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.

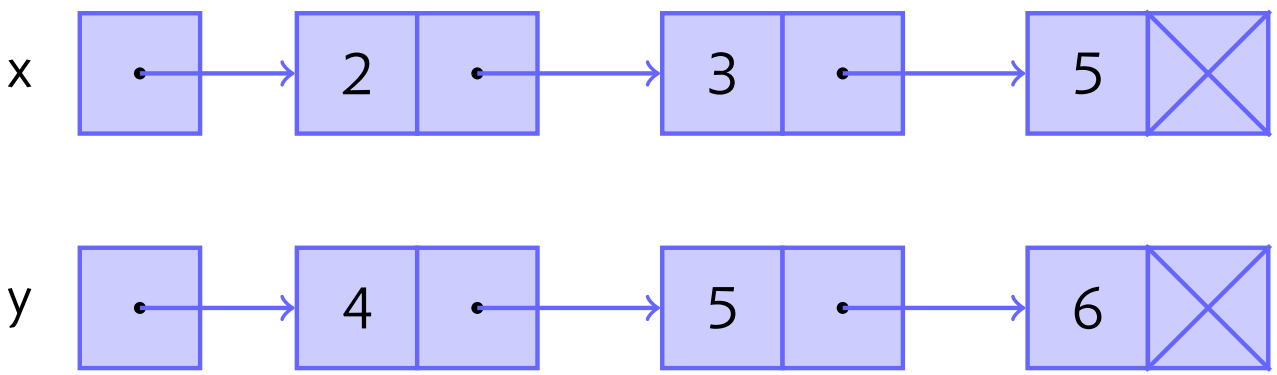
Mutable Data in Clojure

```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

```
1  (defprotocol MConsP
2     (car [this])
3     (cdr [this])
4     (set-car! [this v])
5     (set-cdr! [this v]))
6  (deftype MCons
7        [^{:unsynchronized-mutable true} the-car
8         ^{:unsynchronized-mutable true} the-cdr]
9     MConsP
10    (car [this] the-car)
11    (cdr [this] the-cdr)
12    (set-car! [this v] (set! the-car v))
13    (set-cdr! [this v] (set! the-cdr v)))
```

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

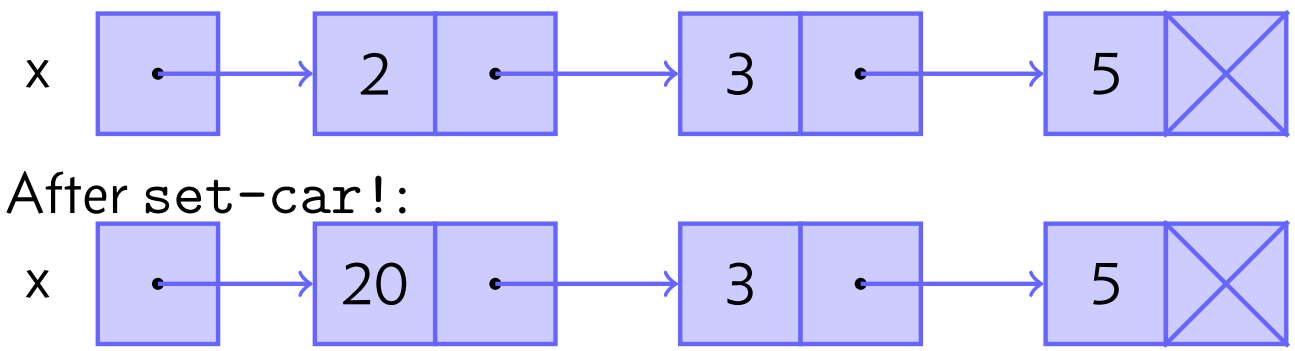


Accessing a Mutable List

```
1 (def x (mlist 2 3 5))
2 (car x)
3 ;; => 2
4 (-> x cdr car)
5 ;; => 3
6 (-> x cdr cdr car)
7 ;; => 5
8 (-> x cdr cdr cdr)
9 nil
```

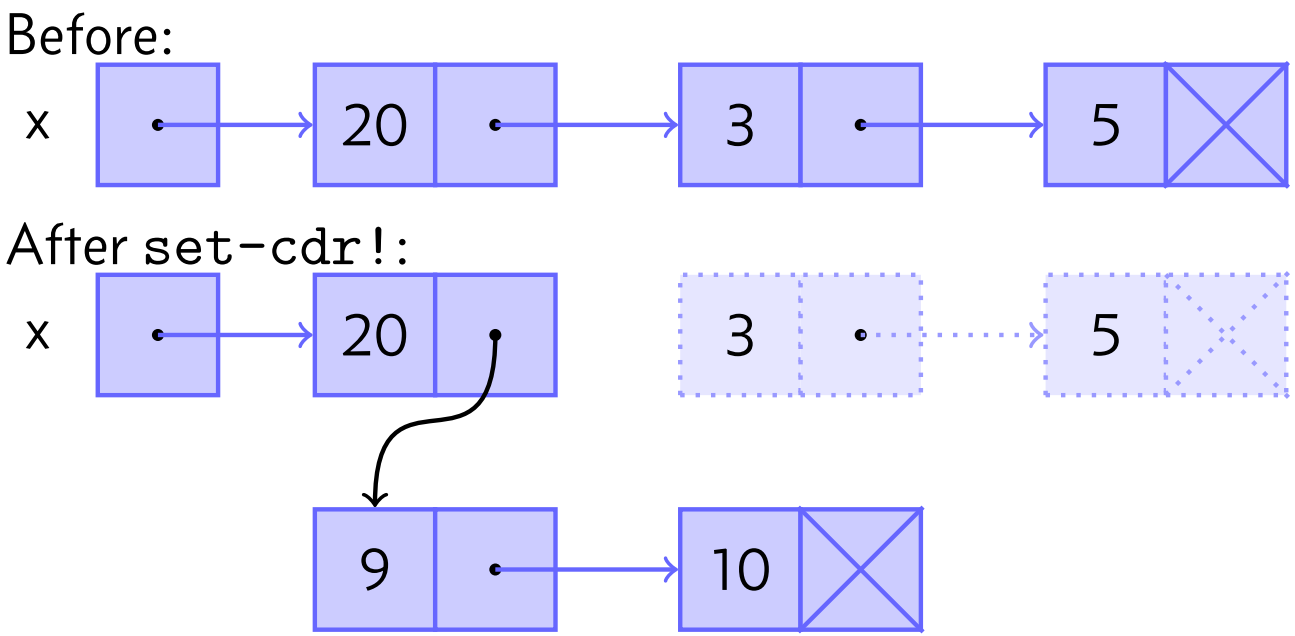
Mutation with set-car!

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



Mutation with set-cdr!

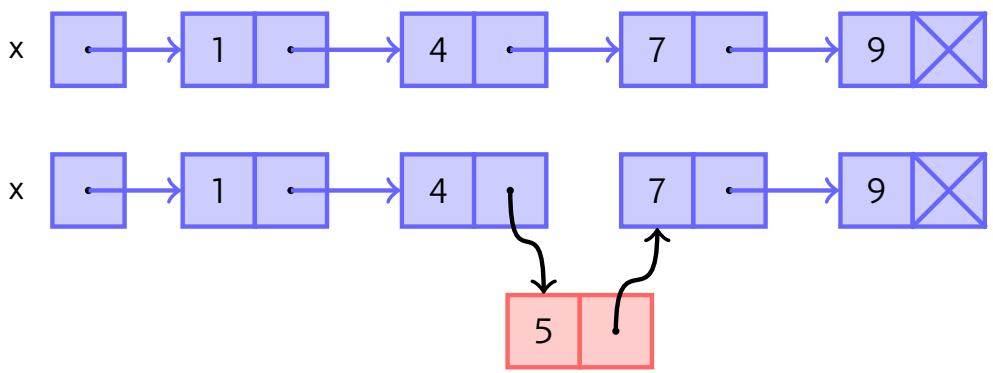
```
1 (set-cdr! x (mcons 9 (mcons 10 nil )))
```



Insertion

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

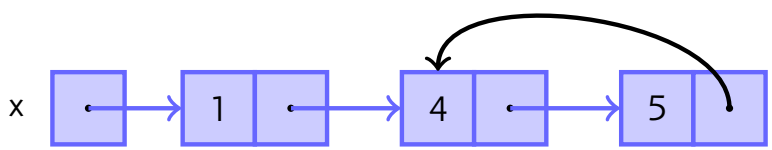
```
1 (defn insert [xx elt]
2    (cond (nil? xx) (mcons elt nil)
3          (< elt (-> xx cdr car))
4                (set-cdr! xx (mcons elt (cdr xx)))
5          :else (insert (cdr xx) elt))))
6 (def x (mlist 1 4 7 9))
7 (insert x 5)
```



Bug Alert

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

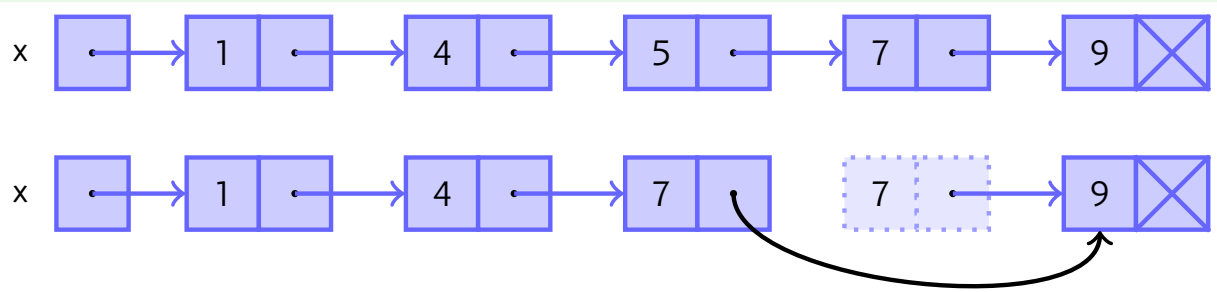
```
1 (defn insert [xx elt]
2    (cond (nil? xx) (mcons elt nil)
3          (< elt (-> xx cdr car))
4                (set-cdr! xx (mcons elt xx))
5          :else (insert (cdr xx) elt)))
6 (def x (mlist 1 4 7 9))
7 (insert x 5)
8 StackOverflowError   java.util.regex.Pattern
```



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?

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



Delete by Relinking

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

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