Scheme Operations

- Scheme provides a number of pre-defined operations on atoms and lists
- The core set is tiny:

  car

  cdr

  cons

  eq?

  atom?

  null?

car

- *car* is an operation on lists:

  car of an atom is undefined

  car of `()` is undefined

  car of `(`x_1 x_2 ... x_n`)` is x_1
- *car* tells you what the first element of list is
- *car* of `(a b c)` is `a`
- *car* of `(a (b c))` is `a`
- *car* of `((a b) c)` is `(a b)`
- *car* of `((a b c))` is `(a b c)`

cdr

- *cdr* is an operation on lists:

  cdr of an atom is undefined

  cdr of `()` is undefined

  cdr of `(`x_1 x_2 ... x_n`)` is `(`x_2 ... x_n`)`
- *cdr* tells you what a list would be without its first element
- *cdr* of `(a b c)` is `(b c)`
- *cdr* of `(a (b c))` is `((b c))`
- *cdr* of `((a b) c)` is `(c)`
- *cdr* of `((a b c))` is `()`

cons

- *cons* is an operation on lists and atoms
- *cons* is defined for any two values, but we'll consider only cases where the second value is a list:

  cons of x with $()$ is (x)

  cons of x_1 with $(x_2 \ \ldots \ x_n)$ is

  $(x_1 \ x_2 \ \ldots \ x_n)$
- *cons* tells you what a list would be with an additional element at the front

cons (cont.)

- *cons* of `a` with `(b c)` is `(a b c)`
- *cons* of `a` with `((b c))` is `(a (b c))`
- *cons* of `(a b)` with `(c)` is `((a b) c)`
- *cons* of `(a b c)` with `()` is `((a b c))`
- *cons* of `a` with `b` is of no interest (at least, not right now)

eq?

- *eq?* is an operation on lists and atoms
- *eq?* is defined for any two values, but we'll consider only cases where <u>at least one value is a symbol</u>:

  eq? of *x* with *y* is `#t`

  if *x* and *y* are the same symbol

  eq? of *x* with *y* is `#f`

  otherwise
- *eq?* tells you if two symbols are equal

eq? (cont.)

- *eq?* of `a` with `a` is `#t`
- *eq?* of `a` with `b` is `#f`
- *eq?* of `(a b)` with `(a b)` is of no interest (at least, not right now)

atom?

- *atom?* is an operation on lists and atoms:

  atom? of *x* is `#t` if *x* is an atom

  atom? of *x* is `#f` if *x* is a list

- *atom?* tells you if a value is an atom or not
- *atom?* of `a` is `#t`
- *atom?* of `()` is `#f`
- *atom?* of `(a b)` is `#f`

null?

- *null?* is an operation on lists and atoms:

  null? of *x* is `#t` if *x* is `()`

  null? of *x* is `#f` otherwise
- *null?* tells you if a value is `()` or not
- *null?* of `a` is `#f`
- *null?* of `()` is `#t`
- *null?* of `(a b)` is `#f`

Terminology

- *car* and *cdr* are referred to as *selectors*, since they give you a piece of an existing thing
- *cons* is referred to as a *constructor*, since it builds a new thing
- *eq?*, *atom?*, and *null?* are referred to as *predicates*, since they always return a true/false result

Behavior

- Assume that *car*, *cdr*, *cons*, *eq?*, *atom?*, and *null?* take O(1) time
- Assume that *car*, *cdr*, *cons*, *eq?*, *atom?*, and *null?* take O(1) space

Behavior (cont.)

- Note that *car*, *cdr*, *cons*, *eq?*, *atom?*, and *null?* are not *mutators* – they do not modify their arguments in any way:
  - *cdr* does not delete the first element from a list: it simply reports what a list without that first element would look like
  - *cons* does not insert a new first element into a list: it simply reports what a list with a new first element would look like
- Remember this!