Programming Languages and Types

Exercise 12

Yi Dai

February 1, 2013

1 Simply-Typed λ-Calculus

1.1 Typing Derivations

1.2 Programming with Extensions

1. Define the subtraction function *sub* in the simply-typed λ-calculus extended with Peano numbers (**0** and **succ**) and fixed point operator **fix**.[1]

$$
\begin{array}{l}
\textbf{fix}\ (\lambda\ fsub : (\textbf{Nat} \to \textbf{Nat} \to \textbf{Nat}) \to (\textbf{Nat} \to \textbf{Nat} \to \textbf{Nat})\ . \\
\quad \lambda\ m : \textbf{Nat}\ .\ \lambda\ n : \textbf{Nat} \\
\quad\quad \textbf{if}\ (\textbf{iszero}\ m) \\
\quad\quad\quad \textbf{then}\ \textbf{0} \\
\quad\quad\quad \textbf{else}\ (\textbf{succ}\ (fsub\ (\textbf{pred}\ m)\ n))\ )
\end{array}
$$

2 System-$\mathcal{F}$

2.1 Parametric Polymorphism

1. Define the polymorphic function I.

2. Define the polymorphic function K.

[1] I apologize for giving the wrong type (**Nat** $\to$ **Nat** $\to$ **Nat**) to the parameter $fsub$ during the exercise session.

3. Define the polymorphic function S.
4. Define I in terms of S and K.

2.2 Typing Church-Encodings

1. Church-numerals
2. Church-booleans
3. Church-pairs