Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I

John Doe

March 22, 2005

Introduction

A bit of context

- The part II of this paper has never been written.
- LISP and Fortran were written for the IBM 704 (available at MIT).
  - "Only computer which can handle complex math" - Wikipedia
- Fortran older than LISP of one year.
  - One of the oldest high level programming language

IBM 704

Good Old Languages

- IPL2 (1956) - List processing in Assembly.
- Fortran (1957) - No list processing
- LISP (TODO when implementation?)

Goals of LISP

- AI (term coined by McCarthy)
  - They thought they were almost there in the 50s / 60s.
  - McCarthy wrote a “funny paper” later “HUMAN-LEVEL AI IS HARDER THAN IT SEEMED IN 1955” (
- Chess player (beginning of CS game theory).
- Programming the Advice Taker (proposed in 1958).

The Paper

Advice Taker

> *"representing information about the world by sentences in a suitable formal language and a reasoning program that would decide what to do by making logical inferences. Representing sentences by list structure seemed appropriate - it still is - and a list processing language also seemed appropriate for programming the operations involved in deduction - and still is." - McCarthy, 1979*

- From the paper: LISP handle declarative and imperative sentences.

Recursion

- Describe formalism for defining function recursively.
- First programming language with recursion.

$$
\begin{aligned}
2! &= (2 = 0 \rightarrow 1, T \rightarrow 2 \cdot (2-1)!) \\
&= 2 \cdot 1! \\
&= 2 \cdot (1 = 0 \rightarrow 1T \rightarrow \cdot (1-1)!) \\
&= 2 \cdot 1 \cdot 0! \\
&= 2 \cdot 1 \cdot (0 = 0 \rightarrow 1, T \rightarrow 0 \cdot (0-1)!) \\
&= 2 \cdot 1 \cdot 1 \\
&= 2
\end{aligned}
$$

Conditionals

- Invented conditional expression from propositional logic
  - Predicate: function returning #T or #F
- Idea of conditional in programming introduced here (not present in Assembly)

Functions

- Precise not the usual mathematics term “function”.
  - Same input potentially lead to different output.
- Inspired by Church’s lambda-notation (1936).
  - Turing Machine (1936) “too complicated”.

S and M expressions

M-Expression (paper)

car [cons [x; y]] = x cdr [cons [x; y]] = y

S-Expression (paper)

(CAR(CONS,x,y)) (CDR(CONS,x,y))

Actual code (Scheme)

```
(car (cons 1 2)) (cdr (cons 1 2))
```

Legacy

The most important

- First functional programming language
- Smalltalk (70s) was very influenced by LISP (today: Pharo)
  - One rule
  - Garbage collection
  - ... but don't like special forms

LISP today

- Common LISP
- Emacs LISP
- Clojure
- Scheme

References

- Slides TODO
- HUMAN-LEVEL AI IS HARDER THAN IT SEEMED IN 1955