The category of simply-typed lambda terms

Recall that in the simply-typed lambda calculus there is a countable set of atomic types and the set $\Phi \to$ of simple types is built up from the atomic types using the connective $\to$, i.e. all atomic types are simple types and if $\alpha, \beta$ are simple types then so is $\alpha \to \beta$. We say type for simple type.

For each type $\beta$ there is a countable set $\mathcal{V}_\beta$ of variables of type $\beta$, and if $\alpha \leftrightarrow \beta$ then $\mathcal{V}_\alpha \cap \mathcal{V}_\beta = \emptyset$. We write $x : \beta$ for $x \in \mathcal{V}_\beta$.

Let $\Lambda'$ denote the set of (untyped) lambda calculus preterms in the variables $\bigcup_{\beta \in \Phi} \mathcal{V}_\beta$. We define a subset $\Lambda'_{wt} \subseteq \Lambda'$ of well-typed preterms, together with a function $t : \Lambda'_{wt} \to \Phi \to$, by induction:

- all variables $x : \beta$ are well-typed and $t(x) = \beta$,
- if $M = (P \ Q)$ and $P, Q$ are well-typed with $t(P) = \beta \to \gamma$, $t(Q) = \gamma$ for some $\beta, \gamma$, then $M$ is well-typed and $t(M) = \gamma$.
- if $M = (\lambda x. N)$ with $N$ well-typed, then $M$ is well-typed and $t(M) = t(x) \to t(N)$.

We define $\Lambda'_\beta := \{ M \in \Lambda'_{wt} \mid t(M) = \beta \}$ and call these preterms of type $\beta$. Next we observe that $\Lambda'_{wt} \subseteq \Lambda'$ is closed under the relation of $\alpha$-equivalence on $\Lambda'$, as long as we understand $\alpha$-equivalence type-by-type (i.e. $\lambda x. M =_{\alpha} \lambda y. M[x := y]$ with $t(x) = t(y)$), and we may therefore define

$$\Lambda_{wt} := \Lambda'_{wt} / \sim_{\alpha}$$
$$\Lambda_\beta := \Lambda'_{\beta} / \sim_{\alpha}$$

so that $\Lambda_{wt}$ is the disjoint union of $\Lambda_\beta$ over all $\beta \in \Phi \to$. We write $M : \beta$ as a synonym for $[M] \in \Lambda_\beta$, and call these terms, of type $\beta$. 

Recall the equivalence relation $\equiv_{\beta}$ on $\Lambda_{\text{wt}}$ for each $\textsf{Z}$, generated by $\beta$-reduction (which is now typed)

\[
(\lambda x. M) \text{ N : } \textsf{J} \xrightarrow{\beta} M[x := \text{N}]
\]

well-typed, so $N : \textsf{i}$
and $t((\lambda x. M)) = \textsf{i} \to \textsf{J}$
with $t(x) = \textsf{i}$, $t(M) = \textsf{J}$

Once we impose $\equiv_{\beta}$, different types become "the same". The right way to make sense of this is to say the types are isomorphic objects of the category $\mathcal{L}$ of simple types and $\beta$-equivalence classes of terms. We begin with a motivating example of such an isomorphism, but first we need $\equiv_{\sharp}$-equivalence.

**Def** Let $\equiv_{\sharp}$ denote the smallest equivalence relation on $\Lambda_{\text{wt}}$ satisfying

- $\lambda x. (Mx) = \equiv_{\sharp} M$ for any $x \notin \text{FV}(M)$, $x : \textsf{Z}$, $M : \textsf{b} \to \textsf{J}$,
- if $M = \equiv_{\sharp} N$ then $\lambda x. M = \equiv_{\sharp} \lambda x. N$ for any variable $x$,
- if $M = \equiv_{\sharp} N$ then $(PM) = \equiv_{\sharp} (PN)$ whenever $P \in \Lambda_{\text{wt}}$ and $(PM), (PN)$ are well-typed
- if $M = \equiv_{\sharp} N$ then $(MP) = \equiv_{\sharp} (NP)$ whenever $P \in \Lambda_{\text{wt}}$ and $(MP), (NP)$ are well-typed.

**Note** There are good reasons to not impose $\equiv_{\sharp}$-equivalence, but for categorical approaches to $\lambda$-calculus (at least at a naive level) it's necessary.

**Def** For every type $\textsf{Z}$ let $\text{id}_{\textsf{Z}} := \lambda x. x$.

**Note** For any term $M : \textsf{Z}$, $(\text{id}_{\textsf{Z}} M) = \beta M$. 
Example  Let $b, t, p$ be types and consider

\[
\begin{align*}
T_1 &:= b \rightarrow (t \rightarrow p) \\
T_2 &:= t \rightarrow (b \rightarrow p)
\end{align*}
\]

we claim these are "isomorphic" types

Here is a term $M_{12}$ of type $T_1 \rightarrow T_2$, and $M_{21}$ of type $T_2 \rightarrow T_1$

\[
\begin{align*}
M_{12} &:= \lambda u. b \rightarrow (t \rightarrow p) \cdot \lambda v. t \cdot \lambda w. b \cdot ((u \cdot w) \cdot v) \\
M_{21} &:= \lambda u. t \rightarrow (b \rightarrow p) \cdot \lambda w. t \cdot \lambda v. b \cdot ((u \cdot v) \cdot w)
\end{align*}
\]

Recall from Sam's lecture that we compose $\lambda$-terms $F, G$ by taking $\lambda x. (F (G x))$ where $x \notin FV(F) \cup FV(G)$. Observe that for $t : T_2$,

\[
\begin{align*}
\lambda t. (M_{12} (M_{21} t)) &= \beta \lambda t. (M_{12} (\lambda w. t \cdot \lambda v. b \cdot ((u \cdot w) \cdot v))) \\
&= \beta \lambda t. \lambda v. \lambda w. ((u \cdot w) \cdot v) \cdot \lambda v. \lambda w. ((u \cdot w) \cdot v) \\
&= \beta \lambda t. \lambda v. ((u \cdot w) \cdot v) \\
&= \gamma \lambda t. (u \cdot w)
\end{align*}
\]

Similarly, for $s : T_1$,

\[
\begin{align*}
\lambda s. (M_{21} (M_{12} s)) &= \gamma \text{id}_{T_1}
\end{align*}
\]

So if we work up to $\beta$-equivalence, $M_{12}$ and $M_{21}$ behave like isomorphisms between $T_1$ and $T_2$, which "secretly" are $(b \times t) \rightarrow p$, $(t \times b) \rightarrow p$, and $b \times t \cong t \times b$, so these two types are "the same".
A category \( \mathcal{C} \) consists of:

1. A class \( \text{ob}(\mathcal{C}) \) whose elements are called objects of the category.

2. For each pair of objects \( A, B \) a set \( \text{C}(A, B) \) whose elements are called morphisms from \( A \) to \( B \) and are written \( f: A \rightarrow B \).
   (also called arrows from \( A \) to \( B \)).

3. For every triple of objects \( (A; B; C) \) a function
   \[ C_{ABC}: \text{C}(B, C) \times \text{C}(A, B) \rightarrow \text{C}(A, C) \]
   called composition and written \( g \circ f = C_{ABC}(g, f) \).

4. For each object \( A \), a morphism \( 1_A \in \text{C}(A, A) \) called the identity on \( A \).

Satisfying the following axioms:

1. **Associativity**: For any tuple \( (A; B; C; D) \) of objects and morphisms as indicated in the diagram
   \[
   A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D
   \]
   we have \( h \circ (g \circ f) = (h \circ g) \circ f \).

2. **Units**: For any morphism \( f: A \rightarrow B \) we have
   \[ 1_B \circ f = f = f \circ 1_A. \]
We take a nonstandard approach to defining a category of $\lambda$-terms. For a standard treatment see P. Taylor "Practical foundations for mathematics" or Lambek & Scott "Introduction to higher-order categorical logic".

We take as our guide the following desiderata for our category $\mathcal{L}$ constructed from simply-typed lambda calculus as defined above:

1. $\text{ob}(\mathcal{L}) = \Xi \to \eta \{1\}$ the set of simple types with an adjoined "empty" type, which we will see is basically forced on us, by 2.

2. Every $\lambda$-term is represented by a morphism in $\mathcal{L}$.

3. The operations on $\lambda$-terms (application and $\lambda$-abstraction) are represented by natural constructions in $\mathcal{L}$.

Notes
(a) We should take $\beta\eta$-equivalence classes of $\lambda$-terms (i.e. the set $\Lambda_{\text{wt}}$ of well-typed terms modulo $=\tau$) as morphisms, not just $\lambda$-terms, since our identities $\text{id}_Z : Z \rightarrow Z$ only work up to $=\tau$. Notice for $M : Z \rightarrow T$

$$\lambda x^k \left( M \left( \text{id}_Z \left( x \right) \right) \right) = \beta \lambda x. \left( M \left( x \right) \right) = \tau M.$$  

(b) Clearly then $\mathcal{L}(\Xi, T) = \Lambda_{Z \rightarrow T} / =\tau$, but what about terms of atomic type? For these we add a new object $1$ and declare

$$\mathcal{L}(1, Z) := \Lambda_{Z \rightarrow =\tau}$$

for any type $Z$ (not just atomic type. This wouldn't work, see below).
Note that with this definition, a single term in \( \Lambda 8 \to J \) is represented as both a morphism in \( \mathcal{L}(8, J) \) and as a morphism in \( \mathcal{L}(J, 8 \to J) \). This is OK.

Here is the formal definition:

**Def.** The category \( \mathcal{L} \) has

- **objects**: \( \text{ob}(\mathcal{L}) = \mathbb{I} \rightarrow \mathbb{I} \{1\} \)

- **morphisms**: for simple types \( 8, J \) we define

\[
\begin{align*}
\mathcal{L}(8, J) &:= \Lambda \beta \to \gamma / = \gamma \downarrow \{*\} \quad \mathcal{L}(8, \mathbb{I}) := \{*\} \\
\mathcal{L}(\mathbb{I}, 8) &:= \Lambda \beta / = \gamma \quad \mathcal{L}(\mathbb{I}, \mathbb{I}) := \{*\}
\end{align*}
\]

- **composition**: for simple types \( 8, J, P \)

\[
\begin{align*}
\mathcal{L}(J, P) \times \mathcal{L}(8, J) \xrightarrow{- \circ -} \mathcal{L}(8, P) \quad N \circ M := \begin{cases} 
\lambda x^8 \cdot (N \cdot (M \cdot x)) & M, N \neq * \\
* & \text{otherwise}
\end{cases}
\end{align*}
\]

where \( x \notin \text{FV}(N) \cup \text{FV}(M) \).

\[
\begin{align*}
\mathcal{L}(J, P) \times \mathcal{L}(\mathbb{I}, J) \xrightarrow{- \circ -} \mathcal{L}(\mathbb{I}, P) \\
\mathcal{L}(\mathbb{I}, P) \times \mathcal{L}(\mathbb{I}, \mathbb{I}) \xrightarrow{- \circ -} \mathcal{L}(\mathbb{I}, P)
\end{align*}
\]

\[
\begin{align*}
N \circ M &:= \begin{cases} 
(M, N) & N \neq * \\
* & \text{otherwise}
\end{cases} \\
N \circ * &:= N.
\end{align*}
\]

In all other cases \( 8 \to J \to P \) with \( 8 \neq \mathbb{I} \) but at least one of \( J, P \) equal to \( \mathbb{I} \), the composite is always \( * \).
Proposition \( \mathcal{L} \) is a category with identities \( \text{id}_b \in \mathcal{L}(b, b) \).

Proof. We need to check associativity and that identities work. For the former, consider the case of simple types \( b, T, P, S \)

\[
\begin{array}{c}
b \\ M \\ \downarrow \\ T \\ N \\ \downarrow \\ P \\ \downarrow \\ S \\
\end{array}
\]

with \( M, N, P \) not \( * \). Then in \( \mathcal{L} \) (\( = \) now means \( =_\gamma \))

\[
P \circ (N \circ M) = \lambda y^b ((P \circ (N \circ M) y))
\]

\[
= \lambda y^b (P ((\lambda x^b (N (M x))) y))
\]

\[
= \lambda y^b (P (N (M y)))
\]

\[
= (P \circ N) \circ M.
\]

The only other nontrivial case is

\[
\begin{array}{c}
1 \\ M \\ \downarrow \\ T \\ N \\ \downarrow \\ P \\ \downarrow \\ S \\
\end{array}
\]

where we calculate (\( M \in \Lambda_7 / =_\gamma \))

\[
P \circ (N \circ M) = P \circ (N M) = (P (N M))
\]

\[
(P \circ N) \circ M = (\lambda y^7 (P (N y)) M)
\]

\[
= (P (N M))
\]

so associativity holds. The identities follow as in (5.1) from \( \gamma \)-equivalence. \( \square \)
Example Returning to p.3, for types $\beta, \gamma, \rho$ we defined

\[ T_1 : = \beta \rightarrow (\gamma \rightarrow \rho), \quad T_2 : = \gamma \rightarrow (\beta \rightarrow \rho) \]

and we constructed $M_{12} \in \mathcal{L}(T_1, T_2), \ M_{21} \in \mathcal{L}(T_2, T_1)$ with

$M_{12} \circ M_{21} = \text{id}, \ M_{21} \circ M_{12} = \text{id}$, so $T_1 \cong T_2$ in $\mathcal{L}$. More generally

for types $\gamma_1, \ldots, \gamma_n$, $\beta, \rho$, we have an isomorphism for any

permutation $\sigma \in S_n$

\[ \gamma_1 \rightarrow \cdots \rightarrow \gamma_n \rightarrow \rho \cong \gamma_{\sigma(1)} \rightarrow \cdots \rightarrow \gamma_{\sigma(n)} \rightarrow \rho \]
It remains to discuss desiderata (3), i.e., how function application and $\lambda$-abstraction are represented in $L$. In the standard approach this is done by putting a Cartesian closed structure on a (different) category of $\lambda$-terms, in which we (a) add product types to our language, and associated constructor/destructor and modified $\beta$-equivalence, and (b) take as objects pairs $(x : \beta, \beta')$, and morphisms $(x : \beta, \beta') \rightarrow (y : \beta, \beta')$ are terms $M : \beta \rightarrow \beta$ with $FV(M) \subseteq \{x\}$.

Function application is just composition. Given $M : \beta$ and $N : \beta \rightarrow \beta$ we have by definition a commutative diagram in $L$

$$
\begin{array}{c}
1 \xrightarrow{M} \beta \\
\downarrow (N, M) \quad \downarrow N \\
\beta \quad \beta
\end{array}
$$

(8.1)

$\lambda$-abstraction is more complicated, and we treat it in Part II of these notes.