CMPSC 461: Programming Language Concepts
Assignment 1 Solution

Problem 1 [6pt] Add parentheses to the following lambda terms so that the grouping of sub-terms becomes explicit. For example, the term $\lambda x.\ x\ \lambda y.\ y$ with parentheses is $\lambda x.\ (x\ (\lambda y.\ y))$.

a) (3pt) $\lambda x.\ \lambda y.\ z\ x\ y$

Solution: $\lambda x.\ (\lambda y.\ ((z\ x)\ y))$

b) (3pt) $\lambda x.\ \lambda y.\ (\lambda x.\ x)\ y\ \lambda z.\ x\ y$

Solution: $\lambda x.\ (\lambda y.\ (((\lambda x.\ x)\ y)\ (\lambda z.\ (x\ y))))$

Problem 2 [6pt] Fully evaluate the following λ-term so that no further β-reduction is possible.

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

Solution:

$$\begin{aligned}
&((\lambda x.\ (\lambda y.\ (y\ y))\ x)\ y)\ (\lambda x.\ x) \\
&= ((\lambda x.\ (\lambda z.\ (z\ z))\ x)\ y)\ (\lambda x.\ x) && (\alpha - \text{reduction}) \\
&= ((\lambda z.\ (z\ z))\ y)\ (\lambda x.\ x) && (\beta - \text{reduction}) \\
&= (y\ y)\ (\lambda x.\ x) && (\beta - \text{reduction})
\end{aligned}$$

An Alternative Solution:

$$\begin{aligned}
&((\lambda x.\ (\lambda y.\ (y\ y))\ x)\ y)\ (\lambda x.\ x) \\
&= ((\lambda x.\ x\ x)\ y)\ (\lambda x.\ x) && (\beta - \text{reduction}) \\
&= (y\ y)\ (\lambda x.\ x) && (\beta - \text{reduction})
\end{aligned}$$

Problem 3 [10pt] Another way of defining addition on Church numbers is the following:

$$+ \triangleq \lambda n_1\ n_2\ f\ z\ .\ (n_1\ f\ (n_2\ f\ z))$$

Show that $(+\ \underline{2}\ \underline{2}) = \underline{4}$ under Church encoding, where $\underline{n} \triangleq \lambda f\ z\ .\ f^n\ z$.

Solution:

$$
\begin{aligned}
& + \ \underline{2}\ \underline{2} \\
&= (\lambda n_1\ n_2\ f\ z\ .\ (n_1\ f\ (n_2\ f\ z)))\ \underline{2}\ \underline{2} && \text{(definition of } +) \\
&= (\lambda f\ z\ .\ (n_1\ f\ (n_2\ f\ z)))\{\underline{2}/n_1\}\{\underline{2}/n_2\} && (\beta - \text{reduction}) \\
&= \lambda f\ z.\ (\underline{2}\ f\ (\underline{2}\ f\ z))) \\
&= \lambda f\ z.\ (\underline{2}\ f\ ((\lambda f\ z\ .\ f^2\ z)\ f\ z))) && \text{(definition of } \underline{2}) \\
&= \lambda f\ z.\ (\underline{2}\ f\ (f^2\ z)) && (\beta - \text{reduction}) \\
&= \lambda f\ z.\ ((\lambda f\ z\ .\ f^2\ z)\ f\ (f^2\ z)) && \text{(definition of } \underline{2}) \\
&= \lambda f\ z.\ ((f^2\ z)\{f/f\}\{(f^2\ z)/z\} && (\beta - \text{reduction}) \\
&= \lambda f\ z.\ (f^2\ (f^2\ z)) \\
&= \lambda f\ z.\ (f\ (f\ (f\ (f\ z)))) && \text{(expand } f^2) \\
&= \underline{4} && \text{(definition of } \underline{4})
\end{aligned}
$$

Problem 4 [12pt] Using the pure λ-calculus, we can encode a pair of λ-terms as where a term `PAIR` takes three λ-terms a, b, p and applies p to a and b. That is, we can define `PAIR` as $(\lambda a\ b\ p.\ p\ a\ b)$.

a) (4pt) Based on this definition, we can encode function `RIGHT` as $\texttt{RIGHT} \triangleq \lambda p.\ (p\ (\lambda t\ f.\ f))$. Show that under such encoding, `RIGHT` (`PAIR` $x\ y$) evaluates to y.

Solution:

$$
\begin{aligned}
& \texttt{RIGHT}\ (\texttt{PAIR}\ x\ y) \\
&= (\lambda p.\ (p\ (\lambda t\ f\ .\ f)))\ (\texttt{PAIR}\ x\ y) && \text{(definition of RIGHT)} \\
&= (\texttt{PAIR}\ x\ y)\ (\lambda t\ f\ .\ f) && (\beta - \text{reduction}) \\
&= ((\lambda a\ b\ p.\ p\ a\ b)\ x\ y)\ (\lambda t\ f\ .\ f) && \text{(definition of PAIR)} \\
&= ((\lambda a\ b.\ \lambda p.\ p\ a\ b)\ x\ y)\ (\lambda t\ f\ .\ f) && \text{(desugar } \lambda a\ b\ p) \\
&= (\lambda p.\ p\ x\ y)\ (\lambda t\ f\ .\ f) && (\beta - \text{reduction}) \\
&= (\lambda t\ f\ .\ f)\ x\ y && (\beta - \text{reduction}) \\
&= y
\end{aligned}
$$

b) (4pt) Try to define a function `LEFT` in λ-calculus so that given a pair, it returns the first term in the pair. For example, `LEFT` (`PAIR` $x\ y$) should evaluate to x under your encoding.

Solution:

$$\texttt{LEFT} \triangleq \lambda p.\ (p\ (\lambda t\ f.\ t)) \quad (\text{or equivalently, } \lambda p.\ (p\ \texttt{TRUE}))$$

c) (4pt) Try to define a function `REPLACE` in λ-calculus so that given a pair, it "replaces" the second component with the first component in a pair. For example, `REPLACE` (`PAIR` $x\ y$) should evaluate to (`PAIR` $x\ x$) (or some equivalent term under α/β-reduction).

Solution:

$$\texttt{REPLACE} \triangleq \lambda p.\ (p\ (\lambda t\ f.\ \texttt{PAIR}\ t\ t))$$

$$(\text{ or }) \ \texttt{REPLACE} \triangleq \lambda p.\ (\texttt{PAIR}\ (\texttt{LEFT}\ p)\ (\texttt{LEFT}\ p))$$

Problem 5 [16pt] In the lecture, we have used the pure λ-calculus to construct natural numbers and encoded some operations on them. In such encoding, a number n is encoded as a function $\lambda f\ z.\ f^n\ z$ (denoted as $\underline{n}$). In this problem, we will define more useful operations on numbers. If you get stuck in any question, try to proceed by assuming the previous function is properly defined. You can reuse any encoding given in lectures and previous questions.

a) (4pt) Define a function ISZERO in λ-calculus, so that given a number $\underline{n}$, it returns TRUE (the encoding of true) if $\underline{n} = \underline{0}$; FALSE (the encoding of false) if $\underline{n} \neq \underline{0}$. Hint: try to define two terms so that applying the second term multiple times to the first term returns FALSE; otherwise, the application returns TRUE.
Solution:
$\texttt{ISZERO} \triangleq \lambda n.\ n\ (\lambda f.\ \texttt{FALSE})\ (\texttt{TRUE})$

b) (4pt) Define a function PRED in λ-calculus, so that given a number $\underline{n}$, the function returns its predecessor, assuming the predecessor of $\underline{0}$ is $\underline{0}$ Hint: follow the idea in Problem 5a. You might need to use PAIR.
Solution:
$\texttt{PRED} \triangleq \lambda n.\ \texttt{RIGHT}\ (n\ (\lambda p.\ \texttt{PAIR}\ (\texttt{SUCC}\ (\texttt{LEFT}\ p))\ (\texttt{LEFT}\ p))\ (\texttt{PAIR}\ \underline{0}\ \underline{0}))$

c) (4pt) Use your encoding of PRED to define a subtraction function MINUS, so that MINUS $\underline{n_1}$ $\underline{n_2}$ returns $\underline{n_1 - n_2}$ when $n_1 \geq n_2$, and $\underline{0}$ otherwise.
Solution:
$\texttt{MINUS} \triangleq \lambda n_1\ n_2.\ n_2\ \texttt{PRED}\ n_1$

d) (4pt) Based on what you have encoded so far, and the Boolean encodings in the lecture, define an equality test, called EQUALS on two numbers, so that it returns TRUE when two numbers are equal, and FALSE otherwise.
Solution:
$\texttt{EQUALS} \triangleq \lambda n_1\ n_2.\ \texttt{AND}\ (\texttt{ISZERO}\ (\texttt{MINUS}\ n_1\ n_2))\ (\texttt{ISZERO}\ (\texttt{MINUS}\ n_2\ n_1))$