Mutable Lists

Dr. Mattox Beckman

Illinois Institute of Technology
Department of Computer Science

Objectives

- Create and update mutable storage using deftype.
- Describe the syntax for creating and modifying mutable lists.
- Implement and diagram mutable insertion and deletion.

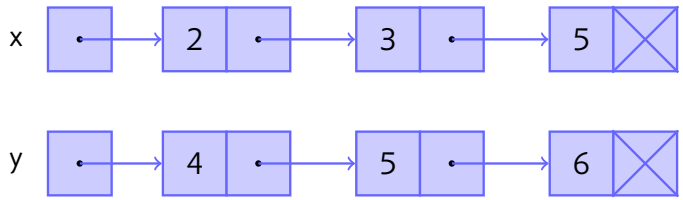
Mutable Data in Clojure

```clojure
(def x (atom 0))
x
;; => #<Atom@5c25d8aa: 0>
@x
;; => 0
(swap! x inc)
;; => 1
@x
;; => 1
(reset! x 20)
;; => 20
```


Creating a Mutable List

```clojure
(defprotocol MConsP
  (car [this])
  (cdr [this])
  (set-car! [this v])
  (set-cdr! [this v]))
(deftype MCons
    [^{:unsynchronized-mutable true} the-car
     ^{:unsynchronized-mutable true} the-cdr]
  MConsP
  (car [this] the-car)
  (cdr [this] the-cdr)
  (set-car! [this v] (set! the-car v))
  (set-cdr! [this v] (set! the-cdr v)))
```

```clojure
(defn mcons [a b] (MCons. a b))
(defn mlist [& xx]
   (if (empty? xx) nil
       (mcons (first xx) (apply mlist (rest xx)))))
(def x (mlist 2 3 5))
(def y (mlist 4 5 6))
```

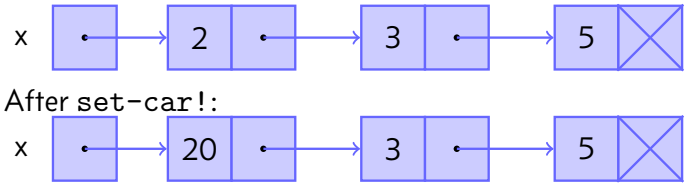


Accessing a Mutable List

```clojure
(def x (mlist 2 3 5))
(car x)
;; => 2
(-> x cdr car)
;; => 3
(-> x cdr cdr car)
;; => 5
(-> x cdr cdr cdr)
nil
```

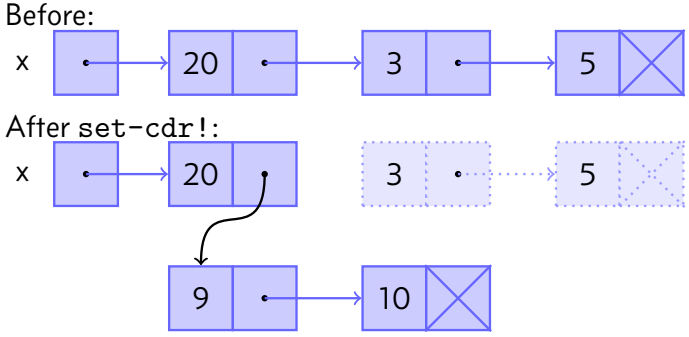

Mutation with set-car!

```clojure
(def x (mlist 2 3 5))
(set-car! x 20)
;; => 20
x
;; => #<MCons linked_list_lab.t_core.MCons@5f6af31>
```



Mutation with set-cdr!

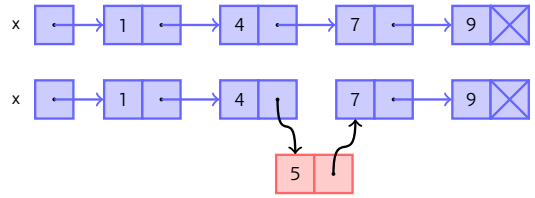
```clojure
(set-cdr! x (mcons 9 (mcons 10 nil )))
```



Insertion

- To insert an item, you first need to find the insertion point.
- Then you need to create a new mcons cell.

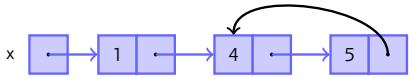
```
(defn insert [xx elt]
   (cond (nil? xx) (mcons elt nil)
         (< elt (-> xx cdr car))
               (set-cdr! xx (mcons elt (cdr xx)))
         :else (insert (cdr xx) elt))))
(def x (mlist 1 4 7 9))
(insert x 5)
```



Bug Alert

- Here's an example of a bug and what it looks like.
- Note the `set-cdr!` uses `xx` instead of `(cdr xx)`.

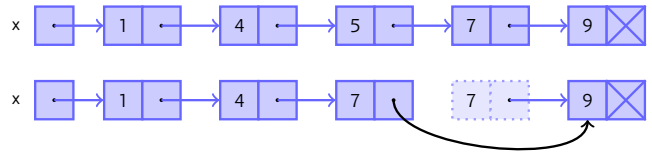
```
(defn insert [xx elt]
   (cond (nil? xx) (mcons elt nil)
         (< elt (-> xx cdr car))
               (set-cdr! xx (mcons elt xx))
         :else (insert (cdr xx) elt)))
(def x (mlist 1 4 7 9))
(insert x 5)
StackOverflowError   java.util.regex.Pattern
```



Deletion by Copying

- There are two ways to delete data.
- Version 1: Clobber the old data!
- Can you see the case where this will not work?

```
(defn delete [xx victim]
   (cond (nil? xx)  nil
         (= (car xx) victim)
            (do (set-car! xx (car (cdr xx)))
                (set-cdr! xx (cdr (cdr xx)))))
         :else (delete (cdr xx) victim))
(def xx (mlist 1 4 5 7 9))
(delete xx 5)
```



Delete by Relinking

- Version 2: Route around the old data!
- Can you see the case where this will not work?

```
(defn delete [xx victim]
   (cond (nil? xx) nil
         (= (car (cdr xx)) victim)
            (set-cdr! xx (cdr (cdr xx)))
         :else (delete (cdr xx) victim)))
(def xx (mlist 1 4 5 7 9))
(delete xx 5)
```