MODELLING COMPUTATIONS: A 2-CATEGORICAL FRAMEWORK

R.A.G. SEELY

Dept. of Mathematics, John Abbott College, P.O. Box 2000
Ste. Anne de Bellevue, Quebec H9X 3L9
and
Dept. of Mathematics, McGill University, Montreal, Quebec

ABSTRACT

An introduction to 2-categories is given by illustrating how the structure of typed lambda calculus may naturally be viewed as a 2-category. In this vein, the structure of computations or conversions gives rise to notions of lax 2-adjointness.

0. Introduction

It has become standard, when modelling the lambda calculus (first or higher order), to treat beta conversion as equality (and frequently eta conversion as well.) In particular, all categorical semantics (eg. for lambda calculus, in terms of C-monoids -- for untyped -- or cartesian closed categories -- for typed -- (LAMBEK-SCOTT [1986]), and for polymorphic lambda calculus, in terms of PL categories (SEELY [1986])) have this property. However, there is no doubt that something is lost with such an approach, since beta conversion

$$(\lambda\ x \text{ in } A.\ a)\ (b) = a[x:=b]$$

explicitly equates each stage in a computation process with the result of the computation.

There are several standard approaches to semantics of lambda calculus which do not make such identifications, particularly for the untyped lambda calculus, and particularly in the setting of domain models. For example, a model of untyped lambda calculus may be given by a triple (D,h,k), where D is a domain, h a map $D \longrightarrow (D \Rightarrow D)$ and k a map $(D \Rightarrow D) \longrightarrow D$, satisfying suitable conditions. (For the moment, let me not specify what "domain" means, nor what kinds of maps h and k are.) In such a context, if h and k are inverse isomorphisms, then D is an extensional model, where both beta and eta conversions are interpreted as identities. Non-extensional models arise from weakening the condition on h, k: for instance, if

$$hk = id(D \Rightarrow D) \quad \text{and} \quad kh \leq id(D)$$

where $\leq$ is defined, usually pointwise, in terms of the order on D, then we get a non-extensional model in which beta conversion is identity, and eta conversion is increasing:

$$(\lambda\ x.\ a(x)) \leq a \quad (\text{for } x \text{ not free in } a).$$

Categorically, we would say that k is left adjoint to h, and so $(D \Rightarrow D)$ is a coreflective subcategory (or subdomain) of D. (SCOTT [1972] calls $(D \Rightarrow D)$ a projection of D.)

Alternatively,

$$hk = id(D \Rightarrow D) \quad \text{and} \quad id(D) \leq kh$$

gives a non-extensional model in which eta conversion is decreasing: ie. h is a left adjoint to k and so $(D \Rightarrow D)$ is a reflective subcategory of D. This is essentially the situation of Scott's $\mathcal{P}\omega$ model, SCOTT [1976].

Finally, replacing $hk = id(D \Rightarrow D)$ with, for instance,

$$hk \leq id(D \Rightarrow D)$$

yields a lambda structure with increasing beta conversion:

$$(\lambda\ x.\ a)(b) \leq a[a:=b].$$

Pairing with this a decreasing eta conversion, $id(D) \leq kh$, yields an adjunction: h is left adjoint to k, (but $(D \Rightarrow D)$ need not be a subcategory of D.)

This last will be the motivating example for the structures of this paper. However, we generalise the usual domain framework by replacing poset structure with categorical structure. We shall work with the typed lambda calculus, for simplicity -- however, I shall indicate at the end how this extends to polymorphic lambda calculus, which is why we use the typed structure. The lambda structure we wish to abstract in this way consists of the types, the terms, and the conversions. This will produce a 2-category, rather than just a category: the types will be the objects, the terms the morphisms, and the conversions will be the 2-cells of this 2-category. The point of this is that the structure of $\Rightarrow$ given by beta and eta conversion amounts to a weak ("lax") notion of adjunction, corresponding in the usual categorical setting to cartesian closedness.

The main purpose of this paper is to introduce 2-categories to computer scientists as a suitable framework for certain types of semantics. Hence a few words of introduction about 2-categories might be suitable. A 2-category is a category enriched with some extra structure: the hom-sets are themselves categories. (This generalises the familiar context in which one has a category, e.g of domains, in which each hom-set inherits a partial order from the orders on the domains. Since a poset is a category, every such category of domains is in fact a 2-category.) The morphisms between morphisms are called "2-cells". There are various axioms which guarantee that the categorical structures on the hom-sets "mesh" well with the original categorical structure (of objects and morphisms). Of course it is my point that the general flavour of this may be gleaned from considering the (typed) lambda calculus: thinking of types as "objects", it is easy to see how a term a of type A with exactly one free variable x of type B may be considered as a map ("morphism") $B \longrightarrow A$. Given another term b of type A with exactly one free variable x of type B, so $b: B \longrightarrow A$ also, then a "2-cell" $p: a \Longrightarrow b$ would be a reduction from a to b. (Notice that p does not affect A, B, though A, B are implicit in any description of p ; p only acts on a to produce b.)

There is an identity reduction $a \Longrightarrow a$ for any term a, and one can compose reductions, in fact in two ways: given terms

$$a,b,c : B \longrightarrow A \quad \text{and} \quad d,e : C \longrightarrow B$$

and given reductions

$$p : a \Longrightarrow b,\ q : b \Longrightarrow c, \quad \text{and} \quad r : d \Longrightarrow e$$

there are evident compositions

$$qp : a \Longrightarrow c \ \text{(between terms } B \longrightarrow A\text{), and}$$
$$pr : ad \Longrightarrow be \ \text{(between terms } C \longrightarrow A\text{),}$$

where ad and be are defined by composition in the category of types and terms -- ie. by substitution:

$$ad = a[x:=d] \quad \text{and} \quad be = b[x:=e].$$

The main axiom of 2-categories is the "interchange law", which asserts that these two kinds of composition must commute with each other.

A final remark: initially we shall suppose that eta conversion is decreasing, rather than increasing. This follows the proof theorist's view that eta conversion is an expansion:

$$a \lesssim (\lambda\ x \text{ in } A.\ a(x)) \ \text{(for } x \text{ not free in } a\text{)}$$

rather than a reduction. As indicated above, this will allow us to regard beta and eta conversions as defining a (lax) adjunction in a fairly standard way. Later, we shall consider the effect of reversing the sense of eta conversion; this allows a different formulation of lax adjunction, but the notion of lax functor becomes less satisfactory.

Acknowledgement: As stated above, this paper is primarily intended as propaganda - 2-categories occur naturally as structures in computer science. (The interested reader should pursue this in more mathematically serious works in "2-categorical logic", particularly those from the "Australian school"; some references are given here.) I have not used the heading "Theorem", but rather "Example", since there are in fact no particularly new ideas or theorems here; this paper is based on SEELY [1979], which gives a similar analysis of first order logic. The main difference between that paper and this, is that in [1979], implication is not successfully treated, whereas here, by concentrating only on implication, those difficulties are avoided.

This work was completed with partial support from Fonds F.C.A.R., Québec.

1. 2-categorical preliminaries

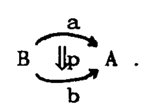
We summarise here the basic notions we need from the general body of 2-category theory; a more comprehensive introduction may be found in KELLY-STREET [1974].

1.1 Definition: A 2-category $\underline{A}$ has the following structure:

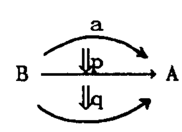
(i) a collection $Ob(\underline{A})$ of objects, or 0-cells: A,B, etc.

(ii) a collection $Mor(\underline{A})$ of morphisms, or arrows or 1-cells: $a: B \longrightarrow A$, etc.

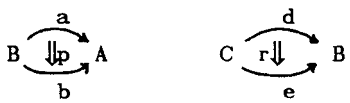
(iii) a collection $Cell(\underline{A})$ of 2-cells: $p: a \Longrightarrow b : B \longrightarrow A$, etc. also denoted



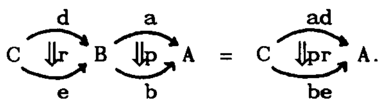
The objects and morphisms form a category $\underline{A}_o$, the "underlying category of $\underline{A}$". For fixed A,B, the morphisms $B \longrightarrow A$ and the 2-cells between them form a category Hom(B,A), also denoted $\underline{A}$(B,A). Composition in this category is known as "vertical composition":



This composite is denoted $q \cdot p$, or qp if no confusion results. Furthermore, given 2-cells



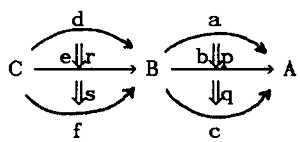
we have a "horizontal composition" giving a 2-cell



This composite is denoted $p*r$, or pr if no confusion results. (Here ad and be are given by composition in the category $\underline{A}_o$.)

The composition $*$ must be associative and have as identities the evident identity 2-cells.

Finally, the compositions must be compatible: given



then $(q\cdot p)*(s\cdot r) = (q*s)\cdot(p*r)$, (this is known as the "interchange law".) (If we view a 2-category as a <u>CAT</u>-enriched category, KELLY [1982], this is part of the "functorality" of composition: $\underline{A}(B,A) \times \underline{A}(C,B) \longrightarrow \underline{A}(C,A)$.)

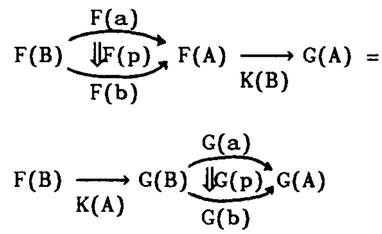
1.2 <u>Examples</u>: The paradigmatic example (for category theorists) is the 2-category <u>CAT</u> of categories, functors, and natural transformations. Indeed, there is an equation (categories: 2-categories) = (<u>SET</u> : <u>CAT</u>), where <u>SET</u> is the (paradigmatic) category of sets and functions.

Other examples can be constructed from various categories of ordered objects, where the hom-sets are themselves ordered naturally, and so are categories. For instance, the category $\underline{QD}_o$ of qualitative domains and stable functions (GIRARD [1986]) becomes a 2-category <u>QD</u> by saying there is a 2-cell $f \Longrightarrow g$ just if $f \leq g$ in the Berry order, for stable functions $f,g : X \longrightarrow Y$, (ie. if $f \leq g$ in $X \Rightarrow Y$.)

Finally, as stated in the introduction, the typed lambda calculus may naturally be viewed as a 2-category, as discussed in section 2.

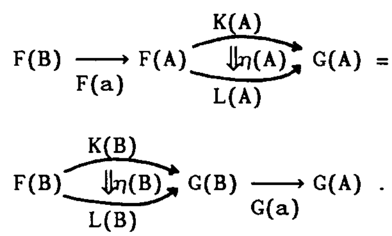
1.3 <u>Definition</u>: A (strict) 2-functor $F : \underline{A} \longrightarrow \underline{B}$ sends objects (respectively morphisms, 2-cells) of $\underline{A}$ to objects (respectively morphisms, 2-cells) of $\underline{B}$, preserving domains, codomains, identities, and compositions.

1.4 Similarly, we can <u>define</u> 2-natural transformations. $K:F \Longrightarrow G:\underline{A} \longrightarrow \underline{B}$ assigns to each object A of $\underline{A}$ a morphism $K(A): F(A) \longrightarrow G(A)$ in $\underline{B}$, natural in the usual sense (for $a:B \longrightarrow A$, $K(A)\cdot F(a) = G(a)\cdot K(B)$), and 2-natural, in that for a 2-cell $p:a \Longrightarrow b:B \longrightarrow A$ in $\underline{A}$, we have



(where this notation means in fact the horizontal composite of, on the left hand side, the identity 2-cell id(K(B)) with F(p), and similarly on the right.) (Note that one frequently identifies an object with its identity map.)

1.5 <u>Definition</u>: A modification $\eta:K \longrightarrow L: F \Longrightarrow G: \underline{A} \longrightarrow \underline{B}$ is a morphism of 2-natural transformations, and assigns to each object A of $\underline{A}$ a 2-cell $\eta(A):K(A) \Longrightarrow L(A)$ so that for $a:B \longrightarrow A$ in $\underline{A}$,



(Again, F(a) means the identity 2-cell, and the equation is between horizontal composite 2-cells.)

(So, <u>2-CAT</u> is really a 3-category!...)

1.6 In sections 3, 4, we shall also examine weakenings of these notions, in discussing lax functors, weak adjunctions, and so on. For a fuller discussion, see KELLY-STREET [1974], KELLY [1982] and GRAY [1974].

2. <u>The 2-category LAMBDA</u>

2.1 I outlined the structure in the introduction, so now I shall be brief, mainly fixing notation and clarifying some technical points. I assume the reader is familiar with the typed lambda calculus, as in LAMBEK-SCOTT [1986]; the following is a brief summary.

Types are closed under the operations $A \& B$, and $A \Rightarrow B$.

Terms include variables for each type, and are closed under:

Proceedings of the Symposium on Logic in Computer Science, 1987
(Computer Society of the IEEE 1987) 65–71

(& I) If a in A, b in B, then ⟨a,b⟩ in A & B.
(& E) If c in A & B, then (1st c) in A, (2nd c) in B.
(⇒ I) If b in B, x a variable in A, then (λ x in A. b) in A ⇒ B.
(⇒ E) If c in A ⇒ B and a in A, then c(a) in B.

Conversions include the following:

(& beta) 1st⟨a,b⟩ = a, 2nd⟨a,b⟩ = b.
(& eta) c = ⟨1st c, 2nd c⟩.

(See below why = appears here instead of ⟹.)

(⇒ beta) (λ x in A. b) (a) ⟹ b[x:=a].
(⇒ eta) c ⟹ λ x in A. c(x) , (where x not free in c).

2.2 LAMBDA is defined as outlined in the introduction: objects are types, morphisms a: B ⟶ A are terms of type A with exactly one free variable x of type B, and a 2-cell between such morphisms is a composition of conversions (a "reduction" -- I shall use this term even though it may seem inappropriate for the increasing eta conversions.)

2.3 I shall treat alpha conversions as identities. Furthermore, for simplicity, I shall concentrate solely on ⇒, and thus shall collapse the 2-categorical structure dealing only with & by regarding (& beta) and (& eta) as identities also. This could be avoided by dropping all reference to &, and generalising the categorical structure to allow morphisms A, B, C,... ⟶ Z with finite sequences of objects as domains: such a morphism should be thought of as an ordinary morphism A & B & C &... ⟶ Z, or equivalently, A ⟶ B ⇒ C ⇒ ... ⇒ Z.

Such notions have been considered by others, but I think that cartesian closed categories are so much more natural that it would be a mistake to omit finite products, (or even a terminal object, for that matter.)

A consequence of this will be that we shall frequently use ordered pairs $\langle x_A, y_B \rangle$ to denote variables of type A & B.

2.4 It is straightforward to check that LAMBDA is in fact a 2-category; most of the details are either implicitly or explicitly in LAMBEK-SCOTT [1986]. Only the interchange law needs comment: in effect we just assume it to be true, introducing an equivalence on reductions. (The validity of this may be checked by considering the corresponding situation in the & ⇒ fragment of first order logic, via the Curry-Howard "types as formulae" isomorphism, where interchange is valid; see SEELY [1979].) The key to the interchange law is this:

2.5 Definition/"Lemma": For p:a ⟹ b:B ⟶ A, r:d ⟹ e:C ⟶ B (as in the introduction), the following reduction sequences are the same:

$$a[x:=d] \xRightarrow{p[d]} b[x:=d] \xRightarrow{b[r]} b[x:=e]$$

$$a[x:=d] \xRightarrow{a[r]} a[x:=e] \xRightarrow{p[e]} b[x:=e].$$

The common composite is p∗r.

2.6 Remark: Notice that the associativity of composition of morphisms is equivalent to the equality

$$a[x_B:=b][y_C:=c] = a[x_B:=b[y_C:=c]]$$

for terms $D \xrightarrow{c} C \xrightarrow{b} B \xrightarrow{a} A$.

3. Laxity

3.1 Definition: Given two 2-categories A and B, by a lax functor F: A ⟶ B we mean a function that sends objects, morphisms, 2-cells of A to, respectively, objects, morphisms, 2-cells of B, which is strictly functorial on 2-cells; instead of functorality for morphisms, we have "comparison 2-cells" as follows:

if a: B ⟶ A, b: C ⟶ B in A, there are 2-cells in B

$$\gamma(F;\ a,b):\ F(a)F(b) \Longrightarrow F(ab)$$

$$\iota(F;\ A):\ id(FA) \Longrightarrow F(idA)$$

(Coherence conditions for these will be discussed in the appendix.)

3.2 Example: Fix a type E: then this induces a lax functor

G: LAMBDA ⟶ LAMBDA, G(A) = (E ⇒ A).

(Exercise: define G on morphisms and 2-cells. Then show that in this case γ is (⇒ beta) and ι is (⇒ eta) .)

3.3 Definition: Given two lax functors F: A ⟶ B , G: B ⟶ A, by a lax semantic adjunction F ⊣ G we mean there is a pair of lax 2-natural transformations

K:B(F-,-) ⟶ A(-,G-) and L:A(-,G-) ⟶ B(F-,-)

so that L is weakly left adjoint to K; this means the following:

(i) (laxity of K,L) Instead of strict naturality of K,L, there are comparison 2-cells. For morphisms $a:A_1 \longrightarrow A$ in A, $b:B \longrightarrow B_1$ in B, there are natural transformations (2-cells in CAT)

Proceedings of the Symposium on Logic in Computer Science, 1987
(Computer Society of the IEEE 1987) 65–71

k,l as shown in figure 1.

(Note that K and L are strict in their first coordinate, lax only in the second.)

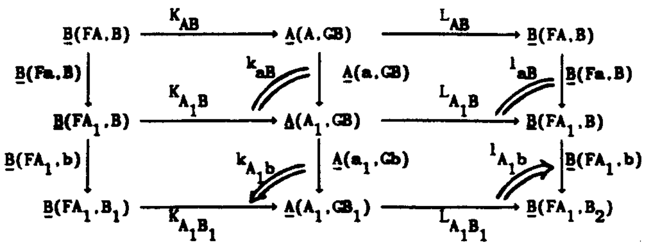


Figure 1

(ii) (L ⊣ K) There are (lax) modifications

$$\eta : id(\underline{A}(-, G-)) \Longrightarrow KL$$
$$\epsilon : LK \Longrightarrow id\ (\underline{B}(F-, -))$$

so that $(K\epsilon)\cdot(\eta K) = K$ and $(\epsilon L)\cdot(L\eta) = L$.

(Again, coherence is relegated to the appendix, where also the meaning of "lax modification" is given.)

3.4 Example: Fix a type E. If G: LAMBDA ⟶ LAMBDA is the lax functor (E ⇒ A) of 3.2, and if F: LAMBDA ⟶ LAMBDA is the (strict) functor F(A) = (A & E), then F ⊣ G.

It is a pity, but setting out this structure in full would take much too much space here; a summary of the relevant ingredients is given in Table 1. As an example, consider η and ϵ: Given objects (types) A,B and a morphism (term) c: A ⟶ (E ⇒ B),

$$K_{AB}L_{AB}(c) = \lambda\ z \text{ in } E.\ c[x_A := 1st\langle x,z\rangle](2nd\langle x,z\rangle)$$
$$= \lambda\ z \text{ in } E.\ c(z).$$

So $\eta_{AB}(c)$ is the eta conversion

$$c \Longrightarrow \lambda\ z \text{ in } E.\ c(z).$$

Similarly, for a term d: (A & E) ⟶ B, with free variable $\langle x_A, z_E\rangle$, $\epsilon_{AB}(d)$ is the beta conversion (λ z in E. d) (z) ⟹ d.

The "triangle identities" are the following principle, which may be viewed as an analogue to beta conversion at the level of reductions:

(BETA) An eta conversion of an occurrence of a logical symbol followed by beta conversion of the same occurrence is an identity operation, (provided the composite is an "endo-operation", so this makes sense.)

F:	(- & E)	k_{aB}:	(& beta)
G:	(E ⇒ -)	k_{Ab}:	(⇒ beta)
ι_G:	(⇒ beta)	l_{aB}:	(& beta)
τ_G:	(⇒ eta)	l_{Ab}:	(⇒ beta)
K:	(&I, ⇒I)	η:	(⇒ eta)
L:	(&E, ⇒E)	ϵ:	(⇒ beta)

TABLE 1

3.5 Usually in defining a notion of adjunction, it is expected that equivalent "semantic" and "syntactic" formulations exist, (the former being in terms of hom-sets, the latter of units and counits.) For lax adjunctions the situation is rather more complicated, and depends on the precise details of the notion of "laxity". In particular, for the notion of 3.3, although no doubt one could "fudge" an equivalent syntactic formulation, what is striking is that the natural such formulation fails. (Section 4 gives a variant - see SEELY [1977] for a discussion of this case.) Further, it is curious to note the role eta conversion plays: for if we reverse the sense of (⇒ eta), then although LAMBDA is no longer an example of 3.3, it gives nevertheless an example of a natural notion of lax syntactic adjunction (which has no natural semantic equivalent.)

4. Reversing eta

4.1 In this section, we briefly consider the situation when eta conversion is increasing. The first remark, of course, is that we must alter the definition (3.1) of lax functor by reversing ι ; τ remains the same.

(It must be admitted that from a 2-categorical viewpoint, this is highly unsatisfactory, in that ι and τ are now going in reversed senses. Indeed, on this observation could be based a fairly convincing argument that this illustrates just why eta ought not to be increasing.)

However, nevertheless, we can give a neat description of the adjoint structure enjoyed by ⇒ in this context as well.

4.2 Definition: Given two lax functors (as in 4.1) F: $\underline{A} \longrightarrow \underline{B}$, G: $\underline{B} \longrightarrow \underline{A}$, by a lax syntactic adjunction F ⊣| G, we mean that there are lax 2-natural transformations

$$\alpha\text{: } id(\underline{A}) \longrightarrow GF \quad\text{and}\quad \beta\text{: } FG \longrightarrow id(\underline{B})\ ,$$

lax in the sense that for morphisms $a\text{: } A_1 \longrightarrow A$

Proceedings of the Symposium on Logic in Computer Science, 1987
(Computer Society of the IEEE 1987) 65–71

of $\underline{A}$, $b: B \longrightarrow B_1$ of $\underline{B}$, there are 2-cells

$$n(a) : FG(a) \cdot \alpha(A_1) \Longrightarrow \alpha(A) \cdot a$$
$$e(b) : \beta(B_1) \cdot FG(b) \Longrightarrow b \cdot \beta(B) \ .$$

Furthermore, there are (lax) modifications

$$\rho : \beta F \cdot F\alpha \Longrightarrow id(F)$$
$$\sigma : G\beta \cdot \alpha G \Longrightarrow id(G)$$

so that

$$\iota G\rho \cdot \alpha = (\sigma F \cdot \alpha)(G\beta F \cdot n\alpha) : G\beta F \cdot GF\alpha \cdot \alpha \Rightarrow \alpha$$

and

$$\beta \cdot F\sigma = (\beta \cdot \rho G)(e\beta \cdot F\alpha G) : \beta \cdot FG\beta \cdot F\alpha G \Rightarrow \beta \ .$$

(Again, coherence is discussed in the appendix.)

4.3 Remark: This situation is somewhat irregular, in that one would expect ρ, σ to have opposite senses. (Indeed, if one were interested in setting up equivalent semantic and syntactic notions of adjunction, in the notations of 3.3 and 4.2, one would expect α, β to correspond to K, L; n, e to correspond to k, l; ρ, σ to correspond to η, ϵ; and the triangle identities to correspond to each other. So in this sense, ρ, σ ought to be unit and counit, and ρ ought to be reversed.) However, for the structure of LAMBDA, this simply is not the case. (However, the above correspondances are more or less correct, and give rise to Table 2.)

4.4 Example: As in 3.4, if F(A) = (A & E), G(A) = (E ⇒ A), then F ⊣ G. Again, the details are but summarized, in Table 2. It seems this formulation is less perspicuous, as may be seen by considering ρ and σ; (part of the "problem" is that in 3.4, our objects are terms, since we are working with hom-categories, whereas here the objects are types.)

Given object A, $\beta(F(A)) \cdot F(\alpha(A))$ = (λ z' in E. $\langle x_A, z' \rangle)(z_E)$ and id(F(A)) is $\langle x_A, z_E \rangle$; $\rho(A)$ is the beta conversion

$$(\lambda\ z'\ \text{in E}.\ \langle x_A, z' \rangle)(z_E) \Longrightarrow \langle x_A, z_E \rangle.$$

Given object B, $G(\beta(B)) \cdot \alpha(G(B))$ = λz in E. (1st(λz' in E.⟨y,z'⟩)(z))(2nd(λz' in E.⟨y,z'⟩)(z)), where y is of type E ⇒ B, so that id(G(B)) = y; $\sigma(B)$ is the reduction: beta applied to each occurrence of (λ z' in E. ⟨y,z'⟩)(z) to produce λ z in E.y(z), (note that (& beta) is also used), and then eta conversion to yield y.

α : (&I, ⇒ I)	e : (& beta, ⇒ beta)
β : (&E, ⇒ E)	ρ : (& beta, ⇒ beta,) & eta
n : (& beta, ⇒ beta)	σ : (& beta, ⇒ beta,) ⇒ eta

TABLE 2

5. Higher order lambda calculus

The structure discussed in sections 3,4 also applies to higher order (polymorphic) lambda calculus. The following brief outline shows how this works for second order lambda calculus.

Types also include indeterminates (variable types) and FORALL t. A.

Terms are also closed under

(FORALL I) If a in A, t not free in the type of a free variable of a, then (Λ t. a) in (FORALL t. A).

(FORALL E) If c in (FORALL t. A), B a type, then c(B) in A[t:=B].

Conversions include

(FORALL beta) (Λ t. a)(B) ⟹ a[t:=B].
(FORALL eta) c ⟹ Λ t. c(t).

We then define an indexed 2-category POLYLAMBDA, along the lines of the PL categories of SEELY [1986]: now each "fibre" will be a 2-category like LAMBDA in section 2 above. The base category will consist, as in SEELY [1986], of "orders" (ie. "kinds" -- in the second order case, just finite powers of TYPE) and "operators" (ie. "constructors"). Over a kind (e.g. TYPE) will be the 2-category of types with the appropriate free indeterminates (eg. exactly one, over TYPE), terms, and reductions.

In such a context, FORALL t.( ) defines a lax functor between fibres (ie. from the fibre over K × TYPE to the fibre over K, for any kind K.) This functor has a lax left adjoint (strict) functor, viz. "add a dummy indeterminate", (this is essentially the K-combinator, as discussed in SEELY [1986].)

APPENDIX (Coherence considerations)

A.1 It is usual, when considering "lax" concepts, to require a host of coherence conditions for the various comparison 2-cells. Without being too precise, it turns out that insofar as LAMBDA is concerned, these can generally be subsumed under two principles: (BETA) of 3.4, and:

(beta comm) Beta conversions applied to different occurrences of logical symbols commute, (ie. it doesn't matter what order the beta conversions are done.)

Proceedings of the Symposium on Logic in Computer Science, 1987
(Computer Society of the IEEE 1987) 65–71

Although a related notion is considered in BARENDREGT [1981], from our point of view this principle seems rather more dubious than, say, (BETA): surely one ought to distinguish the order of steps in making a computation, and not merely the steps themselves. However, all the various naturality and coherence conditions suitable for sections 3,4 do seem to require (beta comm).

A.2 For the record, the coherence conditions referred to are the following. We suppose ι decreasing, as in section 3, and use the notation there for objects, morphisms, and 2-cells.

(For γ, ι):

$$\gamma(id(A),a)\cdot\iota(A)F(a) = id(F(a))$$
$$\gamma(a,id(B))\cdot F(a)\iota(B) = id(F(a))$$
$$\gamma(ab,c)\cdot\gamma(a,b)F(c) = \gamma(a,bc)\cdot F(a)\gamma(b,c)$$
$$\gamma(a',b')\cdot F(p)F(r) = F(pr)\cdot\gamma(a,b)$$

(and similarly for G.)

(For k): $k(A,id(B))\cdot\underline{A}(A,\iota(B))K(A,B) = K(A,B)$

$$k(A,b'b)\cdot\underline{A}(A,\gamma(b',b))K(A,B_2) = k(A,b')\underline{B}(FA,b)\cdot\underline{A}(A,Gb')k(A,b)$$

$$k(A,b')\cdot\underline{A}(A,G(p))K(A,B) = K(A,B_1)\underline{B}(FA,p)\cdot k(A,b)$$

(and similarly for l.)

(For η): $k(A,b)L(A,B)\cdot\underline{A}(A,Gb)\eta(A,B) = K(A,B_1)l(A,b)\cdot\eta(A,B_1)\underline{A}(A,Gb)$

(and similarly for ϵ; these give the "laxity" of the modifications η,ϵ.)

A.3 Similar conditions apply for the situation of section 4, with increasing ι.

(For e): $e(id(B)) = \beta(B)\iota(B)$
$$e(b'b)\cdot\beta(B_2)\gamma(b',b) = b'e(b)\cdot e(b')FG(b)$$
$$p\beta(B)\cdot e(b) = e(b')\cdot\beta(B_1)FG(p)$$

(and similarly for n; note the similarity with the conditions for k,l.)

(for ρ): $\rho(A)F(a)\cdot\beta(F(A))F(n(a)) = F(a)\rho(A_1)\cdot e(F(a))F(\alpha(A_1))$

(and similarly for σ; these give the "laxity" of the modifications ρ,σ.)

REFERENCES

H.P. BARENDREGT, The Lambda Calculus, (North-Holland, 1981).

J.-Y. GIRARD, The system F of variable types, fifteen years later, Theoretical Comp. Sci. 45 (1986) 2, 159-192.

J.W. GRAY, Formal Category Theory: Adjointness for 2-categories, (Springer LNM 391, 1974).

G.M. KELLY, ed., Category Seminar, (Springer LNM 420, 1974).

G.M. KELLY, R. STREET, Review of the elements of 2-categories, in KELLY [1974], 75-103.

G.M. KELLY, Basic Concepts of Enriched Category Theory, London Math. Soc. Lecture Note Series 64, (Cambridge University Press, 1982).

J. LAMBEK, P.J. SCOTT, Introduction to Higher Order Categorical Logic, (Cambridge University Press, 1986).

D.S. SCOTT, Continuous lattices, in F.W Lawvere, ed., Toposes, Algebraic Geometry and Logic, (Springer LNM 274, 1972).

D.S. SCOTT, Data types as lattices, SIAM J. Comput. 5 (1976) 522-587.

R.A.G. SEELY, Hyperdoctrines and Natural Deduction, Ph.D. Thesis, (University of Cambridge, 1977).

R.A.G. SEELY, Weak adjointness in proof theory, in M.P. Fourman, C.J. Mulvey, and D.S. Scott, eds., Applications of Sheaves, (Springer LNM 753, 1979), 697-701.

R.A.G. SEELY, Categorical semantics for higher order polymorphic lambda calculus, Math. Report 86-2, (McGill University 1986). (To appear in J. Symbolic Logic, 1987.)

Proceedings of the Symposium on Logic in Computer Science, 1987
(Computer Society of the IEEE 1987) 65–71