CSE216
Programming Abstraction

Instructor: Zhoulai Fu

State University of New York, Korea

Some slides taken from CMU:
https://www.cs.cmu.edu/~venkatg/teaching/15252-sp20/notes/lambda-calculus-slides.pdf

Thanks!

Today

- Lambda calculus syntax and semantics (more details)

Lambda calculus syntax review

- TERM::= Var //Variables

  | lambda Var. TERM // Definition/Abstraction

  | TERM TERM // Application

Var ::= x | y | z …

Exercise: lambda terms?

- λ x. λ y. x y
- λ x. λ y. y
- λ x. λ y. λ z
- (λ x. x) (λ y. y)
- λ x. x λ y

Solution

- T
- T
- F
- T
- F

Implied parentheses

- Application is left associative
  - x y z same as (x y) z
- Definition is right associative
  - λ x. y z same as λ x. (y z)
- λ x. λ y. x y λ z. z y same as ?

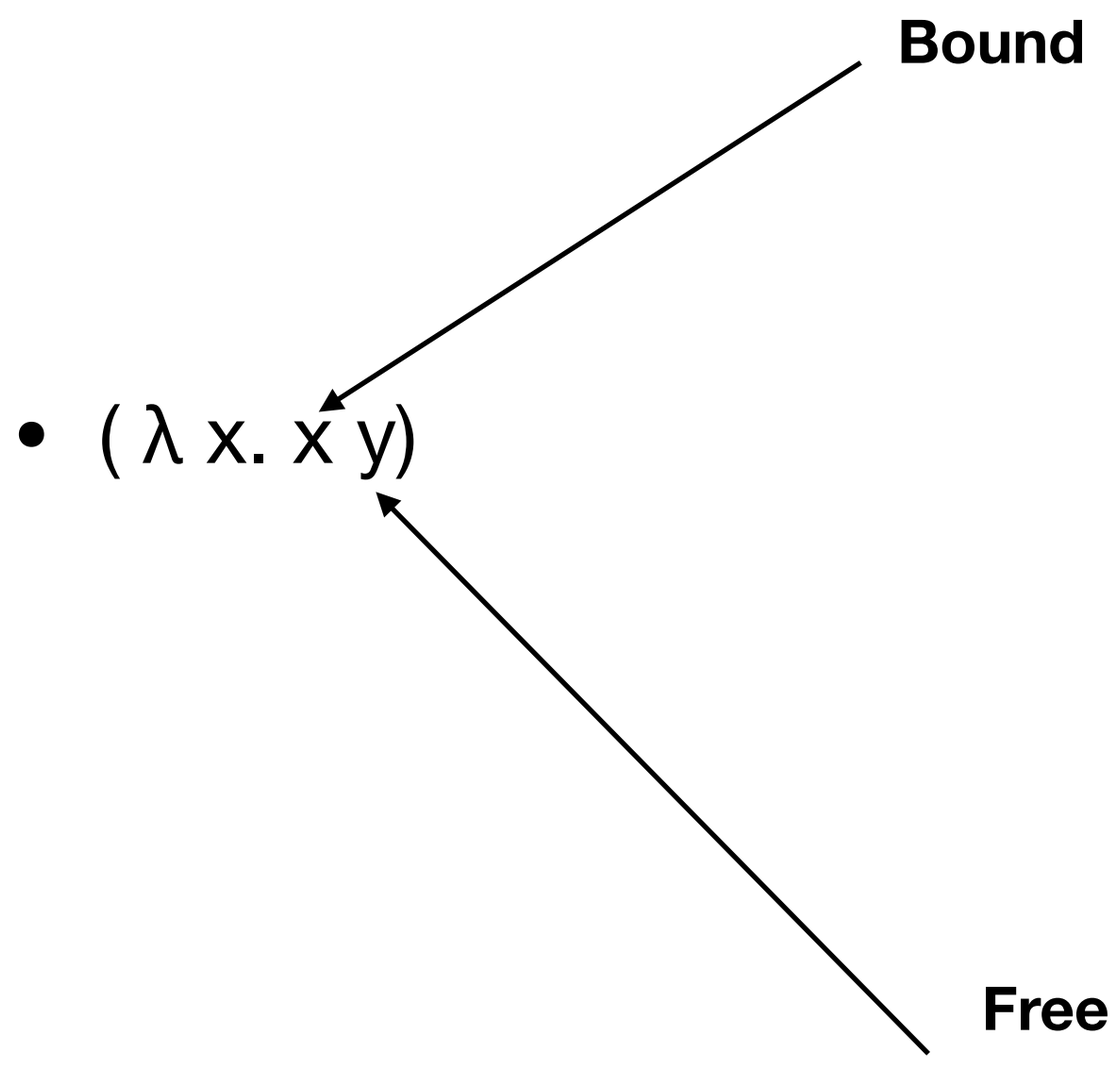
Exercise: Put parentheses

- λ x. λ y. x
- x y λ z. z
- λ x. λ y. x y z

Solution

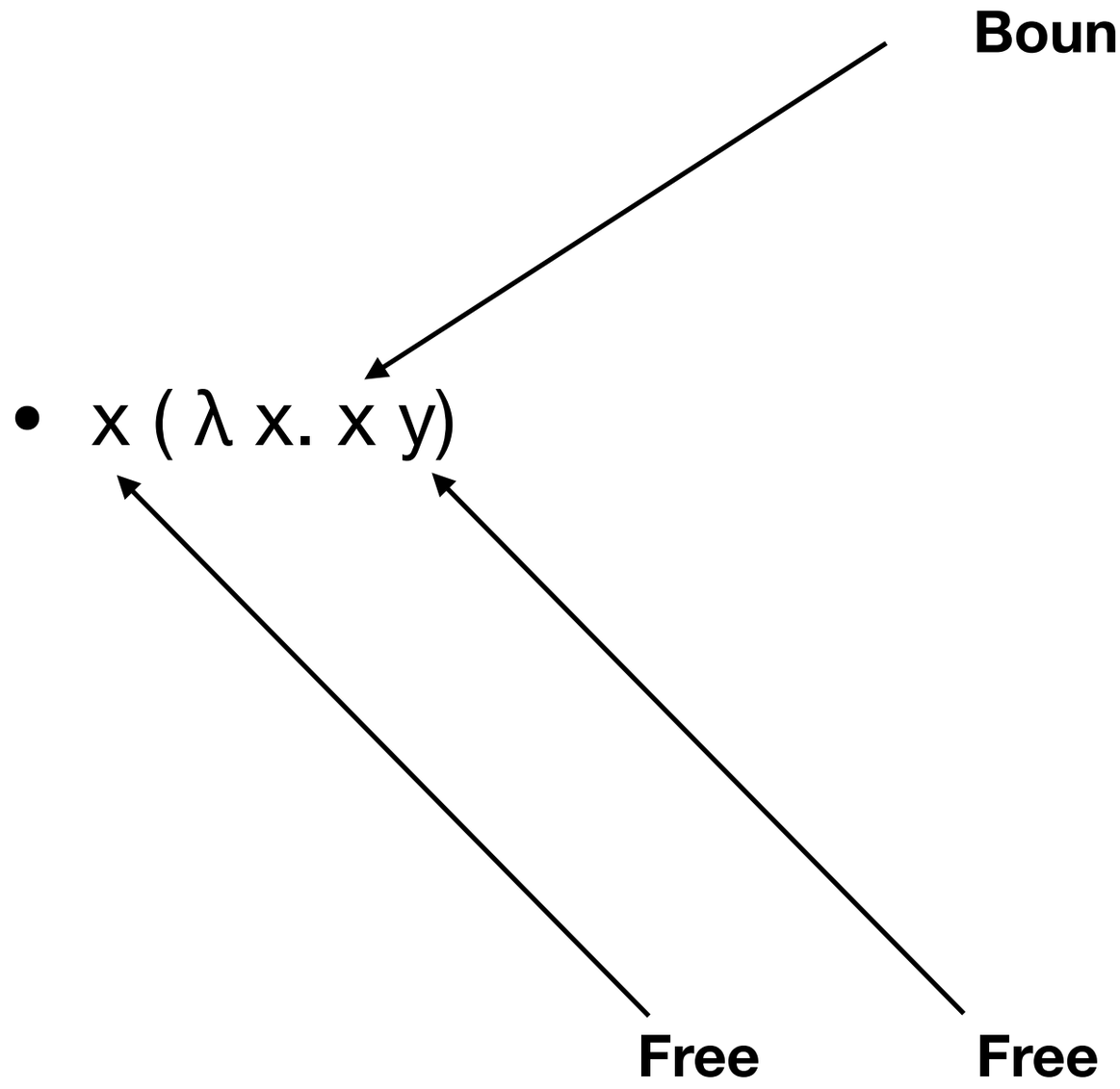
- λ x. (λ y. x)
- (x y) λ z. z
- λ x. (λ y. ((x y) z))

Free and bound variables



- (λ x. x y)

Free and bound variables



Exercise: Find free variables

- λ x. y
- λ x. λ y. x y z
- (λ x.λ y. xy) (λ x. y)
- λ x.λ y. x y λ x. y

Solution

- FV(λ x. y) = { y }
- FV(λ x. λ y. x y z) = { z }
- FV ((λ x.λ y. xy) (λ x. y))= {y} last occurrence of “y”
- FV (λ x.λ y. x y λ x. y) = {} emptyset

β-Reduction (aka λ expression evaluation)

Consider this lambda expression:

$$(\lambda x.\ +\ x\ 1)\ 4$$

β-Reduction (aka λ expression evaluation)

Consider this lambda expression:

$$(\lambda x.\ +\ x\ 1)\ 4$$

This juxtaposition of $(\lambda x.\ +\ x\ 1)$ with 4 means to apply the lambda abstraction $(\lambda x.\ +\ x\ 1)$ to the argument 4. Here's how we do it:

The result of applying a lambda abstraction to an argument is an instance of the body of the lambda abstraction in which bound occurrences of the formal parameter in the body are replaced with copies of the argument.

β-Reduction (aka λ expression evaluation)

Consider this lambda expression:

$$(\lambda x.\ +\ x\ 1)\ 4$$

This juxtaposition of $(\lambda x.\ +\ x\ 1)$ with 4 means to apply the lambda abstraction $(\lambda x.\ +\ x\ 1)$ to the argument 4. Here's how we do it:

The result of applying a lambda abstraction to an argument is an instance of the body of the lambda abstraction in which bound occurrences of the formal parameter in the body are replaced with copies of the argument.

I.e. we put the 4 in for the x in $+\ x\ 1$. The result is $+\ 4\ 1$.

$$(\lambda x.\ +\ x\ 1)\ 4 \rightarrow_{\beta} +\ 4\ 1$$

β-Reduction (some more examples)

$(\lambda x.\ +\ x\ x)\ 5 \rightarrow +\ 5\ 5 \rightarrow 10$

β-Reduction (some more examples)

$(\lambda x.\ +\ x\ x)\ 5 \rightarrow +\ 5\ 5 \rightarrow 10$

$(\lambda x.\ 3)\ 5 \rightarrow 3$

β-Reduction (some more examples)

$(\lambda x.\ +\ x\ x)\ 5 \rightarrow +\ 5\ 5 \rightarrow 10$

$(\lambda x.\ 3)\ 5 \rightarrow 3$

$(\lambda x.\ (\lambda y.\ -\ y\ x))\ 4\ 5 \rightarrow (\lambda y.\ -\ y\ 4)\ 5 \rightarrow -\ 5\ 4 \rightarrow 1$

(Note the currying – we peel off the arguments 4 then 5.)

β-Reduction (some more examples)

$(\lambda x.\ +\ x\ x)\ 5 \rightarrow +\ 5\ 5 \rightarrow 10$

$(\lambda x.\ 3)\ 5 \rightarrow 3$

$(\lambda x.\ (\lambda y.\ -\ y\ x))\ 4\ 5 \rightarrow (\lambda y.\ -\ y\ 4)\ 5 \rightarrow -\ 5\ 4 \rightarrow 1$

(Note the currying – we peel off the arguments 4 then 5.)

$(\lambda f.\ f\ 3)\ (\lambda x.\ +\ x\ 1) \rightarrow (\lambda x.\ +\ x\ 1)\ 3 \rightarrow +\ 3\ 1 \rightarrow 4$

β-Reduction (some more examples)

$(\lambda x.\ +\ x\ x)\ 5 \rightarrow\ +\ 5\ 5 \rightarrow\ 10$

$(\lambda x.\ 3)\ 5 \rightarrow\ \ \ \ 3$

$(\lambda x.\ (\lambda y.\ -\ y\ x))\ 4\ 5 \rightarrow\ (\lambda y.\ -\ y\ 4)\ 5 \rightarrow\ -\ 5\ 4 \rightarrow 1$

(Note the currying – we peel off the arguments 4 then 5.)

$(\lambda f.\ f\ 3)\ (\lambda x.\ +\ x\ 1) \rightarrow\ (\lambda x.\ +\ x\ 1)\ 3 \rightarrow\ +\ 3\ 1 \rightarrow 4$

$(\lambda x.\ (\lambda x.\ +\ (-\ x\ 1))\ \ x\ 3)\ 9 \rightarrow (\lambda x.\ +\ (-\ x\ 1))\ 9\ 3$

$\rightarrow\ +\ (\ -\ 9\ 1)\ 3$

$\rightarrow\ 11$

Exercise: beta reduction

- (λ x. λ y. x y) (λ s. s)
- (λ x. x) (λ y. y)
- (λ x. x x) (λ x. x x)

Solution

- (λ x. λ y. x y) (λ s. s) -> λ y. y

- (λ x. x) (λ y. y) -> λ y. y

- (λ x. x x) (λ x. x x) -> (λ x. x x) (λ x. x x) -> itself -> …
  (always the same)

β-Reduction (contd)

(λx. (λx. + (- x 1)) x 3) 9 → (λx. + (- x 1)) 9 3
→ + (- 9 1) 3
→ 11

In this last example we could have applied the reduction in a different order.

(λx. (λx. + (- x 1)) x 3) 9 → (λx. + (- x 1) 3) 9
→ + (- 9 1) 3
→ 11

β-Reduction (ordering)

So, to evaluate an expression we search for reductions, and make them. When the process stops (there are no more reductions to apply), the expression is said to be in ***normal form***. This is the *value* of the original expression.

Wait… there are different orders in which to do the reductions. Does the order matter? If so, we have a problem.

On the ordering of reductions

Church-Rosser Theorem : No expression can be converted into two distinct normal forms.

Summary

- lambda calculus syntax
- implied parentheses
- bound/free variable
- beta evaluation