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 σ, τ are simple types then so is $\sigma \to \tau$. We say type for simple type. For each type σ there is a countable set Y_σ of variables of type σ, and if $\sigma \neq \tau$ then $Y_\sigma \cap Y_\tau = \phi$. We write $x : \sigma$ for $x \in Y_\sigma$.

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

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

We define $\Lambda'_\sigma := \{ M \in \Lambda'_{wt} \mid t(M) = \sigma \}$ and call these preterms of type σ. Next we observe that $\Lambda'_{wt} \subseteq \Lambda'$ is closed under the relation of α-equivalence on Λ', as long as we understand α-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

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

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

Recall the equivalence relation $=_\beta$ on Λ_σ for each σ, generated by β-reduction (which is now typed)

$$\underbrace{(\lambda x.M)\,N}_{\substack{\text{well-typed, so } N:\sigma \\ \text{and } t((\lambda x.M))=\sigma\to\tau \\ \text{with } t(x)=\sigma,\ t(M)=\tau}} : \tau \longrightarrow_\beta M[x := N]$$

Once we impose $=_\beta$, 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 β-equivalence classes of terms. We begin with a motivating example of such an isomorphism, but first we need η-equivalence.

Def^N Let $=_\eta$ denote the smallest equivalence relation on Λ_{wt} satisfying

- $\lambda x.(Mx) =_\eta M$ for any $x \notin FV(M)$, $x:\sigma$, $M:\sigma\to\tau$,
- if $M =_\eta N$ then $\lambda x.M =_\eta \lambda x.N$ for any variable x,
- if $M =_\eta N$ then $(P\,M) =_\eta (P\,N)$ whenever $P \in \Lambda_{wt}$ and $(P\,M), (P\,N)$ are well-typed
- if $M =_\eta N$ then $(M\,P) =_\eta (N\,P)$ whenever $P \in \Lambda_{wt}$ and $(M\,P), (N\,P)$ are well-typed.

Note There are good reasons to not impose η-equivalence, but for categorical approaches to λ-calculus (at least at a naive level) it's necessary.

Def^N For every type σ let $id_\sigma := \lambda x^\sigma.x$.

Note For any term $M:\sigma$, $(id_\sigma\,M) =_\beta M$.

<u>Example</u> Let σ, τ, ρ be types and consider

$$\left.\begin{aligned} T_1 &:= \sigma \to (\tau \to \rho) \\ T_2 &:= \tau \to (\sigma \to \rho) \end{aligned}\right\} \text{ we claim these are ``isomorphic'' types}$$

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

$$\begin{aligned} M_{12} &:= \lambda u^{\sigma \to (\tau \to \rho)}. \lambda v^{\tau}. \lambda w^{\sigma}. ((u\ w)\ v) \\ M_{21} &:= \lambda u^{\tau \to (\sigma \to \rho)}. \lambda w^{\sigma}. \lambda v^{\tau}. ((u\ v)\ w) \end{aligned}$$

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

$$\begin{aligned} \lambda t. (M_{12}\ (M_{21}\ t)) &=_\beta \lambda t. (M_{12}\ (\lambda w^{\sigma}. \lambda v^{\tau}. ((t\ v)\ w))) \\ &=_\beta \lambda t. \lambda \bar{v}^{\tau}. \lambda \bar{w}^{\sigma}. ((\lambda w^{\sigma}. \lambda v^{\tau}. ((t\ v)\ w))\ \bar{w})\ \bar{v}) \\ &=_\beta \lambda t. \lambda \bar{v}^{\tau}. \lambda \bar{w}^{\sigma}. ((\lambda v^{\tau}. ((t\ v)\ \bar{w}))\ \bar{v}) \\ &=_\beta \lambda t. \lambda \bar{v}^{\tau}. \lambda \bar{w}^{\sigma}. ((t\ \bar{v})\ \bar{w}) \\ &=_\eta \lambda t. \lambda \bar{v}^{\tau}. (t\ \bar{v}) \\ &=_\eta \lambda t. t \\ &= id_{T_2}. \end{aligned}$$

Similarly, for $s : T_1$,

$$\lambda s. (M_{21}\ (M_{12}\ s)) =_\eta id_{T_1}.$$

So if we work up to η-equivalence, M_{12} and M_{21} behave like <u>isomorphisms</u> between T_1 and T_2, which "secretly" are $(\sigma \times \tau) \to \rho$, $(\tau \times \sigma) \to \rho$, and $\sigma \times \tau \cong \tau \times \sigma$, so these two types are "the same".

<u>Def^N</u> A category $\mathscr{C}$ consists of

(1) A class $ob(\mathscr{C})$ whose elements are called <u>objects</u> of the category

(2) For each pair of objects A, B a set $\mathscr{C}(A,B)$ whose elements are called <u>morphisms</u> from A to B and are written $f : A \longrightarrow B$. (also called <u>arrows</u> from A to B).

(3) For every triple of objects (A,B,C) a function

$$c_{ABC} : \mathscr{C}(B,C) \times \mathscr{C}(A,B) \longrightarrow \mathscr{C}(A,C)$$

called <u>composition</u> and written $g \circ f = c_{ABC}(g,f)$.

(4) For each object A, a morphism $1_A \in \mathscr{C}(A,A)$ called the <u>identity</u> on A.

Satisfying the following axioms:

(1) <u>Associativity</u> 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) <u>Units</u> For any morphism $f : A \longrightarrow B$ we have

$$1_B \circ f = f = f \circ 1_A.$$

<u>Note</u> We take a nonstandard approach to defining a category of λ-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 <u>desiderata</u> for our category $\mathcal{L}$ constructed from simply-typed lambda calculus as defined above:

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

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

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

<u>Notes</u> (a) We should take $\beta\eta$-equivalence classes of λ-terms (i.e. the set Λ_{wt} of well-typed terms modulo $=_\eta$) as morphisms, not just λ-terms, since our identities $id_\sigma : \sigma \to \sigma$ only work up to $=_\eta$. Notice for $M : \sigma \to \tau$

$$\lambda x^\sigma.(M\ (id_\sigma\ x)) =_\beta \lambda x.(M\ x) =_\eta M. \qquad (5.1)$$
$$\lambda x^\sigma.(id_\tau\ (M\ x)) =_\beta \lambda x.(M\ x) =_\eta M.$$

(b) Clearly then $\mathcal{L}(\sigma, \tau) = \Lambda_{\sigma\to\tau} / =_\eta$, but what about terms of atomic type? For these we add a new object $\mathbb{1}$ and declare

$$\mathcal{L}(\mathbb{1}, \sigma) := \Lambda_\sigma / =_\eta$$

for any type σ (not just atomic types. This wouldn't work, see below).

(c) Note that with this definition, a single term in $\Lambda_{\sigma\to\tau}$ is represented as both a morphism in $\mathscr{L}(\sigma,\tau)$ and as a morphism in $\mathscr{L}(\mathbb{1},\sigma\to\tau)$. This is OK.

Here is the formal definition:

<u>Def^N</u> The category $\mathscr{L}$ has

- <u>objects</u>: $ob(\mathscr{L}) = \Phi_{\to} \amalg \{\mathbb{1}\}$

- <u>morphisms</u>: for simple types σ,τ we define

$$\mathscr{L}(\sigma,\tau) := \Lambda_{\sigma\to\tau}/{=_\eta} \amalg \{*\} \qquad \mathscr{L}(\sigma,\mathbb{1}) := \{*\}$$
$$\mathscr{L}(\mathbb{1},\sigma) := \Lambda_\sigma/{=_\eta} \qquad \mathscr{L}(\mathbb{1},\mathbb{1}) := \{*\}$$

one should check this is well-defined, even though FV(M), FV(N) is not β-invariant

- <u>composition</u>: for simple types σ,τ,ρ

$$\mathscr{L}(\tau,\rho)\times\mathscr{L}(\sigma,\tau) \xrightarrow{-\circ-} \mathscr{L}(\sigma,\rho) \qquad N\circ M := \begin{cases} \lambda x^\sigma.(N\ (M\ x)) & M,N\neq * \\ * & \text{otherwise} \end{cases}$$

where $x\notin FV(N)\cup FV(M)$.

$$\mathscr{L}(\tau,\rho)\times\mathscr{L}(\mathbb{1},\tau) \xrightarrow{-\circ-} \mathscr{L}(\mathbb{1},\rho) \qquad N\circ M := \begin{cases} (N\ M) & N\neq * \\ * & \text{otherwise} \end{cases}$$

$$\mathscr{L}(\mathbb{1},\rho)\times\mathscr{L}(\mathbb{1},\mathbb{1}) \longrightarrow \mathscr{L}(\mathbb{1},\rho) \qquad N\circ * := N.$$

In all other cases $\sigma\longrightarrow\tau\longrightarrow\rho$ with $\sigma\neq\mathbb{1}$ but at least one of τ,ρ equal to $\mathbb{1}$, the composite is always $*$.

<u>Proposition</u> $\mathcal{L}$ is a category with identities $\mathrm{id}_z \in \mathcal{L}(z,z)$.

<u>Proof</u> We need to check associativity and that identities work. For the former, consider the case of simple types z, τ, ρ, δ

$$z \xrightarrow{M} \tau \xrightarrow{N} \rho \xrightarrow{P} \delta$$

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

$$\begin{aligned} P\circ(N\circ M) &= \lambda y^z.(P\,(N\circ M\;y)) \\ &= \lambda y^z.(P\,(\,(\lambda x^z.(N\,(M\,x)))\;y\,)) \\ &= \lambda y^z.(P\,(N\,(M\,y))) \\ &= (P\circ N)\circ M. \end{aligned}$$

The only other nontrivial case is

$$1 \xrightarrow{M} \tau \xrightarrow{N} \rho \xrightarrow{P} \delta$$

where we calculate ($M \in \Lambda_\tau/{=_\eta}$)

$$P\circ(N\circ M) = P\circ(N\;M) = (P\,(N\;M))$$

$$\begin{tikzcd} z \arrow[r, "\mathrm{id}_z"] & z \\ * \arrow[u, "M:z"] \end{tikzcd} \quad \text{is } (\mathrm{id}_z\;M) = M$$

$$\begin{aligned} (P\circ N)\circ M &= (\lambda y^\tau.(P\,(N\,y))\;M) \\ &= (P\,(N\;M)) \end{aligned}$$

so associativity holds. The identities follow as in (5.1) from η-equivalence. $\square$

<u>Example</u> Returning to p.3, for types δ, τ, ρ we defined

$$T_1 := \delta \longrightarrow (\tau \to \rho), \quad T_2 := \tau \longrightarrow (\delta \longrightarrow \rho)$$

and we constucted $M_{12} \in \mathcal{L}(T_1, T_2)$, $M_{21} \in \mathcal{L}(T_2, T_1)$ with $M_{12} \circ M_{21} = id$, $M_{21} \circ M_{12} = id$, so $T_1 \cong T_2$ in $\mathcal{L}$. More generally for types $\tau_1, \dots, \tau_k, \delta, \rho$ we have an isomorphism for any permutation $\theta \in S_k$

$$\tau_1 \longrightarrow \cdots \longrightarrow \tau_k \longrightarrow \rho \cong \tau_{\theta(1)} \longrightarrow \cdots \longrightarrow \tau_{\theta(k)} \longrightarrow \rho$$

It remains to discuss desiderata 3, i.e. how function application and λ-abstraction are represented in $\mathcal{L}$. In the standard approach this is done by putting a Cartesian closed structure on a (different) category of λ-terms, in which we (a) add <u>product types</u> to our language, and associated constructors/destructors and <u>modified β-equivalence</u> and (b) take as objects pairs $(x:\sigma, \sigma)$, and morphisms $(x:\sigma,\sigma) \longrightarrow (y:\tau,\tau)$ are terms $M:\sigma \to \tau$ with $FV(M) \subseteq \{x\}$.

<u>Function application</u> is just composition. Given $M:\sigma$ and $N:\sigma\to\tau$ we have by definition a commutative diagram in $\mathcal{L}$

$$\begin{tikzcd} \mathbb{1} \arrow[r, "M"] \arrow[dr, "(N\ M)"'] & \sigma \arrow[d, "N"] \\ & \tau \end{tikzcd} \tag{8.1}$$

<u>λ-abstraction</u> is more complicated, and we treat it in Part II of these notes.