TYPE THEORY HW 1

The simply typed lambda calculus was preceded by the *untyped* lambda calculus. There are no types, only terms (or you can think that there is only one type that everything belongs to, never mentioned explicitly, and call the untyped lambda calculus the *unityped* lambda calculus).

The terms consist of the following:

(1) variables $x_1, ..., x_n$,
(2) a term st for every pair of terms s and t,
(3) a term $\lambda x_i.t$ for every variable x_i and term t.

with equations:

(1) $(\lambda x_i.t)s = t[s/x_i]$
(2) $\lambda x_i.(tx_i) = t$ (when x_i does not appear in t).

Turing's fixed point combinator in the untyped lambda calculus is defined to be:

$$\Theta := (\lambda xy.y(xxy))(\lambda xy.y(xxy))$$

We're using two conventions: that lambda terms associate to the left so that abc stands for $(ab)c$, and that $\lambda xy.$ stands for $\lambda x.\lambda y.$.

Exercise 1. What does it mean to be a fixed point of a term in this calculus? Show that Θt is a fixed point of t.

Exercise 2. Church *encoded* natural numbers (which here start at 0) in the untyped lambda calculus by representing $\overline{n}$ as the term that takes any term to its n-fold composition, so that, for example, $\overline{2}\ f$ is what we would denote by $f \circ f$ in usual mathematics. Write down the explicit formula for this.

Exercise 3. Encode multiplication in the untyped lambda calculus. That is, define a term mult such that $\mathsf{mult}\overline{m}\ \overline{n} = \overline{mn}$. Check that $\mathsf{mult}\overline{1}\overline{n} = \overline{n}$ for all natural numbers n.

Exercise 4. Suppose we have the following functions already encoded in the untyped lambda calculus (and if you have time, think about how you would define them):

- $\mathsf{ifzero\_then\_else}$ which has the property that $\mathsf{ifzero\_then\_else}\ \overline{0}fg = f$ and $\mathsf{ifzero\_then\_else}\ \overline{n}fg = g$ whenever $n \neq 0$,
- succ which has the property that $\mathsf{succ}\ \overline{n} = \overline{n+1}$,
- pred which has the property that $\mathsf{pred}\ \overline{0} = \overline{0}$ and $\mathsf{pred}\ \overline{n} = \overline{n-1}$ whenever $n \neq 0$.

You can *recursively* encode addition using the following idea. You want addition to have the following property:

$$\mathsf{add}\ \overline{m}\ \overline{n} = \mathsf{ifzero\_then\_else}\ \overline{n}\ \overline{m}\ (\mathsf{succ}(\mathsf{add}\ \overline{m}\ (\mathsf{pred}\ \overline{n}))).$$

Use Turing's fixed point combinator to turn this into a definition of `add`.

Exercise 5. In the *simply* typed lambda calculus, we can ask whether a type T has a fixed point combinator acting on terms of $T \implies T$. Let $\mathbb{B}$ denote the type defined by the following rules.

$$\frac{}{0 : \mathbb{B}} \qquad \frac{}{1 : \mathbb{B}}$$

$$\frac{T \;\; \text{TYPE} \qquad \Gamma \vdash t_0 : T \qquad \Gamma \vdash t_1 : T}{\Gamma, b : \mathbb{B} \vdash j_{(t_0,t_1)}(b) : T}$$

$$\frac{T \;\; \text{TYPE} \qquad \Gamma \vdash t_0 : T \qquad \Gamma \vdash t_1 : T}{\Gamma \vdash j_{(t_0,t_1)}(0) = t_0 : T} \qquad \frac{T \;\; \text{TYPE} \qquad \Gamma \vdash t_0 : T \qquad \Gamma \vdash t_1 : T}{\Gamma \vdash j_{(t_0,t_1)}(1) = t_1 : T}$$

Intuitively, this just means that there are two terms $0, 1 : \mathbb{B}$, and that given two terms $t_0, t_1 : T$, we can always form a sort of function $j_{(t_0,t_1)}$ from $\mathbb{B}$ to T such that $j_{(t_0,t_1)}(0)$ is the original t_0 that we started with and $j_{(t_0,t_1)}(1)$ is the original t_1 that we started with.

Prove or disprove that $\mathbb{B}$ has a fixed point combinator.

Exercise 6. Given types R, S, T in the simply typed lambda calculus with $\implies$-types, construct a term of the type

$$(R \implies (S \implies T)) \implies (S \implies (R \implies T)).$$

Exercise 7. Given types S, T in the simply typed lambda calculus with $\implies$- and $\wedge$-types, construct a term of the type

$$S \implies (T \implies (S \wedge T)).$$