

Grammar for lambda-calculus expressions:

$\langle \text{LcExpr} \rangle ::= \langle \text{identifier} \rangle \mid$
 $(\text{lambda } (\langle \text{identifier} \rangle) \langle \text{LcExpr} \rangle) \mid$
 $(\langle \text{LcExpr} \rangle \langle \text{LcExpr} \rangle)$

variable use
abstraction
application

CSSE 304 Day 12

1. **Free and bound Examples:** In each of the following expressions, does x occur free and/or occur bound?

a) x

b) t

c) $(x \ t)$

d) $(\text{lambda } (x) (x \ t))$

e) $((\text{lambda } (x) x) x)$

f) $(\text{lambda } (x) (\text{lambda } (t) (t \ x)))$

```
(define occurs-bound?
  (lambda (sym exp)
    (cond
```

Variable x **occurs free** in the LcExp e iff one of the following is true:

F1. e is a variable, and e is the same as x .

F2. e is an abstraction $(\lambda (y) e')$, where y is different from x and x occurs free in e' .

F3. e is an application $(e_1 \ e_2)$, where x occurs free in e_1 or in e_2 .

Variable x **occurs bound** in the LcExp e iff one of the following is true:

B1. e is an abstraction $(\lambda (y) e')$, where x occurs bound in e' , **or** x and y are the same variable and x occurs free in e' .

B2. e is an application $(e_1 \ e_2)$ where x occurs bound in e_1 or in e_2 .

2. The **lexical depth** of a bound occurrence of a variable is the number of levels of nested lambdas and lets between this occurrence and the variable's definition. In

```
(lambda (z)
  (lambda (x)
    (lambda (y) (x y))))
```

the occurrence of y has depth 0 and the occurrence of x has depth 1. There is no occurrence of z .

3. The **lexical address** of a bound occurrence of a variable is a pair $(d \ p)$, where d is that occurrence's lexical depth, and p is the variable's position within its "declaration list". The lexical address of a free variable includes the variable's name and an indication that it is free.

4. Example:

```
In (lambda (x z)
    (lambda (y)
      ((x y) z)))
```

The occurrence of x has depth 1 and position 0.
 The occurrence of y has depth 0 and position 0.
 The occurrence of z has depth 1 and position 1.

5. Example of output from the lexical-address procedure that you will write:

<pre>(lexical-address '(lambda (a b c) (if (eq? b c) ((lambda (c) (cons a c)) a) b)))</pre>	→	<pre>(lambda (a b c) (if ((: free eq?) (: 0 1) (: 0 2)) ((lambda (c) ((: free cons) (: 1 0) (: 0 0))) (: 0 0)) (: 0 1)))</pre>
---	---	--

6. Lexical address exercises (also see example that comes before these in the slides).

```
(lexical-address
'((lambda (x y)
  (((lambda (z)
    (lambda (w y)
      (+ x z w y)))
    (list w x y z))
   (+ x y z)))
 (y z)))
```

```
(lexical-address
'(let ([a 3] [b 4])
  (let ([a (+ b 2)] [c a])
    (+ a b c))))
```

```
> (reverse '())
()
> (reverse '(a b c))
(c b a)
> (let* ([x '(a b c)]
         [y (cdr x)]
         [z (reverse x)])
  (list x z (eq? y (cdr z))))
((a b c) (c b a) #f)
```

```
> (reverse! '())
()
> (reverse! '(a b c))
(c b a)
> (define L '(a b c d))
> (reverse! L)
(d c b a)
> L
(a)
> (let* ([x '(a b c)]
         [y (cdr x)]
         [z (reverse! x)])
  (list x z (eq? y (cdr z))))
((a) (c b a) #t)
```