Deriving Category Theory from Type Theory

Roy L. Crole

March 1993

Abstract

This work expounds the notion that (structured) categories are syntax free presentations of type theories, and shows some of the ideas involved in deriving categorical semantics for given type theories. It is intended for someone who has some knowledge of category theory and type theory, but who does not fully understand some of the intimate connections between the two topics. We begin by showing how the concept of a category can be derived from some simple and primitive mechanisms of monadic type theory. We then show how the notion of a category with finite products can model the most fundamental syntactical constructions of (algebraic) type theory. The idea of naturality is shown to capture, in a syntax free manner, the notion of substitution, and therefore provides a syntax free coding of a multiplicity of type theoretical constructs. Using these ideas we give a direct derivation of a cartesian closed category as a very general model of simply typed λ-calculus with binary products and a unit type. This article provides a new presentation of some old ideas. It is intended to be a tutorial paper aimed at audiences interested in elementary categorical type theory. Further details can be found in [Cro93].

1 Introduction

Typed λ-calculus is a subject very well understood by today's computer scientists, being an embodiment of the basic principles which underlie functional programming. In fact the foundations of λ-calculus were laid many years ago by logicians, but this will not concern us here. The interested reader might care to consult [Chu40] and [Chu41]. Many computer scientists are also aware that the λ-calculus has a formal connection with the notion of a cartesian closed category; this idea is due to Lambek [Lam80]. However, experience shows that this connection is often only fully appreciated by those working in very theoretical areas of computer science. We hope that this article will go some way towards bridging this gap, as well as high-lighting some of the more fundamental connections between category theory and type theory. The techniques described here can be used to derive categorical semantics for quite general type theories and similar logical systems.

2 Deriving a Category from Monadic Type Theory

Readers not familiar with type theory will find that [NPS90] is an excellent reference. There are a number of textbooks now available which cover basic category theory; see for example [BW90] or [Pie91].

Before we embark on our derivation of a category, let us review the traditional set-theoretic semantics which can be given to elementary type theory. The "elementary" type theory which we shall consider is sometime also called *algebraic* type theory. In order to define such a type theory, we shall assume that we are given a *signature Sg* which consists of some *types* and some *function symbols*. Each function symbol f has an *arity* which is a natural number a, and a *sorting* which is a list of $a+1$ types, usually written $f\colon \alpha_1, \ldots, \alpha_a \to \beta$, or just $f\colon \alpha$ in the case that f has arity 0. In the latter case we usually call f a *constant* function symbol. For each type α take a countably infinite set $Var^\alpha \stackrel{\text{def}}{=} \{x_1^\alpha, x_2^\alpha, \ldots\}$ of variables (we assume that such sets of variables are disjoint for different types α) and define the terms of the

algebraic type theory by the rules

$$\frac{x \in Var^{\alpha}}{x\colon \alpha} \qquad \frac{}{k\colon \alpha} \qquad \frac{M_1\colon \alpha_1, \ldots, M_a\colon \alpha_a}{f(M_1, \ldots, M_a)\colon \beta}$$

where $k\colon \alpha$ and $f\colon \alpha_1, \ldots, \alpha_a \to \beta$ are function symbols. A set-theoretic semantics would then be defined by giving a set $[\![\alpha]\!]$ for each type α, an element $[\![k]\!] \in [\![\alpha]\!]$ for each constant $k\colon \alpha$ and a function of the form $[\![f]\!]\colon [\![\alpha_1]\!] \times \ldots \times [\![\alpha_n]\!] \to [\![\beta]\!]$ for each function symbol $f\colon \alpha_1, \ldots, \alpha_n \to \beta$. An environment for the type theory is essentially a function which assigns a meaning to each of the variables. More precisely an environment ρ is a function

$$\rho\colon \bigcup\{\, Var^{\alpha} \mid \alpha \text{ is a type}\} \to \bigcup\{[\![\alpha]\!] \mid \ \alpha \text{ is a type}\}$$

where $\rho(x^{\alpha}) \in [\![\alpha]\!]$ for $x^{\alpha} \in Var^{\alpha}$. Given such an environment ρ, we assign a meaning to the terms by setting $[\![x^{\alpha}]\!] \stackrel{\text{def}}{=} \rho(x^{\alpha})$ and

$$[\![f(M_1, \ldots, M_n)]\!] \stackrel{\text{def}}{=} [\![f]\!]([\![M_1]\!], \ldots, [\![M_n]\!]),$$

so in general if $M\colon \alpha$ then $[\![M]\!] \in [\![\alpha]\!]$.

There are at least two problems with this approach. The first is that each of the types must be interpreted by non-empty sets and that we must restrict to a set of types. For if any of the $[\![\alpha]\!]$ were empty, or we had a proper class of types, then we would not be able to define the function ρ. The second problem is that the semantics is specified in terms of elements of sets. Of course, we might consider replacing the sets by some mathematical object which has an underlying set-theoretic structure. But we would be in a much stronger position if we could interpret our syntax in a more general setting, such as a category. Categories are very general mathematical structures, and category theory is a powerful organisational tool, but are categories suitable worlds in which to interpret syntactical systems such as algebraic theories? We shall see that categories are in fact tailor made mathematical universes for interpreting type theories.

Let us consider the first problem high-lighted above. We can think of a term as giving rise to a function with the variables of the term taking input data and the effect of the term on such data as the output. With this perspective, we can give a meaning to the types by assigning sets to them, and a meaning to the terms as functions. With due regard for this interpretation of the

syntax, we should abandon type tagged variables and present the syntax using terms which have an associated environment of declared variables. We do this using the following three rules

$$\frac{}{\Gamma, x\colon\alpha, \Gamma' \vdash x\colon\alpha} \qquad \frac{}{\Gamma \vdash k\colon\alpha}\ (k\colon\alpha)$$

$$\frac{\Gamma \vdash M_1\colon\alpha_1 \quad \ldots \quad \Gamma \vdash M_a\colon\alpha_a}{\Gamma \vdash f(M_1, \ldots, M_a)\colon\alpha}\ (f\colon\alpha_1, \ldots, \alpha_a \to \alpha)$$

along with the usual structural and substitution rules. Here, each Γ is a *context* of (variable, type) pairs in which the *variables are distinct*, and we refer to a judgement of the form $\Gamma \vdash M\colon\alpha$ as a term-in-context. Thus a term-in-context $x_1\colon\alpha_1, \ldots, x_n\colon\alpha_n \vdash M\colon\beta$ will be modelled by a function $f\colon A_1 \times \ldots \times A_n \to B$, where we think of the n input variables of M being modelled by an element of the cartesian product $A_1 \times \ldots \times A_n$. We may now (if we wish) interpret types by empty sets: we no longer need to define the environment ρ.

Now let us think about terms M with exactly one variable; thus $x\colon\alpha \vdash M\colon\beta$, for example, where x occurs in M. We tacitly assume that we are only dealing with function symbols of arity 1 and will refer to this limited type theory as *monadic*. We shall try to generalise our set-theoretic model using as few assumptions as possible. Suppose we model α and β by "objects" A and B of which we make no assumptions. So we have in mind that A and B are sets, but are not held to this. For the time being the word "object" is just some arbitrary reference. A function is a form of relation; so let us model $x\colon\alpha \vdash M\colon\beta$ as a "relation" m between A and B of which we make no assumptions; we write $A \xrightarrow{m} B$ for this. While we are not at this stage being mathematically precise, we will make things more watertight later on. For the time being we are just trying to set up a very general framework in which to model our monadic type theory.

Now let us think about how our syntactic term language is built up. The crucial point is that terms are built up by *substitution*, in the sense that a raw term $f(M)$ is precisely $f(x)[M/x]$. We now think about the process of substitution in general. Suppose that we have terms-in-context $x\colon\alpha \vdash M\colon\beta$ and $y\colon\beta \vdash N\colon\gamma$. One can prove by induction on the structure of N that there is a derived term-in-context $x\colon\alpha \vdash N[M/y]\colon\gamma$—so how should we model this? Let us just say for the moment that whatever models this term depends on

how we model $x\colon\alpha \vdash M\colon\beta$ and $y\colon\beta \vdash N\colon\gamma$. We can write this as

$$\frac{[\![x\colon\alpha \vdash M\colon\beta]\!] \stackrel{\text{def}}{=} A \xrightarrow{m} B \qquad [\![y\colon\beta \vdash N\colon\gamma]\!] \stackrel{\text{def}}{=} B \xrightarrow{n} C}{[\![x\colon\alpha \vdash N[M/y]\colon\gamma]\!] \stackrel{\text{def}}{=} A \xrightarrow{\Box(n,m)} C}$$

where one reads $[\![\xi]\!] \stackrel{\text{def}}{=} \zeta$ to mean that ξ is modelled by ζ, and $\Box(n,m)$ is some relation depending on n and m. What about the order of substitution of terms for term variables? Let $z\colon\gamma \vdash L\colon\delta$ be a further term-in-context (where we tacitly assume that x, y and z are distinct variables). Well, both of the terms-in-context

$$x\colon\alpha \vdash (L[N/z])[M/y]\colon\delta \qquad \text{and} \qquad x\colon\alpha \vdash L[N[M/y]/z]\colon\delta$$

are syntactically the same (caution - why is this?). So the relations which model them ought to be the same too, namely $A \xrightarrow{\Box(\Box(l,n),m)} C$ and $A \xrightarrow{\Box(l,\Box(n,m))} C$, and we will write $\Box(\Box(l,n),m) = \Box(l,\Box(n,m))$ to indicate this.

Now we shall take a step backward and think about how terms-in-context with at most one variable are formed. We will have to model $x\colon\alpha \vdash x\colon\alpha$ as a relation $A \xrightarrow{\star_A} A$. If we think about how the substitution of terms for variables is modelled, then we deduce that if $E \xrightarrow{e} A$ and $A \xrightarrow{m} B$ then $\Box(\star_A, e) = e$ and $\Box(m, \star_A) = m$. This follows from the observation that $E = x[E/x]$ and $M = M[x/x]$. A term-in-context $x\colon\alpha \vdash f(M)\colon\beta'$, where $f\colon\beta \to \beta'$ is a function symbol, is the term-in-context $x\colon\alpha \vdash f(y')[M/y']\colon\beta'$, and so will be modelled by the relation $A \xrightarrow{\Box(r,m)} B'$, where B' models β' and we *specify* that the term-in-context $x\colon\beta \vdash f(x)\colon\beta'$ is modelled by $B \xrightarrow{r} B'$. The specification of r is just a reflection of the fact that we already have some intended meaning for the function symbol f.

Now we summarise our deductions, writing $n \circ m$ for $\Box(n,m)$ and id_A for $\star_A$:

- Types are interpreted by "objects," say A, $B \ldots$
- Terms-in-context are interpreted by "relations," say $A \xrightarrow{m} B \ldots$
- For each object A there is a relation id_A.
- Given relations $A \xrightarrow{m} B$ and $B \xrightarrow{n} C$, there is a relation $A \xrightarrow{n \circ m} C$.

Interpreting Substitution by Composition

Here, $x{:}\alpha \vdash M{:}\beta$ and $x{:}\gamma \vdash M'{:}\alpha$ are terms of an algebraic type theory, and $[\![x{:}\alpha \vdash M{:}\beta]\!]$ and $[\![x{:}\gamma \vdash M'{:}\alpha]\!]$ are morphisms of a category $\mathcal{C}$ interpreting the proved terms.

$$\underbrace{[\![x{:}\gamma \vdash M[M'/x]{:}\beta]\!]}_{\text{Interpretation of } M[M'/x] \text{ in a category } \mathcal{C}} = \underbrace{[\![x{:}\alpha \vdash M{:}\beta]\!] \circ [\![x{:}\gamma \vdash M'{:}\alpha]\!]}_{\text{Composition of the interpretations of } M \text{ and } M'}$$

Figure 1: A Basic Principle of Categorical Type Theory

- Given relations $E \xrightarrow{e} A$ and $A \xrightarrow{m} B$, then we have $id_A \circ e = e$ and $m \circ id_A = m$.
- For any $A \xrightarrow{m} B$, $B \xrightarrow{n} C$ and $C \xrightarrow{l} D$, we have $l \circ (n \circ m) = (l \circ n) \circ m$.

The above summary amounts to the specification of a category! Thus we have deduced, subject to certain primeval assumptions about how to model function symbols and substitution, that we can interpret algebraic type theory in which at most one variable appears in a term, in an arbitrary *category*. In such a category, *the substitution of raw terms for variables will be interpreted by composition of morphisms*; because of the importance of this idea, we give a special summary: see Figure 1.

3 Categories for Algebraic Type Theory

In the previous section we deduced that monadic type theory could be soundly interpreted in an arbitrary category. What happens when we consider full algebraic type theory? In the original model based around sets and functions, a term-in-context was modelled by a function which took n arguments. There is nothing in the formal rules for giving an algebraic type theory which will allow us to deduce how to model contexts in the same way

that we deduced a framework for modelling substitution. The nearest categorical structure which mimics a set-theoretic function with n arguments is a morphism in a category induced by a finite product.[1] Here we assume that the reader has some knowledge of categories with finite products. The definition can be found in any basic book on category theory, such as [Mac71] or [BW90]. Using such a category we can give a semantics to algebraic type theory as follows:

Let $\mathcal{C}$ be a category with finite products and let Sg be an algebraic signature. A *structure*, $\mathbf{M}$, in $\mathcal{C}$ for Sg is specified by giving

• for every type α of Sg an object $[\![\alpha]\!]$ of $\mathcal{C}$,

• for every constant function symbol $k\colon\alpha$, a morphism $[\![k]\!]\colon 1 \to [\![\alpha]\!]$ where 1 is the terminal object of $\mathcal{C}$, and

• for every function symbol $f\colon\alpha_1,\ldots,\alpha_n \to \beta$ of Sg a morphism

$$[\![f]\!]\colon [\![\alpha_1]\!] \times \ldots \times [\![\alpha_n]\!] \to [\![\beta]\!].$$

Given a context $\Gamma = [x_1\colon\alpha_1,\ldots,x_n\colon\alpha_n]$ we set

$$[\![\Gamma]\!] \stackrel{\text{def}}{=} [\![\alpha_1]\!] \times \ldots \times [\![\alpha_n]\!].$$

Then for every proved term $\Gamma \vdash M\colon\alpha$ we shall specify a morphism

$$[\![\Gamma \vdash M\colon\alpha]\!]\colon [\![\Gamma]\!] \to [\![\alpha]\!]$$

in $\mathcal{C}$. The semantics of proved terms is specified inductively using the rules for generating terms-in-context:

- $[\![\Gamma, x\colon\alpha, \Gamma' \vdash x\colon\alpha]\!] \stackrel{\text{def}}{=} \pi\colon [\![\Gamma]\!] \times [\![\alpha]\!] \times [\![\Gamma']\!] \to [\![\alpha]\!]$ where π is a projection,
- $[\![\Gamma \vdash k\colon\alpha]\!] \stackrel{\text{def}}{=} [\![k]\!] \circ !\colon [\![\Gamma]\!] \to 1 \to [\![\alpha]\!]$, and
- $[\![\Gamma \vdash f(M_1,\ldots,M_a)]\!] \stackrel{\text{def}}{=} [\![f]\!] \circ \langle [\![M_1]\!],\ldots,[\![M_a]\!]\rangle\colon [\![\Gamma]\!] \to (\Pi_1^a [\![\alpha_i]\!]) \to [\![\alpha]\!]$.

The interpretation given to the terms-in-context models our intended meaning of the syntax. For example, the meaning assigned to a term of the form

[1] Apologies to linear categorical logicians; I am just considering one possibility in this article.

$f(M_1, \ldots, M_a)$ comes as a direct generalisation of the idea that substitution is modelled by composition of morphisms where in this case the finite product structure is modelling the n inputs to f as contrasted to the single input in the previous section.

4 Natural Transformations derived from Introduction Rules

Let us suppose that we are now working with a general type theory in which the forms of judgement still take the simple form $\Gamma \vdash M\colon \alpha$. As in Section 1, types will be modelled by objects in a category; and as a simplifying assumption we shall take all categories to be locally small, that is, the collection of morphisms $A \to B$ (where A and B are objects) can be indexed by a set. This set will be denoted by $\mathcal{C}(A, B)$.

As discussed in Section 2, the interpretation of a term-in-context $\Gamma \vdash M\colon \alpha$ is given by a morphism $[\![\Gamma \vdash M\colon \alpha]\!]\colon [\![\Gamma]\!] \to [\![\alpha]\!]$ in $\mathcal{C}$. At the moment we do not know how to define such an interpretation, but by looking at how to soundly interpret the terms and equations of the type theory we will deduce how to do this.

Let us think about the rules of formation of term-in-contexts in general, assuming just one hypothesis. A typical rule looks like

$$\frac{\Gamma \vdash M\colon \alpha}{\Gamma \vdash \mathsf{R}(M)\colon \beta} \quad (\mathsf{R})$$

where $\mathsf{R}(M)$ is a new raw term depending on M. Suppose that $m \stackrel{\text{def}}{=} [\![\Gamma \vdash M\colon \alpha]\!]$ which is an element of $\mathcal{C}([\![\Gamma]\!], [\![\alpha]\!])$. How do we model

$$[\![\Gamma \vdash \mathsf{R}(M)\colon \beta]\!] \in \mathcal{C}([\![\Gamma]\!], [\![\beta]\!])?$$

All we can say at the moment is that this latter morphism will depend on m, and we can model this idea by having a function

$$\Phi_{[\![\alpha]\!],[\![\beta]\!],[\![\Gamma]\!]} : \mathcal{C}([\![\Gamma]\!], [\![\alpha]\!]) \longrightarrow \mathcal{C}([\![\Gamma]\!], [\![\beta]\!])$$

and setting $[\![\Gamma \vdash \mathsf{R}(M)\colon \beta]\!] \stackrel{\text{def}}{=} \Phi_{[\![\alpha]\!],[\![\beta]\!],[\![\Gamma]\!]}(m)$. Now think about how the raw terms are formed. The crucial point is that new raw terms are formed from

old raw terms by substitution; and we can easily see that a derived rule for our type theory is

$$\frac{x\colon\gamma \vdash M\colon\alpha \quad y\colon\gamma' \vdash N\colon\gamma}{y\colon\gamma' \vdash M[N/x]\colon\alpha} \quad \text{(Sub)}$$

Suppose that $x\colon\gamma \vdash M\colon\alpha$ and $y\colon\gamma' \vdash N\colon\gamma$ are any two given term-in-contexts. Using our basic assumption that substitution is modelled by composition of morphisms, if $m \stackrel{\text{def}}{=} [\![x\colon\gamma \vdash M\colon\alpha]\!]$ and $n \stackrel{\text{def}}{=} [\![y\colon\gamma' \vdash N\colon\gamma]\!]$ then we must have that $[\![y\colon\gamma' \vdash M[N/x]\colon\alpha]\!] = m \circ n$. Applying each of (Sub) and (R) in turn, we deduce that there are term-in-contexts

$$y\colon\gamma' \vdash \mathsf{R}(M)[N/x]\colon\beta \qquad \text{and} \qquad y\colon\gamma' \vdash \mathsf{R}(M[N/x])\colon\beta.$$

However, both of the above raw terms should be syntactically identical (by the definition of substitution), and therefore the categorical interpretations should be the same, that is

$$\Phi_{[\![\alpha]\!],[\![\beta]\!],[\![\gamma]\!]}(m) \circ n = \Phi_{[\![\alpha]\!],[\![\beta]\!],[\![\gamma']\!]}(m \circ n). \qquad (*)$$

The reader may notice that $(*)$ looks similar to a naturality condition; in fact we can be certain that it will hold if we demand the following. For every object A and B of $\mathcal{C}$ there is a natural transformation

$$\Phi_{A,B} : \mathcal{C}(-, A) \longrightarrow \mathcal{C}(-, B) : \mathcal{C} \longrightarrow \mathcal{S}et.$$

We can summarise these thoughts in the slogan:

Categorical Modelling of Substitution

The sound categorical interpretation of the notion of substitution of syntax amounts to requiring that certain naturality conditions hold in the categorical model.

5 How to Model Binary Products and the Unit Type

The categorical semantics given to algebraic theories was strongly motivated by traditional set-theoretic semantics. We shall now show how to derive a

semantics for the syntax of binary products and unit. Some readers will know that this syntax can be modelled in categories with finite products. However, we shall present a uniform analysis of the syntax and rules of the unit type and binary product type to discover what, in categorical terms, is the most general interpretation.

Let us recall that the raw syntax of binary products, in a setting which subsumes algebraic type theory. First we specify a collection of *ground types*. The *types* are then given by the grammar $\alpha ::= \gamma \mid \mathit{unit} \mid \alpha \times \alpha$ where γ is any ground type. The *raw terms* are given by

$$M ::= x \mid f(M, \ldots, M) \mid \langle\rangle \mid \mathsf{Fst}(M) \mid \mathsf{Snd}(M) \mid \langle M, N\rangle.$$

The interpretation of a type $\alpha \times \beta$ will depend on the interpretations of α and β. The term-in-contexts will be interpreted by morphisms in a category, and the assumption that the equations-in-context are soundly interpreted will then determine equations which hold between morphisms: this will become clearer later on. In the cases of binary product types, we shall see that the equations between morphisms will determine the objects which model the types up to isomorphism. Finally, recall the basic assumption that all of our syntax is interpreted in a category with (at least) finite products: products are used to model the list of types which appear in contexts.

First we consider the types of Sg. We have to give an object $[\![\gamma]\!]$ of $\mathcal{C}$ to interpret each of the ground types γ, and an object $[\![\mathit{unit}]\!]$ to interpret *unit*—for the moment we cannot say anything more specific about $[\![\mathit{unit}]\!]$. We will assume that the interpretation of binary product types $\alpha \times \beta$ depends on the interpretations of α and β. So there should be operation in $\mathcal{C}$ which gives an object $A \square B$ for all objects A and B so that we can define

$$[\![\alpha \times \beta]\!] \stackrel{\text{def}}{=} [\![\alpha]\!] \square [\![\beta]\!].$$

Having done this, we can now choose a morphism $[\![f]\!] \colon [\![\alpha_1]\!] \times \ldots \times [\![\alpha_n]\!] \to [\![\beta]\!]$ in $\mathcal{C}$ for each function symbol $f \colon \alpha_1 \ldots \alpha_n \to \beta$ of Sg. We will obviously have an intended interpretation of the function symbol f and the choice of the morphism $[\![f]\!]$ will reflect this.

Let us now think about specific types and terms.

First we deal with the type *unit*. The rules for the unit type are

$$\frac{}{\Gamma \vdash \langle\rangle \colon \mathit{unit}} \qquad \frac{\Gamma \vdash M \colon \mathit{unit}}{\Gamma \vdash M = \langle\rangle \colon \mathit{unit}}$$

To interpret the first rule there must always be a morphism

$$u_0 \stackrel{\text{def}}{=} [\![\Gamma \vdash \langle\rangle : \mathit{unit}]\!] : [\![\Gamma]\!] \to [\![\mathit{unit}]\!].$$

To soundly interpret the equation-in-context, whenever there is a morphism $m \stackrel{\text{def}}{=} [\![\Gamma \vdash M : \mathit{unit}]\!]$ in $\mathcal{C}$, we must have $m = u_0$. All this amounts to saying that for every object A of $\mathcal{C}$, there must exist a unique morphism $! : A \to [\![\mathit{unit}]\!]$, that is *up to isomorphism* $[\![\mathit{unit}]\!]$ *is a terminal object* 1 *of* $\mathcal{C}$.

Recall that the rule for introducing product terms is

$$\frac{\Gamma \vdash M : \alpha \quad \Gamma \vdash N : \beta}{\Gamma \vdash \langle M, N\rangle : \alpha \times \beta}$$

In order to soundly interpret this rule we shall need a natural transformation

$$\Phi_{A,B} : \mathcal{C}(-, A) \times \mathcal{C}(-, B) \longrightarrow \mathcal{C}(-, A \square B)$$

for all objects A and B of $\mathcal{C}$. Now let $m : C \to A$ and $n : C \to B$ be morphisms of $\mathcal{C}$. Applying naturality in C at the morphism $\langle m, n\rangle : C \to A \times B$ we deduce

$$(\Phi_{A,B})_C(\pi_A\langle m, n\rangle, \pi_B\langle m, n\rangle) = (\Phi_{A,B})_{A\square B}(\pi_A, \pi_B) \circ \langle m, n\rangle,$$

that is $(\Phi_{A,B})_C(m, n) = (\Phi_{A,B})_{A\square B}(\pi_A, \pi_B) \circ \langle m, n\rangle$. Now let us define the morphism $q_{A,B} : A \times B \to A \square B$ to be $(\Phi_{A,B})_{A\times B}(\pi_A, \pi_B)$. Then we can make the definition

$$[\![\Gamma \vdash \langle M, N\rangle : A \square B]\!] \stackrel{\text{def}}{=}$$
$$[\![\Gamma]\!] \xrightarrow{\langle [\![\Gamma \vdash M : \alpha]\!], [\![\Gamma \vdash N : \beta]\!]\rangle} [\![\alpha]\!] \times [\![\beta]\!] \xrightarrow{q_{[\![\alpha]\!],[\![\beta]\!]}} [\![\alpha]\!] \square [\![\beta]\!].$$

Recall one of the rules for eliminating product types

$$\frac{\Gamma \vdash P : \alpha \times \beta}{\Gamma \vdash \mathsf{Fst}(M) : \beta}$$

Arguing as above, to model this rule we shall need (for each A and B) a natural transformation $\Phi_{A,B} : \mathcal{C}(-, A \square B) \longrightarrow \mathcal{C}(-, A)$. So for any object C, and morphisms $m : C \to A \square B$ and $\theta : C' \to C$ of $\mathcal{C}$, we have

$$(\Phi_{A,B})_C(m) \circ \theta = (\Phi_{A,B})_{C'}(m \circ \theta).$$

By considering the instance of this equation when $m = id_{A\Box B}$, we see that in general we have $(\Phi_{A,B})_C(\theta) = p_{A,B} \circ \theta$ where $p_{A,B} \stackrel{\text{def}}{=} (\Phi_{A,B})_{A\Box B}(id_{A\Box B})$. So now we can define

$$[\![\Gamma \vdash \mathsf{Fst}(P)\colon \alpha]\!] \stackrel{\text{def}}{=} [\![\Gamma]\!] \xrightarrow{[\![\Gamma \vdash P\colon \alpha \times \beta]\!]} [\![\alpha]\!]\Box[\![\beta]\!] \xrightarrow{p_{[\![\alpha]\!],[\![\beta]\!]}} [\![\alpha]\!].$$

Of course we can deduce a semantics for term-in-contexts of the form $\Gamma \vdash \mathsf{Snd}(P)\colon \beta$ in much the same way, involving a morphism $p'_{A,B}\colon A\Box B \to B$. Our last task is to see what information we obtain by soundly interpreting the equations-in-context for product types. These are

$$\frac{\Gamma \vdash M\colon \alpha \quad \Gamma \vdash N\colon \beta}{\Gamma \vdash \mathsf{Fst}(\langle M, N\rangle) = M\colon \alpha} \quad (1) \qquad \frac{\Gamma \vdash M\colon \alpha \quad \Gamma \vdash N\colon \beta}{\Gamma \vdash \mathsf{Snd}(\langle M, N\rangle) = N\colon \beta} \quad (2)$$

$$\frac{\Gamma \vdash P\colon \alpha \times \beta}{\Gamma \vdash \langle \mathsf{Fst}(P), \mathsf{Snd}(P)\rangle = P\colon \alpha \times \beta} \quad (3)$$

If we put $h \stackrel{\text{def}}{=} [\![\Gamma \vdash P\colon \alpha \times \beta]\!]\colon C \to A\Box B$, $m \stackrel{\text{def}}{=} [\![\Gamma \vdash M\colon \alpha]\!]\colon C \to A$ and $n \stackrel{\text{def}}{=} [\![\Gamma \vdash N\colon \beta]\!]\colon C \to B$, and demand that our categorical interpretation satisfies the equations-in-context, this forces

$$\begin{aligned} p_{A,B} \circ q_{A,B} \circ \langle m, n\rangle &= m && (1)\\ p'_{A,B} \circ q_{A,B} \circ \langle m, n\rangle &= n && (2)\\ q_{A,B} \circ \langle p_{A,B} \circ h, p'_{A,B} \circ h\rangle &= h. && (3) \end{aligned}$$

At last we are done, because these equations imply that, up to isomorphism, $A\Box B$ and $A \times B$ are the same. Thus we may *soundly interpret binary product types by binary categorical product.*

6 How to Model Function Types

Now we shall enrich our type theory so that it contains not only product types but also function types $\alpha \Rightarrow \beta$. The raw terms are the same as in Section 5 but also include the terms $\lambda x\colon \alpha.M$ and $M\,N$. To model the type $\alpha \Rightarrow \beta$ we shall need an object $A\Diamond B$ for all pairs of objects A and B of $\mathcal{C}$.

To soundly interpret the introduction rule

$$\frac{\Gamma, x{:}\,\alpha \vdash F{:}\,\beta}{\Gamma \vdash \lambda x{:}\,\alpha.F{:}\,\alpha \Rightarrow \beta}$$

we shall need (for every object A and B) a natural transformation

$$\Phi_{A,B}{:}\,\mathcal{C}(- \times A, B) \longrightarrow \mathcal{C}(-, A \lozenge B),$$

and we can then define

$$[\![\Gamma \vdash \lambda x{:}\,\alpha.F{:}\,\alpha \Rightarrow \beta]\!] \stackrel{\text{def}}{=} (\Phi_{[\![\alpha]\!],[\![\beta]\!]})_{[\![\Gamma]\!]}([\![\Gamma, x{:}\,\alpha \vdash F{:}\,\beta]\!]) : [\![\Gamma]\!] \to ([\![\alpha]\!] \lozenge [\![\beta]\!]).$$

To soundly interpret the elimination rule

$$\frac{\Gamma \vdash M{:}\,\alpha \Rightarrow \beta \quad \Gamma \vdash N{:}\,\alpha}{\Gamma \vdash MN{:}\,\beta}$$

we shall need a natural transformation

$$\Psi_{A,B}{:}\,\mathcal{C}(-, A \lozenge B) \times \mathcal{C}(-, A) \longrightarrow \mathcal{C}(-, B)$$

for all objects A and B of $\mathcal{C}$. Given any two morphisms $m{:}\,C \to A \lozenge B$ and $n{:}\,C \to A$ and applying naturality, we have

$$(\Psi_{A,B})_C(m, n) = (\Psi_{A,B})_{(A \lozenge B) \times A}(\pi, \pi') \circ \langle m, n \rangle$$

where $\pi{:}\,(A \lozenge B) \times A \to A \lozenge B$ and $\pi'{:}\,(A \lozenge B) \times A \to A$. So if we define the morphism $ev_{A,B} \stackrel{\text{def}}{=} (\Psi_{A,B})_{(A \lozenge B) \times A}(\pi, \pi')$, we can make the definition

$$[\![\Gamma \vdash MN{:}\,\beta]\!] \stackrel{\text{def}}{=}$$
$$[\![\Gamma]\!] \xrightarrow{\langle [\![\Gamma \vdash M{:}\,\alpha \Rightarrow \beta]\!], [\![\Gamma \vdash N{:}\,\alpha]\!] \rangle} ([\![\alpha]\!] \lozenge [\![\beta]\!]) \times [\![\alpha]\!] \xrightarrow{ev_{[\![\alpha]\!],[\![\beta]\!]}} [\![\beta]\!].$$

The equations-in-context for the function type are

$$\frac{\Gamma, x{:}\,\alpha \vdash F{:}\,\beta \quad \Gamma \vdash M{:}\,\alpha}{\Gamma \vdash (\lambda x{:}\,\alpha.F)\, M = F[M/x]{:}\,\beta} \qquad (4)$$

$$\frac{\Gamma \vdash N{:}\,\alpha \Rightarrow \beta}{\Gamma \vdash \lambda x{:}\,\alpha.(Mx) = M{:}\,\alpha \Rightarrow \beta} \qquad (5)$$

If our categorical interpretation is to satisfy the equation-in-context (4), then we must have $ev_{A,B}\langle(\Phi_{A,B})_C(f), m\rangle = f\langle id, m\rangle$ for all morphisms $f\colon C\times A \to B$ and $m\colon C \to A$. Using the naturality of $\Phi_{A,B}$ we can show that this equation holds just in case

$$ev_{A,B}((\Phi_{A,B})_C(f) \times id) = f. \qquad (*)$$

Satisfaction of (5) requires that $(\Phi_{A,B})_C(ev_{A,B}(m \times id_A)) = m$ for every morphism $m\colon C \to A\mathbin{\Diamond}B$ and from the naturality of $\Phi_{A,B}$ this holds just in case

$$(\Phi_{A,B})_{A\Diamond B}(ev_{A,B}) = id. \qquad (\dagger)$$

Now we come to the crucial point. If we define a natural transformation

$$\theta\colon \mathcal{C}(- \times A, B) \longrightarrow \mathcal{C}(-, A\mathbin{\Diamond}B)$$

by setting $\theta_C(f) \stackrel{\text{def}}{=} ev_{A,B}\circ(f\times id)$ the equations $(*)$ and $(\dagger)$ imply that θ_C is a natural bijection. *Thus, up to isomorphism in the category $\mathcal{C}$, the object $A\mathbin{\Diamond}B$ is exactly the exponential $A \Rightarrow B$ and of course $ev_{A,B}\colon (A\mathbin{\Diamond}B)\times A \to B$ is the evaluation morphism, and such categorical structure will soundly interpret function types.*

7 Conclusion and Acknowledgements

By considering some of the underlying principles of set-theoretic models of equational type theory, we have set up a very general framework in which such type theories can be modelled, and shown that such a framework corresponds to the notion of a category. We have also shown that introduction rules can be soundly modelled by suitable natural transformations. This very general framework was then used to find a minimal categorical structure for soundly interpreting a specific elementary type theory, namely simply typed λ-calculus with finite products. The methodology of manipulating naturality and soundness equations to compute general categorical structures can be applied to less well known type theories. Examples can be found in [Cro91].

I would like to thank Andrew Pitts for discussions which threw light on my understanding of the way category-theoretic ideas capture slickly the essence of intricate syntactic constructions. I would also like to thank the Science and Engineering Research Council for providing funding in the form of a Research Fellowship.

References

[BW90] M. Barr and C. Wells. *Category Theory for Computing Science*. International Series in Computer Science. Prentice Hall, 1990.

[Chu40] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[Chu41] A. Church. *The Calculi of Lambda Conversion*. Princeton University Press, 1941.

[Cro91] R. L. Crole. *Programming Metalogics with a Fixpoint Type*. PhD thesis, Computer Laboratory, University of Cambridge, 1991.

[Cro93] R. L. Crole. *Categories for Types*. Cambridge Mathematical Textbooks. Cambridge University Press, 1993. xvii+335 pages, ISBN 0521450926HB, 0521457017PB.

[Lam80] J. Lambek. From λ-calculus to cartesian closed categories. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1980.

[Mac71] S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer-Verlag, 1971.

[NPS90] B. Nordström, K. Petersson, and J.M. Smith. *Programming in Martin-Löf's Type Theory*, volume 7 of *Monographs on Computer Science*. Oxford University Press, 1990.

[Pie91] B.C. Pierce. *Basic Category Theory for Computer Scientists*. Foundations of Computing Series. The MIT Press, 1991.