CS-720 Programming Language Principles

Solution to Homework for Week 04.

1. Identify the free variable occurrences in the following AE's:

   (a) fn a. fn c. a b
   (b) (fn a. b) a
   (c) (fn a. fn b. a+b) b 8
   (d) (fn b. fn a. a b) b
   (e) (fn a. ((fn a. fn b. b+b) (a+1) (b+1)) + 3) 7

   Reduce the above AE's.

   (a) fn a. fn c. a b (already reduced)
   (b) b
   (c) 8 + b
   (d) fn a. a b
   (e) b + 1 + b + 1 + 3

2.

   (a) b+1
   (b) 5
   (c) 5
   (d) 3+4+c
   (e) fn b. b+3
   (f) c
   (g) fn a. a 1
   (h) 3
   (i) 3
   (j) 2
   (k) 1
   (l) 4
   (m) 3+b
   (n) ((fn b.b+1)+2)3
   (o) 1 1
   (p) 1 1
   (q) 2

3. In the following, even the most trivial steps are shown.

   (fn b. fn a. -ba) a b

   { [a/b] (fn a.-ba) } b by axiom beta.
   { fn a`. [a/b] ([a`/a] (-ba)) } b by case 3.2.3.
   { fn a`. [a/b] ([a`/a](-b) [a`/a]a) } b by case 2.
   { fn a`. [a/b] (((a`/a)(-) [a`/a]b) [a`/a]a) } b by case 2.
{ fn a`. [a/b] (( (-) [a`/a]b) [a`/a]a) } b by case 1.2.
{ fn a`. [a/b] (( (-) b) [a`/a]a) } b by case 1.2.
{ fn a`. [a/b] (( (-) b) a`) } b by case 1.1.
{ fn a`. [a/b] (-b) ([a/b] a`) } b by case 2.
{ fn a`. ([a/b] (-) [a/b] b) ([a/b] a`) } b by case 2.
{ fn a`. ( - [a/b] b) ([a/b] a`) } b by case 1.2.
{ fn a`. ( - a) ([a/b] a`) } b by case 1.1.
{ fn a`. ( - a) a` } b by case 1.2.
[b/a`] ((-a)a`) by axiom beta.
([b/a`] (-a)) ([b/a`]a`) by case 2.
([b/a`] (-) ([b/a`]a)) ([b/a`]a`) by case 2.
- ([b/a`]a) ([b/a`]a`) by case 1.2.
- a ([b/a`]a`) by case 1.2.
- a b by case 1.1.
4 a) AST:

```
  a
 / \      
 and   b   
    /   \  
   x     c   
      / \  
     y   z
```

b) AST → ST

1. First, standardize the "":
   - `Tree 1` is a sub-tree with `8` and `nil`.
   - `Tree 2` is a sub-tree with `q` and `Tree 2`.
   - `Tree 3` is a sub-tree with `f` and `Tree 2`.

2. Next, standardize `ten-fom`:
   - `12` is a sub-tree.

3. Next, standardize the other `ten-fom`:
   - `x` and `y` are sub-trees.
So, the overall tree, after standardizing @, becomes:

So, now the overall tree becomes:
Standardizes the `and`: 

\[
\text{Tau} \quad \Rightarrow \quad \text{Tree 6}
\]

So, now the overall tree is:

\[
\text{let} \quad \Rightarrow \quad \text{Tree 6}
\]

\[
\text{Tree 6} \quad \Rightarrow \quad \text{Tree 2}
\]

which yields the following hidden tree:
When!
5.
(a) \( x + 1 \)
(b) 11
(c) 1
(d) \( \text{fn} \ a.\ c+2 \)
(e) \( a+6 \)
(f) \( 12 + x \)
(g) 12
(h) 10

6.
\[
(\text{fn} \ a.\ a+a) \ [(\text{fn} \ b.\ b+b) \ {(\text{fn} \ c.\ c+c) \ [(\text{fn} \ d.\ d+d) \ 2]}])
\]
=> \( (\text{fn} \ a.\ a+a) \ [(\text{fn} \ b.\ b+b) \ {(\text{fn} \ c.\ c+c) \ [ \ 2+2 \ ]}]) \]
=> \( (\text{fn} \ a.\ a+a) \ [(\text{fn} \ b.\ b+b) \ { \ 2+2+2+2 \ ]} \]
=> \( (\text{fn} \ a.\ a+a) \ [2+2+2+2 \ +2+2+2+2 \ ] \)
=> \( 2+2+2+2 \ +2+2+2+2 \ +2+2+2+2 \ +2+2+2+2 \ +2+2+2+2 \ +2+2+2+2 \ +2+2+2+2 \ +2+2+2+2 \ +2+2+2+2 \)
(==delta==> 32)

In programming language order, 4 beta reductions are necessary.

In normal order,

\[
\text{# beta reductions}[(\text{fn} \ a.\ a+a) \ E]
= 1+2*(\text{# beta reductions}[E])
\]

\[
(\text{fn} \ a.\ a+a) \ [(\text{fn} \ b.\ b+b) \ {(\text{fn} \ c.\ c+c) \ [(\text{fn} \ d.\ d+d) \ 2]}])
\]
Thus: \( 1+2* \ ( \ 1+2* \ ( \ 1+2* \ ( \ 1+2* \ 0))) \)
= 15 beta-reductions in normal order.