Part one

Tutorials

1

Category Theory for Linear Logicians

Richard Blute and Philip Scott
Department of Mathematics and Statistics
University of Ottawa

Abstract

This paper presents an introduction to category theory with an emphasis on those aspects relevant to the analysis of the model theory of linear logic. With this in mind, we focus on the basic definitions of category theory and categorical logic.

An analysis of cartesian and cartesian closed categories and their relation to intuitionistic logic is followed by a consideration of symmetric monoidal closed, linearly distributive and $*$-autonomous categories and their relation to multiplicative linear logic. We examine nonsymmetric monoidal categories, and consider them as models of noncommutative linear logic. We introduce traced monoidal categories, and discuss their relation to the geometry of interaction. The necessary aspects of the theory of monads is introduced in order to describe the categorical modelling of the exponentials. We conclude by briefly describing the notion of *full completeness*, a strong form of categorical completeness, which originated in the categorical model theory of linear logic.

No knowledge of category theory is assumed, but we do assume knowledge of linear logic sequent calculus and the standard models of linear logic, and modest familiarity with typed lambda calculus.

1.0 Introduction

Category theory arose as an organizing framework for expressing the *naturality* of certain constructions in algebraic topology. Its subsequent applicability, both as a language for simply expressing complex relationships between mathematical structures and as a mathematical theory in its own right, is remarkable. Categorical principles have been put to

good use in virtually every branch of mathematics, in most cases leading to profound new understandings.

Roughly a *category* is an abstraction of the principle that the morphisms between objects of interest are just as important as the objects themselves. So a category will consist of two classes, the class of objects and the class of morphisms between objects. One must have a composition law, and each object must come equipped with a specified identity morphism. This data must satisfy some evident axioms. From this simple definition, an enormous theory follows. For example, one next defines morphisms between categories; these are *functors*. One can go on to define morphisms between functors; these are *natural transformations*, and on and on. There is a remarkably rich interaction between these structures. As expositions of this theory, we highly recommend [53, 21].

Categorical logic begins with the idea that, given a logic, one can form a category whose objects are formulas and whose morphisms are (equivalence classes of) proofs. The question of the proper notion of equivalence is extremely important and delicate. We will examine it in some detail below. There are several benefits to the formation of this category. First, under this interpretation, the logic's connectives are naturally exhibited as functors, and the logic's inference rules are exhibited as natural transformations. Then models of the logic can be simply defined as structure-preserving functors from this syntactic category to a category with the appropriate structure. Second, the category so formed will typically be freely generated in a certain sense, and can thus be used to derive general information about all categories of the same structure. The most well-developped examples of this idea are the relations between intuitionistic logic and cartesian closed categories, and between linear logic and $*$-autonomous categories. Both of these relationships will be described below.

The goal of this paper is to establish sufficient categorical background to understand these relationships and their consequences. We will introduce cartesian closed categories (cccs) and describe the translation between cccs and intuitionistic logic. This is the most well-established example of categorical logic, and is the subject of the book [51]. This is followed by a consideration of monoidal, symmetric monoidal closed, linearly distributive and $*$-autonomous categories and the translation between these structures and multiplicative linear logic. One of the most intriguing aspects of linear logic is that it is sufficiently flexible as a logical system to allow one to define noncommutative versions. With this

in mind, we examine nonsymmetric monoidal categories, and consider them as models of noncommutative linear logic. We will especially focus on examples arising from the representation theory of *Hopf algebras*.

We also introduce traced monoidal categories, which arose independently of linear logic, but were subsequently seen to provide the appropriate framework for the analysis of Girard's geometry of interaction. Computationally, the most important fragment of linear logic is the exponential fragment, and its categorical structure leads one to the notion of *Seely model*. The necessary aspects of the theory of monads is introduced in order to describe the categorical modelling of the exponentials. We conclude by briefly describing the notion of *full completeness*, a strong form of categorical completeness, which originated in the categorical model theory of linear logic [3]. Full completeness is an excellent example of the influence of categorical principles on logical semantics, not just for linear logic, but for general logics.

No knowledge of category theory is assumed, but we do assume knowledge of linear logic sequent calculus and the standard models of linear logic. Also it would help to have a modest familiarity with typed lambda calculus (as in Girard's [39]). This paper may be considered a companion to the article [59], but stressing the linear logic aspects. We note that we only focus on aspects of category theory of immediate relevance to linear logic. So important topics like limits and colimits are omitted.

1.1 Categories, Functors, Natural Transformations

1.1.1 Basics of Categories

A *category* $\mathcal{C}$ consists of two classes, *Objects* and *Arrows*, together with two functions $\mathit{Arrows} \underset{cod}{\overset{dom}{\rightrightarrows}} \mathit{Objects}$ satisfying the following properties (we write $A \xrightarrow{f} B$ for: $f \in \mathit{Arrows}$, $dom(f) = A$ and $cod(f) = B$):

- There are *identity* arrows $A \xrightarrow{id_A} A$, for each object A,
- There is a partially-defined binary *composition* operation on arrows, denoted by juxtaposition,

$$\frac{A \xrightarrow{f} B \quad B \xrightarrow{g} C}{A \xrightarrow{gf} C}$$

(defined only when $dom(g) = cod(f)$) satisfying the following equations:

(i) $fid_A = f = id_B f$, where $A \xrightarrow{f} B$,
(ii) $h(gf) = (hg)f$, where $A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$.

A category is called *large* or *small* depending upon whether its class of objects is respectively a proper class or a set, in the sense of Gödel-Bernays set theory. We denote by $\mathcal{C}(A, B)$ the collection of arrows $A \to B$ in the category $\mathcal{C}$. A category is *locally small* if $\mathcal{C}(A, B)$ is a set, for all objects A, B. It is convenient to represent arrows graphically. Equations in categories are also typically represented graphically and are called *commutative diagrams*.

Many familiar classes of structures in mathematics and logic can be organized into categories. Here are some basic examples. Verification that the set of arrows is closed under composition as well as satisfying the axioms of a category is left as an exercise.

Set: This (large) category has the class of all sets as Objects, with all set-theoretic functions as Arrows. Identity arrows and composition of arrows are defined in the usual way.

Rel: This has the same objects as **Set**, but an arrow $A \xrightarrow{R} B$ is a binary relation $R \subseteq A \times B$. Here composition is given by *relational product*, i.e. given $A \xrightarrow{R} B \xrightarrow{S} C$,

$$A \xrightarrow{SR} C = \{(a, c) \in A \times C \mid \exists b \in B \text{ such that } (a, b) \in R \ \& \ (b, c) \in S\}$$

while the identity arrows $A \xrightarrow{id_A} A$ are given by the diagonal relations: $id_A = \{(a, a) \mid a \in A\}$.

Universal Algebras: Here Objects can be any equational class of algebras (e.g. semigroups, monoids, groups, rings, lattices, heyting or boolean algebras, $\cdots$). Arrows are *homomorphisms*, i.e. set-theoretic functions preserving the given structure. Composition and identities are induced from **Set**. We use boldface notation for the names of the associated categories, e.g. **Group**, **Lat**, **Heyt**, for the categories of groups, lattices, and Heyting algebras, resp.

$\mathbf{Vec_k}$: Here Objects are vector spaces over a field $\mathbf{k}$, and Arrows are $\mathbf{k}$-linear maps. We usually omit the subscript $\mathbf{k}$, and write **Vec** for short. An important subcategory of **Vec** is the category $\mathbf{Vec}_{fd}$ of finite dimensional vector spaces and linear maps. Of course, one can also consider various classes of topological vector spaces and normed spaces, with appropriate notions of map.

Top: Here objects are topological spaces and morphisms are continuous maps. One can also consider various homotopy categories, i.e.

where the morphisms are homotopy equivalence classes of continuous maps. It is from this sort of example that category theory originally arose.

Poset: Here the objects are partially-ordered sets and morphisms are monotone maps. A particularly important example arising in theoretical computer science is the category ω**-CPO** of posets in which ascending countable chains $\cdots a_i \leq a_{i+1} \leq a_{i+2} \leq \cdots$ have suprema, and in which morphisms are poset maps preserving suprema of countable chains. Composition and identities are inherited from **Set**. For an introduction to this and other aspects of domain theory, see [8].

Of course, those areas of mathematics that heavily use category theory, e.g. algebraic topology, algebraic geometry, and homological algebra, are replete with many more sophisticated examples.

The previous examples were large categories, i.e. in which the collections of objects form proper classes in the sense of set theory. We now present some "small" categories, based on much smaller collections of objects and arrows:

One: The category with one object and one (identity) arrow.

Discrete categories: These are categories where the only arrows are identities. A set X becomes a discrete category, by letting the objects be the elements of X, and adding an identity arrow $x \to x$ for each $x \in X$. All (small) discrete categories arise in this way.

A monoid: A monoid M gives a category with one object, call it $\mathcal{C}_M$, as follows: if the single object is $*$, we define $\mathcal{C}_M(*,*) = M$. Composition of maps is multiplication in the monoid. Conversely, note that every category $\mathcal{C}$ with one object corresponds to a monoid, namely $\mathcal{C}(*,*)$.

A preorder: A preordered set $\mathbb{P} = (P, \leq)$ (where $\leq$ is a reflexive & transitive relation) may be considered as a category, whose objects are just the elements of $\mathbb{P}$ and in which we define $\mathbb{P}(a,b) = \{*\}$ if $a \leq b$ and $\mathbb{P}(a,b) = \emptyset$ if $a \not\leq b$. Thus, given two objects $a, b \in \mathbb{P}$, there is at most one arrow from a to b; moreover, there is an arrow $a \to b$ in $\mathbb{P}$ exactly when $a \leq b$. In this case, the category laws are exactly the preorder conditions.

Graphs and finite categories: A *graph* (more precisely, a directed multigraph), consists of a pair of sets, called *Objects* and *Arrows*, together with two functions $\textit{Arrows} \underset{cod}{\overset{dom}{\rightrightarrows}} \textit{Objects}$. Every (small) category has an underlying graph, obtained by simply ignoring the other

data beyond dom, cod. In particular, any finite category can be represented by simply drawing its underlying graph and assuming the existence of all well-defined compositions of arrows. Notice that all vertices in the underlying graph of a category have loops (given by identity arrows). Indeed, another way of looking at a category is as a kind of graph with additional structure (i.e. identity edges, a composition law and equations).

Graphs form a category **Graph** whose objects are graphs and whose arrows are pairs of functions $Arrows_0 \xrightarrow{f} Arrows_1$ and $Objects_0 \xrightarrow{g} Objects_1$ such that $gdom_0 = dom_1 f$ and $gcod_0 = cod_1 f$.

1.1.2 Deductive systems as categories

In the 1960's, Lambek introduced the novel idea of using Gentzen's methods in category theory and linguistics. His new approach involved the use of proof-theoretical methods in constructing free categories and for solving coherence (i.e. decision) problems. At the same time he emphasized a fundamental new idea: arrows in (freely generated) categories are equivalence classes of proofs. Lambek's work raises a question of particular relevance to linear logicians: what should the equations between proofs be? There is no ultimate answer except that Lambek's work would seem to say that the equations should be elegant and natural from the viewpoint of category theory. This section will follow [51] closely. For more on the history of this area, see [51] and the references therein.

Definition 1.1 A *deductive system* is a labelled directed graph (whose objects are called *formulas* and whose arrows are called *labelled sequents*). There are certain specified arrows (called *axioms*) among which are arrows $A \xrightarrow{id_A} A$, for all formulas A, and certain specified rules (called "inference rules") for generating new arrows from old ones, among which is the composition rule called "cut": $\dfrac{A \xrightarrow{f} B \quad B \xrightarrow{g} C}{A \xrightarrow{gf} C}\ cut$, for all formulas A, B, C.

A deductive system freely generates "labelled proof trees" by the following procedure:

- Axioms are proof trees.
- The set of proof trees must be closed under the inference rules.

The root of a proof tree is called a "provable sequent", or "proof" for short, while the leaves of the tree are axioms.

Example 1.2 Let $\mathcal{G}$ be a graph. The *deductive system freely generated from* $\mathcal{G}$ is defined as follows:

(i) The formulas are the objects of $\mathcal{G}$ (also called *atomic formulas*).

(ii) The axioms consist of a distinguished *identity* axiom $A \xrightarrow{id_A} A$, for each formula A, together with all the arrows of $\mathcal{G}$ (the latter are sometimes called *nonlogical axioms*).

(iii) Cut is the only rule of inference.

A deductive system freely generated from $\mathcal{G}$ forms a category $F(\mathcal{G})$, *the category freely generated from* $\mathcal{G}$, whose objects are all the formulas and whose arrows are equivalence classes of proofs. Namely, we impose equations between proof trees by taking the congruence relation generated by the following equations:

$$\frac{A \xrightarrow{f} B \quad B \xrightarrow{id_B} B}{A \xrightarrow{id_B f} B} \quad = \quad A \xrightarrow{f} B$$

$$\frac{A \xrightarrow{id_A} A \quad A \xrightarrow{f} B}{A \xrightarrow{f id_A} B} \quad = \quad A \xrightarrow{f} B$$

$$\frac{A \xrightarrow{f} B \qquad \dfrac{B \xrightarrow{g} C \quad C \xrightarrow{h} D}{B \xrightarrow{hg} D}}{A \xrightarrow{(hg)f} D} \quad = \quad \frac{\dfrac{A \xrightarrow{f} B \quad B \xrightarrow{g} C}{A \xrightarrow{gf} C} \qquad C \xrightarrow{h} D}{A \xrightarrow{h(gf)} D}$$

An important special case is the following:

Example 1.3 (Deductive systems generated by discrete graphs) A graph $\mathcal{G}_0$ is *discrete* if it has no arrows: it may be identified with a set (of objects). The deductive system generated from the set $\mathcal{G}_0$ has only atomic formulas (objects of $\mathcal{G}_0$) and for axioms it has only identity axioms $A \xrightarrow{id_A} A$, for atomic formulas A. $F(\mathcal{G}_0)$ is called the *free category generated from the set of objects* $\mathcal{G}_0$.

We will later consider freely generated categories with additional structure (i.e. with additional operations on formulas, and additional axioms and rules of inference). It is possible conversely to *define* a category as a certain kind of deductive system, although in that case it will not

necessarily be freely generated: the class of objects may simply be specified and the class of arrows merely closed under appropriate operations and equations (see [51]). Moreover, if the category is large, like **Set**, the objects and arrows form proper classes, which is not exactly what logicians are familiar with.

1.1.3 Operations on categories

There are many ways of forming new categories out of old ones. Two basic operations are the following:

Dualization: If $\mathcal{C}$ is a category, so is its *dual* $\mathcal{C}^{op}$, with the same objects, but whose arrows are reversed (i.e. interchange *dom* and *cod*). Clearly $(\mathcal{C}^{op})^{op} = \mathcal{C}$ and reversing all arrows changes commutative diagrams in $\mathcal{C}$ to commutative diagrams in $\mathcal{C}^{op}$. In other words, we have the following bijective correspondence:

$$\frac{f\colon A \to B \text{ in } \mathcal{C}}{f\colon B \to A \text{ in } \mathcal{C}^{op}}$$

Products: If $\mathcal{C}$, $\mathcal{D}$ are categories, so is their cartesian product $\mathcal{C} \times \mathcal{D}$, with the obvious structure: objects are pairs of objects, arrows are pairs of arrows, composition and identities are defined componentwise.

Finally, we end with a useful notion:

Definition 1.4 A *subcategory* $\mathcal{C}$ of a category $\mathcal{B}$ is any category whose class of objects and arrows are contained in those of $\mathcal{B}$, respectively, and which is closed under the "operations" in $\mathcal{B}$ of *domain, codomain, composition, and identity.* $\mathcal{C}$ is a *full* subcategory of $\mathcal{B}$ if for all objects $A, B \in \mathcal{C}$, $\mathcal{C}(A, B) = \mathcal{B}(A, B)$. In other words, a full subcategory is determined by just restricting the class of objects, since the arrows are predetermined by $\mathcal{B}$.

For example, we often consider small subcategories whose objects are of "bounded size" within the large examples above: e.g. the full subcategories of (i) finite sets and (ii) finite dimensional vector spaces, and more generally, for a fixed infinite cardinal κ, sets (resp. vector spaces) of cardinality (resp. dimension) bounded by κ.

1.1.4 Functors

If $\mathcal{C}, \mathcal{D}$ are categories, a *functor* $F : \mathcal{C} \to \mathcal{D}$ is a pair $F = (F_{ob}, F_{arr})$, where $F_{ob} : Objects(\mathcal{C}) \to Objects(\mathcal{D})$, and similarly for arrows, satisfying the following (we omit the subscripts ob, arr): if $A \xrightarrow{f} B$ then $FA \xrightarrow{F(f)} FB$ with: $F(gf) = F(g)F(f)$ and $F(id_A) = id_{FA}$. A functor $F : \mathcal{C}^{op} \to \mathcal{D}$ is sometimes called *contravariant*. From the definition of the opposite category, a contravariant functor F preserves the identity arrows, but reverses composition: $F(gf) = F(f)F(g)$.

Examples 1.5

1. *Forgetful (also called Underlying) Functors.* These include forgetful functors U : **Posets** $\to$ **Set**, U : **Top** $\to$ **Set**, U : **Alg** $\to$ **Set** (where **Alg** is any category of universal algebras and homomorphisms between them). U maps objects and arrows to their underlying set (omitting the other structure).

   Sometimes, one only forgets part of the structure, e.g. there are several forgetful functors on **TopGrp**; we have U_1 : **TopGrp** $\to$ **Grp** which maps a topological group and a continuous group homomorphism to its underlying group (and the underlying group homomorphism), and similarly there is U_2 : **TopGrp** $\to$ **Top**.
2. *Representable (or Hom) Functors.* If $A \in \mathcal{C}$, we have the dual co- and contravariant homs:

   1. *Covariant hom* : $\mathcal{C}(A, -) : \mathcal{C} \to$ **Set** given by:

   $$\begin{array}{rcl} B & \mapsto & \mathcal{C}(A, B) \\ B \xrightarrow{f} C & \mapsto & \mathcal{C}(A, f) : \mathcal{C}(A, B) \to \mathcal{C}(A, C) \\ & & \text{where } \mathcal{C}(A, f)(g) = fg. \end{array}$$

   2. *Contravariant hom*: $\mathcal{C}(-, A) : \mathcal{C}^{op} \to$ **Set** given by:

   $$\begin{array}{rcl} B & \mapsto & \mathcal{C}(B, A) \\ B \xrightarrow{f} C & \mapsto & \mathcal{C}(f, A) : \mathcal{C}(C, A) \to \mathcal{C}(B, A) \\ & & \text{where } \mathcal{C}(A, f)(g) = gf. \end{array}$$

3. *Powerset Functors.* There are co- and contravariant powerset functors on **Set**: