Contents

1 Building Abstractions with Procedures

Definition (computational process)
Abstract beings that inhabit computers.

Definition (data)
Computational processes manipulate other abstract things called ***data*** as they evolve.

Definition (program)
A pattern of rules by which the evolution of a computational process is directed.

Definition (programming language)
That in which programs are carefully composed from symbolic expressions that prescribe the tasks we want our computational processes to perform.

Definition (bug, glitch)
Small errors.

Definition (debug)
Remove bugs.

Programming in Lisp

See the appendix.

Appendix

A Notes on LISP

A.1 McCarthy 1960

Recursive Functions of Symbolic Expressions and Their Computation by Machine

A.1.1 Introduction

LISP:

- "**LIS**t **P**rocessor"
- Developed for the IBM 704 computer by the Artificial Intelligence group at M.I.T.
- Designed to facilitate experiments with a proposed system called the *Advice Tracker*:
  - A programming system for manipulating expressions representing formalized declarative and imperative sentences so that the Advice Tracker system could make deductions.
  - Originally proposed in November 1958.
- Came to be based on a scheme for representing the partial recursive functions of a certain class of symbolic expressions, independent of any electronic computer.

In this article:

1. Describle a formalism for defining functions recursively.
2. Describe S-expressions and S-functions, give examples, and describe the universal S-function `apply` which plays the theoretical role of a universal Turing machine and the practical role of an intepreter.
3. Describe the representation of S-expressions in the memory of the IBM 704 by list structures ... and the representation of S-functions by program.
4. Mention the main features of the LISP programming system for the IMB 704.
5. Another way of describing computations with symbolic expressions.
6. Give a recursive function interpretation of flow charts.

A.1.2 Functions and Function Definitions

Definition (partial function)
A function that is defined only on part of its domain.

Definition (propositional expression)
A *propositional expression* is an expression whose possible values are T (for truth) and F (for falsity)

Definition (predicate)
A function whose range consists of the truth values T and F.

Definition (conditional expression)
A device for expressing the dependence of quantities on propositional quantities, denoted:

$$(p_1 \to e_1, \cdots, p_n \to e_n)$$

where the p's are prositional expressions and the e's are expression of any kind, read "If p_1 then e_1, otherwise if p_2 then e_2, ... otherwise if p_n then e_n" or "p_1 yields e_1, ..., p_n yields e_n."

How to determine the value of an arbitrary conditional statement $(p_1 \to e_1, \cdots, p_n \to e_n)$:

- Let $i = 1$.
- If the value of p_i is undefined, then the value of the conditional expression is undefined.
- If the value of p_i is T, then the value of the conditional expression is the value of the expression e_i.
- If the value of p_i is F, then increment i and evaluate p_i.

Example.

- $(1 < 2 \to 4,\ 1 > 2 \to 3) = 4$

  Starting from the left with the statement $1 < 2 \to 4$, the propositional expression $1 < 2$ is true, therefore the value of the conditional expression is 4.

- $(2 < 1 \to 4,\ 2 > 1 \to 3,\ 2 > 1 \to 2) = 3$

  Starting from the left with the statement $2 < 1 \to 4$, the propositional expression $2 < 1$ is false. Moving on to the next statement $2 > 1 \to 3$, the propositional expression $2 > 1$ is true, therefore the value of the conditional expression is 3.

- $(2 < 1 \to 4,\ T \to 3) = 3$

  The propositional expression $2 < 1$ is false, so move on. The propositional expression T is true, therefore the value of the conditional expression is 3.

- $(2 < 1 \to \frac{0}{0},\ T \to 3) = 3$

  The propositional expression $2 < 1$ is false, so move on. The propositional expression T is true, therefore the value of the conditional expression is 3.

- $(2 < 1 \to 3,\ T \to \frac{0}{0})$ is undefined.

  The propositional expression $2 < 1$ is false, so move on. The propositional expression T is true, therefore the value of the conditional expression is the value of the expression $\frac{0}{0}$ which is undefined.

- $(2 < 1 \to 3,\ 4 < 1 \to 4)$ is undefined.

The propositional expression $2 < 1$ is false, so move on. The propositional expression $4 < 1$ is false, so move on. There's nothing else to move on to, so the value of the conditional expression defaults to being undefined.

Example. Applications:

- $|x| = (x < 0 \to -x,\ T \to x)$
- $\delta_{ij} = (i = j \to 1, T \to 0)$
- $sgn(x) = (x < 0 \to -1,\ x = 0 \to 0,\ T \to 1)$

Example. Recursive applications:

- $n! = (n = 0 \to 1,\ T \to n \cdot (n-1)!)$
- $\gcd(m, n) = (m > n \to \gcd(n, m),\ rem(n, m) = 0 \to m,\ T \to \gcd(rem(n, m), m))$
- $\text{sqrt}(a, x, \epsilon) = (|x^2 - a| < \epsilon \to x,\ T \to \text{sqrt}(a, \frac{1}{2}(x + \frac{a}{x}), \epsilon))$

Example. Propositional connectives defined by conditional expressions:

- $p \wedge q = (p \to q,\ T \to F)$
- $p \vee q = (p \to T,\ T \to q)$
- $\neg p = (p \to F,\ T \to T)$
- $p \supset q = (p \to q,\ T \to T)$

Definition (form)
The expression $y^2 + x$ is an example of a form.

Definition (function)
A function substitutes arguments into a form to evaluate a result. It's necessarily to know which arguments map to which variables of the form. Something like $y^2 + x(3, 4)$ is not a function because it's not clear how $(3, 4)$ should map to x and y. If we make explicit the positions of the variables in the ordered tuple of arguments, then the mapping is clear. So, $\lambda((x, y), y^2 + x)$ is a function, where e.g. $\lambda((x, y), y^2 + x)(3, 4)$ evaluates to $4^2 + 3 = 19$.

A function of n arguments is denoted

$$\lambda((x_1, \ldots, x_n), \mathcal{E})$$

where $\mathcal{E}$ is a form of n variables $x_1, \ldots, x_n$.

Index