Lambda Calculus

Programming Languages
Lecture 8

Adeesha Wijayasiri

Lambda Calculus

- To obtain the "value" of an RPAL program:

  1. Transduce RPAL source to an AST.
  2. Standardize the AST into ST.
  3. Linearize the ST into a lambda-expression.
  4. Evaluate the lambda-expression, using the CSE machine

First, need some theory.

RPAL's flattener grammar: use bracket tree notation: <'root' s_1 … s_n>

RPAL	→	E	
E	→	<'γ' E E >	=> E E
	→	<'λ' V E >	=> 'λ' V '.' E
	→	<id:x>	=> x
	→	<integer:i>	=> i
	→	<string:s>	=> s
	→	'true'	=> 'true'
	→	'false'	=> 'false'
		. . .	

end RPAL.

Result of Tree Flattening: a string

- An Applicative Expression (AE).
- The interpretation of the AE is ambiguous.
- Need parentheses to disambiguate.

Disambiguation Rules

1. Function application is left associative.

2. If an expression of the form $\lambda x.M$ occurs in a larger expression, then M is extended as far as possible (i.e. to the end of the entire expression or to the next unmatched right parenthesis).

Example:

λ x. λ y.+xy 2 3

is equivalent to

λ x.(λ y.+xy 2 3)

- Must be parenthesized to obtain the intended expression:

(λ x. λ y.+xy) 2 3.

Definition

- Let M and N be λ-expressions. An occurrence of x in a λ-expression is free if it can be proved so via the following three rules:
  1. The occurrence of x in λ-expression "x" is free.
  2. Any free occurrence of x in either M or N is free in M N.
  3. Any free occurrence of x in M is free in $\lambda y.M$, if x and y are different.

Practical, Equivalent Definition

- An occurrence of x in M is free if the path from x to the root of M includes no λ node with x on its left.

- Definition:
  - An occurrence of x in a λ-expression M is said to be bound if it is not free in M.

Examples:

a	-	a occurs free
x	-	x occurs free
a x	-	a and x both occur free
(λ x.ax)x	-	a occurs free; x occurs both free and bound
(λ x. λ y.x+y) y 3		
	-	first occurrence of y is not free, second occurrence is free, in the entire expression.

More definitions

- Definition:
  - In an expression of the form λ x.M, x is the bound variable, and M is the body.

- Definition:
  - The scope of an identifier x, in an expression of the form λ x.M, consists of all free occurrences of x in M.

Axiom Delta

Let M and N be AE's that do not contain λ-expressions.

Then M $=>_\delta$ N if Val(M) = Val(N).

We say that M and N are delta-convertible, pronounced “M delta reduces to N”.

Val: Value obtained from ordinary evaluation.

- Example: +35 $=>_\delta$ 8.

Axiom Alpha

Let x and y be names, and M be an AE with no free occurrences of y. Then, in any context, λ x.M $=>_{\alpha}$ λ y.subst[y,x,M]

subst[y,x,M] means "substitute y for x in M".

Axiom Alpha used to rename the bound variable.

•Example: λx.+x3 $=>_{\alpha}$ λ y.+y3

Axiom Beta

- Let x be a name, and M and N be AE's. Then, in any context,

  $(\lambda x.M)\ N \Rightarrow_\beta subst[N,x,M]$.

- Called a "beta-reduction", used to apply a function to its argument.

- Pronounced “beta reduces to”.

Definition (subst):

- Let M and N be AE's, and x be a name.
- Then subst[N,x,M], also denoted as [N/x]M, means:

1. If M is an identifier, then
   1.1. if M=x, then return N.
   1.2. if M is not x, then return M.

2. If M is of the form X Y, then return ([N/x]X) ([N/x]Y).

Definition (subst[N,x,M], cont'd)

3. If M is of the form λ y.Y, then
   - 3.1. if y=x then return λ y.Y.
   - 3.2. if y is not x then
     - 3.2.1. if x does not occur free in Y, then
       return λ y.Y.
     - 3.2.2. if y does not occur free in N, then
       return λ y.[N/x]Y.
     - 3.2.3. if x occurs free in Y,
       and y occurs free in N, then
       return λw.[N/x] ([w/y]Y),
       for any w that does not occur
       free in either N or Y.

Examples

- $[3/x](\lambda\, x.{+}x2)$ $= \lambda\, x.{+}x2$ (by 3.1)
- $[3/x](\lambda\, y.y)$ $= \lambda\, y.y$ (by 3.2.1)
- $[3/x](\lambda\, y.{+}xy)$ $= \lambda\, y.[3/x]({+}xy)$

  $= \lambda\, y.{+}3y$ (by 3.2.2 and 2)
- $[y/x](\lambda\, y.{+}xy)$ $= \lambda\, z.[y/x]([z/y]({+}xy))$

  $= \lambda\, z.[y/x]({+}xz)$

  $= \lambda\, z.{+}yz$ (by 3.2.3, 2, and 2)

Definition

- An AE M is said to be "directly convertible“ to an AE N, denoted
  M => N, if one of these three holds:
    - $M =>_{\alpha} N$, $M =>_{\beta} N$, $M =>_{\delta} N$.

- Definition:

    - Two AE's M and N are said to be equivalent if M =>* N.

Recursion and Fixed-Point Theory

- We know that the meaning of

  let x=P in Q

  is the value of

  $(\lambda x.Q)P$

- So, what's the meaning of (the de-sugared version of)

  let rec f n = P in Q ?

Consider Factorial:

let rec f n = n eq 0 → 1 | n*f(n-1) in f 3

- Without the 'rec', we'd have
  - (λ f.f 3) [λ n.n eq 0 → 1 | n*f(n-1)]
- Note: the last 'f' is free.
- The 'rec' makes the last 'f' bound to the [λ n. ...] expression.

With 'rec', somehow …

$(\lambda\ f.f\ 3)\ [\lambda\ n.n\ eq\ 0 \rightarrow 1 \mid n*f(n-1)\ ]$

$=>_{\beta}$ $(\lambda\ n.n\ eq\ 0 \rightarrow 1 \mid n*f(n-1))\ 3$

$=>_{\beta}$ $3\ eq\ 0 \rightarrow 1 \mid 3*f(3-1)$

$=>_{\delta}$ $3*f(2)$

$=>_{magic}$ $3*(\lambda\ n.n\ eq\ 0 \rightarrow 1 \mid n*f(n-1))\ 2$

$=>_{\beta}$ $3*(2\ eq\ 0 \rightarrow 1 \mid 2*f(2-1))$

$=>_{\delta}$ $3*2*f(1)$

$=>_{magic}$ $3*2*(\lambda\ n.n\ eq\ 0 \rightarrow 1 \mid n*f(n-1))\ 1$

$=>_{\beta}$ $3*2*(1\ eq\ 0 \rightarrow 1 \mid 1*f(1-1))$

$=>_{\delta}$ $3*2*1*f(0)$

$=>_{magic}$ $3*2*1*(\lambda\ n.n\ eq\ 0 \rightarrow 1 \mid n*f(n-1))\ 0$

$=>_{\beta}$ $3*2*1*(0\ eq\ 0 \rightarrow 1 \mid 0*f(0-1))$

$=>_{\delta}$ $3*2*1*1$

$=>_{\delta}$ 6

To dispel the magic, need some math

- Let F be a function. A value w is called a "fixed-point" of F if F w = w.
- Example:

$f = \lambda$ x.x ** 2 - 6 has two fixed points,

3 and -2, because

(λ x.x ** 2 - 6) 3 $=>_\beta$ 3 ** 2 - 6 $=>_\delta$ 3, and

(λ x.x ** 2 - 6) -2 $=>_\beta$ -2 ** 2 - 6 $=>_\delta$ -2

- Only two fixed-points:

solutions to x**2-x-6 = 0.

Fixed-points can be functions

T = λ f. λ ().(f nil)**2-6 has two fixed points,

λ ().3 and λ ().-2, because

(λ f. λ ().(f nil)**2-6) (λ ().3)

$=>_\beta$ λ ().((λ ().3) nil)**2-6

$=>_\beta$ λ ().3**2-6

$=>_\delta$ λ ().3

and

(λ f. λ ().(f nil)**2-6) (λ ().-2)

$=>_\beta$ λ ().((λ ().-2) nil)**2-6

$=>_\beta$ λ ().-2**2-6

$=>_\delta$ λ ().-2

Let's Dispel the Magic

	let rec f n = P in Q	
=>	let rec f = λ n.P in Q	(fcn_form)
<=	let rec f = (λ f. λ n.P) f in Q	(abstraction)
<=	let rec f = F f in Q where F = λ f. λ n.P (abstraction).	

- Now there are no free occurrences of f in F, and only one free occurrence of f overall.

Dispelling magic, (cont’d)

- But, let rec f = F f in Q means we want f to be a fixed point of F !

- So, re-phrase it as

  let f = A_fixed_point_of F in Q.

- The "A-fixed_point_of" operator is called a "fixed point finder". We call it "Y“, or Ystar, or Y*.

Dispelling magic, (cont'd)

- So, we have

$$\begin{array}{ll} & \text{let } f = Y\ F \text{ in } Q \\ \Rightarrow & (\lambda f.Q)\ (Y\ F) \\ = & (\lambda f.Q)\ (Y\ (\lambda f.\ \lambda n.P)) \end{array}$$

- Note: No free occurrences of f !

- So, explaining the purpose of "rec" is the same as finding a fixed point of F.

How do we find a suitable Y ?

- WE DON'T NEED TO !

- We only need to use some of its characteristics:
  1. f = Y F
  2. F f = f

- Substituting 1) in 2), we obtain

F (Y F) = Y F

- This is the fixed point identity.

Let's extend some earlier definitions:

- Definition:
  - Axiom ρ (rho): Y F $=>_{\rho}$ F (Y F).

Let's Do the Factorial

- This time, no magic. Just science.
- $F = \lambda f. \lambda n.n \text{ eq } 0 \to 1 \mid n*f(n-1)$, so

$$
\begin{aligned}
& (\lambda f.f\ 3)\ (Y\ F) \\
\Rightarrow_{\beta}\ & Y\ F\ 3 \\
\Rightarrow_{\rho}\ & F\ (Y\ F)\ 3 \\
=\ & (\lambda f. \lambda n.n \text{ eq } 0 \to 1 \mid n*f(n-1))\ (Y\ F)\ 3 \\
\Rightarrow_{\beta}\ & (\lambda n.n \text{ eq } 0 \to 1 \mid n*\ Y\ F\ (n-1))\ 3 \\
\Rightarrow_{\beta,\delta}\ & 3*\ Y\ F\ (2) \\
\Rightarrow_{\rho}\ & 3*\ F\ (Y\ F)\ 2
\end{aligned}
$$

Let's Do the Factorial (cont'd)

$= \quad 3 * (\lambda f.\ \lambda n.n \text{ eq } 0 \rightarrow 1 \mid n*f(n-1))\ (Y\ F)\ 2$

$=>_{\beta} \quad 3 * (\lambda n.n \text{ eq } 0 \rightarrow 1 \mid n*\ Y\ F\ (n-1))\ 2$

$=>_{\beta,\delta} \quad 3*2*\ Y\ F\ (1)$

$=>_{\rho} \quad 3*2*\ F\ (Y\ F)\ 1$

$= \quad 3*2* (\lambda f.\ \lambda n.n \text{ eq } 0 \rightarrow 1 \mid n*f(n-1))\ (Y\ F)\ 1$

$=>_{\beta} \quad 3*2* (\lambda n.n \text{ eq } 0 \rightarrow 1 \mid n*\ Y\ F\ (n-1))\ 1$

$=>_{\beta,\delta} \quad 3*2*1*\ Y\ F\ (0)$

$=>_{\rho} \quad 3*2*1*\ F\ (Y\ F)\ 0$

$= \quad 3*2*1* (\lambda f.\ \lambda n.n \text{ eq } 0 \rightarrow 1 \mid n*f(n-1))\ (Y\ F)\ 0$

$=>_{\beta} \quad 3*2*1* (\lambda n.n \text{ eq } 0 \rightarrow 1 \mid n*\ Y\ F\ (n-1))\ 0$

$=>_{\beta,\delta} \quad 3*2*1*1$

$=>_{\delta} \quad 6$

Subtree transformation for ‘rec’

- Build AST for factorial, and standardize it.
- RPAL subtree transformation rule for ‘rec’. (remember we skipped it ?)
- Example: factorial (see diagram)

Standardizing 'rec'

```
rec   =>     =
 |          / \
 =         X   gamma
/ \              /  \
X  E         Ystar  lambda
                     /  \
                    X    E
```

Control Structures and CSE Machine Evaluation

- See diagram.

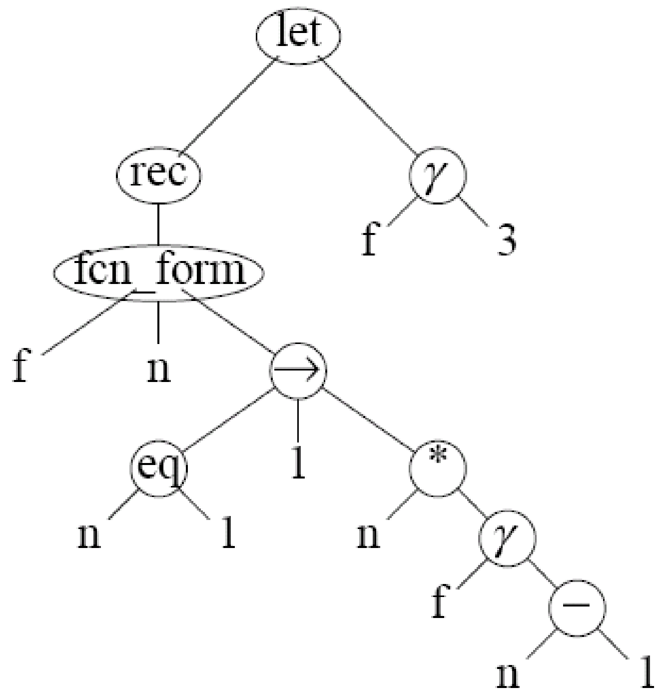
- CSE Rule 12:
  - Applying Y to F
  - Results in in (Y F). We represent it using a single symbol : η

- CSE Rule 13:
  - Applying F to Y F.
  - Results in F (Y F)

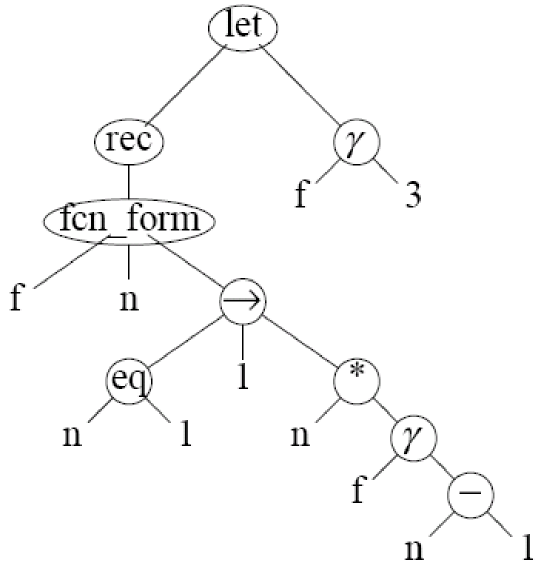
Recursion Example (factorial)

- `let rec f n = n eq 1 -> 1 | n * f (n-1) in f 3`

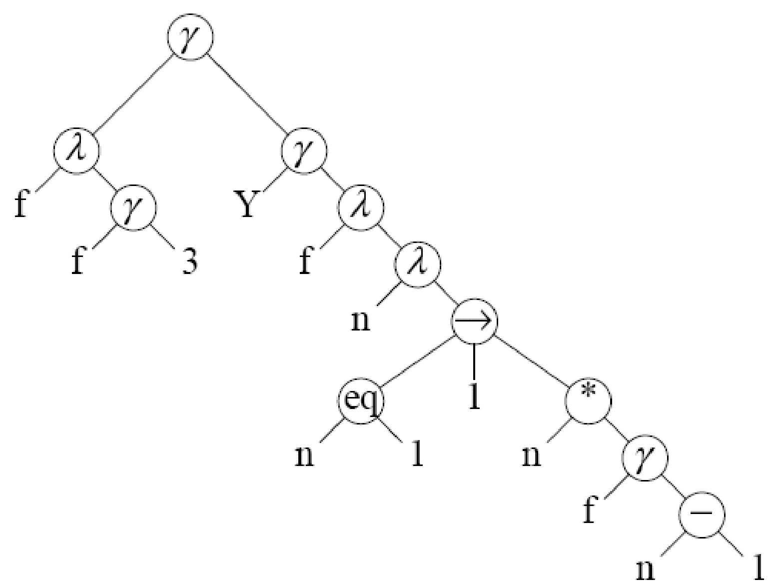
AST:



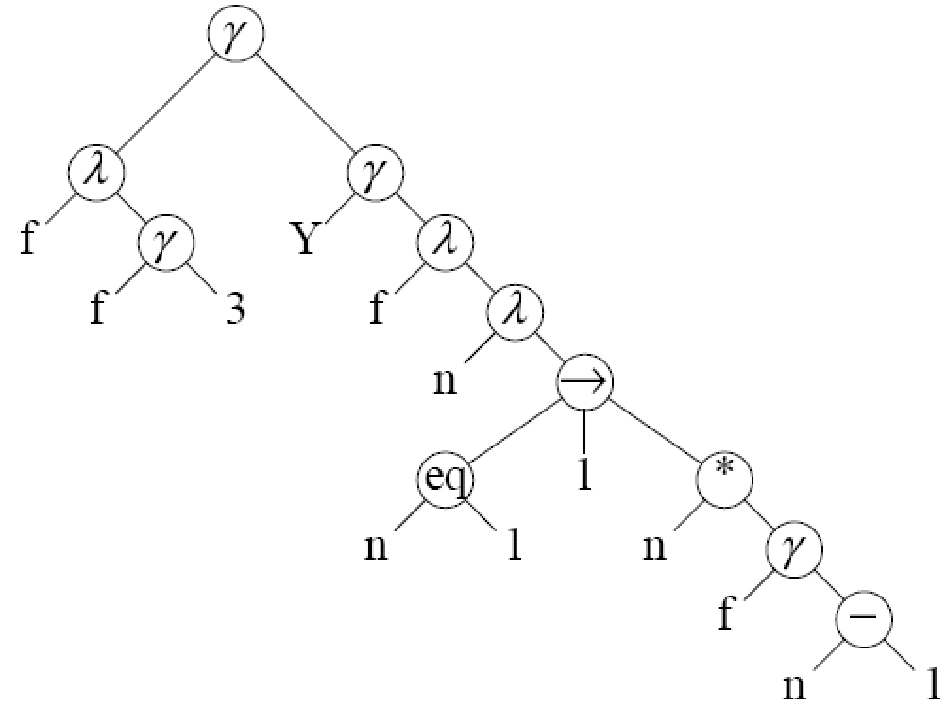
AST:



Standardized tree:



- Standardized tree:



Control Structures:

$$\delta_0 = \gamma\ \lambda_1^f\ \gamma\ Y\ \lambda_2^f$$
$$\delta_1 = \gamma\ f\ 3$$
$$\delta_2 = \lambda_3^n$$
$$\delta_3 = \delta_4\ \delta_5\ \beta\ \text{eq}\ n\ 1$$
$$\delta_4 = 1$$
$$\delta_5 = *\ n\ \gamma\ f\ \text{-}\ n\ 1$$

CSE Rules:

	CONTROL	STACK	ENV
CSE Rule 12	.... γ	$Y\ {}^{c}\lambda_i^v$	
(applying Y)	....	${}^{c}\eta_i^v$	
CSE Rule 13	.... γ	${}^{c}\eta_i^v$ R	
(applying f.p.)	.... $\gamma\ \gamma$	${}^{c}\lambda_i^v\ {}^{c}\eta_i^v$ R	

CSE Evaluation:

RULE	CONTROL	STACK	ENV
2	$e_0\ \gamma\ \lambda_1^f\ \gamma\ Y\ \lambda_2^f$	e_0	e_0=PE
1	$e_0\ \gamma\ \lambda_1^f\ \gamma\ Y$	${}^{0}\lambda_2^f\ e_0$	
12	$e_0\ \gamma\ \lambda_1^f\ \gamma$	$Y\ {}^{0}\lambda_2^f\ e_0$	
2	$e_0\ \gamma\ \lambda_1^f$	${}^{0}\eta_2^f\ e_0$	
4	$e_0\ \gamma$	${}^{0}\lambda_1^f\ {}^{0}\eta_2^f\ e_0$	
1	$e_0\ e_1\ \gamma$ f 3	$e_1\ e_0$	$e_1=[{}^{0}\eta_2^f/f]e_0$
1	$e_0\ e_1\ \gamma$ f	$3\ e_1\ e_0$	
13	$e_0\ e_1\ \gamma$	${}^{0}\eta_2^f\ 3\ e_1\ e_0$	

4	$e_0\ e_1\ \gamma\ \gamma$	${}^0\lambda_2^f\ {}^0\eta_2^f\ 3\ e_1\ e_0$	
2	$e_0\ e_1\ \gamma\ e_2\ \lambda_3^n$	$e_2\ 3\ e_1\ e_0$	$e_2=[{}^0\eta_2^f/f]e_0$
5	$e_0\ e_1\ \gamma\ e_2$	${}^2\lambda_3^n\ e_2\ 3\ e_1\ e_0$	
4	$e_0\ e_1\ \gamma$	${}^2\lambda_3^n\ 3\ e_1\ e_0$	
1	$e_0\ e_1\ e_3\ \delta_4\ \delta_5\ \beta$ eq n 1	$e_3\ e_1\ e_0$	$e_3=[3/n]e_2$
1	$e_0\ e_1\ e_3\ \delta_4\ \delta_5\ \beta$ eq n	$1\ e_3\ e_1\ e_0$	
6	$e_0\ e_1\ e_3\ \delta_4\ \delta_5\ \beta$ eq	$3\ 1\ e_3\ e_1\ e_0$	
8	$e_0\ e_1\ e_3\ \delta_4\ \delta_5\ \beta$	false $e_3\ e_1\ e_0$	
1	$e_0\ e_1\ e_3$ * n γ f - n 1	$e_3\ e_1\ e_0$	
1	$e_0\ e_1\ e_3$ * n γ f - n	$1\ e_3\ e_1\ e_0$	
6	$e_0\ e_1\ e_3$ * n γ f -	$3\ 1\ e_3\ e_1\ e_0$	
1	$e_0\ e_1\ e_3$ * n γ f	$2\ e_3\ e_1\ e_0$	
13	$e_0\ e_1\ e_3$ * n γ	${}^0\eta_2^f\ 2\ e_3\ e_1\ e_0$	
4	$e_0\ e_1\ e_3$ * n $\gamma\ \gamma$	${}^0\lambda_2^f\ {}^0\eta_2^f\ 2\ e_3\ e_1\ e_0$	
2	$e_0\ e_1\ e_3$ * n $\gamma\ e_4\ \lambda_3^n$	$e_4\ 2\ e_3\ e_1\ e_0$	$e_4=[{}^0\eta_2^f/f]e_0$
5	$e_0\ e_1\ e_3$ * n $\gamma\ e_4$	${}^4\lambda_3^n\ e_4\ 2\ e_3\ e_1\ e_0$	
4	$e_0\ e_1\ e_3$ * n γ	${}^4\lambda_3^n\ 2\ e_3\ e_1\ e_0$	

1	$e_0\ e_1\ e_3\ *\ n\ e_5\ \delta_4\ \delta_5\ \beta$ eq n 1	$e_5\ e_3\ e_1\ e_0$	$e_5=[2/n]e_4$
1	$e_0\ e_1\ e_3\ *\ n\ e_5\ \delta_4\ \delta_5\ \beta$ eq n	$1\ e_5\ e_3\ e_1\ e_0$	
6	$e_0\ e_1\ e_3\ *\ n\ e_5\ \delta_4\ \delta_5\ \beta$ eq	$2\ 1\ e_5\ e_3\ e_1\ e_0$	
8	$e_0\ e_1\ e_3\ *\ n\ e_5\ \delta_4\ \delta_5\ \beta$	false $e_5\ e_3\ e_1\ e_0$	
1	$e_0\ e_1\ e_3\ *\ n\ e_5\ *\ n\ \gamma$ f - n 1	$e_5\ e_3\ e_1\ e_0$	
1	$e_0\ e_1\ e_3\ *\ n\ e_5\ *\ n\ \gamma$ f - n	$1\ e_5\ e_3\ e_1\ e_0$	
6	$e_0\ e_1\ e_3\ *\ n\ e_5\ *\ n\ \gamma$ f -	$2\ 1\ e_5\ e_3\ e_1\ e_0$	
1	$e_0\ e_1\ e_3\ *\ n\ e_5\ *\ n\ \gamma$ f	$1\ e_5\ e_3\ e_1\ e_0$	
13	$e_0\ e_1\ e_3\ *\ n\ e_5\ *\ n\ \gamma$	${}^0\eta_2^f\ 1\ e_5\ e_3\ e_1\ e_0$	
4	$e_0\ e_1\ e_3\ *\ n\ e_5\ *\ n\ \gamma\ \gamma$	${}^0\lambda_2^f\ {}^0\eta_2^f\ 1\ e_5\ e_3\ e_1\ e_0$	
2	$e_0\ e_1\ e_3\ *\ n\ e_5\ *\ n\ \gamma\ e_6\ \lambda_3^n$	$e_6\ 1\ e_5\ e_3\ e_1\ e_0$	$e_6=[{}^0\eta_2^f/f]e_0$
5	$e_0\ e_1\ e_3\ *\ n\ e_5\ *\ n\ \gamma\ e_6$	${}^6\lambda_3^n\ e_6\ 1\ e_5\ e_3\ e_1\ e_0$	
4	$e_0\ e_1\ e_3\ *\ n\ e_5\ *\ n\ \gamma$	${}^6\lambda_3^n\ 1\ e_5\ e_3\ e_1\ e_0$	
1	$e_0\ e_1\ e_3\ *\ n\ e_5\ *\ n\ e_7\ \delta_4\ \delta_5\ \beta$ eq n 1	$e_7\ e_5\ e_3\ e_1\ e_0$	$e_7=[1/n]e_6$
1	$e_0\ e_1\ e_3\ *\ n\ e_5\ *\ n\ e_7\ \delta_4\ \delta_5\ \beta$ eq n	$1\ e_7\ e_5\ e_3\ e_1\ e_0$	
6	$e_0\ e_1\ e_3\ *\ n\ e_5\ *\ n\ e_7\ \delta_4\ \delta_5\ \beta$ eq	$1\ 1\ e_7\ e_5\ e_3\ e_1\ e_0$	

8	$e_0\ e_1\ e_3$ * n e_5 * n e_7 δ_4 δ_5 β	true $e_7\ e_5\ e_3\ e_1\ e_0$	
1	$e_0\ e_1\ e_3$ * n e_5 * n e_7 1	$e_7\ e_5\ e_3\ e_1\ e_0$	
5	$e_0\ e_1\ e_3$ * n e_5 * n e_7	1 $e_7\ e_5\ e_3\ e_1\ e_0$	
1	$e_0\ e_1\ e_3$ * n e_5 * n	1 $e_5\ e_3\ e_1\ e_0$	
6	$e_0\ e_1\ e_3$ * n e_5 *	2 1 $e_5\ e_3\ e_1\ e_0$	
5	$e_0\ e_1\ e_3$ * n e_5	2 $e_5\ e_3\ e_1\ e_0$	
1	$e_0\ e_1\ e_3$ * n	2 $e_3\ e_1\ e_0$	
6	$e_0\ e_1\ e_3$ *	3 2 $e_3\ e_1\ e_0$	
5	$e_0\ e_1\ e_3$	6 $e_3\ e_1\ e_0$	
5	$e_0\ e_1$	6 $e_1\ e_0$	
5	e_0	6 e_0	
-	-	6	

Whew !

Exercise:

Show the environment tree.

Thank You!

REFERENCES

- Programming Language Pragmatics by Michael L. Scott. 3rd edition. Morgan Kaufmann Publishers. (April 2009).
- Lecture Slides of Dr.Malaka Walpola and Dr.Bermudez