Rafael Fernández Ortiz

Assignments 1

Exercise 1. Classify the following terms according to α-equivalence:

$$\lambda x.xy,\ \lambda x.xz,\ \lambda y.yz,\ \lambda z.zz,\ \lambda z.zy,\ \lambda f.fy,\ \lambda f.ff,\ \lambda y.\lambda x.xy,\ \lambda z.\lambda y.yz$$

Provide an α-equivalent term where each abstraction uses a different variable name:

$$\lambda x.((x(\lambda y.xy))(\lambda x.x))(\lambda y.yx)$$

Solution

Let e a λ-term and X a set of λ-terms, we can define

$$\bar{e} := X \Big/_{\underset{\alpha}{\rightsquigarrow} e}$$

the set that contains all λ-terms in X such that they are α-equivalent to e. We can consider $\mathcal{O}$ the set of all λ-term in the statement, i.e. $\mathcal{O} = \{\lambda x.xy,\ \lambda x.xz,\ \lambda y.yz,\ \lambda z.zz,\ \lambda z.zy,\ \lambda f.fy,\ \lambda f.ff,\ \lambda y.\lambda x.xy,\ \lambda z.\lambda y.yz\}$.

$$\begin{array}{l}
\lambda x.xy \underset{\alpha}{\rightsquigarrow} \lambda z.zy \\
\lambda z.zy \\
\lambda f.fy \underset{\alpha}{\rightsquigarrow} \lambda z.zy \\
\\
\lambda x.xz \\
\lambda y.yz \underset{\alpha}{\rightsquigarrow} \lambda x.xz \\
\\
\lambda z.zz \\
\lambda f.ff \underset{\alpha}{\rightsquigarrow} \lambda z.zz \\
\\
\lambda y.\lambda x.xy \underset{\alpha}{\rightsquigarrow} \lambda z.\lambda x.xz \underset{\alpha}{\rightsquigarrow} \lambda z.\lambda y.yz \\
\lambda z.\lambda y.yz
\end{array}$$

Therefore

$$\begin{aligned}
\mathcal{O} \Big/_{\underset{\alpha}{\rightsquigarrow} \lambda z.zy} &= \{\lambda x.xy,\ \ \lambda z.zy,\ \ \lambda f.fy\} \\
\mathcal{O} \Big/_{\underset{\alpha}{\rightsquigarrow} \lambda x.xz} &= \{\lambda x.xz,\ \ \lambda y.yz\} \\
\mathcal{O} \Big/_{\underset{\alpha}{\rightsquigarrow} \lambda z.zz} &= \{\lambda z.zz,\ \ \lambda f.ff\} \\
\mathcal{O} \Big/_{\underset{\alpha}{\rightsquigarrow} \lambda z.\lambda y.yz} &= \{\lambda z.\lambda y.yz,\ \ \lambda y.\lambda x.xy\}
\end{aligned}$$

Now we will provide and α-equivalence for the following term $\lambda x.((x(\lambda y.xy))(\lambda x.x))(\lambda y.yx)$ where each abstraction uses a different variable name.

$$\begin{array}{l}
\lambda y.xy \underset{\alpha}{\rightsquigarrow} \lambda z_1.xz_1 \\
\lambda x.x \underset{\alpha}{\rightsquigarrow} \lambda z_2.z_2 \\
\lambda y.yx \underset{\alpha}{\rightsquigarrow} \lambda z_3.z_3x
\end{array}$$

Therefore, $\lambda x.((x(\lambda y.xy))(\lambda x.x))(\lambda y.yx) \underset{\alpha}{\rightsquigarrow} \lambda a.((a(\lambda z_1.az_1))(\lambda z_2.z_2))(\lambda z_3.z_3 a)$

■

Exercise 2. Normalize the following term: $(\lambda x.(\lambda y.xy))y$

Solution

$$(\lambda x.(\lambda y.xy))y \underset{\alpha}{\rightsquigarrow} (\lambda x.(\lambda z.xz))y \underset{\beta}{\rightsquigarrow} \lambda z.yz$$

■

Exercise 3. The *de Brujin index notation* is a way of avoiding the problems related to substitution and variable capture in Church's original presentation of the λ-calculus, thus facilitating its mechanized treatment. The key idea is to replace variable names by numbers denoting the *depth* of the scope of that variable. For example, the familiar terms, $\lambda x.x$, $\lambda x.\lambda y.x$ and $\lambda x.\lambda y.y$ are represented as $\lambda 1$, $\lambda\lambda 2$, $\lambda\lambda 1$ in de Brujin's notation. Free variables are represented by numbers higher than de maximum depth in its location. For example, $\lambda\lambda 3$ is a possible representation for $\lambda x.\lambda y.w$.

a) Represent the term $(\lambda x.\lambda y.\lambda z.xzy)(\lambda x.\lambda y.x)$ in de Brujin's notation.

b) Explain how β-reduction of terms in de Brujin notation can be implemented.

c) Apply your ideas to the application in a).

Solution

a)

$$(\lambda x.\lambda y.\lambda z.xzy)(\lambda x.\lambda y.x) \longrightarrow (\lambda\lambda\lambda 312)(\lambda\lambda 2)$$

b)

In order to find some mechanism to do a "β-Brujin-reduction", we can see some pattern in the way to do β-reduction for our λ-term. Therefore, we will do β-reduction in our expression as follow:

$$\begin{aligned}
&(\lambda x.\lambda y.\lambda z.xyz)(\lambda x.\lambda y.x) \underset{\alpha}{\rightsquigarrow} (\lambda x.\lambda y.\lambda z.xyz)(\lambda x_0.\lambda x_1.x_0) \underset{\beta}{\rightsquigarrow} \\
&\underset{\beta}{\rightsquigarrow} (\lambda y.\lambda z.(\lambda x_0.\lambda x_1.x_0)zy) \underset{\beta}{\rightsquigarrow} \lambda y.\lambda z.(\lambda x_1.z)y \underset{\beta}{\rightsquigarrow} \\
&\underset{\beta}{\rightsquigarrow} \lambda y.\lambda z.z
\end{aligned}$$

We can observe that the idea is to reduce first the leftmost outermost redex in our λ-term. In a Brujin approach we can do something similar:

1. Identify which one is the number related to the first leftmost outermost lambda.
2. Replace this number by the following de Brujin term and drop the correspond lambda.
3. If the de Brujin term has free variable, then reduce the number by 1.
4. If apply a number by some lambda, we increase 1, i.e, $(\underline{\lambda}\lambda\underline{2})\underline{1} = \lambda(1+1) = \lambda 2$

c)

$$\begin{aligned}
(\underline{\lambda}\lambda\lambda\underline{3}12)\underline{(\lambda\lambda 2)} \rightarrow &\quad (\lambda\lambda(\underline{\lambda}\lambda\underline{2})\underline{1}2) \rightarrow \\
\rightarrow &\quad (\lambda\lambda(\lambda(1+1))2) \rightarrow \\
\rightarrow &\quad (\lambda\lambda(\underline{\lambda}2)2) \rightarrow \\
\rightarrow &\quad (\lambda\lambda(2-1)) \rightarrow \\
\rightarrow &\quad (\lambda\lambda 1)
\end{aligned}$$

■

Exercise 4. *Combinators* can be seen as λ-terms without free variables - althought they were actually proposed indepently from the λ-calculus. Given the combinators $\mathrm{S} = \lambda x.\lambda y.\lambda z.(xz)(yz)$, $\mathrm{K} = \lambda x.\lambda y.x$, and $\mathrm{I} = \lambda x.x$, prove the equivalence $\mathrm{SKK} = \mathrm{I}$.

Solution

$$
\begin{aligned}
\mathrm{SKK} = \quad & (\lambda x.\lambda y.\lambda z.(xz)(yz))(\lambda x.\lambda y.x)(\lambda x.\lambda y.x) \underset{\alpha}{\rightsquigarrow} \\
& \underset{\alpha}{\rightsquigarrow} (\lambda x.\lambda y.\lambda z.(xz)(yz))(\lambda z_0.\lambda z_1.z_0)(\lambda x_0.\lambda x_1.x_0) \underset{\beta}{\rightsquigarrow} \\
& \underset{\beta}{\rightsquigarrow} (\lambda y.\lambda z.((\lambda z_0.z_1 z_0)z)(yz))(\lambda x_0.\lambda x_1.x_0) \underset{\beta}{\rightsquigarrow} \\
& \underset{\beta}{\rightsquigarrow} (\lambda z.((\lambda z_0.\lambda z_1.z_0)z)((\lambda x_0.\lambda x_1.x_0)z)) \underset{\beta}{\rightsquigarrow} \\
& \underset{\beta}{\rightsquigarrow} (\lambda z.(\lambda z_1.z)((\lambda x_0.\lambda x_1.x_0)z)) \underset{\beta}{\rightsquigarrow} \\
& \underset{\beta}{\rightsquigarrow} \lambda z.(\lambda z_1.z)(\lambda x_1.z) \underset{\beta}{\rightsquigarrow} \\
& \underset{\beta}{\rightsquigarrow} \lambda z.z = \mathrm{I}
\end{aligned}
$$

■

Exercise 5. Combinators systems are commonly presented as equational theories where a *combinator base* is defined using oriented equational rules and new combinators are created by means of application alone - no abstraction is required. For example, the aforementioned combinators would be defined by the equations $\mathrm{S}mno = mo(no)$, $\mathrm{K}ab = a$ and $\mathrm{I}x = x$. As variales only appear in definitions where no confusion of scope can happen, combinators solve in a natural way many of the nuissances associated with variable names in Church's formulation of the λ-caluclus.

Take as base the following two combinators: $\mathrm{B}fgx = f(gx)$ and $\mathrm{M}x = xx$. Using B and M alone prove the existence of a *narcissistic* combinator n such that $nn = n$.

Solution

ToDo

■

Exercise 6. Using Church's encoding of booleans in the pure λ-calculus, define normalized λ-terms CONJ, DISJ, and NEG to represent conjunction, disjunction and negation, respectively. (**Hint.** define a λ-term COND which behaves as an *if-then-else* and apply β-reduction).

Solution

We can define COND combinator as follow:

$$\mathrm{COND} \quad = \quad \lambda b.\lambda x.\ \lambda y.b\ x\ y$$

Therefore, the simplest idea to define CONJ AND DISJ is using COND:

$$
\begin{aligned}
\mathrm{CONJ} &= \lambda b_1.\lambda b_2.\ \mathrm{COND}\ b_1\ b_2\ \mathrm{FALSE} \\
\mathrm{NEG} &= \lambda b_1.\ \mathrm{COND}\ b_1\ \mathrm{FALSE}\ \ \mathrm{TRUE} \\
\mathrm{DISJ} &= \lambda b_1.\lambda b_2.\ \mathrm{COND}\ b_1\ \mathrm{TRUE}\ \ b_2
\end{aligned}
$$

But, if we do β-reductions, the combinators defined above are β-equivalent to the following lambda terms:

$$
\begin{aligned}
\mathrm{CONJ} &= \lambda b_1.\lambda b_2.\ b_1\ b_2\ \mathrm{FALSE} \\
\mathrm{NEG} &= \lambda b_1.\ b_1\ \mathrm{FALSE}\ \ \mathrm{TRUE} \\
\mathrm{DISJ} &= \lambda b_1.\lambda b_2.\ b_1\ \mathrm{TRUE}\ \ b_2
\end{aligned}
$$

■

Exercise 7. Using Church's encoding of natural numbers define addition, multiplication and exponentiation.

Solution

$$\begin{array}{rcl} \text{ADD} & = & \lambda m.\lambda n.\lambda f.\lambda x.\ mf(nfx) \\ \text{MUL} & = & \lambda m.\lambda n.\ m\ (\text{ADD}\ \ n)\ \ \bar{0} \\ \text{EXP} & = & \lambda m.\lambda n.\ n\ (\text{MUL}\ \ m)\ \ \bar{1} \end{array}$$

■

Exercise 8. Devise an encoding for pairs in the λ-caluclus. Provide λ-terms PAIR (a pair constructor) and the projections FST and SND. What properties would be required in order to check the correctness of the encoding?

Solution

$$\begin{array}{rcl} \text{PAIR} & = & \lambda x.\lambda y.\lambda f.\ f\ x\ y \\ \text{FST} & = & \lambda p.\ p\ \text{TRUE} \\ \text{SND} & = & \lambda p.\ p\ \text{FALSE} \end{array}$$

In order to check the correctness of the encoding, we will prove that FST (PAIR a b) $\underset{\beta}{\rightsquigarrow}$ a and SND (PAIR a b) $\underset{\beta}{\rightsquigarrow}$ b.

$$\begin{array}{rl} \text{FST (PAIR } a\ b) = & \lambda p.\ p\ \text{TRUE (PAIR } a\ b) \underset{\beta}{\rightsquigarrow} \\ & \underset{\beta}{\rightsquigarrow} \text{(PAIR } a\ b)\ \text{TRUE} \underset{\beta}{\rightsquigarrow} (\lambda x.\lambda y.\lambda f.fx\ y)\ a\ b\ \text{TRUE} \underset{\beta}{\rightsquigarrow} \\ & \underset{\beta}{\rightsquigarrow} (\lambda y.\lambda f.f\ a\ y)\ b\ \text{TRUE} \underset{\beta}{\rightsquigarrow} (\lambda f.f\ a\ b)\ \text{TRUE} \underset{\beta}{\rightsquigarrow} \\ & \underset{\beta}{\rightsquigarrow} \text{TRUE}\ a\ b \underset{\beta}{\rightsquigarrow} (\lambda x.\lambda y.x)\ a\ b \underset{\beta}{\rightsquigarrow} (\lambda y.a)\ b \underset{\beta}{\rightsquigarrow} a \end{array}$$

$$\begin{array}{rl} \text{SND (PAIR } a\ b) = & \lambda p.\ p\ \text{FALSE (PAIR } a\ b) \underset{\beta}{\rightsquigarrow} \\ & \underset{\beta}{\rightsquigarrow} \text{(PAIR } a\ b)\ \text{FALSE} \underset{\beta}{\rightsquigarrow} (\lambda x.\lambda y.\lambda f.fx\ y)\ a\ b\ \text{FALSE} \underset{\beta}{\rightsquigarrow} \\ & \underset{\beta}{\rightsquigarrow} (\lambda y.\lambda f.f\ a\ y)\ b\ \text{FALSE} \underset{\beta}{\rightsquigarrow} (\lambda f.f\ a\ b)\ \text{FALSE} \underset{\beta}{\rightsquigarrow} \\ & \underset{\beta}{\rightsquigarrow} \text{FALSE}\ a\ b \underset{\beta}{\rightsquigarrow} (\lambda x.\lambda y.y)\ a\ b \underset{\beta}{\rightsquigarrow} (\lambda y.y)\ b \underset{\beta}{\rightsquigarrow} b \end{array}$$

■

Exercise 9. Define a predecessor function and subtraction for Church numerals. (**Hint.** You may use the pairs just defined).

Solution

ToDo

■

Exercise 10. Imagine that the list $[a_1, a_2, \dots a_n]$ is represented in the lambda calculus by the term

$$\lambda f.\lambda x.fa_1(fa_2(\dots fa_nx))$$

Define:

a) NIL, the empty list constructor

b) APP, a function to concatenate two lists

c) HD, which returns the first element of a nonempty list

d) ISEMPTY, a function to check whether a list is empty

Solution

$$
\begin{array}{lcl}
\text{NIL} & = & \text{FALSE} \\
\text{HD} & = & \lambda l.\ l\ \text{TRUE} \\
\text{APP} & = & \lambda xs.\lambda ys.\lambda f.\lambda x.\ xs\ f\ (ys\ f\ x) \\
\text{ISEMPTY} & = & \lambda xs.\ xs\ (\lambda h.\lambda t.\ \text{FALSE})\ \text{TRUE}
\end{array}
$$

$\blacksquare$