Lisp handout #2 – 2002 August 23

Expressions

Expressions have many uses – not just evaluation. Not all expressions can be evaluated.

A **list** has 1 or more elements (members) which are written between parentheses, separated by spaces. We store many kinds of information in lists. Examples of lists:

(A) a list with the symbol A as its only element
(A B C 49) a list of 3 symbols and a number
(A (B C) D) a 3-element list whose 2nd element is another list
((GEORGIA ATHENS) (FLORIDA TALLAHASSEE)) a 2-element list of 2-element lists

Operations on lists

(FIRST e) where *e* is a list, extracts the first element
Example: (FIRST '(A B C)) ➔ A

(REST e) where *e* is a list, gives the list of all elements except the first
Example: (REST '(A B C)) ➔ (B C)

(CONS x y) where y is a list, makes the list whose FIRST is x and whose REST is y
Example: (CONS 'A '(B C)) ➔ (A B C)

Special forms

A **special form** is like a function except that its arguments are not evaluated. So we have to revise our rule for how to evaluate a list:

To evaluate a list:
Look up the first element. It must be the name of a function or of a special form.
If it's a function, then:
    Evaluate all the remaining elements, in order.
    Pass those values to the function and perform the function.
If it's a special form, then:
    Pass the rest of the elements to the special form, unevaluated.
    The special form will decide whether to evaluate them.

Example of a special form: SETF

(SETF A 23) when evaluated, gives the symbol A the value 23.
(Do you see why SETF has to be a special form, not a function?)
And because (SETF A 23) is an expression, it has a value, namely 23.
Once given a value with SETF, a symbol retains that value until changed.

Example:
(SETF A 23) ➔ 23 Now the value of A is 23
(SETF B 'A) ➔ A Now the value of B is the symbol A
(SETF C 'B) ➔ B Now the value of C is the symbol B
(SETF D A) ➔ 23 But the value of D is the number 23.
A ➔ 23
B ➔ A
(EVAL B) ➔ 23
(EVAL (EVAL C)) ➔ 23