Lecture 3-4 **27-30.1.2025**

Today's agenda:

Lambda calculus:

There are three constructs in Lambda calculus:

--variables
--function abstraction
--function application [**highest precedence**]

Pure Lambda calculus: there are no constant symbols (0,1,2,....), there are no operators like +, *,..

This means that $\lambda x.\ x*x$ $\lambda x.\ x + 1$ are not valid terms of pure LC.

*But there exist pure lambda terms for 0,1,2,..... and +, *, ..., We shall develop these terms later.*

Syntax of Pure Lambda calculus:

M ::= x	variables	term	
| (λx. M)	abstraction	term	
| (M N)	application	term	(M_1 M_2) is same as (M N)

Syntactic Convention:

1. f g h is equiv to (f g) h since function application is **left associative**.

2. λx. M N is equiv to λx. (M N) since function application has highest precedence

3. λx. λy. λz. M is equiv to λxyz. M [now we shall not use this shorthand]

Number of spaces after f,g, dot, M is irrelevant.

M ::= x | (λx. M) | (M N) syntax of pure LC

Examples of some valid lambda terms of pure LC (terms obtained from the above grammar):

1. x

2. (λx. x)

3. (x x)

4. (λx. ((λy. x) y))

Examples of some invalid lambda terms of pure LC:

1. $(\lambda x.\ x)\ y$
2. $x\ (\lambda x.\ x)$
3. $(\ \lambda x.\ x\ \ y)$

Problem1: Given a valid term of pure LC, can we remove one or more parentheses such that the meaning of the term remains same?

Examples:

$(\lambda x.\ x)$ becomes $\lambda x.\ x$ [any problem? No]

$(x\ \ x)$ becomes $x\ \ x$ [any problem? No]

$(\lambda x.\ (\ (\lambda y.\ x)\ \ y)\ )$ becomes $\lambda x.\ \lambda y.\ x\ \ y$ [any problem? yes]

So the question is to what extent the parentheses can be removed so that the meaning of the term remains same.

$t = (\lambda x.\ (\ (\lambda y.\ x)\ \ y)\ )$ removing the outermost ( ) has no effect

So we get $t1 = \lambda x.\ (\ (\lambda y.\ x)\ \ y)$

If we remove the outer parenthesis from t1, does the meaning change? i.e., $t2 = \lambda x.\ (\lambda y.\ x)\ \ y$

Let us examine. Let $M = (\lambda y.\ x)$ so t2 becomes $t3 = \lambda x.\ M\ \ y$

Since function application has the highest precedence so t3 is actually $\lambda x.\ (M\ \ y)$ which is t2.

Can we remove the parentheses from t2?

After removing we get $t4 = \lambda x.\ \lambda y.\ x\ \ y$

Since function application has the highest precedence so t4 is actually $t5 = \lambda x.\ \lambda y.\ (x\ \ y)$

But t5 is different from t3 and hence different from t. so after t2 we cannot remove the parentheses.

Now we do the reverse problem. Given t2, obtain a term as per the grammar of pure LC. From t2 -> t1 -> t

Note: for a better understanding write t as $t = (_1\ \lambda x.\ (_2\ (_3\ \lambda y.\ x)_3\ \ y)_2\ )_1$

for removing brackets, use bracket numbered

Exercise: do the above with numbered parentheses as above.

Problem2: Given an unambiguous lambda term, obtain an equivalent valid term of pure LC.

The syntax of pure LC can be appropriately modified to include the constants, operators, and arithmetic expressions and this class of languages is called LC.

Syntax of LC:

M ::= x | (λx. M) | (M N) | c | op | ...

From now on, for the sake of illustration, we shall use the syntax of LC.

How is a function application done?

((λx. 5x + 2) 2)

= 5.2 + 2 = 10 + 2 = 12

((λx. M) N) means **we look for appropriate places** in M for x that can be replaced or substituted by N

If x does not occur in M, the output of the above is M, e.g., λx. 0 = 0 for any N

By appropriate place we mean that x is not captured by another lambda term in M.

that is, x **occurs free in** M.

Eg, ((λx. (λx. x + 1)) 1) how would this be evaluated? [scope of a variable]

Here M = (λx. x +1)

The outer x is captured by the inner λx, this means that the outer x is not visible inside the inner λx, so we can rewrite it as λx. λy. y+1, so M does not contain x, so the answer would be M i.e., (λy. y+1) which by replacement is (λx. x +1).

End of lecture