1 Lambda calculus encodings

(a) Evaluate \texttt{AND FALSE TRUE} under CBV semantics.

(b) Evaluate \texttt{IF FALSE }\Omega \lambda x. x\texttt{ under CBN semantics. What happens when you evaluated it under CBV semantics?}

(c) Evaluate \texttt{ADD 2 T} under CBV semantics. (Make sure you know what the Church encoding of 1 and 2 are, and check that the answer is equal to the Church encoding of 3.)

(d) In class we made use of a combinator \texttt{ISZERO}, which takes a Church encoding of a natural number \(n\), and evaluates to \texttt{TRUE} if \(n\) is zero, and \texttt{FALSE} if \(n\) is not zero. (We don’t care what \texttt{ISZERO} does if it is applied to a lambda term that is not a Church encoding of a natural number.) Define \texttt{ISZERO}.

2 Recursion

Assume we have an applied lambda calculus with integers, booleans, conditionals, etc. Consider the following higher-order function \(H\).

\[
H \triangleq \lambda f. \lambda n. \begin{cases} 
\text{true} & \text{if } n = 1 \\
\text{false} & \text{if } n = 0 \\
\neg (f (n - 1)) & \text{otherwise}
\end{cases}
\]

(a) Suppose that \(g\) is the fixed point of \(H\). What does \(g\) compute?

(b) Compute \(Y H\) under CBN semantics. What has happened to the function call \(f (n - 1)\)?

(c) Compute \((Y H) 2\) under CBN semantics.

(d) Use the “recursion removal trick” to write another function that behaves the same as the fixed point of \(H\).

3 Evaluation context

Consider the lambda calculus with let expressions and pairs (§1.3 of Lecture 9), and a semantics defined using evaluation contexts. For each of the following expressions, show one step of evaluation. Be clear about what the evaluation context is.

(a) \((\lambda x. x) (\lambda y. y) (\lambda z. z)\)

(b) \texttt{let } x = 5 \texttt{ in } (\lambda y. y + x) 9

(c) \((4, ((\lambda x. x) 8, 9))\)

(d) \texttt{let } x = \#1 ((\lambda y. y) (3, 4)) \texttt{ in } x + 2
4 Definitional translations

Consider an applied lambda calculus with booleans, conjunction, a few constant natural numbers, and addition, whose syntax is defined as follows.

\[ e ::= x \mid \lambda x. e \mid e_1 \ e_2 \mid \text{true} \mid \text{false} \mid \text{e}_1 \text{ and } \text{e}_2 \mid 0 \mid 1 \mid 2 \mid e_1 + e_2 \mid \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \]

Give a translation to the pure lambda calculus. Use the encodings of booleans and natural numbers that we considered in class.