CMSC330 Fall 2013 Practice Problems 8 Solutions

1. Lambda calculus

Make all parentheses explicit in the following λ-expressions

a. λx.xz λy.xy → **(λx.((x z) (λy.(x y))))**

b. (λx.xz) λy.w λw.wyzx → **((λx.(x z)) (λy.(w (λw.((((w y) z) x))))))**

c. λx.xy λx.yx → **(λx.((x y) (λx.(y x))))**

Find all free (unbound) variables in the following λ-expressions

d. λx.x z λy.x y → **(λx.((x z) (λy.(x y))))**

e. (λx. x z) λy. w λw. w y z x → **((λx.(x z)) (λy.(w (λw.((((w y) z) x))))))**

f. λx. x y λx. y x → **(λx.((x y) (λx.(y x))))**

Apply β-reduction to the following λ-expressions as much as possible

g. (λz.z) (λy.y y) (λx.x a) **// β-reduction = body[sym/replacement]**

(λz.z) (λy.y y) (λx.x a) → **// z[z/(λy.y y)] replace z with λy.y y**

(λy.y y) (λx.x a) → **// y y[y/(λx.x a)] replace y with λx.x a**

(λx.x a) (λx.x a) → **// x a[x/(λx.x a)] replace x with λx.x a**

(λx.x a) a → a a **// x a[x/a] replace x with a**

h. (λz.z) (λz.z z) (λz.z y)

(λz.z) (λz.z z) (λz.z y) → **// β-reduction: replace z with λz.z z**

(λz.z z) (λz.z y) → **// β-reduction: replace z with λz.z y**

(λz.z y) (λz.z y) → **// β-reduction: replace z with λz.z y**

(λz.z y) y → y y **// β-reduction: replace z with y**

i. (λx.λy.x y y) (λa.a) b

(λx.λy.x y y) (λa.a) b → **// β-reduction: replace x with λa.a**

(λy.(λa.a) y y) b → **// β-reduction: replace y with b**

(λa.a) b b → b b **// β-reduction: replace a with b**

j. (λx.λy.x y y) (λy.y) y

(λx.λy.x y y) (λy.y) y → **// α-conversion: rename y to a**

(λx.λa.x a a) (λy.y) y → **// β-reduction: replacing x with λy.y**

(λa.(λy.y) a a) y → **// β-reduction: replacing a with y**

(λy.y) y y → y y **// β-reduction: replacing y with y**

k. (λx.x x) (λy.y x) z

(λx.x x) (λy.y x) z → **// β-reduction: replacing x with λy.y x**

(λy.y x) (λy.y x) z → **// β-reduction: replacing y with λy.y x**

(λy.y x) x z → **// β-reduction: replacing y with x**

x x z

l. (λx. (λy. (x y)) y) z

(λx. (λy. (x y)) y) z → **// α-conversion: rename y to a**

(λx. (λa. (x a)) y) z → **// β-reduction: replacing x with z**

(λa. (z a)) y → **// β-reduction: replacing a with y**

z y

m. ((λx.x x) (λy.y)) (λy.y)

((λx.x x) (λy.y)) (λy.y) → **// β-reduction: replacing x with λy.y**
((λy.y) (λy.y)) (λy.y) → **// β-reduction: replacing y with λy.y**
(λy.y) (λy.y) → **// β-reduction: replacing y with λy.y**
λy.y

n. (((λx. λy.(x y))(λy.y)) w)

(((λx. λy.(x y))(λy.y)) w) → **// α-conversion: rename y to a**
(((λx. λa.(x a))(λy.y)) w) → **// β-reduction: replacing x with λy.y**
((λa.((λy.y) a)) w) → **// β-reduction: replacing a with w**
(λy.y) w → **// β-reduction: replacing y with λy.y**
w

Show that the following expression has multiple reduction sequences

o. (λx.y) ((λy.y y y) (λx.x x x))

// β-reduction: replace x in λx.y with ((λy.y y y) (λx.x x x))
// (no x in body, so just discard argument and replace (λx.y) <…> with y)
(λx.y) ((λy.y y y) (λx.x x x))→ y

OR

// β-reduction: replace y in λy.y y y with λx.x x x
(λx.y) ((λy.y y y) (λx.x x x))→ (λx.y) ((λx.x x x) (λx.x x x) (λx.x x x))

Can repeat β-reduction for x as many times as we wish!

2. Lambda calculus encodings

Prove the following using the appropriate λ-calculus encodings

a. not (not true) = true

Given:

not = λx.((x false) true)
true = λx.λy.x
false = λx.λy.y

Proof:

not (not true)	**// replacing 1^{st} not w/ encoding**
= λx.((x false) true) (not true)	**// β-reduction: x → not true**
= ((not true) false) true	**// replacing not w/ encoding**
= ((λx.((x false) true) true) false) true	**// β-reduction: x → true**
= (((true false) true) false) true	**// replace true w/ encoding**
= ((((λx.λy.x) false) true) false) true	**// β-reduction: 1^{st} x → false**
= (((λy.false) true) false) true	**// β-reduction: y → true**
= ((false) false) true	**// replace false w/ encoding**
= ((λx.λy.y) false) true	**// β-reduction: x → false**
= (λy.y) true	**// β-reduction: y → true**
= true	**// not (not true) = true**

b. or false true = true

Given:

or = λx. λy. ((x true) y)
true = λx.λy.x
false = λx.λy.y

Proof:

or false true // replacing or w/ encoding
= λx. λy. ((x true) y) false true // β-reduction: x → false
= λy. ((false true) y) true // β-reduction: y → true
= (false true) true // replace 1st false w/ encoding
= ((λx.λy.y) true) true // β-reduction: x → false
= (λy.y) true // β-reduction: y → true
= true // or false true = true

c. if false then x else y = y

Given:

if a then b else c = a b c
true = λx.λy.x
false = λx.λy.y

Proof:

if false then x else y // replacing if… w/ encoding
= false x y // replacing false w/ encoding
= (λx.λy.y) x y // β-reduction: x → x
= (λy.y) y // β-reduction: y → y
= y // if false then x else y = y

d. succ 2 = 3

Given:

2 = λf.λy.f (f y)
3 = λf.λy.f (f (f y))
succ = λz.λf.λy.f (z f y)

Proof:

succ 2 // replacing succ w/ encoding
= (λz.λf.λy.f (z f y)) 2 // β-reduction: z → 2
= λf.λy.f (2 f y) // expanding 2 w/ encoding
= λf.λy.f ((λf.λy.f (f y)) f y) // β-reduction: 1st f → f
= λf.λy.f ((λy.f (f y)) y) // β-reduction: 1st y → y
= λf.λy.f (f (f y)) // apply encoding for 3
= 3 // succ 2 = 3

e. (* 1 3) = 3

Given:

1 = λf.λy.f y
3 = λf.λy.f (f (f y))
M * N = λx.(M (N x))

Proof:

(* 1 3) // replacing * w/ encoding
= λx.(1 (3 x)) // replacing 3 w/ encoding

= λx.(1 (λf.λy.f (f (f y)) x)) // β-reduction: 1^{st} f → x
= λx.(1 (λy.x (x (x y)))) // replacing 1 w/ encoding
= λx.((λf.λy.f y) (λy.x (x (x y)))) // β-reduction: 1^{st} f w/ λy.x (x (x y))
= λx.(λy.(λy.x (x (x y)))) y // β-reduction: 1^{st} y → y
= λx.λy.x (x (x y)) // α-conversion: replace x with f
= λf.λy.f (f (f y)) // apply encoding for 3
= 3

f. (+ 2 1) = 3

Given:

1 = λf.λy.f y
2 = λf.λy.f (f y)
3 = λf.λy.f (f (f y))
M + N = λx.λy.(M x)((N x) y)

Proof:

(+ 2 1) // replacing + w/ encoding
= λx.λy.(2 x)((1 x) y) // replacing 2 w/ encoding
= λx.λy.((λf.λy.f (f y)) x)((1 x) y) // β-reduction: 1^{st} f → x
= λx.λy.(λy.x (x y))((1 x) y) // replacing 1 w/ encoding
= λx.λy.(λy.x (x y))(((λf.λy.f y) x) y) // β-reduction: 1^{st} f → x
= λx.λy.(λy.x (x y))((λy.x y) y) // β-reduction: 3^{rd} y → y
= λx.λy.(λy.x (x y))(x y) // β-reduction: 2^{nd} y → x y
= λx.λy.x (x (x y)) // α-conversion: replace x with f
= λf.λy.f (f (f y)) // apply encoding for 3
= 3

g. (Y fact) 2 = 2 // you do not need to expand any operators except fact & Y

Given:

Y = λf.(λx.f (x x)) (λx.f (x x))
fact = λf. λn.if n = 0 then 1 else n * (f (n-1))

Proof:

(Y fact) 2 // replacing Y w/ encoding
= (λf.(λx.f (x x)) (λx.f (x x)) fact) 2 // β-reduction: 1^{st} f → fact
= (λx.fact (x x)) (λx.fact (x x)) 2 // β-reduction: 1^{st} x → λx.fact (x x)
= (fact ((λx.fact (x x)) (λx.fact (x x)))) 2
// apply encoding for (Y fact)
// ((λx.fact (x x)) (λx.fact (x x))) → (Y fact)
// we know this is the encoding for (Y fact) from 3^{rd} line of proof
= (fact (Y fact)) 2 // apply encoding for fact
= (λf. λn.if n = 0 then 1 else n * (f (n-1)) (Y fact) 2
// β-reduction: 1^{st} f → (Y fact)
= (λn.if n = 0 then 1 else n * ((Y fact) (n-1))) 2// β-reduction: n → 2
= if 2=0 then 1 else 2 * ((Y fact) (2-1)) // apply if
= 2 * ((Y fact) 1) // showed in class (Y fact) 1 = 1
= 2 * 1 // apply *
= 2