INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION

INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
34100 TRIESTE (ITALY) - P.O.B. 586 - MIRAMARE - STRADA COSTIERA 11 - TELEPHONE: 2240-1
CABLE: CENTRATOM - TELEX 460392-1

SECOND SCHOOL ON ADVANCED TECHNIQUES
IN COMPUTATIONAL PHYSICS
(18 January - 12 February 1988)

SMR.282/22

The LISP Family of Languages

A.C.HEARN

The RAND Corporation, Santa Monica, USA

THE LISP

FAMILY OF LANGUAGES

Anthony C. Hearn
The RAND Corporation

WHAT IS LISP?

Interpreter:

- Data types: integers, bignums, reals, vectors, list cells, identifiers, functions.
- Functions: predicates, structure manipulation, control flow, arithmetic, property list, I/O, evaluation, macros, error handling, debugging.
- Garbage collector.
- Library manager.
- Lisp listener: history queue, error trapping.

LISP DIALECTS

- Cambridge Lisp: BCPL based, IBM 370 series, MC 68000's. Based on Standard LISP.

- Common Lisp: Large standard for Lisp (1984). Not widely available yet, but use and implementations growing. IBM 370's, IBM PC/RT, IBM PC, many MC68000 systems, Data General, Prime, etc.

- Franz Lisp: Subset of MacLisp, distributed with Berkeley UNIX. Commercial version also available.Constructed in C. Portability based on C.

- Interlisp: Xerox/BBN. Large programming support environment. Runs on Xerox and DEC equipment.

- PSL: University of Utah. Efficient for application programs, highly portable. Cray, IBM 370's, VAX, Decsystem 10/20, MC68000's.

- Zetalisp: Successor to MacLisp, superset of Common Lisp. Texas Instruments, Symbolics, LMI.

2

Other Options:

- Editing.
- Document processing.
- Slide preparation.
- Electronic mail.
- Network support.

Compilation:

- Speedup of 5 to 100 times.
- Compiled and interpreted code can be mixed freely.
- Optimizations: Local variable names removed, macros expanded, loops unrolled, tail recursion removed, functions open-coded.
- Fast load libraries: separately compiled modules.
- Block compilation: optimizations within a module.

Debugging:

- Backtrace: function calls, variable assignments.
- Tracing: variable assignment, function invocation.
- Breakpoints: on error, user definable.
- Single stepping.
- Application dependent explanation: object browsers, text browsers.
- Data flow analysis: Master-(Micro-)Scope.

ATOMS

- **Numbers:**

  The usual format, any number of digits. +,- sign, E for exponents. Odd characters following a number will usually turn it into an identifier (i.e. 1+).

- **Strings:**

  Generally any sequence of characters enclosed in quotation marks (" not '). Double double quotes used to include a quote mark (PSL, Franz), or prefixed by an escape character (% in Interlisp).

- **Identifiers**

  These are also called "symbols" (Franz Lisp), "literal atoms" (Interlisp). Any sequence of characters that can't be interpreted as a number or a string and excluding parentheses, period, and square brackets. Generally starts with an alphabetic character or punctuation mark and terminates with a blank or right parentheses. ! (PSL), % (Interlisp), and back slash in Franz are used to include special characters in an identifier. |...| is used by Franz and ZetaLisp to enclose identifiers with one or more special characters. Either mechanism is used to freeze lower case characters into an identifier.

DOTTED-PAIRS

(CAR x) returns first element of dotted-pair
(CDR x) returns second element of dotted-pair
(LIST a ... z) constructs a list of elements
(CONS a b) returns (a . b)

List Notation: (a b c) is a list constructed of dotted-pairs in which the CAR of each pair points to an element of a list and the CDR points to the next list element or NIL if there are no more.

Dot Notation: is used whenever the CDR of a dotted-pair is not another dotted-pair.

CAR/CDR Composites: To keep from writing large numbers of nested CARs and CDRs, Lisp provides composites (up to 4 deep usually). Thus you can also write (CAR (CDR (CAR x))) as (CADAR x).

QUOTE Programs and data are constructed from dotted-pairs. QUOTE destinguishes between the two stopping evaluation of data that looks like a program. The QUOTE function call can be abbreviated with the ' mark.

LISTS

- Dot-Notation

$$(<\text{car}> \ . \ <\text{cdr}>)$$

The <car> or <cdr> can be:

- ☞ an atom
- ☞ a dotted-pair
- ☞ a list

- List Notation

$$(<\text{elt}_0> \ \ldots \ <\text{elt}_n>)$$

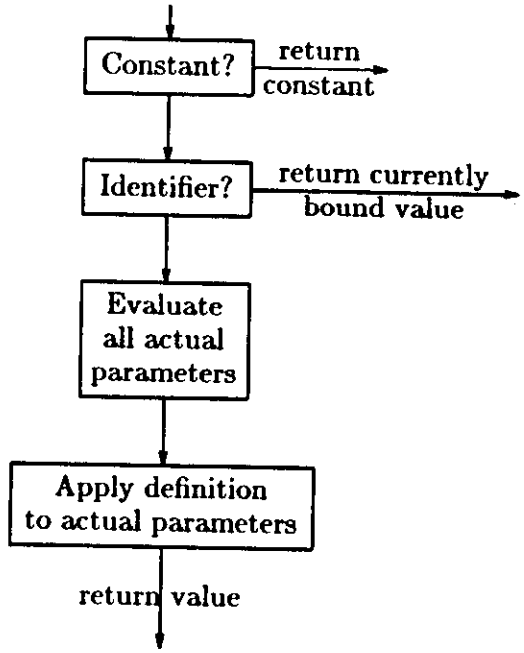
An $<\text{elt}_i>$ can be:

- ☞ an atom
- ☞ a dotted-pair
- ☞ a list

FUNCTIONS

$$(<\text{name}> \ <\text{actual}_0> \ \ldots \ <\text{actual}_n>)$$

Evaluation Strategy



ARITHMETIC FUNCTIONS

(plus a ... z) returns sum of all arguments
(difference a b) returns a - b
(times a ... z) returns product of all arguments
(quotient a b) returns a / b
(remainder a b) returns remainder of a / b
(add1 a) returns a + 1
(sub1 a) returns a - 1
(minus a) returns -a
(abs a) returns absolute value of a
(expt n m) returns n ** m
(fix r) converts floating to integer
(float n) converts integer to floating
(max a ... z) returns largest of a - z
(min a ... z) returns smallest of a - z
(sqrt a) returns square root of a

PREDICATES

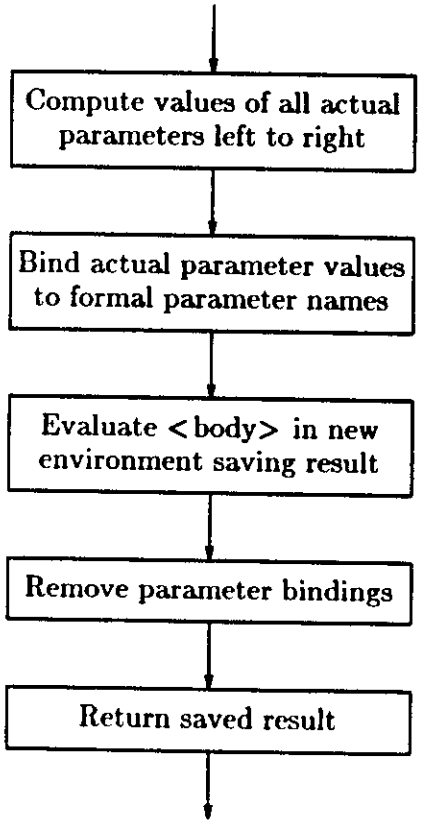
NIL = false

anything else = true

(greaterp a b) true if a is greater than b
(lessp a b) true if a is less than b
(equal a b) true if a equal to b (all types)
(idp x) T if x is an identifier (symbolp:Franz, litatom:Interlisp)
(stringp x) T if x is a string
(numberp x) T if x is a number
(atom x) T if x is not a dotted-pair
(pairp x) T if x is a dotted-pair
(null x) T if x is NIL

DEFINING FUNCTIONS

(defun <name>
(<a_0> ... <a_n>)
<body>)



COND

(COND (<$antecedent_0$> <$consequent_0$>)
.
.
.
(<$antecedent_n$> <$consequent_n$>))

COND evaulates successive antecedents until one returns a non-NIL value and then evaluates the corresponding consequent which is then returned as the value of the expression. If no antecedent answers non-NIL, NIL is returned as the COND's value.

(AND <a_0> ... <a_n>)

Evaluate each <a_i> in order until one is found that is NIL, then return NIL and don't evaluate any of the rest. If all return non-NIL, returns the value of <a_n>.

(OR <a_0> ... <a_n>)

Evaluate each <a_i> in sequence and return the value of the first one that does not return NIL and don't evaluate any after that. If all return NIL, the value is NIL.

EQ - EQUAL

EQ compares addresses, EQUAL compares values.

RECURSIVE FUNCTIONS

Some rules:

☞ Check for termination and return "empty" object if so.

☞ Do something to part of one of the arguments and combine this with the result of calling the same function changing at least one of the arguments.

EXAMPLES

```
(defun append (a b)
  (cond ((null a) b)
        (t (cons (car a) (append (cdr a) b)))))

(defun length (x)
  (cond ((atom x) 0)
        (t (add1 (length (cdr x))))))

(defun member (a l)
  (cond ((null l) nil)
        ((equal a (car l)) l)
        (t (member a (cdr l)))))

(defun assoc (a l)
  (cond ((null l) nil)
        ((equal a (caar l)) (car l))
        (t (assoc a (cdr l)))))
```


LISP TEXTBOOKS

Abelson, H. and G.J. Sussman, "Structure and Interpretation of Computer Programs", MIT Press, 1985.

Allen, J.A., "Anatomy of LISP", McGraw-Hill, 1978.

Steele, G.L. Jr., "Common LISP: The Language", Digital Press, 1984.

Touretsky, D.S., "LISP, A Gentle Introduction to Symbolic Computation", Harper and Row, 1984.

Winston, P.H. and B.K.P. Horn, "LISP" (Second Edition), Addison-Wesley, 1984.