Czech Technical University
Faculty of Informatics
Department of Theoretical Informatics

BI-PPA 2017

Cvi Introduction to the λ-calculus

Date: 3. 11. 2017

We are going to cover Functions in λ-calculus, α-, β- a η-reductions, order of evaluation (normal vs. aplicative), the normal form, arithmetics in λ-calculus

Rules of the λ-calculus (from last exercise):

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.

$\{(,), ., \lambda, a, \ldots, z\}$, starting nonterminal is `<exp>` and the rules are

Simplifying the notation (from last exercise:

- 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. 1.** Transform the following.

a) (λx. + x 1)3 **Solution:** (λx. + x 1)3 . . . substitute 3 for x
$\rightarrow$ + 3 1 $\rightarrow$ 4

b) (λxy. − x y)3 5 **Solution:** (λxy. − x y)3 5 . . . substitute 3 for x
$\rightarrow$ (λy. − 3 y) 5 . . . substitute 5 for y
$\rightarrow$ − 3 5 $\rightarrow$ −2

c) (λx. (λy. − x y)3) 5, ! diffenent than the example above. **Solution:** (λx. (λy. − x y)3) 5 . . . substitute 5 for x!!!
$\rightarrow$ (λy. − 5 y) 3 . . . substitute 3 for y
$\rightarrow$ − 5 3 $\rightarrow$ 2

d) (λx. + x 1)((λy. + y 2)3) **Solution:** (λx. + x 1)((λy. + y 2)3) . . . substitute ((λy. + y 2)3) for x
$\rightarrow$ + ((λy. + y 2)3) 1 . . . substitute 3 for y
$\rightarrow$ + (+ 3 2) 1 $\rightarrow$ + 5 1 $\rightarrow$ 6

e) (λfx.f x)(λy. + y 1) **Solution:** (λfx.f x)(λy. + y 1) …substitute (λy. + y 1) for f
→ (λx. (λy. + y 1) x) …substitute x for y
→ (λx. + x 1) …cannot be transformed further

f) (λfx.f x)(λy. + y 1) 5 **Solution:** (λfx.f x)(λy. + y 1) 5 …substitute (λy. + y 1) for f
→ (λx. (λy. + y 1) x) 5 …substitute x for y
→ (λx. + x 1) 5 …substitute 5 for x
→ + 5 1 → 6.

g) (λf x. fx)(λy.y) **Solution:** (λf x. fx)(λy.y) …substitute (λy.y) for f
→ (λx. (λy.y) x) … λ. x cannot be further transformed, so substitute x for y in function (λy.y)
→ (λx. x)

h) (λx.(+ x ((λx. + x 1)3)))2 **Solution:** (λx.(+ x ((λx.+ x 1)3)))2 …substitute 2 for x – the leftmost one
→ (+ 2 ((λx. + x 1)3)) …3 substituujeme for x
→ + 2 (+ 3 1) → + 2 4 → 6

i) ((λx.λy.x)y)z, be careful about free and bound variables. **Solution:** ((λx.λy.x)y)z …substitute y for x, which creates binding for the free y!!! we must therefore first rename
→ ((λx.λt.x)y)z …and then substitute y for x,
→ (λt.y)z …substitute z for t,
→ y **Solution:** BE WARY OF THE COMMON MISTAKE:
((λx.λy.x)y)z …substitute y for x, (not noticing the collision of y)
→ (λy.y)z …then substitute z for y, (formerly free y is now bound with function λy.)
→ z …and the resylt is incorrect!

j) (λs. λq. s q q) (λq.q) q 5**Solution:** (λs. λq. s q q) (λq.q) q …we must first rename λq. to λt
→ (λs. λt. s t t) (λq.q) q …and only then substitute (λq.q) for s
→ (λt. (λq.q) t t) q …no need to rename now (but we can), substitute q for t
→ (λq.q) q q …and again rename λq. to for instance λz.
→ (λz.z) q q …substitute q for z
→ q q …second q remains unchanged and we have the result

Reductions, normal form:

- **β-reduction** (Appplication): (λx. E) A, all bound uses of x in the expression E will be replaced with A. AS LONG AS A does not have other free variables that might collide.
  Example: (λxy. (x y)) (ay) CANNOT transform to (λy. ((ay) y)), because y would become bound variable in (ay) (the solution is to use α-reduction). But (λxy. (x y)) (az) is fine and will look as (λy. ((az) y)) after the application.

- **α-reduction** (Renaming): Based on the principle that (λx.x) and (λy.y) are identical functions, because variable names alone do not matter. It is therefore only renaming of all bound uses of x to y, but ONLY IF y was not free in E, in which case we must select different letter. **Example:** (λxy. (x y)) y IS NOT (λy. (y y)), because y would be bound. We'll rename first using (α-reduction) y to t: (λxt. (x t))y, and then can apply y to x with result: (λt. (y t)). This reduction is used before β-reduction in which free variables might be incorrectly bound.

- **η-reduction** (Optimalization): Special case of the β-reduction, where instead of application, we just delete the lambda. It is only useful for expression in the form of (λx.A x), where bound x is the rightmost element in the function definition and there are no uses of x in A. For example (λx.(A x))B replaces lambda with B and the result is AB. Using η-reduction x and λ will be deleted and connected to the rest of the form after the parentheses, i.e. we will remove x and λ from the function (λx.A x)B and end up with having AB as well.

Functions in **normal form** are those functions for which there is no further reduction possible using either β-rductions or η-reductions, only renamings (α-reductions).p. **Ex. 2**. Further examples: What is the result of the following expressions?

a) (λy. + 8 y)((λx. + x 1) 3)
Solution: ((λx. + x 1)3) substitute for y
→ + 8 ((λx. + x 1) 3) ...3 substitute for x
→ + 8 (+ 3 1)

b) (λx. (λx. (λy. * x y)3) ((λz.+ x z)2))1
Solution: substitute bound use of x for 1 with first λ(to x v λz.)
→ (λx. (λy. * x y)3)((λz.+ 1 z)2)...substitute ((λz.+ x z)2) for x
→ (λy. * ((λz.+ 1 z)2) y) 3 ...substitute 3 for y
→ * ((λz.+ 1 z)2) 3 → * (+ 1 2) 3

c) (λx. x i)((λz. (λq. q) z) h)
Solution: celou závorka ((λz. (λq. q) z) h) substitute for x
→ ((λz. (λq. q) z) h) i ...h subsitutes for z → ((λq. q) h) i ...h is substituted for z
→ ((λq. q) h) i → (h)i ...h substitutes for q

d) (λx. x i)((λz. (λq. q z)) h)
Solution: substitute x for the entire parenthesis ((λz. (λq. q z)) h)
→ ((λz. (λq. q z)) h) i ...substitute x for h
→ ((λq. q h)) i...If we do not allow ellimination of paretheses, this is the nofmrl form, otherwise we cal substitute q for i
→ (ih)

e) (λx. x o j)((λy. (λz. z h)y)a)
Solution: ((λy. (λz. zh)y)a) susbtitute for x
→ ((λy. (λz. zh)y)a)oj...substitute for y
→ ((λz. zh)a)oj ...substitute for z
→ (ah)oj

f) (λx. (λx. (λx. xxx)(bx)x)(ax))c
Solution: substitute x for c
→ (λx. (λx. xxx)(bx)x)(ac) ...substitute (ac) for x
→ (λx. xxx)(b(ac))(ac) ...substitute (b(ac)) for x
→ (b(ac))(b(ac))(b(ac))(ac)

g) try writing lambda function that "prints" expression.

h) (λw.(λx.(λy.w y a) (u w)) b) y
Solution: → (λx.(λy.y y a) (uy)) b ...is this substitution of y for w correct? NO! We have bound previously free w in function λy. We must therefore first rename variable y to say t.
→ (λw.(λx.(λt.w t a) (u w)) b) y ...Now we can proceed with substitution of y for w.
→ (λx.(λt. y t a) (u y)) b ...b is replaced with x
→ (λt.y t a) (u y) ...(u y) for t
→ y (u y) a

i) (λp. (λq. (λp. p (p q))(λr. + p r))(+ p 4))2 ... normal vs applicative evaluation

Solution: 2 substitute for x. The bound use of x is in the last two parentheses.

→ (λq. (λp. p (p q))(λr. + 2 r))(+ 2 4) ... (+ 2 4) substitute for q
→ (λp. p(p (+ 2 4)))(λr. + 2 r)... (λr. + 2 r) substitute for p
→ (λr. + 2 r)((λr. + 2 r)(+2 4))... ((λr. + 2 r)(+2 4)) substitute for r
→ + 2 ((λr. + 2 r)(+2 4))... (+2 4) substitute for r
→ + 2 (+ 2 (+2 4))

There are many ways to simplify this example:

Solution: Applicative evaluation: Apply 2 on x.

→ (λq. (λp. (p(p q)))(λr. (+ 2 r)))(+ 2 4) ... (+ 2 4) **evaluate** to 6 and substitute for q
→ (λp. (p(p 6)))(λr. (+ 2 r))... (λr. (+ 2 r)) substitutes p
→ (λr. (+ 2 r))((λr. (+ 2 r)) 6)... 6 substitutes r
→ (λr. (+ 2 r))((+ 2 6))... (+ 2 6) evaluates to 8
→ (λr. (+ 2 r))(8)... 8 substitutes r
→ (+ 2 8)

Normal and applicative evaluation are two differet ways how to tranform λexpressions. **Normal** evaluation proceeds always from the left (i.e. what we did before). It has only one rule, which is to **strictly apply from the left** (i.e. substitute what is after the parentheses with λ) without any modifications. This means that the resulting expression might be more complicated than necessary. On the other hand, the **applicative** evaluation attempts to simplify an expression **before** ts application (substitution) on λ.

- An example of normal evaluation:
  (λx. + xx) ((λp. + p 4)3) leads to
  → + ((λp. + p 4)3)((λp. + p 4)3) ... **only substitute**, no other updates for the expression ((λp. +p 4)3) are made
  → + (+ 3 4) ((λp. + p 4)3)... and again evaluate from left
  → + (+ 3 4) (+ 3 4) → + 7 7 → 14
- An example of applicative evaluation:
  (λx. + xx) ((λp. + p 4)3) leads to
  → (λx. + xx) (+ 3 4) ... **ssimplify** second parenthesis before its application (because it is possible), and apply the result of the simplification
  → (λx. + xx) 7 ... update more and the subsstitute x for 7
  → + 7 7 → 14

While the second example looks much more elegant, the disadvantage of applicative evaluation is that there are cases in which applicative transformations may not lead to a normal form (but if it does, it gives the same result as normal evaluation - proved in 1936 in the Church-Rosser theorem). This is shown in the next example: dávat stejný výsledek.

a) (λx. (λw.(λy. wyw)b))((λx. xxx)(λx. xxx)) ((λz.z)a) ... transform using both normal and applicative evaluation order

Solution: entire parenthessis ((λx. xxx)(λx. xxx)) is subsstituted for x,but there is no use of x.

→ (λw.(λy. wyw)b)((λz.z)a)... ((λz.z)a) ssubstitute for w
→ (λy. ((λz.z)a) y ((λz.z)a)) b... b subsitute for y
→ ((λz.z)a) b ((λz.z)a) ... and for left z
→ (a) b ((λz.z)a) ... and for right z
→ (a) b (a)

Solution: And now applicative evaluation: the parenthesis ((λx. xxx)(λx. xxx)) can be ssimplified before substitution for x. The result of that is
((λx. xxx)(λx. xxx)(λx. xxx))...which can still be simplified by substituting x in the firsst parenthesis for (λx. xxx). The reesult is
((λx. xxx)(λx. xxx)(λx. xxx)(λx. xxx)(λx. xxx)(λx. xxx)(λx. xxx))...which can again be applied, and thesse applications can go forever, while normal evaluation did not encounter this problem.

b) (λxy. (+ x y))((λx. ((λx. λy. (+ y((λz. (* x z))3)))7 5)) x) 4
⇒ (+ (+ 5 (* 7 3)) 4) **Solution:** entire parentheesis ((λx. ((λx. λy. (+ y((λz. (* x z))3)))7 5)) x) is substituted for x
→ (λy. (+ ((λx. ((λx. λy. (+ y((λz. (* x z))3)))7 5)) x) y)) 4...4 is substituted for y
→ (+ ((λx. ((λx. λy. (+ y((λz. (* x z))3)))7 5)) x) 4) ...x subsstituted for x, which has no bound use.
→ (+ (((λx. λy. (+ y((λz. (* x z))3)))7 5)) 4)...7 substituted for x
→ (+ ((λy. (+ y((λz. (* 7 z))3)) 5)) 4) ...5 substituted for y
→ (+ ((+ 5((λz. (* 7 z))3))) 4) ...3 substituted for z
→ (+ ((+ 5((* 7 3)))) 4)...remove double parentheses
→ (+ (+ 5(* 7 3)) 4)

c) ((λx. xx)(λx. xx))

d) ((λx. R(xx))(λx. R(xx)))

Ex. 3. Transform the following expressions into normal forms using normal and then applicative order of evaluation.

a) (λs. (λq. s q q) (λq.q)) q **Solution:** both evaluations are te same: (λs. (λt. s t t) (λq.q)) q → (λt. q t t) (λq.q) → (λt. q t t) (λz.z) → q (λz.z) (λz.z)

b) (λx. ((λx y. (* x y)) 2 (+ x y))) y **Solution:**

c) (λx. λy. (x y)) (λz. (z y)) **Solution:**

d) (λx. y (λt. t t) (λx. x x)) **Solution:** infinite loop in both evaluaions: Nothing to do with the first λx, but the λt can be replaced with (λx. x x)
(λx. y (λx. x x) (λx. x x)) and again, the second λx can be replaced with (λx. x x) and so on.

e) (λx. y) ((λt. t t) (λx. x x)) **Solution:** substitute λx. with the entire expression ((λt. t t) (λx. x x)), but x is not in any bound use, so the result is y