Assignment III (40 pts)

Burak Ekici

Assigned : May the 15th, 23h55
Due : May the 22nd, 23h55

1 Pairs

A **pair** of terms in pure (untyped) λ-Calculus (UTLC) is represented by the following λ-term

$$\texttt{lPair} := \lambda e_1.\lambda e_2.\lambda p.p\ e_1\ e_2$$

along with the projections

$$\begin{aligned}\texttt{lProj}_1 &:= \lambda u.u\ \texttt{lTrue}\\ \texttt{lProj}_2 &:= \lambda u.u\ \texttt{lFalse}\end{aligned}$$

2 Lists

Similarly, a **list** of terms in UTLC could be encoded with the constructors stated below.

$$\begin{aligned}\texttt{lCons} &:= \lambda t_1.\lambda t_2.\texttt{lPair lFalse (lPair}\ t_1\ t_2\texttt{)}\\ \texttt{lNil} &:= \lambda l.l\end{aligned}$$

To obtain the **head** and the **tail** of a given list, one could employ the following λ-terms:

$$\begin{aligned}\texttt{lHead} &:= \lambda l.\texttt{lProj}_1\ \texttt{(lProj}_2\ l\texttt{)}\\ \texttt{lTail} &:= \lambda l.\texttt{lProj}_2\ \texttt{(lProj}_2\ l\texttt{)}\end{aligned}$$

In addition, the **nullary check** (checking whether a given list is empty) could be captured by the λ-term

$$\texttt{lIsNil} := \texttt{lProj}_1$$

One can then develop **list operations** such as

$$\begin{aligned}\texttt{lLength} &:= \lambda f.\lambda l.\texttt{lIte (lIsNil}\ l\texttt{) (lZero) (lAdd (lOne) (}f\ \texttt{(lTail}\ l\texttt{)))}\\ \texttt{lAppend} &:= \lambda f.\lambda l_1.\lambda l_2\texttt{lIte (lIsNil}\ l_1\texttt{) (}l_2\texttt{) (lCons (lHead}\ l_1\texttt{) (}f\ \texttt{(lTail}\ l_1\texttt{)}\ l_2\texttt{))}\\ \texttt{lReverse} &:= \lambda f.\lambda l_1.\lambda l_2.\texttt{lIte (lIsNil}\ l_1\texttt{) (}l_2\texttt{) (}f\ \texttt{(lTail}\ l_1\texttt{) (lCons (lHead}\ l_1\texttt{) (}l_2\texttt{)))}\end{aligned}$$

thanks to the λ-terms listed above and those imported from the provided modules.

Note.

Note that the term `reverse` is implemented in a tail recursive fashion, therefore the parameter l_2 must be `nil`.

3 Tasks

Implement in OCaml, the following UTLC terms.

1. a) (**3 pts**) `lPair (t1: term) (t2: term) : term`
   b) (**3 pts**) `lProj1 (t: term) : term`
   b) (**3 pts**) `lProj2 (t: term) : term`

2. a) (**3 pts**) `lCons (t1: term) (t2: term) : term`
   b) (**1 pts**) `lNil : term`
   c) (**3 pts**) `lHead (t: term) : term`
   d) (**3 pts**) `lTail (t: term) : term`
   e) (**1 pts**) `lIsNil (t: term) : term`

3. (**6 pts**) `lLength (t: term) : term`
4. (**6 pts**) `lAppend (t1: term) (t2: term) : term`
5. (**8 pts**) `lReverse (t: term) : term`

Important.

Download the accompanying library `assignment3.zip` from the course DYS page and include terms listed in item 1 above into the `pairs.ml`, the rest (listed in items 2 and 3) into `lists.ml`.

Do not remove the topmost lines that import previously implemented modules. E.g., `open Booleans`, `open Church`, and etc.

4 Sanity Check

To sanity check your implementations, you could execute below commands and compare obtained results with the expected ones:

```
let t1 = (lCons (lNumeral 1) (lCons (lNumeral 4) (lCons (lNumeral 2) lNil)))
let t2 = (lCons (lNumeral 3) (lCons (lNumeral 1) lNil))
```

Command	Expected Output

`refl_trans_beta t1`	`(λp.[[p (λa.(λb.b))](λp.[[p (λs.(λz.[s z]))]`
	`(λp.[[p (λa.(λb.b))](λp.[[p (λs.(λz.[s [s [s [s z]]]]))]`
	`(λp.[[p (λa.(λb.b))](λp.[[p (λs.(λz.[s [s z]]))]`
	`(λl.l)])])])])

`refl_trans_beta (length t1)`	`(λs.(λz.[s [s [s z]]]))`

`refl_trans_beta (reverse t1)`	`(λp.[[p (λa.(λb.b))](λp.[[p (λs.(λz.[s [s z]]))]`
	`(λp.[[p (λa.(λb.b))](λp.[[p (λs.(λz.[s [s [s [s z]]]]))]`
	`(λp.[[p (λa.(λb.b))](λp.[[p (λs.(λz.[s z]))]`
	`(λl.l)])])])])

`refl_trans_beta (append t1 t2)`	`(λp.[[p (λa.(λb.b))](λp.[[p (λs.(λz.[s z]))]`
	`(λp.[[p (λa.(λb.b))](λp.[[p (λs.(λz.[s [s [s [s z]]]]))]`
	`(λp.[[p (λa.(λb.b))](λp.[[p (λs.(λz.[s [s z]]))]`
	`(λp.[[p (λa.(λb.b))](λp.[[p (λs.(λz.[s [s [s z]]]))]`
	`(λp.[[p (λa.(λb.b))](λp.[[p (λs.(λz.[s z]))]`
	`(λl.l)])])])])])])])])])

Important Notice.

- Collaboration is strictly and positively prohibited; lowers your score to 0 if detected.
- Any submission after 23h55 on May the 22nd will NOT be accepted. Please beware and respect the deadline!