CSCI 334: Principles of Programming Languages

Lecture 2: Lisp

Instructor: Dan Barowy
Williams

How this course works

• Lots of new languages
• Not enough class time to cover all features (e.g., Java over the course of 134-136: 1 year!)
• Read the assigned book chapters, read course tutorials, and even seek additional material on your own (on the Googles)
• TAs/office hours

John McCarthy

IBM 704
Lisp was invented for AI research

704 Assembly (circa 1954)
(From Coding the MIT-IBM 704 Computer)

FORTRAN (circa 1956)
(From NASA Technical Note D-1737)

LISP (circa 1958)

(defun fact (n)
  (cond ((eq n 0) 1)
        (t (* n (fact (- n 1))))))

LISP is a “functional” language

- programs are modeled after math. functions
- no statements, only expressions
- no “mutable” variables, only declarations
- therefore, the effect of running a program (“evaluation”) is purely the effect of applying a function to an input.

LISP is a “functional” language

(defun add-one (n) n + 1)

Big functions are “composed” of little functions

(defun cleaning-robot (dirt) ...)

(dirty house)

(clean house)
Program correctness is easier to achieve

I.e., whole is correct if pieces are correct.

LISP is inspired by the lambda calculus

- everything either a value or a function of a value
  value: 1
  function of two values: (+ 1 1)
- syntax is (mind-numbingly) regular
  functions: (function-name arguments ...)
  values: anything that is not a function
- evaluating a function produces a value:
  (+ 1 1) = 2

Statements vs. expressions

- A **statement** is an operation that changes the
  state of the computer
  Java: i++
  value stored at location i incremented by one
- An **expression** is a combination of functions and
  values that yields a new value
  Lisp: (+ i 1)
  evaluating + with i and 1 returns i + 1

**REPL**
(read-eval-print loop)
Batch mode

Mutable variables

- If you can update a variable in a language, you have mutable variables
  - Java:
    ```java
    int i = 3;
    i = 4;
    ```
- Notice that both lines of code are statements
- (pure) Lisp does not have mutable variables

Immutable variables

- Variables cannot be updated in Lisp
  - Lisp:
    ```lisp
    (defun my-func (i) ...)
    (my-func 3)
    ```
  - or the shorter
    ```lisp
    ((lambda (i) ...) 3)
    ```
- Notice that all of the above are expressions
- In fact, functions are the only way to bind values to names in (pure) Lisp

Lisp syntax: atoms

- An atom is the most basic unit in Lisp: data
  ```lisp
  4
  112.75
  "hello"
  'foo
  t
  nil
  ```
Lisp syntax: cons cells

• The “cons cell” allows “composing” values

```
(cons "hello" 4)
```

Lisp syntax: lists

• E.g., lists in Lisp are just made out of cons cells

```
(cons 1 (cons 2 (cons 3 nil)))
```

• Lisp has a shorthand for this:

```
'(1 2 3)
```

Lisp syntax: car and cdr

• Access the first element of a cons cell with car

```
(car (cons 1 2)) = 1
```

• Access the second element with cdr

```
(cdr (cons 1 2)) = 2
```

• What’s the value of the following expression?

```
(car '(1 2 3))
```

• What about this?

```
(cdr '(1 2 3))
```

Lisp syntax: functions

• Everything else is a function (or special form)

• There are a bunch of built-in functions

```
(car ...)
(cdr ...)
(append ...) , etc.
```

• And you can define your own

```
(defun my-func (arg) (value))
```
In Lisp, there is no **if/else**

```lisp
(cond ((test) (value)) …)
```

E.g.,

```lisp
(cond ((eq 1 x) (cons x xs)) …)
```

Does the same as the Java

```java
if (x == 1) {
    xs.add(x);
}
```

---

**cond** is more general than **if/else**.

```lisp
(cond ((test1) (value1))
     ((test2) (value2))
     …)
```

```lisp
(defun insert (x l)
    (cond ((eq l nil) (cons x nil))
          ((< x (car l)) (cons x l))
          (t (cons
              (car l)
              (insert x (cdr l)))))))
```

That’s pretty much it!

- **Note that the book uses slightly different syntax**
  - You should use:
    - `t` instead of `true`
    - `(defun ...)` instead of `(define ...)`
    - `(mapcar #'f xs)` instead of `(maplist f xs)`
  - See "334 Lisp FAQ" for all the syntax you need to know on course webpage

---

**Activity**

list length
Activity

Cow
  Badger  Eel
    Aardvark  Donkey  Fox

Memory management

- C:
  When you want to use a variable, you have to allocate it first, then deallocate it when done.
  
  ```c
  MyObject *m = malloc(sizeof(MyObject));
  m->foo = 2;
  m->bar = 3;
  ... do stuff with m ...
  free(m);
  ```

Memory management

- Lisp and Java:
  You barely need to think about this at all.
  
  ```java
  MyObject m = new MyObject(2,3);
  ... do stuff with m ...
  (cons 2 3)
  ```

Lisp memory model

Cons cell:  | Address | Decrement |
---|---|---|
Atom:  | Atom | value |

(cons 'A (cons 'B (cons 'C nil)))
(a) Sharing data                                    (b) Sharing data

- Both structures could be printed as \((A.B) \cdot (A.B)\)
- What is result of evaluating

\((\text{cons} (\text{cons} 'A 'B) (\text{cons} 'A 'B))\) ?
2. Mark reachable cells
2. Mark reachable cells

3. Free unreachable cells