# Lisp Machines and the Analysis of Their High-Level Language Computer Architecture

Dominic Dabish and Evan Bradley

Oakland University

{*dadabish, edbradley*}*@oakland.edu*

April 2016

# Overview

- Lambda Calculus introduced in 1930s by Alonzo Church

- Fortran in 1957

- No programming languages optimized for artificial intelligence

- Lisp designed in 1958 by John McCarthy

- Lisp code implemented on IBM 170 months after

- Lisp machines released in mid-1970s, became popular in 1980s
- Manufactured by Symbolics, Lisp Machines, Inc., Xerox, TI
- Offered GUIs, advanced programmability, flexibility
- Noncompetitive hardware
- Eventually became outperformed by general-purpose computers
- Vendors went bankrupt in 1990s

# Functions in Lisp

Consider a simple function g, which takes one argument x.

**Example (Function in mathematics)**

```
g(x)
```

In Lisp, this is written in the following manner:

**Example (Function in Lisp)**
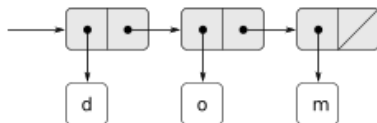
```
(g x)
```

# Lists

Consider a simple array of characters in Java:

```
char[] myArray = ['d', 'o', 'm'];
```

In Lisp, we can write the elements as symbols, and our data structure is a `list`.

```
'(d o m)
```

`car` and `cdr` are primitive operations for lists.

- `car` extracts the first element from the list
- `cdr` extracts the rest of the list

### Example (car)

```
(car '(d o m)) => 'd
```

### Example (cdr)

```
(cdr '(d o m)) => '(o m)
```

Let us define a factorial function in Lisp using recursion:

## Example (Factorial function in Lisp)

```
(defun factorial(n)
    (if (= n 0)
        1
        (* n (factorial (- n 1))
    )
)
```

Consider the following function call:

```
(factorial 5)
```

Here is its evaluation path:

| Function call | Evaluation 1 | Eval 2 | Eval 3 |
|---|---|---|---|
| (factorial 5) | (* 5 (factorial 4)) | (* 5 24) | 120 |
| (factorial 4) | (* 4 (factorial 3)) | (* 4 6) | 24 |
| (factorial 3) | (* 3 (factorial 2)) | (* 3 2) | 6 |
| (factorial 2) | (* 2 (factorial 1)) | (* 2 1) | 2 |
| (factorial 1) | (* 1 (factorial 0)) | (* 1 1) | 1 |

Table: Complete recursive evaluation

Is the argument `n` is equal to 0 (the base case)?

- **YES**: the function evaluates to 1 and terminates.
- **NO**: the function will multiply the argument by the factorial with `n-1` as the argument.

This creates a chain, and evaluates to the factorial of `n`.
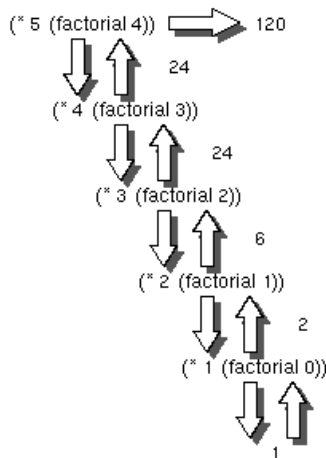
Lisp doesn't run very efficiently on general-purpose hardware. Problems include:

- Evaluating functions

- List representation

- Environment management

- Heap management

LISP is now the second oldest programming language in present widespread use (after FORTRAN). Lisp owes its survival to the fact that its programs are lists, which is actually a disadvantage.

- John McCarthy, 1979

- Lisp machines dominated in AI (their most popular domain)
- Worked very well for domain-specific programming
- Introduced new features: Dynamic creation of new objects, dynamically sized-lists, garbage collector
- On-the-fly changes - no need to recompile
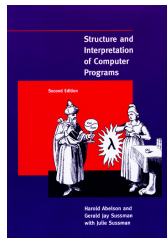
# Legacy of Lisp machines

Lisp continues to be a popular tool in

- education
- research
- general programming

*Lisp and Lisp machines influenced...* Haskell, JavaScript, Lua, Mathematica, ML, Nim, Perl, Python, R, Ruby, Scala, Smalltalk

*Structure & Interpretation of Computer Programs* is a classic programming textbook, and uses a Lisp variant to teach programming in universities nationwide, including Oakland University.

# Legacy of Lisp machines

Though Lisp machines are no longer a commercial enterprise,

- Lisp has continued interest and usage
- Lisp machines serve an important place in history
- Many with interest in retro-computing have recreational interest in Lisp machines

… even many years after the last manufacturers ceased to exist



Figure: These *retro keyboards*, that once were used with then-modern Lisp machines, are today collected by hobbyists.

Thank you!

Questions?