Narendra Singh Rana
Lecture Notes of

13-01-2000

1 Curried Functions

1.1 some curried functions :

INCLEFT Increases the left of a PAIR n times
INCLEFT takes in a pair and replaces it's RIGHT by LEFT and increases LEFT by 1. Subsequent calls to INCLEFT repeat the process mentioned above.

INCLEFT	$\equiv \lambda$ **P(PAIR(INC(LEFT P))(LEFT P))**
(INCLEFT(PAIR $\overline{x}$ $\overline{y}$**))**	$\equiv$ **(PAIR** $\overline{x+1}$ $\overline{x}$**)**

apply INCLEFT to (PAIR $\overline{0}$ $\overline{0}$) n times.This returns a new PAIR with left as n and right as n-1

DEC returns the RIGHT of a (PAIR $\overline{n}$ $\overline{n-1}$**)**

DEC	$\equiv$ (λ $\overline{n}$**(RIGHT(**$\overline{n}$ **INCLEFT (PAIR** $\overline{0}$ $\overline{0}$**))))**
$\overline{n}$	$\equiv$ (λ**(f a)**f^n **a)**

TIMES returns a product of m and n

TIMES	$\equiv$ (λ($\overline{m}$ $\overline{n}$)($\overline{m}$**(PLUS** $\overline{n}$**))**$\overline{0}$)
ONESTEP	$\equiv \lambda$ **P(PAIR (DEC(LEFT P))(TIMES(LEFT P)(RIGHT P)))**

example:

(ONESTEP(PAIR $\overline{n}$ $\overline{1}$**))**	$\equiv$ **(PAIR** $\overline{n-1}$ $\overline{n}$**)**
(ONESTEP(PAIR $\overline{n}$ $\overline{1}$**))**	$\equiv$ **(PAIR** $\overline{n-2}$ $\overline{n(n-1)}$**)**

Using ONESTEP , factorial can now be defied as

FACT	$\equiv \lambda$ $\overline{n}$**(RIGHT (** $\overline{n}$ **ONESTEP (PAIR** $\overline{n}$ $\overline{1}$**)))**

2 Reduction

example:

((λ a(a a))(λ b (b b)))

2.1 Normal order reduction

In normal form reduction, each reduction step reduces the leftmost outermost parents.

It is also called 'lazy evaluation' as it leaves the inner application to be reduced later.

consider a lambda expression :

((λ a(a a))(λ b(c b))d)

a parse tree for this lambda expression is given below:

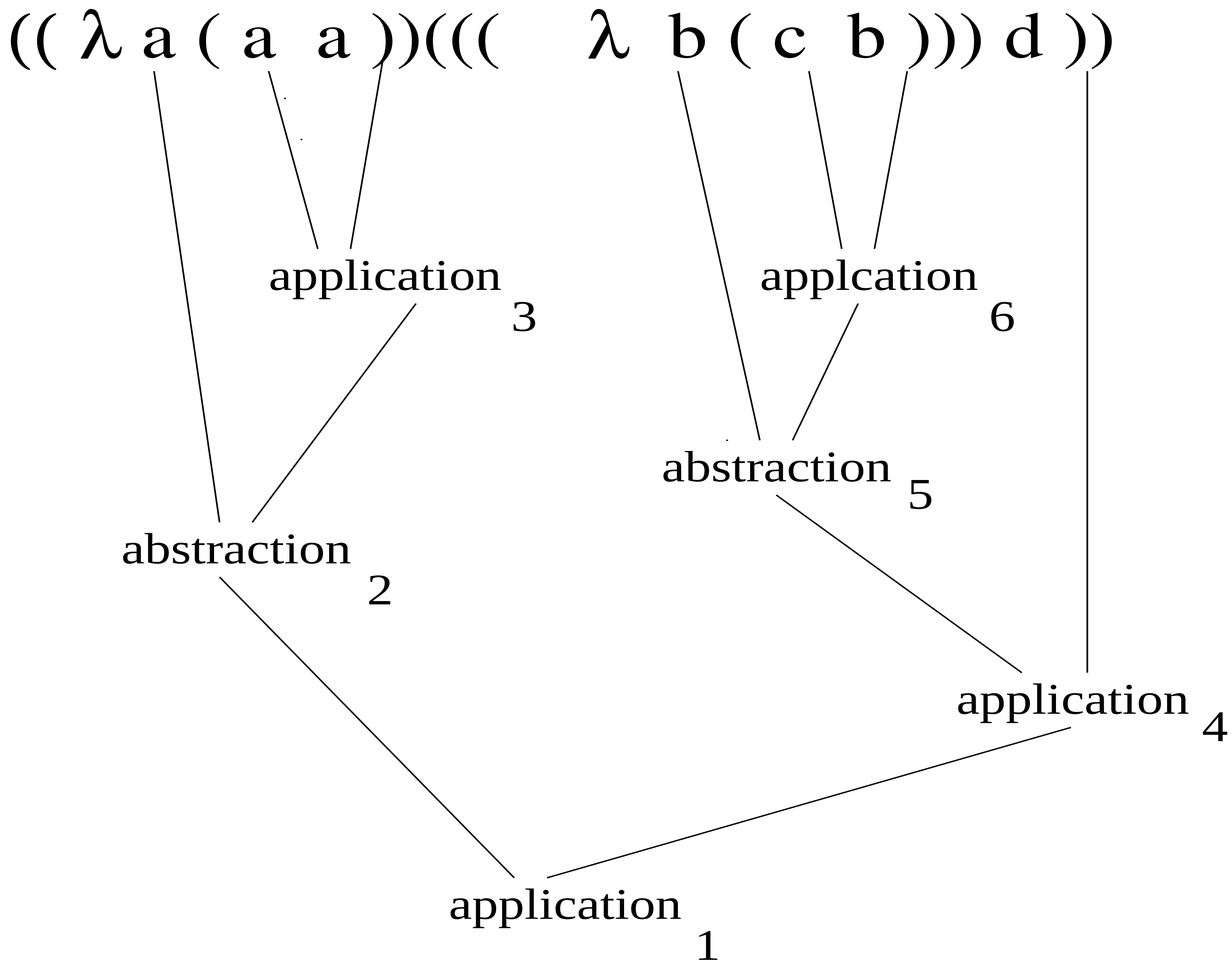

fig 1. **Parse tree for a lambda expression**

In the dfs tree shown above the outer most application where we can apply β reduction is the leftmost outermost β reduction.

The Reduction of the λ expression given above can be done using Normal order reduction as shown below:

$$
\begin{aligned}
((\lambda\ a(a\ a))(\lambda\ b(c\ b))d) \quad & \xRightarrow{\beta} ((\lambda\ b(c\ b))d((\lambda\ b(c\ b))d)) \\
& \xRightarrow{\beta/\eta} ((c\ d)((\lambda\ b(c\ b))d)) \\
& \xRightarrow{\beta/\eta} ((c\ d)(c\ d))
\end{aligned}
$$

2.2 Applicative order reduction

In applicative order reduction, the operator and the operand of the leftmost outermost β reducts are reduced to normal form and then β reduction is done.

It is also called eager evaluation as it is faster than Normal order reduction.

Applying Applicative order reduction to the λ expression given earlier we can reduce it to normal form as shown below:

$$
\begin{aligned}
((\lambda\ a(a\ a))(\lambda\ b(c\ b))d) \quad & \xRightarrow{\beta} ((\lambda\ a(a\ a))(c\ d)) \\
& \xRightarrow{\beta} ((c\ d)(c\ d))
\end{aligned}
$$

A parse tree comparing Normal order reduction and Applicative order reduction
is shown below:

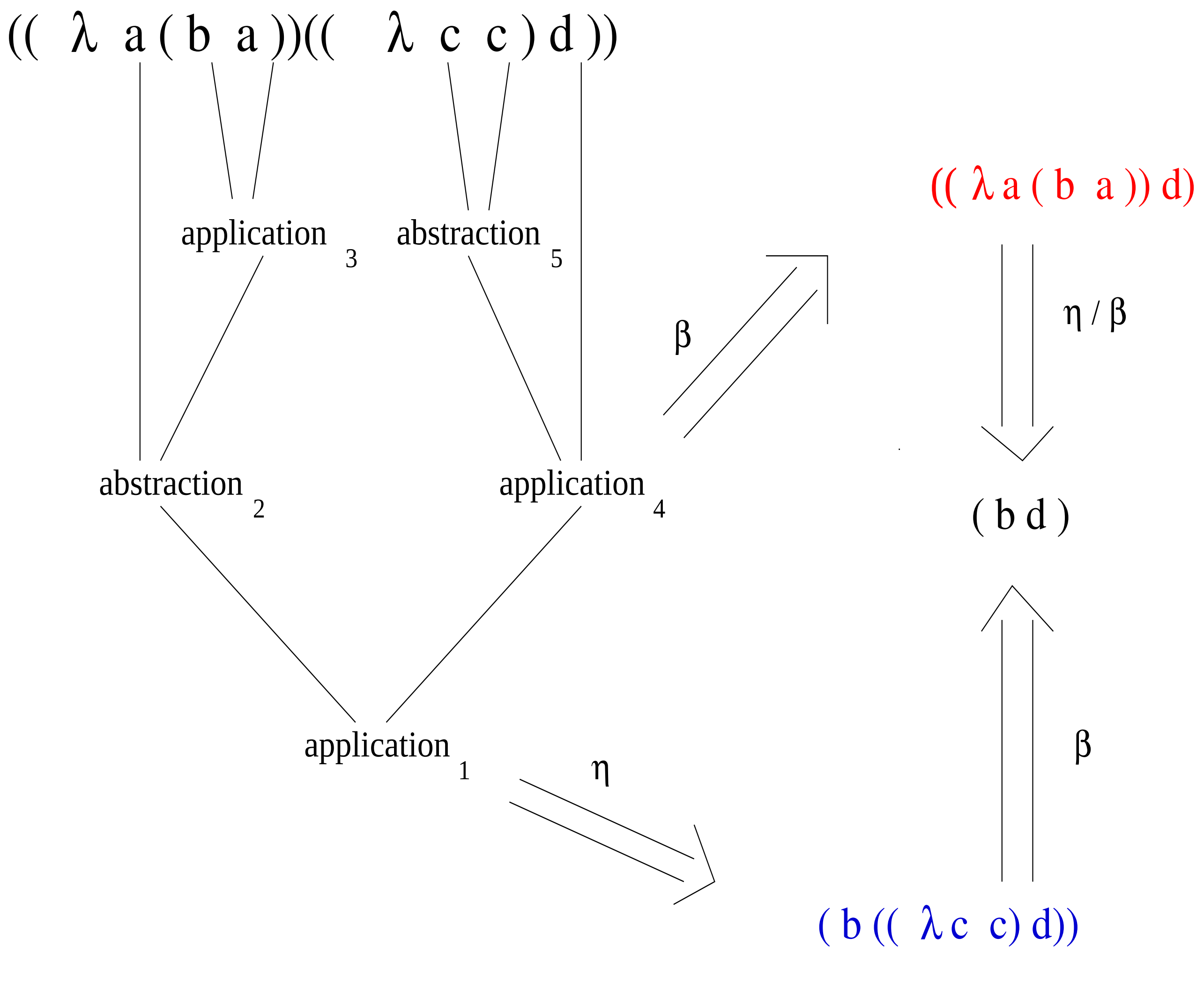

fig 2. **Applicative order reduction** **and** **Normal order reduction**

3 Church Rosser Theorem. 1

If T_i is a term and

If $T_1 \Leftrightarrow T_2$ then $\exists$ T_3

such that $T_1 \stackrel{*}{\Rightarrow} T_3$

and $T_2 \stackrel{*}{\Rightarrow} T_3$

3.1 Corollary

4 Church Rosser Theorem. 2

A term T can't be reduced to different normal forms(Although the normal forms may be α reducible to each other)

5 Church Rosser Theorem. 2

If $T_1 \stackrel{*}{\Rightarrow} T_2$
and T_2 is in normal form(can't be reduced further)
then $\exists$ a Normal order reduction from T_1 to T_2.

6 Partial functions

Domain (D) $\equiv$ a finite λ expression.
Rangen (R) $\equiv$ a finite λ expression.
Total function = (D,R,G)
G = (a,b),a ϵD,b ϵR
such that, for each of dϵD $\exists$ a unique (d,y) in G

Domain - D
Range - R = D$\bigcup\{\perp\}$
here $\perp$ is the bottom marker and can be interpreted as
non terminating computation.