# Foundations1 assignments I 2020
# Assignment I: Submit by Wednesday of week 5 (14 October 2020) by 15:30 pm.
# worth 15%
# Submit only typed material through assessment on vision
# No handwritten material
# SUBMIT ONE SINGLE FILE IN PDF ONLY FOR THE ASNWERS+ Submit your entire SML program AS ONE SINGLE SML FILE
# NO ZIP FILES. NO MULTIPLE FILES. JUST ONE PDF FOR ANSWERS AND ONE SML FOR ALL THE CODE.

## Background

- The syntax of the classical $\lambda$-calculus is given by $\mathcal{M} ::= \mathcal{V} \mid (\lambda\mathcal{V}.\mathcal{M}) \mid (\mathcal{M}\mathcal{M})$.
  We assume the usual notational conventions in $\mathcal{M}$ and use the reduction rule:
  $\underline{(\lambda v.P)Q} \rightarrow_\beta P[v := Q]$.
- The syntax of the $\lambda$-calculus in item notation is given by $\mathcal{M}' ::= \mathcal{V} \mid [\mathcal{V}]\mathcal{M}' \mid \langle\mathcal{M}'\rangle\mathcal{M}'$.
  We use the reduction rule: $\underline{\langle Q'\rangle[v]P'} \rightarrow_{\beta'} [v := Q']P'$
  where $[v := Q']P'$ is defined in a similar way to $P[v := Q]$.
- In $\mathcal{M}$, $(PQ)$ stands for the application of function $P$ to argument $Q$.
  In $\mathcal{M}'$, $\langle Q'\rangle P'$ stands for the application of function $P'$ to argument $Q'$ (note the reverse order).
  For example:

$(\lambda x.x)y$ in $\mathcal{M}$ becomes $\langle y\rangle[x]x$ in $\mathcal{M}'$.
$(\lambda x.(\lambda y.xy)z)(\lambda z'.z')$ in $\mathcal{M}$ becomes $\langle[z']z'\rangle[x]\langle z\rangle[y]\langle y\rangle x$ in $\mathcal{M}'$.

- The syntax of the classical $\lambda$-calculus with de Bruijn indices is given by
  $\Lambda ::= \mathbb{N} \mid (\lambda\Lambda) \mid (\Lambda\Lambda)$.
  For example: $(\lambda 1)$ is a term (it is equivalent to $\lambda x.x$ and $\lambda y.y$, etc).
  Also $(\lambda(\lambda 1\ 2))$ is also a term. It stands as you will see for $\lambda xy.yx$.
  Also, $(\lambda(\lambda 2\ 1))$ is also a term. It stands as you will see for $\lambda xy.xy$.
  We will use similar parenthesis convention as in $\mathcal{M}$, so $A''B''C''$ stands for $((A''B'')C'')$, but we cannot combine many $\lambda$s into one. So, $\lambda\lambda A''$ cannot be written as $\lambda A''$, but we know that $\lambda v.\lambda v'.A$ can be written as $\lambda vv'.A$.

- For $[x_1, \cdots, x_n]$ a list (not a set) of variables, we define $\omega_{[x_1,\cdots,x_n]} : \mathcal{M} \mapsto \Lambda$ by:

  1. $\omega_{[x_1,\cdots,x_n]}(v_i) = \min\{j : v_i \equiv x_j\}$
  2. $\omega_{[x_1,\cdots,x_n]}(AB) = \omega_{[x_1,\cdots,x_n]}(A)\omega_{[x_1,\cdots,x_n]}(B)$
  3. $\omega_{[x_1,\cdots,x_n]}(\lambda x.A) = \lambda\omega_{[x,x_1,\cdots,x_n]}(A)$

  Hence

  - $\omega_{[x,y,x,y,z]}(x) = 1$
  - $\omega_{[x,y,x,y,z]}(y) = 2$
  - $\omega_{[x,y,x,y,z]}(z) = 5$.
  - Also $\omega_{[x,y,x,y,z]}(xyz) = 1\ 2\ 5$.
  - Also $\omega_{[x,y,x,y,z]}(\lambda xy.xz) = \lambda\lambda 2\ 7$.

- **Translation from $\mathcal{M}$ to $\Lambda$:** If our variables are ordered as $v_1, v_2, v_3, \cdots$, then we define $\omega : \mathcal{M} \mapsto \Lambda$ by

  $$0.\ \ \omega(A) = \omega_{[v_1,\cdots,v_n]}(A) \text{ where } FV(A) \subseteq \{v_1, \cdots, v_n\}.$$

  So for example, if our variables are ordered as

  $$x, y, z, x', y', z', \cdots$$

  then the translation of $\omega(\lambda xyx'.xzx')$ from $\mathcal{M}$ to $\Lambda$ gives the term $\lambda\lambda\lambda 3\ 6\ 1$. This can be seen as follows:

$\underline{\omega(\lambda xyx'.xzx')} =^0$
$\underline{\omega_{[x,y,z]}(\lambda xyx'.xzx')} =^3$
$\underline{\lambda\omega_{[x,x,y,z]}(\lambda yx'.xzx')} =^3$
$\lambda\underline{\lambda\omega_{[y,x,x,y,z]}(\lambda x'.xzx')} =^3$
$\lambda\lambda\underline{\lambda\omega_{[x',y,x,x,y,z]}(xzx')} =^2$
$\lambda\lambda\lambda\underline{\omega_{[x',y,x,x,y,z]}(xz)}\omega_{[x',y,x,x,y,z]}(x') =^2$
$\lambda\lambda\lambda\underline{\omega_{[x',y,x,x,y,z]}(x)}\omega_{[x',y,x,x,y,z]}(z)\omega_{[x',y,x,x,y,z]}(x') =^1$
$\lambda\lambda\lambda 3\ \underline{\omega_{[x',y,x,x,y,z]}(z)}\omega_{[x',y,x,x,y,z]}(x') =^1$
$\lambda\lambda\lambda 3\ 6\ \underline{\omega_{[x',y,x,x,y,z]}(x')} =^1$
$\lambda\lambda\lambda 3\ 6\ 1.$

- Assume the following SML datatypes which implement $\mathcal{M}$, $\Lambda$ and $\mathcal{M}'$ respectively (here, if `e1` implements $A_1'$ and `e2` implements $A_2'$, then `IAPP(e1,e2)` implements $\langle A_1' \rangle A_2'$ which stands for the function $A_2'$ applied to argument$A_1'$):

```
datatype LEXP =
   APP of LEXP * LEXP | LAM of string *  LEXP |  ID of string;

datatype BEXP =
   BAPP of BEXP * BEXP | BLAM of BEXP |  BID of int;

datatype IEXP =
   IAPP of IEXP * IEXP | ILAM of string *  IEXP |  IID of string;
```

Recall the printing function on LEXP:

```
(*Prints a term in classical lambda calculus*)
fun printLEXP (ID v) =
    print v
  | printLEXP (LAM (v,e)) =
    (print "(\\";
     print v;
     print ".";
     printLEXP e;
     print ")")
  | printLEXP (APP(e1,e2)) =
    (print "(";
     printLEXP e1;
     print " ";
```

```
printLEXP e2;
print ")");
```

- At `http://www.macs.hw.ac.uk/~fairouz/foundations-2018/slides/sml-week1.sml` and `http://www.macs.hw.ac.uk/~fairouz/foundations-2018/slides/sml-week2.sml`, and `http://www.macs.hw.ac.uk/~fairouz/foundations-2018/slides/sml-week1` and `http://www.macs.hw.ac.uk/~fairouz/foundations-2018/slides/sml-week2`, you find an implementation in SML of the set of terms $\mathcal{M}$ and many operations on it like substitution and printing and free variables etc. You can use all of these in your assignment. Anything you use from elsewhere has to be well cited/referenced.

## Questions

1. For each term $A$ of the terms below, give its translation $\omega(A)$ from $\mathcal{M}$ to $\Lambda$ showing all the steps, their number and underlining all the parts you are working on, just like we did in the above example:

   (a) $(\lambda xz.xz)$. (1)

   $\underline{\omega(\lambda xz.xz)} =^0$
   .....

   (b) $(\lambda xy.xy)$. (1)

   $\underline{\omega(\lambda xy.xy)} =^0$
   ...

   (c) $xz(\lambda xy.xy)$. (1)

   $\underline{\omega(xz(\lambda xy.xy))} =^0$
   ...

   (d) $(\lambda xy.xy)xz$. (1)

   $\underline{\omega((\lambda xy.xy)xz)} =^0$
   ...

2. Give a translation function $f$ from $\mathcal{M}$ to $\mathcal{M}'$ that will translate terms in $\mathcal{M}$ to terms in $\mathcal{M}'$ so for example:
   $f((\lambda x.x)y) = \langle y\rangle[x]x$
   $f((\lambda x.(\lambda y.xy)z)(\lambda z'.z')) = \langle[z']z'\rangle[x]\langle z\rangle[y]\langle y\rangle x.$ (1)
   $f(v) =??$
   $f(\lambda v.A) =??$
   $f(AB) =??.$

3. Use your translation function $f$ of Question 2, to translate all the terms in Question 1 above into terms of $\mathcal{M}'$. That is, give $f(\lambda xz.xz)$ and $f(\lambda xy.xy)$ and $f(xz(\lambda xy.xy))$ and $f((\lambda xy.xy)xz)$ showing all the steps. (2)

   - $f(\lambda xz.xz) = .. = .. = .....$
   - $f(\lambda xy.xy) = .. = .. = .....$
   - $f(xz(\lambda xy.xy)) = .. = .. = .....$
   - $f((\lambda xy.xy)xz) = .. = .. = .....$

4. For each of BEXP and IEXP write a printing function printBEXP (respectively printIEXP) that prints its elements nicely just like we wrote printLEXP which prints nicely the elements of LEXP. (2)

```
(*Prints a term in item lambda calculus*)
fun printIEXP (IID v) =
........

(*Prints a term in classical lambda calculus with de Bruijn indices*)
fun printBEXP (BID n) =
......
```

5. For each term below, write it in LEXP and print it using printLEXP, write its translation by $f$ into IEXP and print it using printIEXP, and its translation by $\omega$ into BEXP and print it using printBEXP,

   (a) $(\lambda xz.xz)$. (1.5)

```
  val vx = (ID "x");
....

printLEXP t101 gives
.....

  val Ivx = (IID "x");
....

printIEXP It101 gives
....

  val bv1 = (BID 1);
```

....

printBEXP Bt101 gives
....

(b) $(\lambda xy.xy)$. (1.5)

.....

(c) $xz(\lambda xy.xy)$. (1.5)

.....

(d) $(\lambda xy.xy)xz$. (1.5)

.....