

Section Notes 8

CS51—Spring 2009

Week of March 2, 2009

Outline

1. Box and Pointer Diagrams
2. OO Practice

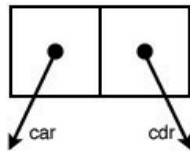
1 Goals for today:

At the end of today's section, you should be able to do the following:

1. Understand mcons, set-mcar! and set-mcdr!.
2. Draw a box and pointer diagram for lists.
3. Be able to understand and implement basic objects with inheritance

2 Box and Pointers

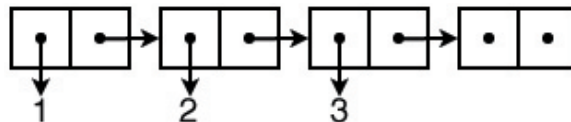
A *box and pointer diagram* is a visual representation of a cons cell.



a single empty cons cell as a box and pointer diagram

2.1 Representing Structures

Lists can be represented like this:



(list 1 2 3) as a box and pointer diagram

Draw the following lists with box and pointer diagrams:

```
(list (list "hi" "hello") "good-evening")
```

```
(list (list "cars" "trucks" "buses") (cons empty (list "traffic" "light")))
```

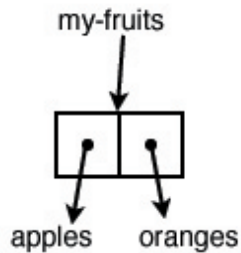
```
(list (list (list (list empty) empty empty) empty))
```

2.2 Mutable lists in PLT Scheme

In the dialect of Scheme that we are using, normal `cons` is not mutable. However, we are provided with `mcons`, `mcar`, and `mcdr`, which are mutable.

How does `mcons` work?

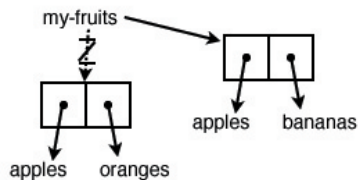
```
(define my-fruits (mcons 'apples 'oranges))
```



Say we want to change `my-fruits` to a pair of `'apples` and `'bananas`. How might we do this?

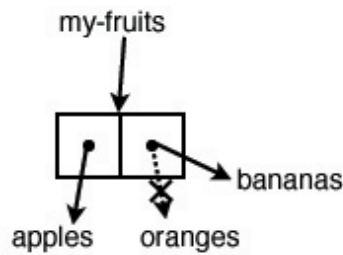
We could `set!` the whole thing:

```
(set! my-fruits (mcons 'apples 'bananas))
```



or we could `set-mcdr!`:

```
(set-mcdr! my-fruits 'bananas)
```



2.3 Sample exam question

Fill in the ... below so that the expression evaluates to 42

```
(define f ...)
```

```
(if (= (f) 42) 0 (f))
```

3 OO practice

We are going to practice some basic OO programming by setting up some objects for a very basic game. In this game there are blobs that wander around and eat each other.

Let's write a basic blob class.

We want our blob to have a `size`, and a boolean value `alive`, a public accessor function for the size `getSize`, and a method `eat` which takes another blob, and if that blob is smaller than this blob, we want to kill the other blob, which will require a `kill` method:

```
(define blob%  
  (class object%
```

Recall that one of the nice things about OO programming is that we can inherit from classes to make other classes.

We can also make a blob that lies about its size. When another blob tries to eat this blob, it will look like it is twice its actual size:

```

(define scary-blob%
  (class blob% ; this shows that we are inheriting from blob%
    (super-new)
    (define/override (getSize) (* 2 (super getSize)))
;critically, this cannot be (* 2 (send this getSize)) why not?
  )
)

```

Let's write a new blob, `growing-blob`. When a `growing-blob` eats another blob, adds the size of that blob to itself:

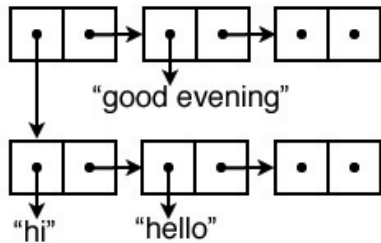
```

(define growing-blob%
  (class blob%

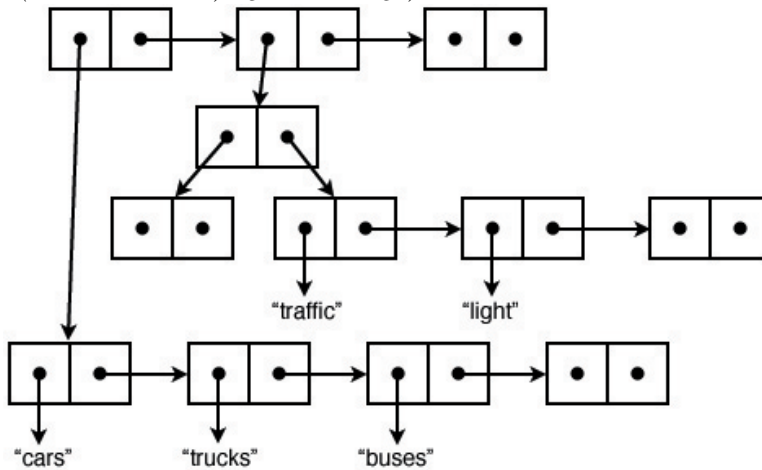
```

4 Solutions

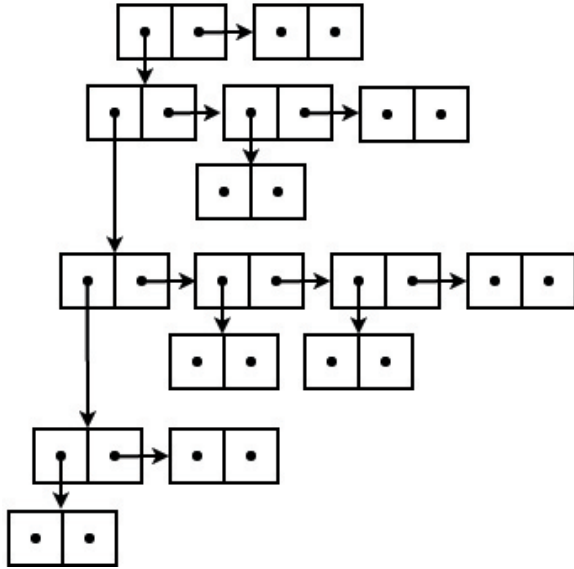
4.1 boxes and pointers



(list (list "hi" "hello") "good-evening")



```
(list (list "cars" "trucks" "buses") (cons empty (list "traffic" "light")))
```



```
(list (list (list (list empty) empty empty) empty))
```

4.2 one possible way to code blobs

```
(define blob%
  (class object%
    (init size)
    (define alive #t)
    (define blob-size size)
    (super-new)
    (define/public (alive?) alive)
    (define/public (kill) (set! alive #f))
    (define/public (getSize) blob-size)

    ; yes I cheated and added this to the underlying blob
    (define/public (setSize s) (set! blob-size s))
    (define/public (eat other-blob)
      (if (> blob-size (send other-blob getSize))
          (begin (display "omm nom nom!\n")
                 (send other-blob kill)
                 'success)
          (begin (display "too big\n")
                 'fail)))
  )
)
```

```
(define growing-blob%
  (class blob%
    (super-new)
    (define/override (eat other-blob)
      (let ([x (send other-blob getSize)])
        (if (eq? 'success (super eat other-blob))
            (begin (display "growth +")
                   (display x) (display "\n")
                   (send this setSize (+ x (send this getSize))))
            (void))))
  )
)
```