Introduction to LISP

LISP language

LISP: LISt Processing language
• An AI language developed in 1958 (J. McCarthy at MIT)
• Special focus on symbolic processing and symbol manipulation
  – Linked list structures
  – Also programs, functions are represented as lists
• At one point special LISP computers with basic LISP functions implemented directly on hardware were available (Symbolics Inc., 80s)

LISP today:
• Many AI programs now are written in C,C++, Java
  – List manipulation libraries are available
LISP language

LISP Competitors:
- Prolog, Python
- but LISP keeps its dominance among high level (AI) programming languages

Current LISP:
- Common Lisp
- Scheme

are the most widely-known general-purpose Lisp dialects

Common LISP:
- Interpreter and compiler
- CLOS: object oriented programming

LISP tutorial
LISP tutorial

Syntax:
• Prefix notation
  – Operator first, arguments follow
  – E.g. (+ 3 2) adds 3 and 2
A lots of parentheses
• These define lists and also programs
• Examples:
  – (a b c d) a list of 4 elements (atoms) a,b,c,d
  – (defun factorial (num)
    (cond ((<= num 0) 1)
    (t (* (factorial (- num 1)) num)))

LISP tutorial: data types

Basic data types:
• Symbols
  – a
  – john
  – 34

• Lists
  – ()
  – (a)
  – (a john 34)
  – (lambda (arg) (* arg arg))
LISP tutorial

For each symbol lisp attempts to find its value
> (setq a 10)  ;; sets a value of symbol a to 10
10
> a             ;; returns the value of a
10

Special symbols:
> t      ;; true
T
> nil    ;; nil stands for false or
NIL
> ( )    ;; an empty list
NIL

Lists represent function calls as well as basic data structures
> (factorial 3)
6
> (+ 2 4)
6

> (setq a ‘(john peter 34))
(john peter 34)
> (setq a ‘((john 1) (peter 2)))
((john 1) (peter 2))
LISP tutorial: lists

List representation:
• A singly linked list

> (setq a ‘(john peter))
  (john peter)
> (car a)
  john
> (cdr a)
  (peter)

LISP tutorial: list

List building functions
> (cons ‘b nil) ;; quote means: do not eval the argument
  (b)
> (setq a (cons ‘b (cons ‘c nil)) ;; setq a is a shorthand for set ‘a
  (b c)
> (setq v (list ‘john 34 25))
  (john 34 25)
> (setq v (list a 34 25))
  ((b c) 34 25)
> (append ‘(1 2) ‘(2 3))
  (1 2 2 3)
LISP tutorial

List copying

> (setq foo (list 'a 'b 'c))
  (a b c)
> (setq bar (cons 'x (cdr foo)))
  (x b c)
> foo
  (a b c)  ;; (cdr foo) makes a copy of the remaining list before cons
> bar
  (x b c)
• Car and cdr operations are nondestructive.

LISP tutorial: lists

> (setq bar '(a b c))
  (a b c)
> (setq foo (cdr bar))
  (b c)
> (rplaca foo 'u)
  (u c)
> foo
  (u c)
> bar
  (a u c)
> (rplacd foo '(v))
  (u v)
> bar
  (a u v)
LISP tutorial

The same effect as with rplaca and rplacd can be achieved with setf

> (setq bar '(a b c))
   (a b c)
> (setq foo (cdr bar))
   (b c)
> (setf (cadr bar) ‘u)
   u
> bar
   (a u c)
> foo
   (u c)

---

LISP tutorial

Evaluation rules:
- A symbol value is sought and substituted
- A quoted value is kept untouched

> (setf a 12)
   12
> (setf b (+ a 4))
   16
> (setf b ‘(+ a 4))
   (+ a 4)
> (eval b) ;; explicit evaluation call
   16
LISP tutorial: functions and predicates

Some useful functions and predicates:

```lisp
> (setq a '(1 2 3 4 5))
(1 2 3 4 5)
> (length a)
5
> (atom 'a)
T
> (atom a)
NIL
> (listp 'a)
NIL
> (listp a)
T
```

LISP tutorial: function definition

Definition of a function

```
(defun <f-name> <parameter-list> <body>)
```

```lisp
> (defun square (x)
    (* x x))
SQUARE
> (square 2)
4
> (square (square 2))
16
```
LISP tutorial

Definition of a function

(defun <f-name> <parameter-list> <body>)

<body> can be a sequence of function calls, the function returns the value of the last call in the sequence

> (defun foo (a)
    (setq b (+ a 1))
    (setq c (+ a 2))
    c)
FOO
> (foo 2)
4

LISP tutorial: conditionals

Cond statement: sequentially tests conditions, the call associated with the first true condition is executed

> (defun abs (a)
    (cond ((> a 0) a)
          (t (- a)))))
ABS
> (abs 2)
2
> (abs -3)
3
LISP tutorial

if statement:

(if <test> <then> <else>)

> (defun abs (a)
  (if (> a 0) a (- a)))
ABS
> (abs 2)
  2
> (abs -3)
  3

LISP tutorial: equality

4 equality predicates: =, equal, eq, eql

> (= 2 4/2) ;; used for numerical values only
  T
> (setf a '(1 2 3 4))
  (1 2 3 4)
> (setf b '(1 2 3 4))
  (1 2 3 4)
> (setf c b)
  (1 2 3 4)
> (equal a b) ;; equal is true if the two objects are isomorphic
  T
> (equal c b)
  T
LISP tutorial: equalities

> (eq a b) ;; eq is true if the two arguments point to the same object
NIL
> (eq b c)
T

LISP tutorial: nil

Nil represents False and an empty list

> (null nil)
T
> (null ())
T
> (null '(a b))
NIL
> (not '(a b))
NIL
LISP tutorial: functions

Logical operators: and, or

> (and NIL T)
NIL
> (and T 2 3)
3
> (or nil (= 5 4))
NIL
> (or nil 5)
5

LISP tutorial: recursion

Recursive function definitions are very common in LISP

> (defun factorial (num)
  (cond ((<= num 0) 1)
        (t (* (factorial (- num 1)) num))
  )
  FACTORIAL
> (factorial 4)
24
LISP tutorial: local and global variables

> (setq a 12)
12
> (defun foo (n)
  (setq a 14)
  (+ n 2))
FOO
> a
12
> (foo 3)
5
> a
14