CM20256 - Coursework 2

Adam Jaamour (aj645)

02/05/2017

Contents

1 Part 1 (10%)

Show that the term *"[3,2,1] times 1"* β-reduces to 6:

$$
\begin{aligned}
[3,2,1] \text{ times } 1 &\triangleq (\underline{\lambda c.\lambda n.\ c\ 3\ (c\ 2\ (c\ 1\ n))\ \text{times}})\ 1 \\
&\to_\beta (\underline{\lambda n.\ \text{times}\ 3\ (\text{times}\ 2\ (\text{times}\ 1\ n)))\ 1} \\
&\to_\beta \text{times}\ 3\ (\text{times}\ 2\ (\underline{\text{times}\ 1\ 1})) \\
&\to_\beta \text{times}\ 3\ (\text{times}\ 2\ (1 \times 1)) \\
&= \text{times}\ 3\ (\underline{\text{times}\ 2\ (1)}) \\
&\to_\beta \text{times}\ 3\ (2 \times 1) \\
&= \underline{\text{times}\ 3\ (2)} \\
&\to_\beta 3 \times 2 \\
&= 6
\end{aligned}
$$

"times $m\ n\ \to_\beta n \times m$" was used for this part.

2 Part 2 (15%)

Reduce *"cons 3 [2,1]"* to *"[3,2,1]"*:

$$
\begin{aligned}
\mathit{cons}\ 3\ [2,1] &= (((\underline{\lambda x.\lambda l.\lambda c.\lambda n\ c\ x\ (l\ c\ n))\ 3})\ [2,1] \\
&\to_\beta (\underline{\lambda l.\lambda c.\lambda n\ c\ 3\ (l\ c\ n))\ [2,1]} \\
&\to_\beta \lambda c.\lambda n\ c\ 3\ (\underline{[2,1]}\ c\ n) \\
&\equiv_\alpha \lambda c.\lambda n\ c\ 3\ (\underline{\lambda d.\lambda m.\ d\ 2\ (d\ 1\ m)\ c}\ n) \\
&\to_\beta \lambda c.\lambda n\ c\ 3\ (\underline{\lambda m.\ c\ 2\ (c\ 1\ m)\ n}) \\
&\to_\beta \lambda c.\lambda n\ c\ 3\ (c\ 2\ (c\ 1\ n)) \\
&= [3,2,1]
\end{aligned}
$$

"$\mathit{cons} \triangleq \lambda x.\lambda.l.\lambda x.\lambda n.\ c\ x\ (l\ c\ n)$" was used for this part.

3 Part 3 (15%)

Define terms *"head"* and *"empty"* such that:

$$\begin{array}{rcrl} head & [N, ...] & \rightarrow_\beta^* & N \\ empty & [\ \ ] & \rightarrow_\beta^* & true \\ empty & [N, ...] & \rightarrow_\beta^* & false \end{array}$$

The λ-term found for *"head"* is:

$$head \triangleq \lambda l.l\ (\lambda.x.\lambda y.x)\ n$$

The λ-term found for *"empty"* is:

$$empty \triangleq \lambda l.l\ (\lambda.a.\lambda b.false)\ true$$

Proof by example using the λ-terms found for *"head"* and *"empty"*:

$$\begin{aligned} head\ [2,1] \triangleq\ & \underline{(\lambda l.l\ (\lambda.x.\lambda y.x)\ n)\ \lambda c.\lambda n.\ c\ 2\ (c\ 1\ n)} \\ \rightarrow_\beta\ & (\underline{\lambda c.\lambda n.\ c\ 2\ (c\ 1\ n))\ (\lambda.x.\lambda y.x}) \ n \\ \rightarrow_\beta\ & \underline{\lambda n.\ ((\lambda.x.\lambda y.x)\ 2)\ (((\lambda.x.\lambda y.x)\ 1)\ n)\ n} \\ \rightarrow_\beta\ & ((\lambda.x.\lambda y.x)\ 2)\ ((\underline{(\lambda.x.\lambda y.x)\ 1})\ n) \\ \rightarrow_\beta\ & ((\lambda.x.\lambda y.x)\ 2)\ (\underline{(\lambda y.1)\ n}) \\ \rightarrow_\beta\ & (\underline{(\lambda.x.\lambda y.x)\ 2})\ 1 \\ \rightarrow_\beta\ & (\underline{\lambda y.2)\ 1} \\ \rightarrow_\beta\ & 2 \end{aligned}$$

$$
\begin{aligned}
empty\ [\,] \ \triangleq\ & \underline{(\lambda l.l\ (\lambda .a.\lambda b.false)\ true)\ \lambda c.\lambda n.n} \\
\rightarrow_\beta\ & \underline{\lambda c.\lambda n.n\ (\lambda .a.\lambda b.false)}\ true \\
\rightarrow_\beta\ & (\underline{\lambda n.n)\ true} \\
\rightarrow_\beta\ & true
\end{aligned}
$$

$$
\begin{aligned}
empty\ [2,1] \ \triangleq\ & \underline{(\lambda l.l\ (\lambda .a.\lambda b.false)\ true)\ \lambda c.\lambda n.\ c\ 2\ (c\ 1\ n)} \\
\rightarrow_\beta\ & \underline{(\lambda c.\lambda n.\ c\ 2\ (c\ 1\ n))\ (\lambda .a.\lambda b.false)}\ true \\
\rightarrow_\beta\ & \underline{(\lambda n.\ ((\lambda .a.\lambda b.false)\ 2)\ (((\lambda .a.\lambda b.false)\ 1)\ n))\ true} \\
\rightarrow_\beta\ & ((\lambda .a.\lambda b.false)\ 2)\ ((\underline{(\lambda .a.\lambda b.false)\ 1})\ true) \\
\rightarrow_\beta\ & ((\lambda .a.\lambda b.false)\ 2)\ ((\underline{\lambda b.false)\ true}) \\
\rightarrow_\beta\ & (\underline{(\lambda .a.\lambda b.false)\ 2})\ false \\
\rightarrow_\beta\ & (\underline{\lambda b.false)\ false} \\
\rightarrow_\beta\ & false
\end{aligned}
$$

" $[\,]\ =\ \lambda c.\lambda n.n$" , where [ ] is the empty list, was used for this part.

4 Part 4 (35%)

4.1 List represented by L_m

What does $L_m = \lambda \mathbf{c}.\lambda \mathbf{n}.L'_m$ (for any m) represent?

The list represented by the term L_m corresponds to the list of descending natural numbers from m to 1, excluding 0, such as m $\in$ $\mathbb{Z}^*$.

Proof by example, using $m = 4$:

$$\begin{aligned} L_4 &= \lambda c.\lambda n.\ L'_4 \\ &= \lambda c.\lambda n.\ c\ 4\ (L'_3) \\ &= \lambda c.\lambda n.\ c\ 4\ (c\ 3\ (L'_2)) \\ &= \lambda c.\lambda n.\ c\ 4\ (c\ 3\ (c\ 2\ (L'_1))) \\ &= \lambda c.\lambda n.\ c\ 4\ (c\ 3\ (c\ 2\ (c\ 1\ (L'_0)))) \\ &= \lambda c.\lambda n.\ c\ 4\ (c\ 3\ (c\ 2\ (c\ 1\ n))) \\ &= [5, 4, 3, 2, 1] \end{aligned}$$

4.2 Inductive proof

Prove by induction on m that L'_m [times/c , 1/n] $\rightarrow^*_\beta$ m!

<u>Base case</u>: $m = 0$

$$\begin{aligned} L'_0\ [\ \text{times}/c\ ,\ 1/n\ ] = &\quad n\ [\ \text{times}/c\ ,\ 1/n\ ] \\ &\rightarrow_\beta 1 \end{aligned}$$

Since $0! = 1$, the base case is therefore true.

<u>Inductive case</u>:

The inductive hypothesis is L'_m [times/c , 1/n] $\rightarrow^*_\beta$ m!, and it is assumed to be true. If it can be proved with $m + 1$, then L'_m [times/c , 1/n] $\rightarrow^*_\beta$ m! will be true for all m.

For $m+1$:

$$\begin{aligned}
& L'_{m+1} \text{ [times}/c \text{ , } 1/n \text{]} \\
& = \quad (c\ (m+1)\ L'_{m-1+1}) \text{ [times}/c \text{ , } 1/n \text{]} \\
& \rightarrow_\beta \text{times } (m+1)\ (L'_m \text{ [times}/c \text{ , } 1/n \text{]}) \\
& = \quad \text{times } (m+1)\ m! \\
& \rightarrow_\beta (m+1) \times m! \\
& = \quad (m+1)!
\end{aligned}$$

On the second line of solution, L'_m [times/c , $1/n$] is substitued with $m!$ throughout the inductive hypothesis.

"times m n $\rightarrow_\beta$ $n \times m$" was also used for this part.

True for $m+1$, therefore L'_m [times/c , 1/n] $\rightarrow^*_\beta$ is true for all m.

Based on previous answer, prove that L_m times 1 $\rightarrow^*_\beta$ m!

$$\begin{aligned}
L'_m \text{ times } 1 &= (\lambda c.\lambda n.\ L'_m \text{ times})\ 1 \\
&= \lambda n.\ L'_m \text{ [times / } c\text{] } 1 \\
&= L'_m \text{ [times}/c \text{ , } 1/n\text{]} \\
&= m!
\end{aligned}$$

Previous proof that L'_m [times/c , 1/n] $\rightarrow^*_\beta$ m! used to find $m!$

5 Part 5 (25%)

5.1 foldr

Give a λ-term corresponding to Haskell function *foldr* such as:

$$foldr\ f\ u\ [N_1\ , ...,\ N_k] \quad \rightarrow^*_\beta \quad f\ N_1\ (f\ N_2\ (\ ...\ (f\ N_k\ u)))$$

The λ-term found for *"foldr"* is:

$$foldr \triangleq \lambda a.\lambda b.\lambda l.(l\ a\ b)$$

The function was created by analyzing the property it should have, which was provided by the coursework specification. Looking at the term, it can be seen that $foldr$ takes three inputs: the function f, the accumulator u and the list $[N_1, ..., N_k]$. In Haskell, $foldr$ is defined as the function that combines the accumulator u to each list element using the function f, starting from the right and moving to the left.

Looking more closely at the resulting term $f\ N_1\ (f\ N_2\ (\ ...\ (f\ N_k\ u)))$, which is obtained after performing the beta reductions, it is obvious that the term follows the same pattern as the one of a list: $c\ N_1(c\ N_2(...(c\ N_k\ n)...))$, with c being replaced by f and n being replaced by u.

This is achieved by substituting the inputs a, b, and l from the term found for $foldr$ with their corresponding inputs to get a term with the list first, followed by f and u, of the form $([N_1, ..., N_k]\ f)\ u$. After extending the list, beta-reductions can be perfomed on the *cons* and the *nil* of the extended list, thus replacing each *cons* by f and each *nil* by u, which corresponds to the term $foldr$ should beta-reduce to.

Proof by example using the λ-terms found for *"foldr"*:

non-empty list:

$foldr\ f\ u\ [3,2,1]$ should return $f\ 3\ (f\ 2\ (f\ 1\ u))$

$$\begin{aligned}
foldr\ f\ u\ [3,2,1] &\triangleq ((\underline{\lambda a.\lambda b.\lambda l.(l\ a\ b)\ f})\ u)\ [3,2,1] \\
&\to_\beta (\underline{\lambda b.\lambda l.(l\ f\ b)\ u})\ [3,2,1] \\
&\to_\beta \underline{\lambda l.(l\ f\ u)\ [3,2,1]} \\
&\to_\beta ([3,2,1]\ f)\ u \\
&= (\underline{\lambda c.\lambda n.\ c\ 3\ (c\ 2\ (c\ 1\ n))\ f})\ u \\
&\to_\beta \underline{\lambda n.\ f\ 3\ (f\ 2\ (f\ 1\ n))\ u} \\
&\to_\beta f\ 3\ (f\ 2\ (f\ 1\ u))
\end{aligned}$$

empty list:

$foldr\ f\ u\ [\,]$ should return u, since nil is replaced by the accumulator u:

$$\begin{aligned}
foldr\ f\ u\ [\,] &\triangleq ((\underline{\lambda a.\lambda b.\lambda l.(l\ a\ b)\ f})\ u)\ [\,] \\
&\to_\beta (\underline{\lambda b.\lambda l.(l\ f\ b)\ u})\ [\,] \\
&\to_\beta \underline{\lambda l.(l\ f\ u)\ [\,]} \\
&\to_\beta ([\,]\ f)\ u \\
&= ((\underline{\lambda c.\lambda n.\ n)\ f})\ u \\
&\to_\beta (\underline{\lambda n.n)\ u} \\
&\to_\beta u
\end{aligned}$$

5.2 map

Give a λ-term corresponding to Haskell function *map* such as:

$$map\ f\ [N_1\ ,...,\ N_k] \quad \to_\beta^* \quad [f\ N_1\ , f\ N_2\ , ..., f\ N_k]$$

The λ-term found for *"map"* is:

$$map \triangleq \lambda a.\lambda l.\lambda c.\ l\ (\lambda x.\ c\ a\ x)$$

Looking at the property *map* should have (given by the coursework specification), the objective is to build a function which applies a function f to each element of a list. This also corresponds to the definition of the *map* function in Haskell. Looking at the term, *map* takes two inputs: the function f which needs to be applied to each element of the list $[N_1, ..., N_k]$, which is the second input.

The resulting term corresponds to a list where f is applied to each of its elements. Consequently, the function *map* should build a list of the type $\lambda c.\lambda n.c\ N_1(c\ N_2(...(c\ N_k\ n)...))$, inserting f between each c and N for each element of the list to get a term of the form $\lambda c.\lambda n.c\ N_1(c\ N_2(...(c\ N_k\ n)...))$.

This can be done by substituting inputs a and l from the term found for *map* with their corresponding inputs to get a term starting with λc, followed by the list and ending with a term $(\lambda x.c\ f\ x)$ which keeps the form (cfx) when beta-reduced, with x corresponding to the rest of the list. Performing beta-reductions from left to right builds a list with f between c and N, until the end of the list is reached.

Proof by example using the λ-terms found for *"map"*:

non-empty list:

map f $[3,2,1]$ should return $[f\ 3\ ,\ f\ 2\ ,\ f\ 1]$

$$\begin{aligned}
map\ f\ [3,2,1] &\triangleq\ (\underline{\lambda a.\lambda l.\lambda c.\ l\ (\lambda x.\ c\ a\ x)\ f})\ [3,2,1] \\
&\rightarrow_\beta \underline{\lambda l.\lambda c.\ l\ (\lambda x.\ c\ f\ x)\ [3,2,1]} \\
&\rightarrow_\beta \lambda c.\ \underline{[3,2,1]}\ (\lambda x.\ c\ f\ x) \\
&\equiv_\alpha\ \lambda c.\ (\underline{\lambda d.\lambda n.\ d\ 3\ (d\ 2\ (d\ 1\ n))\ (\lambda x.\ c\ f\ x)}) \\
&\rightarrow_\beta \lambda c.\ (\lambda n.\ (\underline{\lambda x.\ c\ f\ x)\ 3}\ ((\lambda x.\ c\ f\ x)\ 2\ ((\lambda x.\ c\ f\ x)\ 1\ n))) \\
&\rightarrow_\beta \lambda c.\ (\lambda n.\ c\ f\ 3\ ((\underline{\lambda x.\ c\ f\ x)\ 2}\ ((\lambda x.\ c\ f\ x)\ 1\ n))) \\
&\rightarrow_\beta \lambda c.\ (\lambda n.\ c\ f\ 3\ (c\ f\ 2\ ((\underline{\lambda x.\ c\ f\ x)\ 1\ n}))) \\
&\rightarrow_\beta \lambda c.\ (\lambda n.\ c\ f\ 3\ (c\ f\ 2\ (c\ f\ 1\ n))) \\
&=\ \ \lambda c.\lambda n.\ c\ f\ 3\ (c\ f\ 2\ (c\ f\ 1\ n))) \\
&= [\ f\ 3\ ,\ f\ 2\ ,\ f\ 1\ ]
\end{aligned}$$

empty list:

map f $[\,]$ should return the empty-list $[\,]$

$$\begin{aligned}
map\ f\ [\,] &\triangleq\ (\underline{\lambda a.\lambda l.\lambda c.\ l\ (\lambda x.\ c\ a\ x)\ f})\ [\,] \\
&\rightarrow_\beta \underline{\lambda l.\lambda c.\ l\ (\lambda x.\ c\ f\ x)\ [\,]} \\
&\rightarrow_\beta \lambda c.\ \underline{[\,]}\ (\lambda x.\ c\ f\ x) \\
&\equiv_\alpha\ \lambda c.\ (\underline{\lambda d.\lambda n.n)\ (\lambda x.\ c\ f\ x}) \\
&\rightarrow_\beta \lambda c.\lambda n.n \\
&=\ \ [\,]
\end{aligned}$$

5.3 infinite list

Give a λ-term corresponding to infinite list $[0, 1, 2, ...]$**:**

The infinite list can be written as $[0, 1, 2, 3, ...]$.

It is considered to start from the point 0 onwards.

To begin with, the infinite list can be written as $infinite \triangleq infinite\ m$. There are two cases to take into account: when $m = 0$, and $m \neq 0$.

Combined with the term for a finite list $\lambda c.\lambda n.c\ N_1(c\ N_2(...(c\ N_k\ n)...))$, a first idea of the logic of the infinite list can be written:

$if\ m = 0\ then\ (\lambda c.\ c\ m\ infinite(succ\ m))$
$else\ if\ m \neq 0\ (c\ m(infinite(succ\ m)))$

Note that the n is not included since the empty list is impossible in an infinite list.

Using the logic stated previously, the term for $infinite$ can be written:

$$infinite\ =\ \lambda m.\ ifthen(iszero\ m)(\lambda c.\ c\ m\ infinite(succ\ m))\ (c\ m(infinite(succ\ m)))$$

where: $ifthen \triangleq \lambda a.\lambda x.\lambda y.\ (a\ x\ y)$
where: $iszero \triangleq \lambda n.n\ (\lambda w.false\ true)$
where $succ \triangleq \lambda n.\lambda f.\lambda x.\ f(n\ f\ x)$

To end, the Y-combinator is applied to this term to get an infinitely recursive list.