) *$\mathbb{A}$ is combinatory complete*

(ii) *The $\mathbb{A}$-computable morphisms form a cartesian restriction category over $\mathbb{C}$.*

Proof: We sketch the proof, which is a straightforward extension of the total case proved in [20].

Assume first that A is combinatory complete. By definition, this means that every polynomial map is computable. In particular, the application map is computable, as is every total element. Now if $f : A^n \longrightarrow A$ is computable, we may express f as a polynomial, since it has a code a such that $f = \bullet^{(n)}(a \times 1)$, expresses f as a composite of polynomials. Thus the polynomial maps and the computable maps coincide, and hence the computable maps form a cartesian restriction category.

For the converse, assume that computable maps form a category; we first note that this implies that the identity map $A \longrightarrow A$ has a code, say i. Then i is also a code for the application map, and hence the application map is computable. Now the maps k_0, s_0 defined above are built from the application and the cartesian structure, so are in the category of computable maps. By the previous theorem this implies combinatory completeness. $\square$

In this proof, one aspect has been hidden: given a morphism f, how do we see that $\overline{f}$ is again computable? Consider the diagram

$$\begin{tikzcd}
 & A \times A \times A \arrow[r, "\bullet"] & A \\
A \arrow[r, "\langle 1, f\rangle"'] & A \times A \arrow[u, "\mathsf{k} \times A \times A"] \arrow[ur, "\pi_0"'] &
\end{tikzcd}$$

By definition of k, the triangle commutes; the composite $\pi_0\langle 1, f\rangle$ equals the domain $\overline{f}$. Since all maps are computable, so is $\overline{f}$.

Of course, categorically what we are using here is that in any cartesian restriction category, the restriction structure is definable in terms of the products, namely as $\overline{f} = \langle 1, f\rangle\pi_0$

We pause for a moment to contemplate what the elements of a PCA are. We defined an element $p : 1 \longrightarrow A$ to be computable if it has a $\bullet^{(0)}$-index; an equivalent formulation would be: an element $p : 1 \longrightarrow A$ is computable if the composite $A \xrightarrow{!} 1 \xrightarrow{p} A$ is computable. The first formulation has the advantage that it explains the relation between the collection of total and of partial elements of a PCA. We state this in a lemma:

Lemma 6.16 *Let $\mathbb{A} = (A, \bullet)$ be a PCA in $\mathbb{C}$. Then every partial element $p : 1 \longrightarrow A$ factors through $\bullet^{(0)}$ via a total element.*

PROOF: Of course, the desired total element is simply a $\bullet^{(0)}$-index for p. $\square$

In set-theoretic notation, this means that every partial element p may be written $p = u \bullet u$, for some total u.

The main point is, that this reveals more of the role of the family $\bullet^{(n)}$ and why $\bullet^{(0)}$ fits into it: it provides a normal form for morphisms in $\mathsf{comp}(\mathbb{A})$. Indeed, every morphism may be given by a total element followed by one of the $\bullet^{(n)}$.

So far, we have shown that a PCA $\mathbb{A} = (A, \bullet)$ in $\mathbb{C}$ gives a faithful category over $\mathbb{C}$ on the computable maps. Given the preceding discussion of combinatory completeness, it should not come as a surprise that $\mathsf{comp}(\mathbb{A})$ is in fact a Turing category with Turing morphism $\bullet$, regarded as a morphism $\underline{2} \longrightarrow \underline{1}$. The only thing left to verify is that every object $\underline{n}$ is a retract of $\underline{1}$. We show this for $\underline{2}$, and the others will follow. The point is that combinatory completeness entails that one can construct a computable pairing map $p : A \times A \longrightarrow A$, together with computable unpairings $p_0, p_1 : A \longrightarrow A$ (see, e.g. [19]). Then $A \times A$ is a retract of A in $\mathbb{C}$ via the pair of computable maps p and $\langle p_0, p_1 \rangle$). Thus we have shown:

Theorem 6.17 *For any PCA $\mathbb{A} = (A, \bullet)$ in $\mathbb{C}$, the category $\mathsf{comp}(\mathbb{A}, \mathbb{C})$ is a Turing category.*

Although it is probably clear from the construction, it is important to remark that the Turing category $\mathsf{comp}(\mathbb{A}, \mathbb{C})$ is faithful over $\mathbb{C}$, via the functor $\underline{n} \mapsto A^n$. In fact, we have found a way of generating a Turing categories over a fixed base category by considering PCAs in the base category. This is classically how the notion of computability arises.

Remark. One can of course consider the category $\mathbb{R}(\mathbb{A})$ of **realizable maps** (those maps f for which there exists a realizer p such that $h \leq (p \times 1)\bullet$. Everything above generalizes immediately, and we find that every PCA gives rise to a Kleene category (as discussed in Example 6.1.5). Moreover, $\mathbb{R}(\mathbb{A})$ is obtained from $\mathsf{comp}(\mathbb{A})$ by formally adding all domains from $\mathbb{C}$.

We end this section with an elementary observation which allows us to transport PCAs along cartesian functors. If $\mathbb{A} = (A, \bullet)$ is an applicative system in $\mathbb{C}$ and $F : \mathbb{C} \longrightarrow \mathbb{D}$ is a cartesian restriction functor, then $F\mathbb{A} = (FA, F\bullet)$ is an applicative system in $\mathbb{D}$. Moreover, if we can make a choice for k, s in $\mathbb{A}$, then $F\mathsf{k}, F\mathsf{s}$ will turn $F\mathbb{A}$ into a PCA. Thus cartesian restriction functors preserve PCAs as long as in the external world we are able to make a choice of combinators.

6.5 Examples of PCAs

Given the connection between Turing categories and PCAs, it is now clear that each example of a Turing object in a Turing category is an example of a PCA (in that Turing category). Nevertheless, we give some more examples to provide a more complete the picture.

Kleene's first model. The prime example of a PCA is Kleene's model $\mathbb{K}_1$; its underlying set is $\mathbb{N}$, and application is Kleene-application as defined in the previous section. Combinators k and s are found using the Parameter Theorem (see [5]) for a precise account). Of course, $\mathsf{comp}(\mathbb{K}_1)$ is the classical recursion category (first example in the previous section).

Oracles. One may generalize this example by considering computability relative to an oracle A; the resulting model is denoted $\mathbb{K}_1^A$. It has the same underlying set $\mathbb{N}$, but application is given by $n \bullet^A m = \phi_{n,A}(m)$, where $\phi_{0,A}, \phi_{1,A}, \ldots$ is a suitable enumeration of the class of unary partial functions which are computable with help of the oracle A.

Basic recursive function theory. Generalizing the situation even further, there is the notion of a so-called *Basic Recursive Function Theory*, BRFT for short (see [29, 30]). Essentially, a BRFT consists of an infinite set B, a collection of partial functions $\mathcal{F}$ on B, and, for each n, an enumeration function $\Phi^{(n)}$, which plays the role of the universal function of n variables. It is required that $\mathcal{F}$ contains the identity, constant functions and the parametrization functions, as well as decision functions $d(x, y, u, v)$, which are supposed to satisfy

$$d(x, y, u, v) = \begin{cases} u & \text{if } x = y \\ v & \text{otherwise.} \end{cases}$$

Now we can use the enumeration function $\Phi^{(1)}$ to define an application on B, and use the parametrization functions to define the combinators k, s. Thus, each BRFT can be made into a PCA. It can be shown that if the underlying set of a BRFT is the natural numbers, then the class $\mathcal{F}$ must contain all partial recursive functions. Thus, this example does generalize some important aspects of elementary recursion theory over $\mathbb{N}$; indeed it has been shown that most basic aspects of recursion theory can be proved in this setting.

Effective applicative structures. When, in the definition of a BRFT, one omits the requirements that the decision functions be present, one arrives at what Asperti and Longo ([1]) call an *effective applicative structure*. It is easily seen that this notion coincides with that of a combinatory complete applicative structure which is non-trivial (because we insist on the underlying set being infinite). Note that the requirement that the identity function and the constant functions be present is already ensured by combinatory completeness. Thus, after a series of generalizations from the recursive structure on the natural numbers, we have come back to PCAs (in the category of sets, that is).

Syntactical models. Combinatory logic (CL) is an equational theory whose models are total combinatory algebras (i.e. applicative systems whose application map is total) which come equipped with a specific choice of combinators k, s. Both the open and closed term models of CL are therefore examples of (total) PCAs. It is well-known that CL admits a rewriting presentation; therefore it makes sense to look at strongly normalizing terms. The set of strongly normalizing terms modulo provable equality is a PCA; for details, see [5].

Partial combinatory logic (PCL) is a partial equational theory whose models are PCAs (in which one has chosen combinators). As indicated in Example 6.1.4, it gives rise to a classifying category $\mathbb{C}_{PCL}$ containing a generic model; if we are given a PCA in an arbitrary cartesian restriction category $\mathbb{C}$ then there is an essentially unique cartesian restriction functor which sends the generic PCA to the given model. In particular, we can apply the global sections functor to the generic PCA, and the result is (up to isomorphism) the PCA of strongly normalizing CL-terms.

Models of lambda calculus. Any model of the (partial) lambda calculus is a PCA; in particular, a reflexive object in a (partial) cartesian closed category is a PCA.

7 Recursion categories

The standard settings for computability (also known as recursion theory) has much more structure than that of a mere Turing category. Unlike Turing categories computability settings can never be total. Furthermore, they always support the discussion of *recursive* properties – that is those properties which can be decided by a computable function. A crucial component of this is identifying certain key undecidable properties.

The aim of this section is to provide a basic framework for computability and to establish some of the basic results of computability theory. To prosecute this we introduce *recursion categories*: these are very closely related (but not quite the same) as the setting originally proposed by Di Paola and Heller. Below we shall show how the standard results of computability theory can be developed in any recursion category.

Of particular interest is the manner in which the results on undecidability are expressed. Their expression is slightly different from the usual statement as we shall prove that if certain predicates are decidable then this leads to the collapse of the recursion category to a "trivial" recursion category which has essentially only one object and one map. Thus these results state that in any *non-trivial* recursion category a certain predicate cannot be decidable.

It is also worth being absolutely upfront about the fact that one cannot hope to get all the results of recursion theory in a general recursion category: the standard setting really does have many more properties. Part of our purpose here is to make more explicit the relationship between the special properties of the classical setting and the results in these settings. What is particularly significant is that many of the fundamental results, as we have already seen, are true in much greater generality.

7.1 Recursiveness

A **recursion category** is a discrete Turing category with (finite) joins. Explicity this means it is a cartesian restriction category, which possesses a Turing structure, which also has joins and meets.

The classical setting for computability (recursion) theory is, as we shall shortly see, an example of a recursion category which also possesses a number of additional features. One important way in which recursion categories differ from the classical setting concerns the status of the equality predicate: in a recursion category equality on the Turing object is only assumed to be "semi-decidable", an "enumerable set", or, more abstractly given by a restriction idempotent. Thus, these settings are still much broader than the classical setting (in which equality is definitely decidable).

Given a recursion category it is clear that we can split the restriction idempotents (which means, of course, all idempotents split) to obtain a new recursion category. It simplifies arguments greatly to work in a split recursion category and we will, therefore, feel free to assume that idempotents split in the remainder of this section. Of course, as the original (non-split) recursion category embeds in the split one in a full and faithful manner all the results we shall prove are preserved and reflected by splitting.

Coproducts are crucial for building general data types, a remarkable fact about recursive categories is that they (once split) already have coproducts:

Theorem 7.1 *Split recursion categories have coproducts and are distributive restriction categories.*

PROOF: The idea of the proof is as follows: if we had a boolean object so that $1 \xrightarrow{\text{true}} \mathsf{Bool} \xleftarrow{\text{false}} 1$ is a coproduct then by taking the product with the Turing object, T we would get a coproduct

$$1 \times T \xrightarrow{\text{true} \times 1_T} \mathsf{Bool} \times T \xleftarrow{\text{false} \times 1_T} 1 \times T$$

and so be able to take coproducts of the Turing object. However, as every object occurs as a retract of the Turing object it now follows easily that there are now coproducts for all objects.

Recall that in the join restriction setting coproducts are absolute for join preserving functors of which $\_ \times T$ is one. This fact also guarantees that he coproducts are distributive.

Therefore the key thing to show is that we do have a Boolean object in these categories. First we remark that that every total element $a : 1 \longrightarrow A$ is also a restriction monic as clearly $!a$ is an idempotent and so a splits the idempotent $(!a) \cap 1_A$. Our objective is to find two elements which intersect at the zero object (also, of course, the initial object in the total category). Consider the elements i and z:

$$\begin{tikzcd}
1 \times A \arrow[r, "\pi_1"] \arrow[d, "\mathsf{i} \times 1_A"'] & A \\
A \times A \arrow[ur, "\bullet"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
1 \times A \arrow[r, "0"] \arrow[d, "\mathsf{z} \times 1_A"'] & A \\
A \times A \arrow[ur, "\bullet"'] &
\end{tikzcd}$$

The pullback of these subobjects exists, call it P:

$$\begin{tikzcd}
P \arrow[r, "p"] \arrow[d, "q"'] & 1 \arrow[d, "\mathsf{i}"] \\
1 \arrow[r, "\mathsf{z}"'] & A
\end{tikzcd}$$

we wish to show that $P = 0$. Consider

$$\begin{tikzcd}
P \times 1 \arrow[r, "p \times 1"] \arrow[d, "q \times 1"'] & 1 \times 1 \arrow[d, "\mathsf{i} \times \mathsf{i}"] \arrow[ddr, bend left, "\pi_1 \mathsf{i}"] & \\
1 \times 1 \arrow[r, "\mathsf{z} \times \mathsf{i}"'] \arrow[drr, bend right, "0"'] & A \times A \arrow[dr, "\bullet"] & \\
& & A
\end{tikzcd}$$

then $(p \times 1)\pi_1 \mathsf{i} = 0$ as $\pi_1 \mathsf{i}$ is monic $p \times 1 = 0$ but p is a restriction monic which forces $P = 0$. $\square$

Having coproducts allows us to talk about "recursive" predicates. Of course, the predicates of a object A in restriction category are its open sets $\mathcal{O}(A)$ or more simply the set of restriction idempotents on A. In a distributive restriction category to say that a predicate $p = \overline{p}$ on an object A is **recursive** is to say that p has a complement, in the sense that there is a $p' = \overline{p'}$ with $pp' = 0$ and $p \vee p' = 1_A$. This, in fact, means that $p \xrightarrow{p} A \xleftarrow{p'} p'$ is a coproduct and so there is a unique map χ_p

$$\begin{tikzcd}
p \arrow[r, "p"] \arrow[d, "!"'] & A \arrow[d, dotted, "\chi_p"] & p' \arrow[l, "p'"'] \arrow[d, "!"] \\
1 \arrow[r, "\mathsf{true}"'] & \mathsf{bool} & 1 \arrow[l, "\mathsf{false}"]
\end{tikzcd}$$

which makes this diagram of total maps a pullback in each square (this property is called "extensivity"). The map χ_p is the **decider** of the recursive property p.

Saying that a property is **undecidable** is asserting that such a map χ_p does not exist. Shortly we shall start investigating these properties. However, before that let us briefly discuss how recursion categories can arise.

7.2 Finite interleaving

In any restriction category with meets and joins a map h is an **interleaving** of a pair of parallel arrows $f, g : A \longrightarrow B$ if there is an arrow $h : A \longrightarrow B$ such that $\overline{f} \leq \overline{h}$, $\overline{g} \leq \overline{h}$, and $h = (h \cap f) \vee (h \cap g)$. Intuitively we might think of h as a computation arrived at by interleaving the computation of f with g, thus, the result is a computation which always behaves either like f or g, although where they overlap could behave like either (classically this depends on which takes less time to halt). In the special case where f and g are compatible this means that we should expect the interleaving to be the join:

Lemma 7.2 *In a restriction category with meets and joins, an interleaving of compatible maps is the join of those maps.*

PROOF: Suppose f and g are compatible and h is an interleaving of f and g then certainly $h \leq f \vee g$ as $(h \cap f) \leq f \leq f \vee g$ and $(h \cap g) \leq g \leq f \vee g$. but also $f \leq h$ and $g \leq h$, so that $h = f \vee g$, as

$$\begin{aligned}
\overline{f}\, h &= \overline{f}\,((h \cap f) \vee (h \cap g)) = (\overline{f}\,(h \cap f)) \vee (\overline{f}\,(h \cap g)) \\
&= ((\overline{f}\, h) \cap (\overline{f}\, f)) \vee ((\overline{f}\, h) \cap (\overline{f}\, g)) = (f \cap f) \vee (f \cap \overline{g}\, f) \\
&= f \vee (\overline{g}\, f) = f
\end{aligned}$$

□

This means interleaving gives a way of *constructing* the join, and thus when interleaving is provided by a construction this allows an algebraic description for forming the join. A recursion category in which all parallel pairs of maps have an interleaving will, naturally, be called a recursion category with **(finite) interleaving** . We shall not require that this interleaving need be uniformly given. Thus, the interleaving operation itself need not be given be an operation on codes within the Turing category. In the classical setting, of course, interleaving *is* uniformly present: that is code for the interleaved computation can be constructed in a computable manner from the original codes.

A partial combinatory algebra, $(A, \bullet, \mathsf{k}, \mathsf{s})$, in a cartesian restriction category is said to be **interleaving** if A is a discrete object and comes equipped with two codes and an operation. The two codes are total elements: $\mathsf{z} : 1 \longrightarrow A$ the "zero" code and $\mathsf{e} : 1 \longrightarrow A$ the "equality" code. The operation is a total map $\ominus : A \times A \longrightarrow A$ called the "interleaving" such that:

[IntL.1] The zero is the code for the zero map: $(\mathsf{z} \times 1)\bullet = 0$;

[IntL.2] The equality code is the code for the partial inverse to the diagonal map: $(\mathsf{e} \times 1 \times 1)(\bullet \times 1)\bullet = \Delta^{(-1)}$;

[IntL.3] The operation $\ominus : A \times A \longrightarrow A$, is such that $(\ominus \times 1_A)\bullet : A \times A \times A \longrightarrow A$ is an interleaving of $(\pi_0 \times 1_A)\bullet$ and $(\pi_1 \times 1_A)\bullet$.

It is important to note that we have not required that the construction of the interleaving code should itself be something which is A-computable (although this may be the case and even desirable). Thus, we allow for the possibility that the code for interleaving of two codes can be constructed in the more powerful external fashion (or non-uniform) but not in a computable manner.

The main result is:

Theorem 7.3 *In any cartesian restriction category with joins whenever A is an interleaving partial combinarory algebra, $\mathsf{Comp}(A)$ is a recursion category.*

PROOF: Given any cartesian restriction category it is easy to see that the full subcategory of discrete objects is a discrete subcategory. Thus, the construction can be done within this subcategory. However, now the proof is immediate as $\mathsf{Comp}(A)$ by design is discrete and because of the interleaving and the lemma just proven inherits the joins of the ambient setting. □

In the standard model of computation interleaving of computations is a standard technique and can be done uniformly. This certainly shows that the standard model of recursion theory is indeed a recursion category.

7.3 Undecidability

One of the first and most basic results of recursion theory is to prove that the halting set is undecidable. Recall that we already know from general Turing considerations that it is complete, however, without joins (and coproducts) it was not even a well-posed question to ask about undecidability.

To express undecidability categorically we must first surmount a minor embarrassment. There are perfectly good recursion categories in which the halting set *is* decidable: namely any recursion category which is equivalent to the category with one object and one map! We call such a recursion category **trivial** and we are of course interested in recursion categories which are non-trivial. An important observation is:

Lemma 7.4 *Any cartesian restriction category with joins, in which there is a zero map, $0 : 1 \longrightarrow A$ from the final object to any object, which is total, is trivial.*

PROOF: This immediately forces the identity map on the final object to be the zero map making $1 \cong 0$. But now $A \cong A \times 1 \cong A \times 0 \cong 0$ ($A \times \_$ always preserves coproducts including the empty coproduct). Thus, the category is trivial. □

Now we can state more precisely, for a recursion category, what it means for the halting set to be undecidable:

Theorem 7.5 *In a recursion category, $\mathbb{X}$, the halting predicate K has a complement if and only if $\mathbb{X}$ is trivial.*

PROOF: Let $K' = \overline{K'}$ be an idempotent with $K'K = 0$. Set v to be a code for K':

$$\begin{tikzcd} A \arrow[r, "K'"] \arrow[d, dotted, "v\times 1"'] & A \\ A \times A \arrow[ur, "\bullet"'] & \end{tikzcd}$$

this means that v is a total element. However, we now have

$$vK = v\bullet^{(0)} = v\Delta\bullet = \langle v, v\rangle\bullet = vK'$$

so that $vK' = vK'K' = vKK' = 0 = vK'K = vKK' = vK$ so that if $K \sqcup K' = 1$ then

$$0 = \overline{0} = \overline{(vK) \sqcup (vK')} = \overline{v(K \sqcup K')} = \overline{v} = 1$$

But this collapses the final object to zero and make the whole category trivial. $\square$

Note that we have shown that K is "creative" that is given $e = \overline{e}$ with $Ke = 0$ there is a point p_e with $pK = 0 = pe$. Clearly, therfore, any creative idempotent in a recursion category has a complement only when the category is trivial.

This has a number of consequences on the logic of predicates in a recursion category. Clearly, whatever the logic is, it is certainly not boolean - that is we cannot have each $\mathcal{O}(A)$ a boolean algebra without collapsing the category. (A restriction category in which this is the case is called a *classical* restriction category.) However, worse, the logic cannot even be intuitionistic, that is $\mathcal{O}(A)$ cannot be a Heyting algebra because given any creative predicate p (such as the halting predicate) we can always find an element between p and any q with $pq = 0$. However, a Heyting negation $p \Rightarrow 0$ cannot have any non-zero predicate between it and p. This shows:

Corollary 7.6 *Every recursion category whose Turing object T has $\mathcal{O}(T)$ a Heyting algebra is trivial.*

So for example there is no non-trivial recursion category which has arbitrary joins.

Corollary 7.7 *Every recursion category with arbitrary joins is trivial.*

This reminds us on the one hand that computability is quite a delicate business – too much structure easily collapses it – and on the other hand that many of the tools used in traditional logic simply do not apply! This latter is one reason why the subject should be of special interest to the categorically minded!

Another classical result concerns recursive inseparability: two disjoint restriction idempotents $e_0, e_1 : X \longrightarrow X$ are **recursively inseparable** if there is no complemented idempotent e such that $e_0 \leq e$ and $e_1 e = 0$. A classical result of recursion theory is that there are inseparable predicates and it is true in every non-trivial recursion category.

Theorem 7.8 (F. Lengyel) *Every non-trivial recursion category has recursively inseparable restriction idempotents.*

PROOF: The above assures us that we may find two total points $p_0, p_1 : 1 \longrightarrow T$ (where T is the Turing object) such that, viewed as predicates $p_0 \cap 1, p_1 \cap 1 : T \longrightarrow T$, they are disjoint (so that $(p_0 \cap 1), (p_1 \cap 1) = 0)$. Any pair of such points will do.

Set $k_i = \overline{\Delta \bullet (p_i \cap 1)} : T \longrightarrow T$. The predicate k_i then is those codes which when applied to themselves evaluate to p_i. Note that k_0 and k_1 are clearly disjoint as

$$k_0 k_1 = \overline{\overline{\Delta \bullet (p_0 \cap 1)}\, \Delta \bullet (p_1 \cap 1)} = \overline{\Delta \bullet (p_0 \cap 1)(p_1 \cap 1)} = 0.$$

Now suppose that $k_i \leq u_i$ and $u_0u_1 = 0$. We wish to now show that assuming that $u_0 \vee u_1 = 1_T$ forces the category to be trivial.

Consider the map $q = u_0p_1 \vee u_1p_0$, note that it is total (as $\overline{q} = \overline{u_0p_1} \vee \overline{u_1p_0} = u_0 \vee u_1 = 1$) and it is, given our assumption, essentially a recursive decider for u_0. Define q' to be a code for q, so that $(q' \times 1)\bullet = q$. Observe that

$$\begin{aligned}
q'k_0 &= q'\overline{\Delta \bullet (p_0 \cap 1)} = \overline{q'\Delta \bullet (p_0 \cap 1)}\, q' = \overline{q'(q' \times 1) \bullet (p_0 \cap 1)}\, q' \\
&= \overline{q'(u_0p_1 \vee u_1p_0)(p_0 \cap 1)}\, q' = q'\overline{u_0p_1(p_0 \cap 1) \vee u_1p_0(p_0 \cap 1)} \\
&= \overline{0 \vee u_1p_0}\, q' = \overline{q'u_1p_0}\, q' = q'u_1
\end{aligned}$$

and similarly $q'k_1 = q'u_0$. This shows $q'u_1 = q'u_1u_1 = q'k_0u_1 = 0$ and similarly $q'u_0 = 0$ This is obviously bad and gives the following calculation to clinch it:

$$1_1 = \overline{q'} = \overline{q'(u_0 \vee u_1)} = \overline{q'u_0 \vee q'u_1} = 0.$$

This suffices to show the category is trivial! $\square$

7.4 The recursion theorem and Rice's theorem

The recursion theorems hold in any Turing category. There is the "second" recursion theorem:

Theorem 7.9 *In any Turing category, for any partial operation $f : T \times T \longrightarrow T$ on the Turing object T there is a (total) point $p : 1 \longrightarrow T$ such that $(p \times 1)\bullet = (p \times 1)f$.*

Proof: Set $h = (\Delta \times 1)(\bullet \times 1)f$ then there is a code, $h^\bullet$, for h, with $(h^\bullet \times 1)\bullet$ total and $(h^\bullet \times 1 \times 1)(\bullet \times 1)\bullet = h$ – as in the diagram – and setting $p = (h^\bullet \times h^\bullet)\bullet$ makes

$$\begin{tikzcd}
T \arrow[rrr, bend left, "p\times 1"] \arrow[dr, "h^\bullet\times 1"] \arrow[ddrr, bend right, "p\times 1"'] & & T\times T\times T \arrow[r, "\bullet\times 1"] & T\times T \arrow[dr, "f"] & \\
& T\times T \arrow[ur, "\Delta\times 1"] \arrow[rrr, "h"] \arrow[d, dotted, "h^\bullet\times 1\times 1"] & & & T \\
& T\times T\times T \arrow[r, "\bullet\times 1"'] & T\times T \arrow[urr, "\bullet"'] & &
\end{tikzcd}$$

commutative. Note that e is total as $(h^\bullet \times 1)\bullet$ is total. $\square$

This result is often stated in a rather different form which is worth recording:

Corollary 7.10 *In any Turing category, for any total map $g : T \longrightarrow T$ there is a (total) point p such that $((pg) \times 1)\bullet = (p \times 1)\bullet$.*

Proof: To obtain this form of the result from the above note that the total map, g, allows one to define $f = (g \times 1)\bullet$ from which the result is immediate. $\square$

This form of the result suggests that there is a "fixed point" for g in the (rather special) sense that the code p has the same computational effect as $g(p)$. In extensional Turing categories (that is when p is uniquely determined by $(p \times 1)\bullet$) this gives a real fixed point which explains the preoccupation with fixed points.

This "recursion theorem"is a crucial ingredient in obtaining a version of Rice's theorem for recursion categories.

We say that a restriction idempotent e on the Turing object of a recursion category is **extensional** (with respect to a given choice of Turing structure!) in case the following implication holds for all parallel maps f and g

$$((fe) \times 1)\bullet = g \times 1 \bullet \quad \Rightarrow \quad \overline{fe}\, g = \overline{fe}\, ge.$$

If take f and g to be (total) codes wuch that $fe = f : 1 \longrightarrow A$, so that f is in e and $\overline{fe} = 1$. The first equality then requires the behaviours of the codes f and g, when applied to an input, to be the same. For e to be extensional then it must follow that g also lies in e, that is $ge = g$. The condition is stated more generally as it must be applicable beyond codes.

An example of an extensional predicate in a recursion category is $e = \overline{(1 \times p_0) \bullet (p_1 \cap 1)}$ where p_0 and p_1 are total points. This predicate picks out all the codes c which when applied to p_0 give p_1 (that is $c \bullet p_0 = p_1$). Suppose $((fe) \times 1)\bullet = (g \times 1)\bullet$ then here is the calculation:

$$\begin{aligned}
\overline{fe}\, ge &= \overline{fe}\, g\overline{(1 \times p_0) \bullet (p_1 \cap 1)} \\
&= \overline{fe}\, \overline{g(1 \times p_0) \bullet (p_1 \cap 1)}\, g \\
&= \overline{fe}\, \overline{(1 \times p_0)(g \times 1) \bullet (p_1 \cap 1)}\, g \\
&= \overline{fe}\, \overline{(1 \times p_0)((fe) \times 1) \bullet (p_1 \cap 1)}\, g \\
&= \overline{fe}\, \overline{((fe)(1 \times p_0) \bullet (p_1 \cap 1)}\, g \\
&= \overline{fe}\, \overline{fee}\, g \\
&= \overline{fe}\, g
\end{aligned}$$

Clearly there are many others. An example for which the calculation for extensionality is similar is the codes which terminate on a particular point $e = \overline{(1 \times p_0)\bullet}$. Once we have a supply we may use the following observations to generate new extensional predicates:

Lemma 7.11

(i) The empty predictate 0 *and the true predicate* 1_T *are always extensional;*

(ii) If e_1 *and* e_2 *are extensional that* $e_1 e_2$ *and* $e_1 \vee e_2$ *are extensional;*

(iii) If e *is extensional and has a complement* e' *then* e' *is extensional.*

PROOF:

(i) This is immediate for the identity and zero predicate as the conclusion of the extension requirement is always true.

(ii) For the first $fe_1e_2 = fe_2e_2$ and so $(fe_1e_2 \times 1)\bullet = (g \times 1)\bullet$ by extensionality of e_2 gives $\overline{fe_1e_2}\,g = \overline{fe_1e_2}\,ge_2$ but by extensionality of e_1 gives $\overline{fe_1e_2}\,g = \overline{fe_1e_2}\,ge_1$ which easily gives $\overline{fe_1e_2}\,g = \overline{fe_1e_2}\,ge_1e_2$.

Suppose now that $(f(e_1 \vee e_2) \times 1)\bullet = (g \times 1)\bullet$ and e_1 and e_2 are extensional then

$$\begin{aligned} \overline{f(e_1 \vee e_2)}\,g &= \overline{fe_1}\,g \vee \overline{fe_2}\,g \\ &= \overline{fe_1}\,ge_1 \vee \overline{fe_2}\,ge_2 \\ &= \overline{fe_1}\,ge_1(e_1 \vee e_2) \vee \overline{fe_2}\,ge_2(e_1 \vee e_2) \\ &= \overline{fe_1}\,g(e_1 \vee e_2) \vee \overline{fe_2}\,g(e_1 \vee e_2) \\ &= \overline{f(e_1 \vee e_2)}\,g(e_1 \vee e_2). \end{aligned}$$

(iii) Suppose $(fe' \times 1)\bullet = (h \times 1)\bullet$ then we always have

$$\overline{fe'}\,h = \overline{fe'}\,h(e \vee e') = (\overline{fe'}\,he) \vee (\overline{fe'}\,he')$$

so that it suffices to show $\overline{fe'}\,he = 0$.

Consider

$$\begin{aligned} (\overline{fe'}\,he \times 1)\bullet &= (\overline{fe'}\,\overline{he} \times 1)(h \times 1)\bullet \\ &= (\overline{fe'}\,\overline{he} \times 1)(fe' \times 1)\bullet \\ &= (\overline{he}\,fe' \times 1)\bullet \end{aligned}$$

Now using extensionality of e:

$$\overline{\overline{fe'}\,he}\,\overline{he}\,fe' = \overline{\overline{fe'}\,he}\,\overline{he}\,fe'e = 0$$

but this means $\overline{he}\,fe' = 0$ so that $\overline{\overline{he}\,fe'} = \overline{he}\,\overline{fe'} = \overline{fe'}\,\overline{he} = 0$ and whence $\overline{fe'}\,he = \overline{fe}\,\overline{he}\,he = 0$.

$\square$

We shall say a restriction idempotent e on the Turing object is **non-trivial** in case there are two points, p_0 and p_1 with $p_0e = p_0$ and $p_1e = 0$.

Theorem 7.12 (Rice's theorem) *In a non-trivial recursion category no non-trivial extensional idempotent is complemented.*

PROOF: Suppose e is nontrivial and extensional. We wish to prove that if e has a complement e' then the setting must be trivial. Note that e' is necessarily extensional (so both are) and non-trivial (so both are). Thus, there are points p_0 and p_1 with $p_0e = p_0$ and $p_1e' = p_1$. Using the second recursion theorem define a point h by:

$$(h \times 1)\bullet = (h \times 1)((e \times 1)\pi_1(p_1 \times 1)\bullet \vee (e' \times 1)\pi_1(p_0 \times 1)\bullet)$$

where

$$\begin{aligned} (h \times 1)((e \times 1)\pi_1(p_1 \times 1)\bullet \vee (e' \times 1)\pi_1(p_0 \times 1)\bullet) &= (he \times 1)\pi_1(p_1 \times 1)\bullet \vee (he \times 1)\pi_1(p_0 \times 1)\bullet \\ &= (\overline{he} \times 1)(p_1 \times 1)\bullet \vee (\overline{he'} \times 1)(p_0 \times 1)\bullet \end{aligned}$$

then

$$\begin{aligned}
(he \times 1)\bullet &= (\overline{he} \times 1)(h \times 1)\bullet \\
&= (\overline{he} \times 1)((\overline{he} \times 1)(p_1 \times 1)\bullet \vee (\overline{he'} \times 1)(p_0 \times 1)\bullet) \\
&= (\overline{he} \times 1)(p_1 \times 1)\bullet = (\overline{he}\, p_1 \times 1)\bullet
\end{aligned}$$

so using extensionality we have:

$$\overline{he}\,\overline{he}\, p_1 = \overline{he}\,\overline{he}\, p_1 e.$$

However, $p_1 e = 0$ so that $\overline{he}\, p_1 = 0$ so that $he = 0$. By symmetry $he' = 0$ as well implying $h = h(e \vee e') = 0$ forcing the category to be trivial. □

It should be remarked that there is a term logic for these categories and this makes reasoning in them much more natural. The diagram chasing proofs, while solid, become much harder to follow once the number of steps in the proof becomes large. Indeed, this last result while classical still needs understanding categorically ... here I have rather slavishly simply translated the classical proofs. But developing more conceptual proofs is crucial to understanding these settings abstractly.

7.5 Gödel incompleteness

From Rice's theorem we know we cannot have a non-trivial extensional idempotent which is complemented and maintain consistency. However, this does not rule out the possibility of an extensional idempotent which is *not* complemented. Indeed, such a predicate would simply say that we can, in some sense, enumerate equal maps: this, of course, is an eminently reasonable hypothesis in any system which is generated by a finite set of axioms.

However, famously this is where Gödel's incompleteness theorem has something to say. For, if such system is consistent, it must be "incomplete". That is there are propositions which can be proven neither true nor false. Gödel's results caused a considerable philosophical furor at the time. It fuelled, for example, inappropriate speculation as to the limits of computation as compared to that of human thought.

It is, however, a Mathematical result about the relationship between "axiomatizability" and "computability" and, as such it is appropriate to search for an expression of it in terms of Turing categories and recursion categories. Pleasingly, it turns out that the result has a very elegant and very general expression in Turing categories.

This way of categorically interpreting Gödel's incompleteness result was first explored by Andre Joyal in his work on Arithmetic Universes: he observed that Gödel's theorem implied that in the initial Arithmetic Universe there are subobjects of the final object which are neither initial or the full subobject. Thus, as subobjects of the final object viewed as the truth values of the setting, Gödel result, as we express it below, states that if one is in a computational setting (in particular, a Turing category) in which one has a certain extensional predicate then one cannot be in a classical setting, in the sense that there only two truth values.

Given that mathematics is determinedly built on the assumption that everything is classical, this is a shattering enough observation in itself!!!

Given a Turing category, $\mathbb{X}$, with Turing object $(T, \bullet)$ a **provability predicate** is a restriction idempotent $e_{sfpf} \in \mathcal{O}(T)$ such that points k_1 and k_2 are in e_{pf} if and only if $\langle !k_1, 1\rangle\bullet \leq \langle !k_2, 1\rangle\bullet$.

Recall that a point of T is a total map $k : 1 \longrightarrow T$ with $\overline{k} = 1_1$, and that a point of T is to be thought of as "code" for the, "program" $\langle !k, 1\rangle \bullet : T \longrightarrow T$ (we have been writing $f^\bullet$ for a point and the $f : T \longrightarrow T$ for the program. To say points k_1 and k_2 are *in* e_{pf} is to say $\langle k_1, k_2\rangle e_{\mathsf{eq}} = \langle k_1, k_2\rangle$. Given a provability predicate e_{pf} one can immediately obtain an **extensional equality predicate** $E_{\mathsf{pf}} = e_{\mathsf{pf}}\overline{c_\times e_{\mathsf{pf}}}$: where k_1 and k_2 are in E_{pf} if and only if $\langle !k_1, 1\rangle \bullet = \langle !k_2, 1\rangle \bullet$.

Proposition 7.13 *A Turing category has an extensional equality predicate if and only if it has a provability predicate.*

PROOF: In a restriction category $f \leq g$ if and only if $\overline{f}\, g = f$ this turns the inequality into an equality test. In order to realize this idea at the level of codes we must be able to compose codes and take their restriction. The composition of codes is given by:

$$\begin{tikzcd} T \times T \arrow[rr, "\bullet"] & & T \\ T \times T \times T \arrow[u, dotted, "\mathsf{comp}\times 1"] \arrow[urr, "(1\times\bullet)\bullet"'] & & \end{tikzcd}$$

To see that one can take the restriction of codes is a bit more challenging.

The idea is to use the the following equality $\overline{f} = \langle f, 1\rangle \pi_0$ and the fact that $T \times T \lhd_{(s,r)} T$. Then $P_i : 1 \longrightarrow T$ is a code for $T \xrightarrow{r} T \times T \longrightarrow \pi_i T$. and the pairing of two codes is given by $\langle f^\bullet, g^\bullet\rangle^\bullet = (\langle f, g\rangle s)^\bullet$. $\square$

In a recursion category due to Rice's theorem the provability predicate cannot be recursive: if it were we could easily produce a non-trivial recursive extensional predicate as those codes extensionally equal to a particular point (say the code for the empty–never terminating–program). Of course, this does not preclude the possibility of having a non-recursive provability predicate. And, indeed, this can happen.

Theorem 7.14 *(Gödel's theorem for Turing categories) Every non-trivial Turing category with joins (or minimally a restriction zero) and a provability predicate has a predicate of the terminal object which is neither* $0 \in \mathcal{O}(1)$ *nor* $1_1 \in \mathcal{O}(1)$.

PROOF: We will use the second recursion theorem and the provability predicate: first use $e_{\mathsf{pf}} \in \mathcal{O}(T \times T)$ to generate $e_{\mathsf{pf}}! : T \times T \longrightarrow 1$ and whence by composition with a point of T a map $e_{\mathsf{pf}}!k : T \times T \longrightarrow T$. We need some other helpful functions: let $0^\bullet$ be a code of the zero map and $0^{\bullet\bullet}$ be a code for $!0^\bullet$ so that

$$\begin{tikzcd} T \times T \arrow[rr, "\bullet"] & & T \\ 1 \times T \arrow[u, dotted, "0^\bullet\times 1"] \arrow[urr, "0"'] & & \end{tikzcd} \qquad \begin{tikzcd} T \times T \arrow[rr, "\bullet"] & & T \\ 1 \times T \arrow[u, dotted, "0^{\bullet\bullet}\times 1"] \arrow[urr, "!0_\bullet"'] & & \end{tikzcd}$$

commute. Also, from codes for two programs, one can construct a code for the composite as discussed above.

We now set $f = (\langle 1, !0^{\bullet\bullet}\rangle \mathsf{comp}) \times 1)e_{\mathsf{pv}}!k : T \times T \longrightarrow T$ where $k : 1 \longrightarrow T$ Now by the second recursion theorem, Theorem 7.9, there is a point $g^\bullet : 1 \longrightarrow T$ so that $g = \langle !g^\bullet, 1\rangle \bullet = \langle !g^\bullet, 1\rangle f : 1 \longrightarrow T$ and we can turn this into a map $g^\bullet! : 1 \longrightarrow 1$. We have:

$$
\begin{aligned}
0^\bullet g! &= \langle g^\bullet, 0^\bullet\rangle f! \\
&= \langle g^\bullet, 0^\bullet\rangle (\langle 1, !0^{\bullet\bullet}\rangle \mathsf{comp}) \times 1)e_{\mathsf{pv}}!k! \\
&= \langle g^\bullet, 0^{\bullet\bullet}\rangle \mathsf{comp}), 0^\bullet\rangle e_{\mathsf{pv}}! \\
&= \langle (!0^\bullet g)^\bullet, 0^\bullet\rangle e_{\mathsf{pv}}! \quad \text{as } e_{\mathsf{pv}} \text{ is extensional.}
\end{aligned}
$$

Now consider $0^\bullet g! : 1 \longrightarrow 1$: if there were only two predicates of the final object this would have to be either 0 or 1. Suppose $0^\bullet g! = 0$ then we know $\langle (0^\bullet g)^\bullet, 0^\bullet\rangle e_{\mathsf{pv}} = \langle (0^\bullet g)^\bullet, 0^\bullet\rangle$ and $\langle (!0^\bullet g)^\bullet, 0^\bullet\rangle! = 1_1$ showing that $0 = 1$ and that the category is trivial contrary to assumption.

Conversely, suppose $0^\bullet g! = 1_1$ then $0^\bullet g$ is total so that $\langle\langle g^\bullet, (0^\bullet)^\bullet\rangle \mathsf{comp}, 0^\bullet\rangle e_{\mathsf{pf}}!$ is 1_1 and so the category cannot be consistent. Conversely, suppose it is $0^\bullet g! = 1$ then we know $g \not\leq 0$ and so $\langle (0^\bullet g)^\bullet, 0^\bullet\rangle e_{\mathsf{pv}}$ cannot be total, in fact, if there are only two truth values this must be 0 as it cannot be 1. Thus again $0 = 1$ and contrary to assumption the category collapses. □

7.6 General interleaving

An important property that the classical setting for computability possesses is a much stronger version of finite interleaving which, given an enumeration of codes allows one to interleave their application. This clearly introduces unions for compatible families with codes which can be enumerated. Thus, although one does *not* have arbitrary joins for compatible families one certainly does have joins over families of this special form. Classically this is used to show one can approximate a partial recursive function by its finite effects.

A Turing category has (general) **interleaving** in case given any $e \in \mathcal{O}(T)$, where T is the Turing object, then there is an $f_e : T \longrightarrow T$ (an interleaving of e) and a $k_e : T \longrightarrow T$ such that

$$
\begin{tikzcd}
T \times T \arrow[r, "e \times 1"] & T \times T \arrow[r, "\bullet"] & T \\
T \arrow[u, "\langle k_e, 1\rangle"] \arrow[urr, "f_e"'] & &
\end{tikzcd}
$$

commutes and $\overline{(e \times 1)\bullet} \leq \overline{\pi_1 k_e}$.

We immediately observe:

Lemma 7.15 *If a Turing category has interleaving then*

(i) $\overline{k_e} = \overline{f_e}$*;*

(ii) $\langle k_e, 1\rangle$ *is a partial section of* $\overline{(e \times 1)\bullet}\,\pi_1$*;*

(iii) $k_e e = k_e$.

A **partial section** of a map $f : A \longrightarrow B$ is a restriction category is a map $g : B \longrightarrow A$ such that $gf = \overline{g}$ and $fgf = f$. Note that the second requirement can be restated, in view of the first requirement, as $f\overline{g} = f$.

Proof:

(i) $\overline{k_e} \leq \overline{f_e}$ as $\overline{k_e}\ \overline{f_e} = \overline{\overline{k_e}\,\langle k_e, 1\rangle(1 \times e)\bullet} = \overline{\langle k_e, 1\rangle(1 \times e)\bullet} = \overline{f_e}$. But $\overline{f_e} \leq \overline{k_e}$ as

$$\begin{array}{rcl} \overline{f_e} & = & \overline{\langle k_e, 1\rangle(1 \times e)\bullet} = \overline{\langle k_e, 1\rangle\overline{(1 \times e)\bullet}} \\ & \leq & \overline{\langle k_e, 1\rangle\overline{\pi_1 k_e}} = \overline{\langle k_e, 1\rangle\pi_1 k_e} = \overline{k_e}\,. \end{array}$$

(ii)

$$\begin{array}{rcl} \langle k_e, 1\rangle\overline{(1 \times e)\bullet}\,\pi_1 & = & \overline{\langle k_e, 1\rangle(1 \times e)\bullet}\,\langle k_e, 1\rangle\pi_1 \\ & = & \overline{f_e}\ \overline{k_e} = \overline{k_e} = \overline{\langle k_e, 1\rangle} \\ \overline{(1 \times e)\bullet}\,\pi_1\overline{\langle k_e, 1\rangle} & = & \overline{(1 \times e)\bullet}\,\pi_1\overline{k_e} \\ & = & \overline{(1 \times e)\bullet}\,\overline{\pi_1 k_e}\,\pi_1 \\ & = & \overline{(1 \times e)\bullet}\,\pi_1 \quad \text{as } \overline{(1 \times e)\bullet} \leq \overline{\pi_1 k_e} \end{array}$$

(iii) $k_e e \leq k_e$ clearly for the converse

$$k_e = \overline{f_e}\,k_e = \overline{\langle k_e e, 1\rangle\bullet}\,k_e \leq \overline{\langle k_e e, 1\rangle}\,k_e = \overline{k_e e}\,k_e = k_e e.$$

□

As $\langle k_e, 1\rangle$ is a partial section of $\overline{(e \times 1)\bullet}\,\pi_1$ it follows that $\overline{\langle k_e, 1\rangle}$ is the absolute range (or image) of $\overline{(e \times 1)\bullet}\,\pi_1$. Thus, $(e \times 1)\bullet$ is defined on values precisely for the predicate $\overline{k_e}$.

This leads to a a simpler statement of the requirement for a Turing category to have interleaving:

Lemma 7.16 *A Turing category has interleaving if and only if each $\overline{(e \times 1)\bullet}\,\pi_1$ has a partial section.*

PROOF: We must prove the existence of a partial section suffices: we can assume it of the form $\langle k_1, k_2\rangle : T \longrightarrow T \times T$ where

$$\langle k_1, k_2\rangle\overline{(e \times 1)\bullet}\,\pi_1 = \overline{\langle k_1, k_2\rangle} \quad \text{and} \quad \overline{(e \times 1)\bullet}\,\pi_1\overline{\langle k_1, k_2\rangle} = \overline{(e \times 1)\bullet}\,\pi_1$$

We therefore have

$$\begin{array}{rcl} \overline{k_1}\,k_2 & = & \overline{\langle k_1, k_2\rangle}\,k_2 = \overline{\langle k_1, k_2\rangle\overline{(e \times 1)\bullet}\,\pi_1}\,k_2 \\ & = & \overline{\langle k_1, k_2\rangle(e \times 1)\bullet}\,k_2 = \overline{\langle k_1, k_2\rangle(e \times 1)\bullet}\,\langle k_1, k_2\rangle\pi_1 \\ & = & \langle k_1, k_2\rangle\overline{(e \times 1)\bullet}\,\pi_1 = \overline{\langle k_1, k_2\rangle} = \overline{k_1}\,\overline{k_2} \end{array}$$

and whence $\langle k_1, k_2\rangle = \langle\overline{k_2}\,k_1, 1\rangle$. So we may set $k_e = \overline{k_2}\,k_1$ and $f_e = \langle k_e, 1\rangle(e \times 1)$. Finally, $\overline{(e \times 1)\bullet} \leq \overline{\pi_1 k_e}$ as

$$\begin{array}{rcl} \overline{(e \times 1)\bullet}\ \overline{\pi_1 k_e} & = & \overline{\overline{(e \times 1)\bullet}\,\pi_1 k_e} = \overline{\overline{(e \times 1)\bullet}\,\pi_1\overline{\langle k_1, k_2\rangle}} \\ & = & \overline{\overline{(e \times 1)\bullet}\,\pi_1\overline{\langle k_1, k_2\rangle}} = \overline{\overline{(e \times 1)\bullet}\,\pi_1} = \overline{(e \times 1)\bullet} \end{array}$$

□

We would expect that if a recursion category has general interleaving that it should have finite interleaving. This is the case as we now show:

Proposition 7.17 *If a recursion category has general interleaving then it has finite interleaving.*

Proof: Given $f, g : T \longrightarrow T$ we want to show that there is an $h : T \longrightarrow T$ which is an interleaving of f and g, that is:

$$(h \cap f) \vee (h \cap g) = h \ \text{ and } \ \overline{h} = \overline{f} \vee \overline{g}\,.$$

Define (total) points $f^\bullet, g^\bullet : 1 \longrightarrow T$ by:

$$\begin{tikzcd} T\times T \arrow[r, "\bullet"] & T \\ 1\times T \arrow[u, dotted, "f^\bullet\times 1"] \arrow[ur, "\pi_1 f"'] & \end{tikzcd} \qquad \begin{tikzcd} T\times T \arrow[r, "\bullet"] & T \\ 1\times T \arrow[u, dotted, "g^\bullet\times 1"] \arrow[ur, "\pi_1 g"'] & \end{tikzcd}$$

Set $e = (1\cap!f^\bullet) \vee (1\cap!g^\bullet)$. This restriction idempotent which isolates exactly the codes for the two maps: In fact, the map $f^\bullet : 1 \longrightarrow T$ has a restriction inverse $(f^\bullet)^{(-1)}$ so that $(f^\bullet)^{(-1)} f^\bullet = 1\cap!f^\bullet$ and $f^\bullet(f^\bullet)^{(-1)} = 1_1$ and similarly for $g^\bullet$. By general interleaving we have maps k and h such that

$$\begin{tikzcd} T\times T \arrow[r, "e\times 1"] & T\times T \arrow[r, "\bullet"] & T \\ T \arrow[u, "\langle k,1\rangle"] \arrow[urr, "h"'] & & \end{tikzcd}$$

commutes and $\overline{(e\times 1)\bullet} \leq \overline{\pi_1 k}$. The claim is now that h is an interleaving of f and g. To prove this we need to verify:

(1) $\overline{h} = \overline{f} \vee \overline{g}$**:**

$$\begin{aligned}
\overline{h} &= \overline{\langle k,1\rangle(e\times 1)\bullet} = \overline{\langle k,1\rangle(((1\cap!f^\bullet)\vee(1\cap!g^\bullet))\times 1)\bullet} \\
&= \overline{\langle k,1\rangle((1\cap!f^\bullet)\times 1)\bullet} \vee \overline{\langle k,1\rangle((1\cap!g^\bullet)\times 1)\bullet} \\
&= \overline{\langle k,1\rangle(((f^\bullet)^{(-1)}f^\bullet)\times 1)\bullet} \vee \overline{\langle k,1\rangle(((g^\bullet)^{(-1)}g^\bullet)\times 1)\bullet} \\
&= \overline{\langle k,1\rangle((f^\bullet)^{(-1)}\times 1)\pi_1 f} \vee \overline{\langle k,1\rangle((g^\bullet)^{(-1)}\times 1)\pi_1 g} \\
&= \overline{\overline{k(f^\bullet)^{(-1)}}\, f} \vee \overline{\overline{k(g^\bullet)^{(-1)}}\, g} \\
&\leq \overline{f} \vee \overline{g}
\end{aligned}$$

But also we have

$$\begin{aligned}
\overline{f} &= \overline{\langle !f^\bullet,1\rangle\bullet} \\
&= \overline{\langle !f^\bullet,1\rangle(((1\cap!f^\bullet)\vee(1\cap!g^\bullet))\times 1)\bullet} \\
&\qquad \text{as } !f^\bullet = !f^\bullet(1\cap!f^\bullet) = !f^\bullet(1\cap!f^\bullet)((1\cap!f^\bullet)\vee(1\cap!g^\bullet)) = !f^\bullet((1\cap!f^\bullet)\vee(1\cap!g^\bullet)) \\
&= \overline{\langle !f^\bullet,1\rangle(e\times 1)\bullet} = \overline{\langle !f^\bullet,1\rangle\overline{(e\times 1)\bullet}} \\
&\leq \overline{\langle !f^\bullet,1\rangle\overline{\pi_1 k}} \qquad \text{as } \overline{(e\times 1)\bullet} \leq \overline{\pi_1 k} \\
&= \overline{\langle !f^\bullet,1\rangle\pi_1 k} = \overline{\overline{!f^\bullet}\, k} = \overline{k} \quad \text{as } f^\bullet \text{ is total.} \\
&= \overline{h}
\end{aligned}$$

and similarly $\overline{g} \leq \overline{h}$ so that $\overline{f} \vee \overline{g} \leq \overline{h}$ establishing equality.

(2) $(h \cap f) \vee (h \cap g) = h$**:**

Here first note that it suffices to prove $h \leq (h \cap f) \vee (h \cap g)$. Next note that

$$\begin{array}{rcl} \langle k,1\rangle((1\cap !f^\bullet) \times 1) \bullet \cap f & = & \langle k,1\rangle((f^\bullet)^{(-1)} f^\bullet \times 1) \bullet \cap f \\ & = & \langle k,1\rangle((f^\bullet)^{(-1)} \times 1)\pi_1 f \cap f \\ & = & \overline{k(f^\bullet)^{(-1)}}\, f \cap f \\ & = & \overline{k(f^\bullet)^{(-1)}}\, f \\ & = & \langle k,1\rangle((1\cap !f^\bullet) \times 1)\bullet \end{array}$$

and similarly $\langle k,1\rangle((1\cap !g^\bullet) \times 1) \bullet \cap g = \langle k,1\rangle((1\cap !g^\bullet) \times 1)\bullet$. This permits the following calculation:

$$\begin{array}{rcl} h & = & \langle k,1\rangle(e \times 1) \bullet \\ & = & \langle k,1\rangle(((1\cap !f^\bullet) \vee (1\cap !g^\bullet)) \times 1) \bullet \\ & = & \langle k,1\rangle((1\cap !f^\bullet) \times 1) \bullet \vee \langle k,1\rangle((1\cap !g^\bullet) \times 1) \bullet \\ & = & (\langle k,1\rangle(((1\cap !f^\bullet) \times 1)\bullet) \cap f) \vee ((\langle k,1\rangle((1\cap !g^\bullet) \times 1)\bullet) \cap g) \\ & \leq & ((\langle k,1\rangle(((1\cap !f^\bullet) \vee (1\cap !g^\bullet)) \times 1)\bullet) \cap f) \vee ((\langle k,1\rangle(((1\cap !f^\bullet) \vee (1\cap !g^\bullet)) \times 1)\bullet) \cap g) \\ & = & ((\langle k,1\rangle(e \times 1)\bullet) \cap f) \vee ((\langle k,1\rangle(e \times 1)\bullet) \cap g) \\ & = & (h \cap f) \vee (h \cap g) \end{array}$$

$\square$

8 Exercises for Part II

1. Show that in a restriction category the relation $f \smile g$ need not be transitive (if it is the category is said to be unitary).

2. Is the category of sets and relations a restriction category where restriction gives the domain of the relation?

3. Show that the category of meet semilattices with maps which preserve the binary meet but not necesarily the top element (stable maps) is a *co*restriction category.

4. Show that the categry of frames (= locales) with join preserving stable maps is a join *co*restriction category.

5. Prove that in any restriction category with meets that for any parallel endomorphisms f and g, $(\overline{g} \cap f)(g \cap \overline{f}\,) = (f \cap g \cap 1)$.

6. In any category do the sections form a stable system of monics? Provide a counter-example.

7. Prove that in the opposite of the category of commutative rings the class of maps which are localizations forms a system of monics. Ponder what $\mathsf{Par}(\mathsf{Ring}^{\mathrm{op}}, \mathsf{loc})$ looks like: what is the restriction?

8. In sets pushouts of epics are epic: therefore in the opposite category these maps form a system of monics what does resulting category $\mathsf{Par}(\mathsf{Set}^{\mathrm{op}}, \mathsf{epic})$ look like?

9. Prove that in a Turing category the restriction of every $\bullet^{(n)}$ is complete.

10. Prove that in a recursion category every complete set is undecidable.

11. Prove that in a recursion category if p_0 and p_1 are disjoint (total) points that $\overline{(1 \times p_0) \bullet (1 \cup p_0)}$ is inseperable from $\overline{(1 \times p_0) \bullet (1 \cup p_1)}$.

References

[1] A. Asperti, A. Ciabattoni, *Effective applicative structures, in: Category Theory and Computer Science*, 1995.

[2] Asperti, A. & Longo G.*Categories, Types, and Structures* MIT-Press (1991)

[3] Barr M. & Wells C., 1990, "Category Theory for Computer Science" Prentice Hall (now CRM, Montreal)

[4] I. Bethke, *Notes on partial combinatory algebras*, Ph.D. Thesis, Universiteit van Amsterdam, 1988.

[5] I. Bethke, J.W. Klop, R. de Vrijer, *Completing partial combinatory algebras with unique head-normal forms*, in: Logic in Computer Science, IEEE Computer Society Press, 1996, pp. 448-545.

[6] Cockett, R. & X. Guo *Stable Meet Semilattice Fibrations and Free Restriction Categories* Theory and Application of Categories, Vol. 16 (2006) 307-341.

[7] J.R.B. Cockett, P.J.W. Hofstra *introduction to Turing categories* Annals of Pure and Applied Logic 156 (2008) 183-209.

[8] Cockett, R. & Lack S. *Restriction Categories I: Categories of partial maps*, Theoretical Computer Science, Vol.270 (2002) 223-259.

[9] Cockett, R. & Manes E. *Boolean and Classical Restriction Categories* Mathematical Structures in Computer Science, Vol. 19-2 (2009) 357-416.

[10] P.-L. Curien, A. Obtulowitz, *Partiality, cartesian closedness and toposes*, Information and Computation 80 (1989) 50-95.

[11] Davis M., Sigal R. & Weyuker E. *Computability, Complexity, and Languages* Morgan Kaufmann (1994)

[12] Di Paola & Heller R. *Dominical categories: recursion theory without elements* Journal of Symbolic Logic, Vol. 53(3) (1987) 594-6635.

[13] Eilenberg, S. & Mac Lane, S., 1945, "General Theory of Natural Equivalences", Transactions of the American Mathematical Society, 58, 231-294.

[14] F. Lengyel, *More existence theorems for recursion categories*, Annals of Pure and Applied Logic 125 (1-3) (2004) 1-41

[15] Freyd P. & Scedrov A., 1990 "Categories and Allegories" North Holland.

[16] Freyd P., 1964, "Abelian Categories" Harper and Row.

[17] Kan, D. M., 1958, "Adjoint Functors", Transactions of the American Mathematical Society, 87, 294-329.

[18] Lambek, J. & Scott, P.J., 1986, “Introduction to Higher Order Categorical Logic”, Cambridge: Cambridge University Press.

[19] J. Longley, *Realizability toposes and language semantics*, Ph.D. Thesis, University of Edinburgh, 1994

[20] G. Longo, E. Moggi, A category theoretic characterization of functional completeness, Theoretical Computer Science 70 (2) (1990) 193-211.

[21] Lawvere, F. W., 1969, “Adjointness in Foundations”, Dialectica, 23, 281-295.

[22] Mac Lane, S., 1971, “Categories for the Working Mathematician”, New York: Springer Verlag.

[23] Manes E. *Cockett-Lack Restriction Categories, Semigroups, and Topologies* FMCS Ottawa. www.site.uottawa.ca/ phil/lfc/fmcs2003/Restriction.pdf¿ (2003).

[24] E. Moggi, *The partial lambda calculus*, Ph.D. Thesis, Edinburgh University, 1988.

[25] P. Odifreddi, *Classical Recursion Theory*, in: Studies in Logic, vol. 125, North-Holland, 1989.

[26] E. Robinson, G. Rosolini, *Categories of partial maps*, Information and Computation 79 (1988) 94-130.

[27] L. Schrder, *The logic of the partial-calculus with equality*, in: J. Marcinkowski, A. Tarlecki (Eds.), Computer Science Logic, CSL 04, in: Lecture Notes in Computer Science, vol. 3210, Springer Verlag, 2004.

[28] Seely R. *Linear Logic: Autonomous Categories and Cofree Coalgebras* in “Categories in Computer Science and Logic,” proceedings of the AMS-IMS-SIAM Joint Summer Research Conference, Boulder, Colorado Gray J. & Scedrov A. (Eds.) American Mathematics Society (1989) 371-382.

[29] H.R. Strong, *Algebraically generalized recursive function theory*, IBM Journal of Research and Development 12 (1968) 465-475.

[30] E.G. Wagner, *Uniform reflexive structures: On the nature of Gödelization and relative computability*, Transactions of the American Mathematical Society 144 (1969) 1-41.