CS3L:
Introduction to Symbolic Programming

Lecture 20:
Tree Recursion

Summer 2008

Colleen Lewis
colleenL@berkeley.edu

Today

- Lists
- Lists vs sentences
- **`List`**
- **`Append`**
- **`Cons`**
- Trial by error coding = bad

Ice cream flavors

- I want to list ice cream flavors
- (chocolate mint chip mocha almond fudge)
- (chocolate (mint chip) (mocha almond fudge))
- This is a list! Here is how it was created:

(**`list`** ‘chocolate ‘(mint chip) ‘(mocha almond fudge))

`lists vs sentences`

- Sentences can be made of
  - words/numbers
- Lists can have be made of
  - words/numbers
  - sentences
  - #f or #t
  - procedures
  - other lists!!!

sentence

- Examples
  - **`(se ‘cat ‘dog)`** ➔ **`‘(cat dog)`**
  - **`(se ‘(cat) ‘(dog))`** ➔ **`‘(cat dog)`**
  - **`(se ‘cat ‘(dog))`** ➔ **`‘(cat dog)`**
  - **`(se ‘(cat) ‘dog)`** ➔ **`‘(cat dog)`**
  - **`(se ‘cat ‘())`** ➔ **`‘(cat)`**
  - **`(se ‘() ‘dog)`** ➔ **`‘(dog)`**

accessors

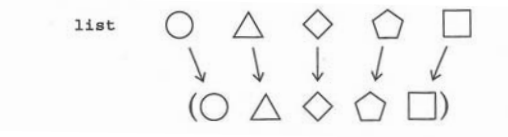
sentence & word stuff	list stuff
first	car
butfirst	cdr
last	☹
Butlast	☹

other procedures

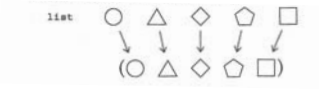
sentence & word stuff	list stuff
empty?	null?
sentence?	list?
item	list-ref
sentence	list

list

- Takes any number of arguments and puts them in a list



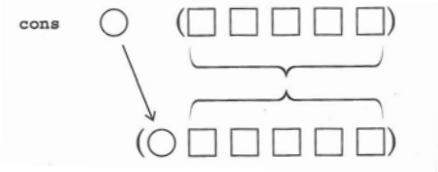
list



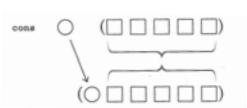
- Examples
  - `(list 'cat 'dog)` → `'(cat dog)`
  - `(list '(cat) '(dog))` → `'((cat) (dog))`
  - `(list '(cat) 'dog)` → `'((cat) dog)`
  - `(list 'cat '(dog))` → `'(cat (dog))`
  - `(list 'cat '())` → `'(cat ())`
  - `(list '() 'dog)` → `'(() dog)`
  - `(list '(cat ()) 'dog)` → `'((cat ()) dog)`

cons

- Takes two arguments
- Makes the first arg the car of the new list
- Makes the second arg the cdr of the new list
- The second argument MUST be a list



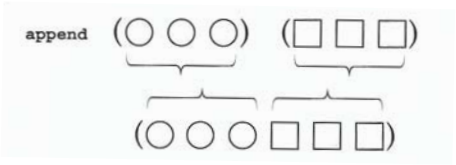
cons



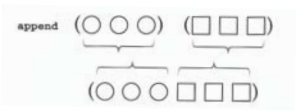
- Examples
  - `(cons 'cat '( dog ))` → `'(cat dog)`
  - `(cons '(cat) '( dog ))` → `'((cat) dog)`
  - `(cons 'cat '())` → `'(cat)`
  - `(cons '() '( () dog ))` → `'(() () dog)`
  - `(cons '(cat) 'dog` → `'((cat) . dog)`

append

- Takes two lists and turns them into one
- Both arguments MUST be lists



append



- Examples
  - `(append '(cat) '(dog)) ➔ '(cat dog)`
  - `(append '(cat) '()) ➔ '(cat)`
  - `(append '() '(dog)) ➔ '(dog)`
  - `(append '(cat) '(())) ➔ '(cat ())`
  - `(append '(()()) '(dog)) ➔ '(() () dog)`

Trial by error coding

Trial by error coding

```
(define (pluralize L)
  (if (null? L)
      '()
      (cons
              (plural (car L))
              (pluralize (cdr L)))))
```