

## last time.

- Grammar for  $\lambda$ -calculus
- Disambiguation rules
- ... not quite finished

## Today

- Disambiguation rules (syntax)
- Bound and free variable (semantics)
- Reducible expressions (syntax)
- $\beta$ -reductions

---

## Reminder.

$$\begin{aligned}t &\rightarrow x \\t &\rightarrow \lambda x. t \quad // \text{abstraction} \\t &\rightarrow t \ t \quad // \text{application} \\t &\rightarrow (t)\end{aligned}$$

The grammar is ambiguous.

## Two disambiguation rules.

1. Abstractions extend as far to the right as possible without crossing a right parentheses that is part of a pair of matching parentheses enclosing the  $\lambda x.$  of the abstraction.

of the abstraction.

2. Application is left associative

---

Example  $x \ (x \ \lambda x. \ x \ (\lambda x. \ x \ x) \ x) \ x$

$((a \ b) \ c) \ d$

---

### Parsing General Expressions

1. Identify the bodies of all abstractions
2. If an abstraction does not have parentheses around it, add parentheses
3. Within the body of each abstraction group terms using left associative grouping. Treat any terms within parentheses as one term.
4. Within any pair of parentheses, group terms using left associative grouping
5. Outside all abstractions and parentheses

5. Outside all abstraction and parentheses group terms using left associative grouping.

---

### Example

$$(\lambda x. (x (\lambda x. ((x (\lambda x. x)) (\lambda x. ((x x) (\lambda x. x)))))))$$

$$(((a \ c) \ b) \ d)$$

$$(((\lambda x. x) (\lambda x. x)) (\lambda x. x))$$

$$(\lambda x. x \ (\lambda x. x \ (\lambda x. x)))$$

$$((x (\lambda x. x)) x)$$

$$((x \ x) (\lambda x. x))$$

$$(((x \ ((x \ x) (\lambda x. ((x (\lambda x. x)) x)))) x) x)$$

---

Bound and free variables (semantics)

# Bound and free variables (semantics)

## Aside

### Syntax vs semantics

int x;

int y;

x = ~~z~~;

vs.

int x;

int y;

x = y;

Syntax  
correct, but  
not semantics

## End Aside

**AFTER WE GROUP THE TERM,** we determine bound and free variables

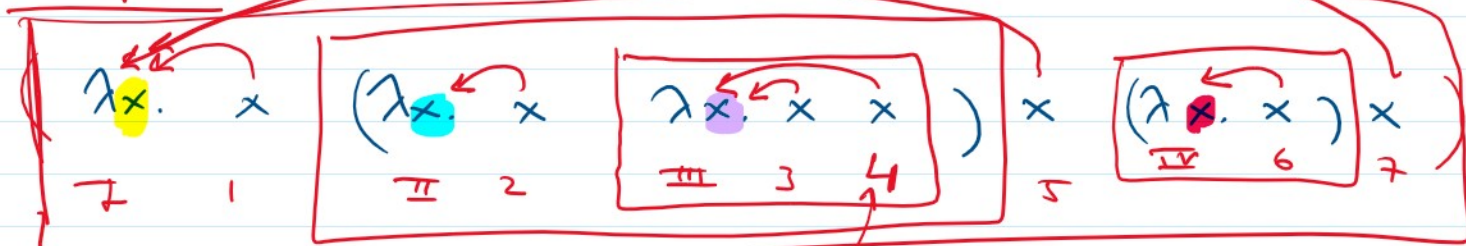
x is bound to  $\lambda x$  if:

!   
 ---   
 same name

- x is in the body of the abstraction on  $\lambda x$ .
- and -  $\lambda x$  is the closest  $\lambda x$  to the left of x in whose body x appears

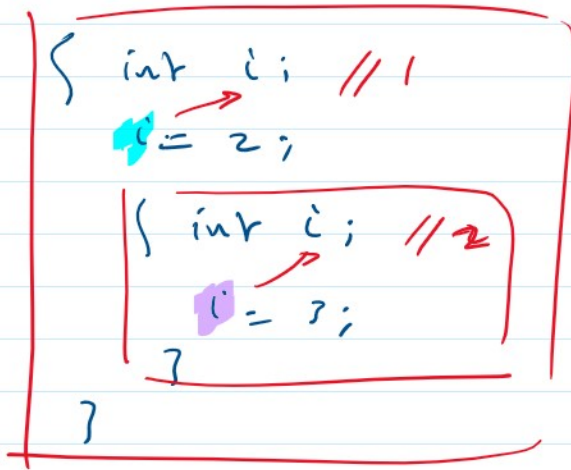
If x is not bound to any  $\lambda x$ , it is free

Example 1.

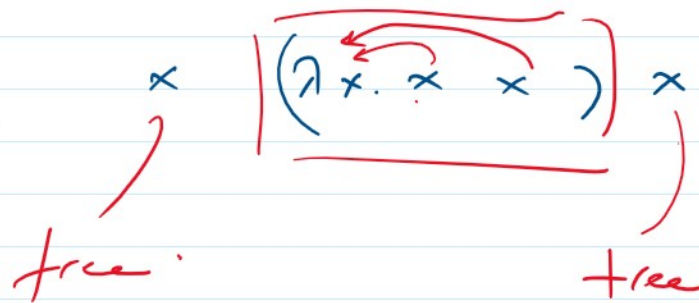




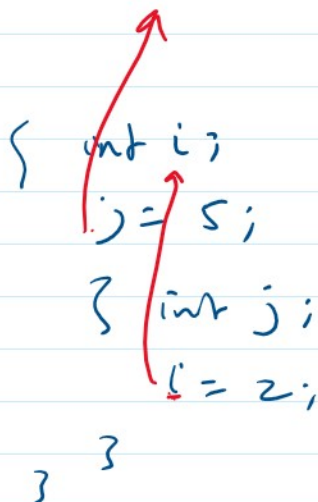
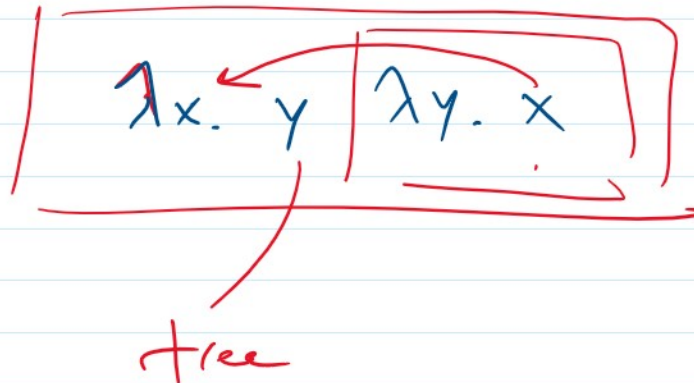
four = 1+1+1+1  
 ↗  
 not seven



Example 2

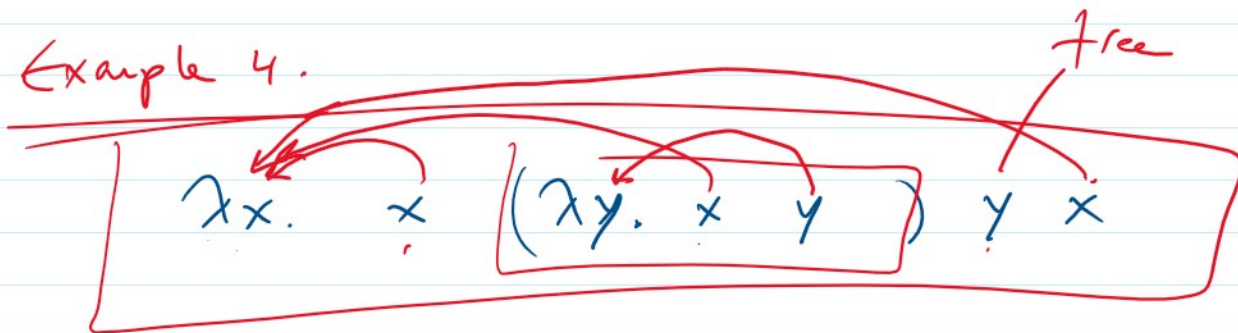


Example 3.

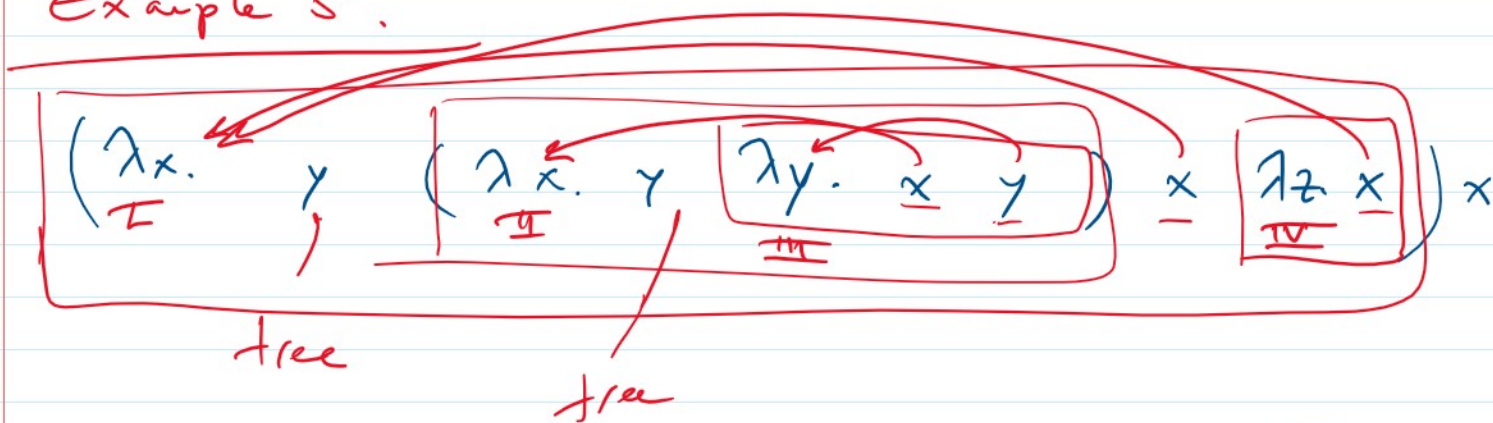




Example 4.



Example 5.



Recap So far

- Grammar (Syntax)
- Disambiguation rules (Syntax)
- bound and free variables (Semantics)

Next. - Reducible Expressions (Syntax)  
 -  $\beta$ -reductions (Semantics)

DEFINITION. A Reducible Expression, also called **redex** is a **term** of the

also called **redex**, is a term of the form  $(\lambda x. t) t'$  where  $t$  and  $t'$  are terms.



### Examples.

1.  $(\lambda x. (x \ x))$  No redex

2.  $((\lambda x. x) x)$  No redex

3.  $(x (\lambda x. (x (\lambda x. x))))$  No redex

$((\lambda x. t) t')$

4.  $((\lambda x. y) z)$  One redex

5.  $((((\lambda x. x) (\lambda x. x)) (\lambda y. y)) (\lambda y. y))$   
⏟  
 Redex

$$(((a \ b) \ c) \ d)$$

$$(((\lambda x. \ x \ x) \ (x \ y)) \ x) \ .$$

$$(\lambda x. \ (x \ x)) \ (\lambda x. \ x \ y \ (\lambda x \ x))$$

$$((\lambda x. \ (t)) \ (t'))$$

$$(((\lambda x. \ x) \ x) \ x)$$

$$(((\lambda x. \ (((\lambda x. \ (x \ x)) \ x) \ x)) \ x) \ (\lambda x. \ x)) \ x)$$

two redexes

$$((\lambda x. \ (x) \ (\lambda x. \ x)) \ (\lambda y. \ y))$$