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 $F(A) = (A \& E)$, $G(A) = (E \Rightarrow A)$, then $F \dashv G$. 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, $\beta(F(A))\cdot F(\alpha(A)) = (\lambda\ z'$ 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, $G(\beta(B))\cdot\alpha(G(B)) = \lambda z$ in E. (1st($\lambda z'$ in E.$\langle y,z'\rangle$)(z))(2nd($\lambda z'$in E.$\langle y,z'\rangle$)(z)), where y is of type $E \Rightarrow B$, so that id(G(B)) = y; $\sigma(B)$ is the reduction: beta applied to each occurrence of (λ z' in E. $\langle y,z'\rangle$)(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) $\Longrightarrow$ a[t:=B].
(FORALL eta)	c $\Longrightarrow$ 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.)