Lecture 06
Map Filter Reduce

T. METIN SEZGIN

1

Announcements

1. New etutor assignment coming (Oct 19th)
2. Reading SICP 1.2 (pages 79-126)

2

Lecture 05
Lists and recursion – Review

T. METIN SEZGIN

3

Lecture Nuggets

- Two main patterns when dealing with lists
  - Consing up – to build
  - Cdring down – to process
- Higher order procedures

4

Nugget

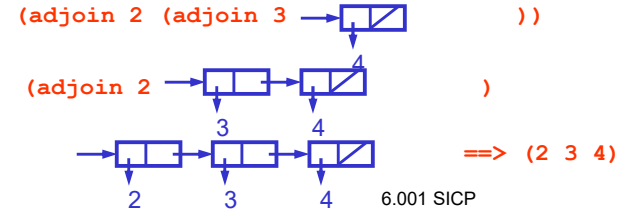
Two patterns for dealing with lists

5

Common Pattern #1: cons'ing up a list

```
(define (enumerate-interval from to)
  (if (> from to)
      nil
      (adjoin from
              (enumerate-interval
                (+ 1 from)
                to))))
```

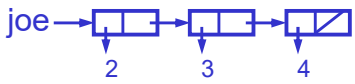
```
(e-i 2 4)
(if (> 2 4) nil (adjoin 2 (e-i (+ 1 2) 4)))
(if #f nil (adjoin 2 (e-i 3 4)))
(adjoin 2 (e-i 3 4))
(adjoin 2 (adjoin 3 (e-i 4 4)))
(adjoin 2 (adjoin 3 (adjoin 4 (e-i 5 4))))
(adjoin 2 (adjoin 3 (adjoin 4 nil)))
```



6

Common Pattern #2: cdr'ing down a list

```
(define (list-ref lst n)
  (if (= n 0)
      (first lst)
      (list-ref (rest lst)
                (- n 1))))
```




```
(list-ref joe 1)
```

```
(define (length lst)
  (if (null? lst)
      0
      (+ 1 (length (rest lst)))))
```


7

Nugget

Higher order procedures

8

Other common patterns

- $1 + 2 + \ldots + 100 = (100 * 101)/2$ ← $\sum_{k=1}^{100} k$
- $1 + 4 + 9 + \ldots + 100^2 = (100 * 101 * 201)/6$ ← $\sum_{k=1}^{100} k^2$
- $1 + 1/3^2 + 1/5^2 + \ldots + 1/101^2 = \pi^2/8$ ← $\sum_{k=1,odd}^{101} k^{-2}$

```
(define (sum-integers a b)
  (if (> a b)
      0
      (+ a (sum-integers (+ 1 a) b))))
(define (sum-squares a b)
  (if (> a b)
      0
      (+ (square a)
         (sum-squares (+ 1 a) b))))
(define (pi-sum a b)
  (if (> a b)
      0
      (+ (/ 1 (square a))
         (pi-sum (+ a 2) b))))
```

```
(define (sum term a next b)
  (if (> a b)
      0
      (+ (term a)
         (sum term (next a) next b))))
```


9

Let's check this new procedure out!

```
(define (sum term a next b)
  (if (> a b)
      0
      (+ (term a)
         (sum term (next a) next b))))
```

A higher order procedure!!

What is the type of this procedure?

(number → number, number, number→ number, number) → number

procedure

procedure

procedure

10

Higher order procedures

- A higher order procedure:
  takes a procedure as an argument or returns one as a value

```
(define (sum-integers1 a b)
        (sum (lambda (x) x) a (lambda (x) (+ x 1)) b))
(define (sum-squares1 a b)
        (sum square a (lambda (x) (+ x 1)) b))
(define (pi-sum1 a b)
     (sum (lambda (x) (/ 1 (square x))) a
          (lambda (x) (+ x 2)) b))
```

```
(define (sum term a next b)
 (if (> a b)
    0
    (+ (term a)
      (sum term (next a) next b))))
```


11

Lecture 06
Map Filter Reduce

T. METIN SEZGIN

12

Lecture Nuggets

- Three new patterns used when dealing with lists
  - Map
  - Filter
  - Reduce
- Map applies a function to all elements of a list
- Filter selects elements satisfying a condition
- Reduce computes a single result while operating on the list

13

Nugget

Map applies a function to all elements of a list

14

Common Pattern #1: Transforming a List

```
(define (square-list lst)
  (if (null? lst)
      nil
      (cons (square (car lst))
            (square-list (cdr lst)))))

(define (double-list lst)
  (if (null? lst)
      nil
      (cons (* 2 (car lst))
            (double-list (cdr lst)))))

(define (MAP proc lst)
  (if (null? lst)
      nil
      (cons (proc (car lst))
            (map proc (cdr lst)))))

(define (square-list lst)
  (map square lst))

(define (double-list lst)
  (map (lambda (x) (* 2 x))
lst))
```


15

Nugget

Filter selects elements satisfying a condition

16

Common Pattern #2: Filtering a List

```
(define (keep-it-odd lst)
 (cond ((null? lst) nil)
       ((odd? (car lst))
        (cons (car lst) (keep-it-odd (cdr lst))))
       (else (keep-it-odd (cdr lst)))))
```

Filter selects elements satisfying a condition

Reduce computes a single result while operating on the list

```
(define (filter pred lst)
  (cond ((null? lst) nil)
        ((pred (car lst))
         (cons (car lst)
               (filter pred (cdr lst))))
        (else (filter pred (cdr lst)))))
```


17

Nugget

Reduce computes a single result while operating on the list

18

Common Pattern #3: Accumulating Results

```
(define (add-up lst)
  (if (null? lst)
      0
      (+ (car lst)
         (add-up (cdr lst)))))

(define (mult-all lst)
  (if (null? lst)
      1
      (* (car lst)
         (mult-all (cdr lst)))))

(define (REDUCE op init lst)
  (if (null? lst)
      init
      (op (car lst)
          (reduce op init (cdr lst)))))

(define (add-up lst)
  (reduce + 0 lst))
```