Cartesian Closed Categories and Lambda-Calculus

Gérard Huet

INRIA

The purpose of these notes is to propose an equational framework for the formalization, and ultimately the mechanization, of categorical reasoning. This framework is explained by way of example in the axiomatization of cartesian closed categories. The relationship with intuitionistic sequent calculus and lambda calculus is explained.

1 The equational nature of category theory

Category theory reasoning proves equality of arrow compositions, as determined by diagrams. The corresponding equality is given in the model, i.e. in the category under consideration. But the proofs do not appeal to any particular property of the equality relation, such as extensionality. All we assume is that equality is a congruence with respect to the arrow operators.

However we are not dealing with simple homogeneous equational theories, but with typed theories. For instance, every arrow is equipped with its type $f : A \to B$. Here A and B are expressions denoting objects. These expressions are formed in turn by functorial operations and constants representing distinguished objects. The object terms can be considered untyped only within the context of one category. As soon as several categories are concerned, we must type the objects as well, with sorts representing categories. We thus have implicitly two levels of type structure.

The main difference between typed theories and untyped ones is that in untyped (homogeneous) theories one usually assume the domain of discourse to be non-empty. For instance, a first-order model has a non-empty carrier. Thus a variable always denotes something. In typed theories one does not usually make this restriction. Thus we do not want to impose the Hom-set $A \to B$ to be always non-empty for every A and B in the category, in the same way that we want to consider partial orderings.

This has an unfortunate consequence: the law of substitution of equals for equals does not hold whenever one substitutes an expression containing a variable universally quantified over an empty domain by an expression not containing this variable, since we replace something that does not denote by something which may denote. For instance, consider the signature $H : A \to B,\ T : B,\ F : B$, and the equations $H(x) = T$ and $H(x) = F$. These equations are valid in the model where A is the empty set, H is the empty function, and B is a set of two elements $\{0, 1\}$, with T interpreted as 1 and F interpreted as 0. In this model we *do not* have $T = F$. We shall have to keep this problem in mind in the following.

1.1 The general formalism

We have thus a formalism with four levels. At the first level, we have the alphabet of categories $\mathbf{Cat} = \{\mathbf{A}, \ldots, \mathbf{Z}\}$. At the second level, we have the alphabet of object operators. Every category is defined over an object alphabet Φ of operators given with an arity. Φ is where the (internal) functors live. We then form *sequents* by pairs of terms $M \to N$, with $M, N \in \mathbf{T}(\Phi, V)$. V is a set of variables denoting arbitrary objects of the category. At the third level we have the alphabet Σ

of arrow operators. An operator from Σ is given as an *inference rule* of the form:

$$S_1, \ldots, S_n \vdash S$$

where the S_i's and S are sequents. Such an operator is *polymorphic* over the free variables of the S_i's and S, which are supposed to be universally quantified over the inference rule. Such operators are familiar from logic, either as schematic inference rules, or as (definite) Horn clauses. Of course the arrows with domain M and codomain N are represented as terms over $\mathbf{T}(\Sigma, F)$ of type $M \to N$. Here F is a set of arrow variables, indexed by sequents $A \to B$. Finally, at the fourth level we have the *proofs* of arrow equalities. The alphabet consists of a set $\mathcal{R}$ of conditional rules of the form:

$$f_1 =_{S_1} g_1, \ldots, f_n =_{S_n} g_n \models f =_S g.$$

Here the f's and g are arrow expressions of type S, and similarly for the f_i's and g_i's. All object and arrow variables appearing in the rule are supposed to be universally quantified in front of the rule.

1.2 A simplified formalism

From now on, we shall assume that we are in one category of discourse which is left implicit. We shall therefore deal only with the last three levels. Furthermore, we shall assume that the only proof rules are:

$$Refl: \ f =_{A \to B} f$$

$$Trans: \ f =_{A \to B} g \ , \ g =_{A \to B} h \models f =_{A \to B} h$$

$$Sym: \ \ f =_{A \to B} g \models f =_{A \to B} g$$

together with the rules stating that $=$ is a congruence with respect to the operators in Σ, all other rules being given by simple identities, i.e. by rules with an empty set of premisses (n=0).

The further simplification comes from the realization that we are not really obliged to completely specify the types of all variable arrows and equalities, since there is a lot of redundancy. This fact exploits unification, and the following:

Meta-theorem. Let Σ be an arbitrary arrow signature, and E be an arbitrary term formed by operators from Σ and untyped variables. If there is an assignment of types to the variables of E that makes E well-typed with respect to Σ, there is a most general such assignment, independent in each variable, and furthermore the resulting type of E is most general. Here "more general" means "has as substitution instance". We call this assignment, together with the resulting type of E, the *principal* typing of E. More generally, for every type sequent S, if there is an assignment of types which makes E of a type some instance of S, there is a principal such assignment.

The meta-theorem above is most useful. It permits to omit most of the types. When we write an equation $E = E'$, we shall implicitly refer to the principal typing giving E and E' the same type $A \to B$. So from now on, all equations in $\mathcal{R}$ are written without type, the types being implicit from the principality assumption.

1.3 The initial theory Categ

We are now ready to start Category Theory. The initial theory **Categ** has $\Phi = \emptyset$, and $\Sigma = \{Id, \_;\_\}$, given with respective signatures:

$$Id \ : \ A \to A$$

$$\_;\_ \ : A \to B \ , \ B \to C \ \vdash A \to C.$$

The notation $\_;\_$ means that we use the infix notation $f\,;g$ for the composition of arrows f and g. We can read f *then* g, and follow arrow composition along diagrams with semi-colon as concatenation of the labels. But since more people are accustomed to the standard set-theoretic composition notation, we shall below use $f \circ g$ as an abbreviation for $g; f$.

The equations $\mathcal{R}$ of **Categ** are simply the laws of a monoïd:

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

$$Idl \ : \ Id \circ f \ = \ f$$

$$Idr \ : \ f \circ Id \ = \ f$$

It is to be noted that the identification of the Hom-set symbol $\to$ with the sequent entailment arrow is not fortuitious. Actually Id and ; are well-known inference rules of intuitionistic propositional calculus. However, the logic is quite poor at this stage: we have no propositional connective whatsoever, just the basic mechanism for sequent composition, stating that entailment is reflexive and transitive. The rules of $\mathcal{R}$, considered as a left-to-right rewriting system, define a normal form on the sequent calculus proofs, i.e. on the arrow expressions.

Before we embark on more complicated theories, let us give a recipe on how to cook an equational presentation from a categorical statement.

1.4 What the category theorists don't say

Open a standard book on category theory, and consider a typical categorical definition. It usually reads: "Mumble, such that the following diagram commutes." Similarly, a typical categorical result states: "If $diagram_1$ and ... and $diagram_n$ commute, then $diagram$ commutes." The first step in understanding such statements is to determine exactly their universality: what is exactly quantified, universally or existentially, what depends on what, what are exactly the parameters of the (frequent) unicity condition. The next step is to realize that the diagram states conditional equalities on arrows, and that it is enough to state the equalities of the inside diagrams in order to get all equalities.

A uniform compilation of such statements as an equational theory proceeds as follows. First write completely explicitly the quantification prefix of the statement, in two lines, one for the objects and one for the arrows. Then Skolemize the statement independently in the first and the second line. That is, for every existentially quantified variable x following the universally quantified $y_1, \ldots y_n$, introduce a new n-ary operator X and substitute throughout x by $X(y_1, \ldots y_n)$. The Skolemization of the object variables determines Φ. The Skolemization of the second line, together with the types implicit from the diagram determine Σ. Finally, following arrows around the inner diagrams determines $\mathcal{R}$. This concerns the *existential* part. For the *unicity* part, proceed as follows. Let f be the arrow whose unicity is asserted. The existence part provided by Skolemization an $F(g_1, \ldots, g_k)$ in place of f. Write a supplementary arrow h on the diagram parallel to f, and use the commutation conditions to eliminate all the g_i's as $G_i(h)$. Add an extra equation $F(G_1(h), \ldots, G_k(h)) = h$.

Once we have convinced ourselves that the category theoretic statements and proofs are of an equational nature, we may ask: why do the category theorists use diagrams at all? The reason is that diagram chasing is a sophisticated way of doing complex equality reasoning, using several equations simultaneously, on a shared data structure (the graph underlying the diagram). So diagrammatic reasoning may be considered a good tool for high level equational reasoning. On the other hand, equality reasoning techniques such as rewrite rules analysis is good for mechanical implementation, and this is why we stress here the equational theories hidden behind the diagrams.

Remark. Let us finally remark that more general categorical concepts than the simple universal statements that we shall now consider may force us to generalize the basic formalism. For instance, more complicated limit constructions such as pullbacks force the dependence of objects on arrows. The Skolemization cannot be effected separately on the object and the arrow variables, and we shall have to place ourselves in a more complicated type theory with dependent types.

2 Products

2.1 The theory Prod

We shall apply the recipe above to the definition of *product* in a category. We recall that a category possesses a product if for all objects A, B there exists an object C and there exist arrows fst, snd such that for every object D and every arrows f, g there exists a unique arrow h such that the following diagram commutes:

$$\begin{tikzcd}
A & C \arrow[l, "fst"'] \arrow[r, "snd"] & B \\
& D \arrow[ul, "f"] \arrow[u, "h"] \arrow[ur, "g"'] &
\end{tikzcd}$$

We now get the theory **Prod** by enriching **Categ** as follows. The Skolemization of C gives the binary functor $\times$, and we write with the infix notation $A \times B$ in place of C. So now $\Phi = \{\times\}$. Similarly, we add to Σ the following operators, issued respectively from fst, snd and h:

$$Fst \ : \ A \times B \to A$$

$$Snd \ : \ A \times B \to B$$

$$< \_, \_ > \ : \ D \to A \ , \ D \to B \vdash D \to A \times B$$

and we now have the usual diagram:

$$\begin{tikzcd}
A & A \times B \arrow[l, "Fst"'] \arrow[r, "Snd"] & B \\
 & D \arrow[ul, "f"] \arrow[u, "<f,g>"] \arrow[ur, "g"'] &
\end{tikzcd}$$

The existence of h, that is the commutation of the two triangles, gives two new equations in $\mathcal{R}$:

$$\pi_1 \;:\; Fst \circ < f,g > \;=\; f$$

$$\pi_2 \;:\; Snd \circ < f,g > \;=\; g.$$

Unicity of h gives one last equation:

$$UniPair \;:\; < Fst \circ h \, , \, Snd \circ h > \;=\; h.$$

The arrow part of the functor $\times$ may be defined as a derived operator as follows:

$$\_ \times \_ \;:\; A \to B \;,\; C \to D \;\vdash\; A \times C \to B \times D$$

$$Def \times \;:\; f \times g \;=\; < f \circ Fst \, , \, g \circ Snd >$$

Remark. There is a possible source of confusion in our terminology. We talked about the elements of Φ as functors. Actually these are just function symbols denoting object constructors. Skolemization of a diagram will determine certain such function symbols, but there is no guarantee that there will be a corresponding functor. For instance, for product, we had to define the arrow part of $\times$ above, and to verify that indeed it obeys the functoriality laws.

2.2 The logical point of view

From the logical point of view, specifying a product amounts to defining conjunction. Read $A \times B$ as $A \wedge B$, and recognize Fst, Snd and $< \_ , \_ >$ as respectively $\wedge$-elim-left, $\wedge$-elim-right, and $\wedge$-intro respectively [**?**, **?**, **?**].

The rules of $\mathcal{R}$ have a computational meaning: they specify how to reduce a proof to its normal form. Here we may apply known results from the theory of term rewriting systems, in order to complete $\mathcal{R}$ to a *canonical* system [**?**, **?**].

The Knuth-Bendix completion procedure, when applied to theory **Prod**, generates two additional rewrite rules:

$$IdPair \;:\; < Fst, Snd > \;=\; Id$$

$$DistrPair \;:\; < f,g > \circ h \;=\; < f \circ h \, , \, g \circ h > .$$

The resulting system $\mathcal{R}$ is canonical, and can be used to decide the equality of arrows in the theory **Prod**. Of course, the equations above do not modify the theory, since they have been obtained by equational reasoning.

Finally, let us note that other presentations of the same theory is possible. For instance, we could have obtained product as the right adjoint of the diagonal functor. The unit of this adjunction is the *duplicator*, which may be defined here as:

$$D \;=\; < Id, Id > .$$

Note that type-checking imposes that the two identities are the same, so that $D_A : A \to A \times A$. As its name suggests, the duplicator duplicates, in the sense that we can prove $D \circ f \;=\; < f, f >$. The co-unit of the adjunction is the pair of projections (Fst, Snd).

2.3 Finite products

We say that a category admits all finite products if it admits products and a terminal object. Equationally, this amounts to enrich the theory **Prod** to a theory **Prods** by adding a constant 1 to Φ, a polymorphic constant $Nil \;:\; A \to 1$ to Σ, and a unicity equation $Uni1$ to $\mathcal{R}$:

$$Uni1 \;:\; h \;=\; Nil.$$

Note that this does not make the equational theory inconsistent: variable h above is principally typed to $A \to 1$. However this equation brings up two problems. The first one is the one mentioned in the beginning of these notes, since variable h appears on the left but not on the right of $Uni1$. The second problem is that $Uni1$ cannot be considered as a term rewriting rule in the usual sense, since it would rewrite Nil to itself and therefore does not satisfy the finite termination criterion. Note that $Uni1$ entails with the other equations two consequences:

$$Zero \;:\; f \circ Nil \;=\; Nil$$

$$Id1 \;:\; Id \;=\; Nil.$$

Again, $Id1$ does not identify every Id with every Nil, but only (restoring explicit types) Id_1 with Nil_1. Now it may be checked that $Uni1$ is actually a consequence of $Zero$ and $Id1$. The rule $Zero$ is a bona fide rewrite rule, which leaves the special equality $Id1$ to be dealt with in an ad hoc fashion.

Using operators $\times$ and 1 we may now construct n-tuples of objects, which we shall call *contexts*. 1 is the empty context, and if E is a context of length n and A an object term, $E \times A$ is a context of length $n+1$. We write $|E|$ for the length of a context. If $\mathcal{C}$ is the current set of (representable) objects, i.e. $\mathbf{T}(\Phi, V)$, we denote by $\mathcal{C}^*$ be the set of contexts.

If $1 \leq i \leq |E|$, we define the i-th component E_i of E recursively, as:

$$\begin{array}{lll} (E \times A)_i & = A & \text{if } i = 1 \\ & E_{i-1} & \text{if } i > 1. \end{array}$$

If E and E' are contexts, we define their concatenation $E@E'$ as a context recursively:

$$\begin{array}{l} E@1 \;=\; E \\ E@(E' \times A) \;=\; (E@E') \times A. \end{array}$$

Similarly, using operators $< \_, \_ >$ and Nil we may construct lists, or n-tuples of arrows of same domain D. The empty arrow list is Nil, of length 0 and if $L : D \to E$ is an arrow list of length n then $< L, f >: D \to E \times A$ is an arrow list of length $n+1$, for every $f : D \to A$. Finally, for every object list and every n, with $1 \leq n \leq |E|$ we define recursively the projection arrow $\pi_E(n) \;:\; E \to E_n$, as:

$$\begin{array}{lll} \pi_E(n) & = Snd & \text{if } n = 1 \\ & \pi_{E'}(n-1) \circ Fst & \text{if } n > 1 \text{ and } E \;=\; E' \times A. \end{array}$$

3 CCC

3.1 The theory Exp

We obtain the theory **Exp** by enriching the theory **Prods** as follows. First, we add a binary operator $\Rightarrow$ to Φ. Next we add two operators to Σ, the constant App (application) and the unary operator $[]$ (abstraction):

$$App \;:\; (B \Rightarrow C) \times B \;\rightarrow\; C$$

$$[] \;:\; A \times B \rightarrow C \;\vdash\; A \rightarrow (B \Rightarrow C)$$

Finally, we add the following equations to $\mathcal{R}$:

$$ExAbs \;:\; App \circ ([]f \times Id) \;=\; f$$

$$UniAbs \;:\; [](App \circ (f \times Id)) \;=\; f$$

As before, this equational theory may be mechanically generated from the following diagram:

$$\begin{tikzcd}
B \Rightarrow C & \times & B \arrow[r, "App"] & C \\
A \arrow[u, "{[]f}"] & \times & B \arrow[u, "Id"] \arrow[ur, "f"] &
\end{tikzcd}$$

The logical point of view is here that $\Rightarrow$ is the (intuitionistic) implication. The operator App is $\Rightarrow$-introduction. It plays the rôle of the Modus Ponens inference rule (although here it is a constant, and not a binary operator). Abstraction is $\Rightarrow$-elimination, and plays somewhat the rôle of the deduction theorem.

Let us give a few equational consequences of the theory **Exp**:

$$IdExp \;:\; []App \;=\; Id$$

$$Red_1 \;:\; App \circ < []f \circ y \,,\, x > \;=\; f \circ < y \,,\, x >$$

$$Red \;:\; App \circ < []f \,,\, x > \;=\; f \circ < Id \,,\, x >$$

$$DistrAbs \;:\; []f \circ g \;=\; [](f \circ (g \times Id))$$

We can also show that abstraction is a bijection between the arrows of $A \times B \rightarrow C$ and those of $A \rightarrow B \Rightarrow C$, with inverse:

$$[]^{-1} f \;=\; App \circ < f \times Id > .$$

Thus we could have presented **Exp** in terms of $[]$ and $[]^{-1}$, and defined App as $[]^{-1} Id$. This corresponds to the fact that we could have rather axiomatized exponentiation by an adjunction to the product, whose co-unit is App (the unit being $[]Id$).

Finally, we define the arrow part of the functor $\Rightarrow$ (which is contravariant in its first argument) as:

$$f \Rightarrow g \;=\; [](g \circ App \circ (Id \times f)).$$

4 Lambda-calculus

Sometimes $\texttt{[]}$ is called Curryfication, in honor of Curry. In fact there is an important relation between combinatory logic and CCC's, which we shall exhibit on λ-calculus.

4.1 The λ-terms

We assume that the current theory is an extension of **Exp**. We define recursively a relation $E \vdash M : A$, read "M is a *term* of *type* A in *context* E", where $A \in \mathcal{C}$ and $E \in \mathcal{C}^*$, as follows:

Variable : If $1 \leq n \leq |E|$ then $E \vdash n : E_n$
Abstraction : If $E \times A \vdash M : B$ then $E \vdash \texttt{[}A\texttt{]}M : A \Rightarrow B$
Application : If $E \vdash M : A \Rightarrow B$ and $E \vdash N : A$ then $E \vdash (M\ N) : B$

Thus a term may be a natural number, or may be of the form $\texttt{[}A\texttt{]}M$ with A an object and M a term, or may be of the form $(M\ N)$ with M, N two terms.

We thus obtain λ-terms typed with objects of the CCC currently axiomatized. Variables are coded as de Bruijn's indexes [**?**], i.e. as integers denoting their reference depth (distance in the tree to their binder). This representation avoids all the renaming problems associated with actual names (α conversion), but we shall use such names whenever we give examples of terms. For instance, the term $\texttt{[}A\texttt{]}(1\ \texttt{[}B\texttt{]}(1\ 2))$ shall be presented under a concrete representation such as $\texttt{[}x : A\texttt{]}(x\ \texttt{[}y : B\texttt{]}(y\ x))$. In Church's original notation, the left bracket was a λ and the right bracket a dot, typing being indicated by superscripting, like: $\lambda x^A \cdot (x\ \lambda y^B \cdot (y\ x))$.

Note that the relation $E \vdash M : A$ is functional, in that A is uniquely determined from E and M. Thus the definition above may be interpreted as the recursive definition of a function $A = \tau_E(M)$.

4.2 A translation from λ-terms to CCC arrows

We shall now show how to translate λ-terms to CCC arrows. More precisely, to every term M such that $E \vdash M : A$ we associate an arrow $F_E(M) : E \rightarrow A$ as follows:

$$F_E(n) = \pi_E(n)$$

$$F_E(\texttt{[}A\texttt{]}M) = \texttt{[]}F_{E\times A}(M)$$

$$F_E((M\ N)) = App \circ < F_E(M)\, ,\, F_E(N) >$$

It can be easily proved by induction that $F_E(M)$ is a well-typed arrow expression.

Example. The closed term $M = \texttt{[}f : nat \Rightarrow nat\texttt{]}\texttt{[}x : nat\texttt{]}(f\ (f\ x))$ of type $A = (nat \Rightarrow nat) \Rightarrow (nat \Rightarrow nat)$ in the empty context $E = 1$, gets translated to:

$$F_E(M) = \texttt{[]}\texttt{[]}(App \circ < Snd \circ Fst\, ,\, App \circ < Snd \circ Fst\, ,\, Snd >>) : 1 \rightarrow A.$$

4.3 The syntactic theory of terms

The advantage of the name-free terms is that we have no name conflict. The disadvantage is that we have to explicitate relocation operations for terms containing free variables. For instance, let us define for every term M the term M^{+n} obtained in incrementing its free variables by n. Let $M^{+n} = R^0_n(M)$, with:

$$
\begin{array}{lll}
R_n^i(k) & = & k \quad\quad\;\; \text{if } k \leq i \\
 & & k+n \;\; \text{if } k > i \\
R_n^i(\texttt{[}A\texttt{]}\,M) & = & \texttt{[}A\texttt{]}\,R_n^{i+1}(M) \\
R_n^i((M\ N)) & = & (R_n^i(M)\ R_n^i(N)).
\end{array}
$$

The reader will check that $E \vdash M : A$ iff $E@E' \vdash M^{+n} : A$, where E' is an arbitrary context of length n.

We now define *substitution* to free variables. Let $E \times A \vdash M : B$, and $E \vdash N : A$. We shall define a term $M\{N\}$, and show that $E \vdash M\{N\} : B$. First we define recursively:

$$
\begin{array}{lll}
\Sigma_N^n(k+1) & = & k+1 \quad\quad \text{if } k < n \\
 & & N^{+n} \quad\quad\;\, \text{if } k = n \\
 & & k \quad\quad\quad\;\;\, \text{if } k > n \\
\Sigma_N^n(\texttt{[}A\texttt{]}\,M) & = & \texttt{[}A\texttt{]}\,\Sigma_N^{n+1}(M) \\
\Sigma_N^n((M\ M')) & = & (\Sigma_N^n(M)\ \Sigma_N^n(M')).
\end{array}
$$

It is easy to show that substitution preserves the types, in the sense that $(E \times A)@E' \vdash M : B$ and $E \vdash N : A$ implies $E@E' \vdash \Sigma_N^n(M) : B$, with $n = |E'|$.

Now we define $M\{N\} = \Sigma_N^0(M)$, and we get that $\tau_E(M\{N\}) = \tau_{E \times A}(M)$, with $A = \tau_E(N)$.

We are now ready to define the *computation* relation $\triangleright$ as follows:

$$(\texttt{[}A\texttt{]}\,M\ N) \triangleright M\{N\} \tag{β}$$

$$M \triangleright M' \Longrightarrow \texttt{[}A\texttt{]}\,M \triangleright \texttt{[}A\texttt{]}\,M' \tag{ξ}$$

$$M \triangleright M' \Longrightarrow (M\ N) \triangleright (M'\ N)$$

$$M \triangleright M' \Longrightarrow (N\ M) \triangleright (N\ M').$$

It is clear that computation preserves the types of terms. But it also preserves their values, in the sense of the translation to CCC arrows: If $E \vdash M : A$ and $M \triangleright N$, then $F_E(M) = F_E(N)$ in the theory **Exp**, as we shall show.

The computation relation presented above is traditionally called (strong) β-reduction. It is confluent and nœtherian (because of the types!), and thus every term possesses a canonical form, obtainable by iterating computation non-deterministically. Another valid conversion rule is η-conversion:

$$\texttt{[}x : A\texttt{]}\,(M\ x) \ = \ M \tag{η}$$

whenever x does not appear in M. Let us show that it corresponds to $UniAbs$, using our translation above.

First we define the *relocation* combinators $\rho(i)$ as follows.

$$
\begin{array}{lll}
\rho(0) & = & Fst \\
\rho(i+1) & = & \rho(i) \times Id
\end{array}
$$

It is easy to show that (with appropriate types):

$$
\begin{array}{lll}
\pi(k) \circ \rho(i) & = & \pi(k) \quad\quad\;\; \text{if } k \leq i \\
 & & \pi(k+1) \;\; \text{if } k > i
\end{array}
$$

and thus that $R_1^i(M) \;=\; M \circ \rho(i)$. As a particular case we get $M^{+1} \;=\; M \circ Fst$ and thus we can read the law $UniAbs$ as $[x](M^{+1}\ x) \;=\; M$. Whenever x does not occur in M the expressions M and M^{+1} are concretely identical, and we obtain the η conversion rule. Note however that $UniAbs$ is an algebraic law, whereas η makes sense only relatively to concrete representations.

We are now going to show that Red validates the β-reduction rule. First we define the *substitution* combinators as follows.

$$\begin{aligned}\sigma_N(0) \;&=\; < Id, N > \\ \sigma_N(n+1) \;&=\; \sigma_N(n) \times Id\end{aligned}$$

Next we check that for every λ-terms M,N and every integer n the following equation is provable in **Exp** (confusing M with $F(M)$, and assuming types are correct):

$$\Sigma_N^n(M) \;=\; M \circ \sigma_N(n).$$

This suggests defining in **Exp** the derived operator:

$$\_\{\_\} \;:\; A \times B \to C \;,\; A \to B \;\vdash A \to C$$

with defining equation:

$$Subst \;:\; f\{x\} \;=\; f\circ < Id, x >$$

and now the rule Red reads:

$$App \circ < [\,]\, f, x > \;\;=\; f\{x\}$$

which clearly validates the computation relation $\rhd$.

CCC arrows are richer than λ-terms. This suggests enriching λ-calculus with further operators $fst, snd, pair, nil$ with appropriate supplementary reduction rules, and to allow "varstruct" binding in order to have variables correspond to arbitrary sequences of Fst and Snd, as opposed to just integers coded up in unary notation. For instance, ML (without recursion) can be translated into CCC arrows by a simple extension of the translation F above. Actually, such a translation is the basis for an efficient implementation of the language [**?**].

4.4 The CCC word problem

The **Exp** theory above is decidable [**?**]. Unfortunately, no canonical system is known for the full theory. An interesting sub-theory is obtained by restricting $\mathcal{R}$ to the set

$$\mathcal{R}_1 \;=\; \{Ass, Idl, Idr, \pi_1, \pi_2, DistrPair, IdPair, UniPair, Red, DistrAbs\}$$

Considered as a (typed) term-rewriting system, $\mathcal{R}_1$ is locally confluent. This can be mechanically checked by the Knuth-Bendix decision procedure [**?**]. Note that this is possible because a system such as $\mathcal{R}_1$ may be considered an (untyped) equational theory in the ordinary sense, by mixing the arrow structure and the object structure as follows: every arrow sub-term M of type $A \to B$ is represented as $:(M, A, B)$, where $:$ is a special ternary function symbol. Note that variables in the type subparts get instanciated by matching and unification, in the same way as the variables in the arrow subparts. This supports our view of the polymorphic nature of the categorical combinators.

We conjecture $\mathcal{R}_1$ to be nœtherian, and thus confluent. It would be interesting to have the termination argument formulated as to showing that the rewriting relation is a well-partial-ordering. However, note that this argument cannot work on the Σ-trees alone, since the corresponding untyped system is strong enough to code the β-reductions of untyped λ-calculus. We remark that

Klop has shown [?] that λ-calculus with surjective pairing is not Church-Rosser; however, his proof cannot be easily adapted to show that the untyped version of $\mathcal{R}_1$ is not confluent.

Another interesting locally confluent subtheory is:

$$\mathcal{R}_2 \;=\; \{Ass, Idl, Idr, \pi_1, \pi_2, DistrPair, IdPair, UniPair, Red, Red_1\}.$$

The state of the art in the theory of categorical combinatory algebra and its relations with various λ-calculi can be found in Curien's extensive monography [?].

4.5 Possible Extensions

It is also possible to enrich the type structure with sums, corresponding to categorical co-product. The **Exp** theory is enriched with injections and a conditional operator. The corresponding models are the bi-cartesian closed categories.

Finally, we may postulate the existence of a universal object U and build the full untyped λ-calculus in the manner of Scott, as described in [?].

That is, we postulate a retract pair between U and $U \Rightarrow U$:

$$Quote : (U \Rightarrow U) \to U$$

$$Eval : U \to (U \Rightarrow U)$$

verifying:

$$Retract : Eval \circ Quote = Id.$$

Let us call **Univ** the theory obtained by the corresponding enrichment of **Exp**.

We may now translate any $M \in \lambda_n$ as an arrow $A_n(M) : U^n \to U$ as follows:

$$A_n(k) \;=\; \pi_{U^n}(k)$$

$$A_n(\texttt{[]}\,M) \;=\; Quote \circ \texttt{[]}\,A_{n+1}(M)$$

$$A_n((M\ N)) \;=\; App \circ < Eval \circ A_n(M)\,,\, A_n(N) > .$$

We leave it to the reader to check that the β rule is still an equational consequence of **Univ**. However, note that the η rule is not valid anymore, since it would entail that *Eval* and *Quote* define an isomorphism between U and $U \Rightarrow U$.

Caution. Some combinations of the above extensions may be incompatible, in that they may lead to an inconsistency, in the sense that the only model of the extended theory is the trivial category **1**. For instance, Lawvere showed that the theory of bi-cartesian closed categories with fixpoints is inconsistent [?, ?].

Acknowledgements

The general formalization was developped in 1983, after listening in Sophia-Antipolis to a talk by P.L. Curien on categorical combinators. The translation from 4.2 was derived after the author noticed the close connection between the categorical operators of Lambek [?] and de Bruijn's nameless notation for λ-calculus [?]. Instrumental to this was a close study of the ML compiler developed by Lockwood Morris for the LCF project [?]. The syntactic theory from 4.3 is similar to the formalization of Automath by Jutting [?], except that the absence of dependent types makes things a little simpler. This treatment of λ-calculus computation was the basis of the author's implementation in ML of the Calculus of Constructions [?]. The author acknowledges numerous discussions on categorical combinators with P.L. Curien, A. Poigné and D. Scott.

References

[1] H. Barendregt "The Lambda-Calculus: Its Syntax and Semantics." North-Holland (1980).

[2] N.G. de Bruijn "Lambda-Calculus Notation with Nameless Dummies, a Tool for Automatic Formula Manipulation, with Application to the Church-Rosser Theorem." Indag. Math. **34,5** (1972), 381–392.

[3] G. Cousineau, P.L. Curien and M. Mauny "The Categorical Abstract Machine." In Functional Programming Languages and Computer Architecture, Ed. J. P. Jouannaud, Springer-Verlag LNCS 201 (1985) 50–64.

[4] Th. Coquand, G. Huet "Constructions: A Higher Order Proof System for Mechanizing Mathematics." EUROCAL85, Linz, Springer-Verlag LNCS 203 (1985).

[5] P. L. Curien "Categorical Combinatory Logic." ICALP 85, Nafplion, Springer-Verlag LNCS 194 (1985).

[6] P. L. Curien "Categorical Combinators, Sequential Algorithms and Functional Programming." Monography to appear, Pitman (1985).

[7] G. Gentzen "The Collected Papers of Gerhard Gentzen." Ed. E. Szabo, North-Holland, Amsterdam (1969).

[8] M. J. Gordon, A. J. Milner, C. P. Wadsworth "Edinburgh LCF" Springer-Verlag LNCS **78** (1979).

[9] G. Huet "Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems." J. Assoc. Comp. Mach. **27,4** (1980) 797–821.

[10] G. Huet "Initiation à la Théorie des Catégories." Polycopié de cours de DEA, Université Paris VII (Nov. 1985).

[11] G. Huet. "Deduction and Computation." in Fundamentals in Artificial Intelligence, Eds. W. Bibel and Ph. Jorrand, Springer-Verlag Lecture Notes in Computer Science vol. 232 (1986) 39–74.

[12] G. Huet. "Formal Structures for Computation and Deduction." In preparation.

[13] H. Huwig and A. Poigné "A note on inconsistencies caused by fixpoints in a cartesian closed category." Personal communication (April 1986).

[14] L.S. van Benthem Jutting "The language theory of Λ_∞, a typed λ-calculus where terms are types." Personal communication (1984).

[15] S.C. Kleene "Introduction to Meta-mathematics." North Holland (1952).

[16] J.W. Klop "Combinatory Reduction Systems." Ph. D. Thesis, Mathematisch Centrum Amsterdam (1980).

[17] D. Knuth, P. Bendix "Simple word problems in universal algebras". In: Computational Problems in Abstract Algebra, J. Leech Ed., Pergamon (1970) 263–297.

[18] J. Lambek "From Lambda-calculus to Cartesian Closed Categories." in To H. B. Curry: Essays on Combinatory Logic, Lambda-calculus and Formalism, Eds. J. P. Seldin and J. R. Hindley, Academic Press (1980).

[19] J. Lambek and P. J. Scott "Aspects of Higher Order Categorical Logic." Contemporary Mathematics **30** (1984) 145–174.

[20] F. W. Lawvere "Diagonal Arguments and Cartesian Closed Categories." in Category Theory, Homology Theory and their Applications II, Springer-Verlag Lect. Notes in Math. **92** (1969).

[21] S. Mac Lane "Categories for the Working Mathematician." Springer-Verlag (1971).

[22] C. Mann "The Connection between Equivalence of Proofs and Cartesian Closed Categories." Proc. London Math. Soc. **31** (1975) 289–310.

[23] A. Poigné "On Semantic Algebras ." Universitat Dortmund (March 1983).

[24] D. Prawitz "Natural Deduction." Almqist and Wiskell, Stockolm (1965).

[25] D. Scott "Relating Theories of the Lambda-Calculus." in To H. B. Curry: Essays on Combinatory Logic, Lambda-calculus and Formalism, Eds. J. P. Seldin and J. R. Hindley, Academic Press (1980).

[26] S. Stenlund "Combinators λ-terms, and proof theory." Reidel (1972).

[27] M. E. Szabo "Algebra of Proofs." North-Holland (1978).