Czech Technical University
Faculty of Informatics
Department of Theoretical Informatics

BI-PPA 2017 **Cvi Introduction to λ-calculus** Date: 3. 11. 2017

What are we going to cover Arithmetic expressions and their evaluation, relation to binry trees. Introduction to the λ-calculus, basics of the syntax and evaluations. Examples of functions and their construction.

Ex. 1. For the following simple expressions, create the expression tree and transform them to prefix notation: 147, $3 + 4, 3 + (7 * 5), (8 + (4 * x)) + (3 * y)$.

Ex. 2. Write expressions for the following operations and then transform them into prefix notation.

a) Increment variable x by one.

b) Multiply two variables x a y.

c) Add squares of variables x and y.

Ex. 3. Write the expressions from previous example in C language and then into λ-calculus.

λ-calculus syntax:

- λ... defines a function (and its "name")
- x ... is **bound** variable (input argument for λfunctions)
- + x 1 ... example of expression definition in prexix notation
- (λx. (+ x 1)) ... example of function definition in λ-calculus which corresponds to the expression above.

  ... Don't forget the enclosing parentheses.
- (λx. (+ x y)) ... Definition of function with an argument (**bound** variable) x, which will be added to y (**free** variable, uspecified argument)
- (λx. (+ x 5))3 ... denotes substitution of value 3 for variable x (**application**) in the function above (i.e. adds x (replaced with 3) to 5). The result is (+ 3 5) and therefore 8.
- (λx. (λy. (+ x y))5)3 ... is **application** of function (where x = 3 is added to y = 5. The result is again 8.

Rules of the λ-calculus:

1. Variable is a valid expression in λ-calculus (any lowercase letter from english alphabet).
2. If M and N are valid λ-calculus expressions, then the following are also valid expressions:
   - (M) ... enclosing an expression in parentheses,
   - $\lambda id.\ M$ ... so called **abstraction**, where id is any variable,
   - MN ... so called **application**, where N is applied to M.

Free and Bound Variables

- (λx. x), x is bound
- (λx. (x y)), x is bound, y is free
- (λx. (x x))x, x is bound in the two inner uses, free in the outer one.
- (λy. (+ x y))(λx. (+ x 1)) ... which variables and their uses are bound and which are free?
- ((λy. (yxx)) y x)
- ((λx.(λx.(λx.x)x)x)x)
- (λx. y(λy. x(λx. xz(λy.yx))))

Ex. 4. Think in λ-calculus! Define own λ-functions for the expressions in exercise 1 and more. Some examples for your inspiration:

a) 147 $\Rightarrow$ (λx. x) (147),

b) $3 + 4 \Rightarrow$ (λx. (λy. (+ x y)) 4) 3,

c) $3 + (7 * 5) \Rightarrow$ (λx. (λy. (+ 3 (* x y))) 5) 7, or as two operations + and * with value substitution: (λx. (λy. (+ x y)) ((λl. (λr. (* l r)) 5) 7)) 3

d) $(8 + (4 * x)) + (3 * y)$ for values x = 4 and y = 3 $\Rightarrow$ (λl. (λr. (+ l r)) (λy. (* 3 y))) (λx. + 8 (*4 x)), with values (λl. (λr. (+ l r)) ((λy. (* 3 y)) 3)) ((λx. + 8 (*4 x)) 4). Try your own solution.

Simplifying the notation:

- Expressions in the form of (((((AB)C)D)E)F),
- (λx. (λy. (λz. ((x y) z)))) can be written as (λx.λy.λz. ((x y) z)) and then as (λxyz. (x y z)).
- Analogically we will use (λxyz. (x y z)) 1 2 3 in the meaning of (λx. (λy. (λz. ((x y) z)) 3) 2) 1.
- Discarding the inner parentheses, i.e. instead of (λxyz. (x y z)) we can use (λxyz. x y z) and instead of (λxyz. (+ x (− yz))) we will write (λxyz. + x (− y z)).

Example: (λx. (λy. (+ x y)) 4) 3 in simplified form (λxy. + x y) 3 4 will be transformed in the following way:
(λxy. + x y) 3 4 $\rightarrow$ (λy. + 3 y) 4 $\rightarrow$ (+ 3 4) $\rightarrow$ 7.

Ex. 5. Remove extra parentheses in the following expressions: Calculus

a) (λx. (λy. (λz.((xz)(yz))))) **Solution:** (λxyz.(xz)(yz))

b) (((ab)(cd))((ef)(gh))) **Solution:** only outer parentheses: ((ab)(cd))((ef)(gh))

c) (λx. ((λy. (yx)) (λv.v)z)u)(λw.w) **Solution:** (λx. (λy. yx) (λv.v) z u)(λw.w)

Ex. 6. Insert parentheses so that the following expressions are valid:

a) xxxx, **Solution:** (((xx)x)x)

b) λx. x.λy.y **Solution:** (λx. x(λy.y))

c) λx. (x λy. yxx)x **Solution:** (λx. (x (λy. (yx)x))x)

Ex. 7. Guess what will be the result of the following expressions.

a) (λx. (λy. (− x y)) 2) 5 . . . What will be the result? 5 − 2 or 2 − 5? **Solution:** 5-2

b) (λx. (λy. (− x y)) 5) 2 . . . What will be the result? 5 − 2 or 2 − 5? **Solution:** 2-5

c) (λxy. (− x y)) 5 2 . . . Mind the order of application. What will be the result? 5 − 2 or 2 − 5? **Solution:** 5-2

Ex. 8. Transform the following λ-calculus expressions, write the respective steps as expression trees.

a) (λx. (* (+ 3 x) (− x 4))) 5 **Solution:** (* (+ 3 5) (− 5 4)) → (* 8 (− 5 4)) → (* 8 1) → 8

b) (λx y. (* (+ y x) (− x 4))) 5 2 **Solution:** (λy. (* (+ y 5) (− 5 4))) 2 → (* (+ 2 5) (− 5 4)) . . .

c) (λx . (λy. (* (+ y x) (− x 4))) 2) 5 **Solution:** (λy. (* (+ y 5) (− 5 4)) 2) → (* (+ 2 5) (− 5 4)) . . .

d) (λx . (λy. (* (+ y x) (− x 4))) 5) 2 **Solution:** (λy. (* (+ y 2) (− 2 4))) 5) → (* (+ 5 2) (− 5 4)) . . .

e) (λz .z) (λq. qq) (λs.sa) = ((λz .z) (λq. qq)) (λs.sa) **Solution:** (λq. qq) (λs.sa) → (λs.sa)(λs.sa) → (λs.sa)a → aa

f) (λz .zz) (λz .z) (λz .z q) **Solution:** (λz .z) (λz .z) (λz .z q) → (λz .z)(λz .z q) → (λz .z q)

g) (λs q .s q q) (λa.a) b **Solution:** (λq .(λa.a) q q) b → (λa.a) b b → b b

h) (λs. ss) (λq.q) (λq.q) **Solution:** (λq.q) (λq.q) (λq.q) → (λq.q) (λq.q) → (λq.q)

i) (λf. (λx. f(f(x)))) (λy. * y y) 2 **Solution:** (λx. (λy. * y y) ((λy. * y y) (x))) 2 → (λy. * y y) ((λy. * y y) (2))
→ * ((λy. * y y) (2)) ((λy. * y y) (2))→ * (* 2 2) ((λy. * y y) (2))
→ * (* 2 2) (*2 2)→ * 4 4 → 16

j) (λfxy. f x y) (λga.ggga) (λhb.hb). **Solution:** (λxy. (λga.ggga) x y) (λhb.hb) → (λy. (λga.ggga) (λhb.hb) y)
→ (λy. (λa.(λhb.hb)(λhb.hb)(λhb.hb)a) (λhb.hb) y)
→ (λy. (λhb.hb)(λhb.hb)(λhb.hb) (λhb.hb) y)
→ (λy. (λb. (λhb.hb) b)(λhb.hb) (λhb.hb) y)
→ (λy. (λhb.hb) (λhb.hb) (λhb.hb) y)
→ (λy. (λb.(λhb.hb) b) (λhb.hb) y)
→ (λy. (λhb.hb) (λhb.hb) y)
→ (λy. (λb.(λhb.hb) b) y)
→ (λy. (λhb.hb) y)
→ (λy. (λb.yb)) → (λyb. yb)

Homework 1. Find online λexpression solvers and play with various expressions. For example:

- http://www.nyu.edu/projects/barker/Lambda/
- http://www.cburch.com/dev/lambda/index.html
- http://www.itu.dk/people/sestoft/lamreduce/lamframes.html