(& 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

$$\gamma(F;\ a,b)\colon F(a)F(b) \Longrightarrow F(ab)$$
$$\iota(F;\ A)\colon id(FA) \Longrightarrow F(idA)$$

(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