Do Not Borrow

March 30, 1959

Artificial Intelligence Project---RLE and MIT Computation Center

Memo 11

RECURSIVE FUNCTIONS OF SYMBOLIC EXPRESSIONS AND THEIR COMPUTATION BY MACHINE

by J. McCarthy

"This memorandum is a continuation of Memo 8"

5. Recursive Functions and Computer Programs

5.1 Recursive Functions of Numerical Arguments

Conditional expressions and recursive function definitions as devices to assist in describing procedures are applicable to other kinds of programming than symbolic manipulations. For example, the Newton algorithm for computing the square root of a number a starting with an initial approximation x may be described by

$$sqrt[a;x] = \left[\frac{|x^2-a|}{a} < \epsilon \rightarrow x; 1 \rightarrow sqrt\left[a; \frac{1}{2}\left[x+\frac{a}{x}\right]\right]\right]$$

Another example is the Euclidean algorithm for the greatest common division of two integers m and n.

$$gcd[m;n] = [m<n \rightarrow gcd[n;m]; rem[m/n]=0 \rightarrow n; 1 \rightarrow gcd[n; rem[m/n]]]$$

where by rem[m/n] we mean the remainder left on dividing m by n. This function in turn may be defined by

$$rem[m/n] = [m<n \rightarrow m; 1 \rightarrow rem[[m-n]/n]]$$

The universality of system of functions of the preceding sections is dependent more on the use of conditional expressions and recursion than on the base functions and predicates used. We can, for example, base the recursive functions of non-negative integers on the successor function (the successor of n' is denoted by n') and the predicate equality. We define

$$dim[m] = dim[m;0]$$

where

$$dim[m;n] = [m=0 \rightarrow 0; n'=m \rightarrow n; 1 \rightarrow dim[m;n']]$$

We further define

$$m+n = [n=0 \rightarrow m; 1 \rightarrow m' + dim[n]]$$

and

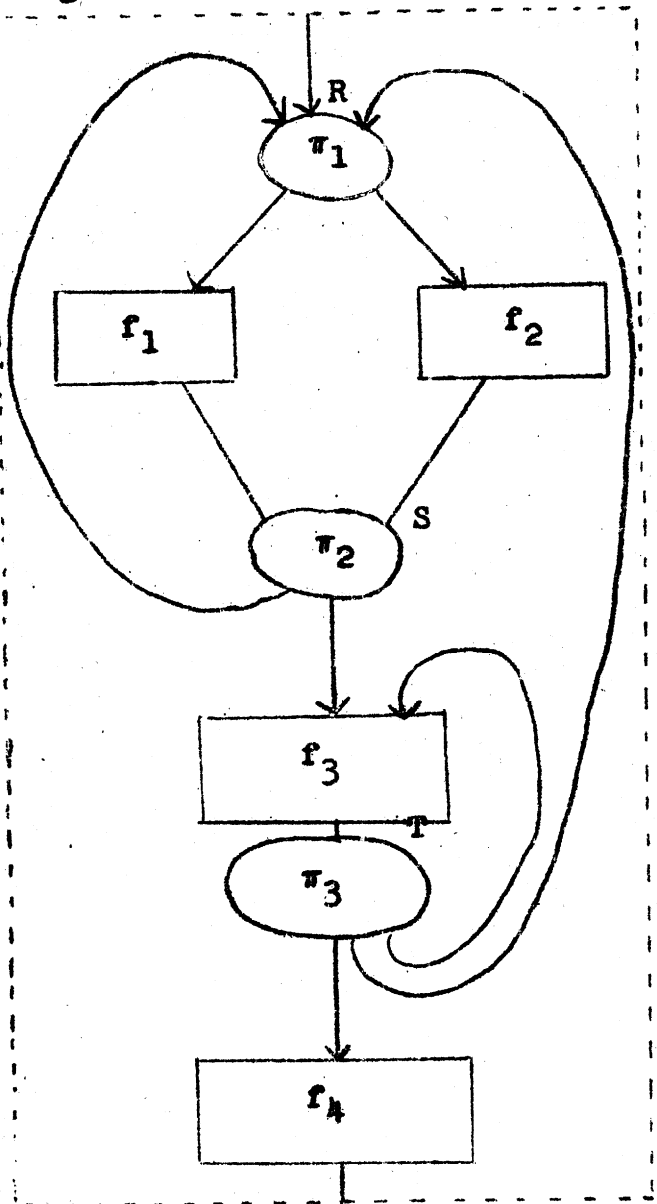
$$m*n = [n=0 \rightarrow 0, 1 \rightarrow m+m*dim[n]]$$

5.2 Flow Charts and Recursion

Since both the usual form of computer program and recursive function definitions are universal computationally it is interesting to display the relation between them. The translation of recursive symbolic functions into computer programs will be the subject of most of the remainder of this paper. In this section we show how to go the other way, at least in principle.

The state of the machine at any time during a computation is given by the values of a number of variables. Let these variables be combined into a vector ξ. Consider a block of program with one entrance and one exit. It defines and is

essentially defined by a certain function f which takes one machine configuration into another, i.e. f has the form $\xi' = f(\xi)$. Let us call f the associated function of the program block. Now let a number of such blocks be combined into a program by decision elements π which after each block is completed decide which block will be entered next. However, let the whole program still have one entrance and one exit. We shall give as an example a flow chart.

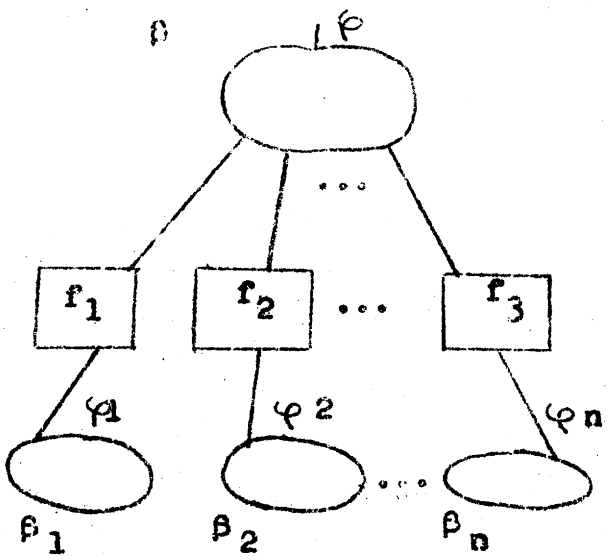


Let us describe the function $r[\xi]$ which gives the transformation of the vector ξ between entrance and exit of the whole block. We shall define it in conjunction with functions $s[\xi]$ and $t[\xi]$ which give the transformations ξ undergoes between the points S and T respectively and the exit. We have

$$r[\xi] = [\pi_{11}[\xi] \to s[f_1[\xi]]; 1 \to s[f_2[\xi]]]$$
$$s[\xi] = [\pi_{21}[\xi] \to r[\xi]; 1 \to t[f_3[\xi]]]$$
$$t[\xi] = [\pi_{31}[\xi] \to f_4[\xi]; \pi_{32}[\xi] \to r[\xi]; 1 \to t[f_3[\xi]]]$$

Given a flow chart with a single entrance and a single exit it is easy to write down the recursive function which gives the

transformation of the state vector from entrance to exit in terms of the corresponding functions for the computation blocks and the predicates of the branch points. In general, we proceed as follows:



In the diagram let β be an n-way branch point, let $f_1, \ldots f_n$ be the following computations heading to branch points $\beta_1, \beta_2 \ldots \beta_n$. Let φ be the function transforming ξ between β and the exit of the chart and let $\varphi_1, \ldots, \varphi_n$ be the corresponding functions for $\beta_1, \ldots, \beta_n$. We then write

$$\varphi[\xi] = [p_1[\xi] \rightarrow \varphi_1[f[\xi]]; \ldots; p_n[\xi] \rightarrow \varphi_n[f_n[\xi]]]$$

There are, however, problems connected with the descriptions of the functions f describing the computation blocks. These are

1. Each block usually affects only a few of the components of the vector ξ and is best described in terms which involve only the components affected and those on which they depend. Such a description is not affected by adding other components to the total vector ξ for other parts of the program.

The so-called arithmetic or replacement statement of Fortran, It or IAL which has the form

$$A = \mathcal{E}\{B, C \ldots\}$$

is of this kind. Even input and output statements can be included in this scheme provided the states of external media are included in the vector ξ.

2. A subblock in a flow chart may also be recursively defined. Usually the vector for the subblock will be conveniently described as having a different set of components from the program as a whole. There will be both variables (i.e. components) internal to the subblock and variables not involved in the subcalculation. To adequately describe this, requires a notation for extensions and reductions of vector functions.

3. Even with a notation for extensions and reductions, subroutines require additional treatment. A subroutine occurring at

several places in the program will not usually be applied to the same components at each place. In order to use the same subroutine at different places it seems to have a notation for selection and permutation of variables.

CS-TR Scanning Project
Document Control Form

Date : 11/30/95

Report # AIM-11

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☐ Technical Report (TR) ☒ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 5 (9-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☐ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☐ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☒ Other: MIMEOGRAPH (POOR)

Check each if included with document:
☐ DOD Form ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP: (1-5) UNNUMBERED TITLE PAGE, 1-4	
(6-9) SCANCONTROL, TRGT'S (3)	

Scanning Agent Signoff:
Date Received: 11/30/95 Date Scanned: 12/14/95 Date Returned: 12/14/95

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 12/14/1995

M.I.T. Libraries
Document Services