This is one of a series of memos concerning a logical system for proof-checking. It is not self-contained, but belongs with future memos which will describe a complete formal system with its intended interpretation and application. This memo also assumes familiarity with LISP and with "A Basis for a Mathematical Theory of Computation" by John McCarthy.

July 11, 1963
Primitive Recursion

by Michael Levin

Primitve recursive functions are a subset of general recursive functions having the following properties.

1. They can be defined using a particular recursive scheme given below.

2. They are always total functions, that is they are defined on their entire natural domain.

3. Most common arithmetic functions are primitive recursive.

4. Important parts of the arithmetization of metamathematics can be expressed using primitive recursive functions. In particular, suppose we define

\[ \text{theorem } \{x\} = \exists \text{proof}[\text{proofcheck } \{x; \text{proof}\}] \]

Proofcheck is a primitive recursive predicate which is true if \( x \) is a theorem and proof is its proof. Theorem is not primitive recursive. It is not even generally recursive although it is recursively enumerable.

Primitive recursive functions of the natural integers can be defined as follows.

1. constant functions e.g. \( \lambda(x,y,z) \) \( 4 \)

2. identity functions e.g. \( \lambda(x,y,z,w) \) \( z \)

3. the successor function \( \lambda(x) \) \( x + 1 \)

4. equality \( \lambda(x,y) \) \( x = y \)

5. composition e.g. \( \lambda(x,y) f(x, g(y,x,x)) \)

where \( f \) and \( g \) are primitive recursive.
6. recursive definition by the following schema only.
\[ f(y;x_1 \ldots x_n) = (\text{if } y = 0 \text{ then } h(x_1 \ldots x_n) \text{ else } \text{g}(y;x_1 \ldots x_n; f(y+1;x_1 \ldots x_n))) \]

where g and h are primitive recursive.

Primitive recursive functions of S-expressions are similarly defined.

1. constant functions e.g. \( \lambda[[u;v] \lt A B C] \)
2. identity functions e.g. \( \lambda[[u;v;x;y] x] \)
3. \( \lambda[[x;y] \text{ cons } [x;y]] \)
4. \( \lambda[[x;y] \text{ equal } [x;y]] \)
5. composition
6. recursive definition by the following schema only.
\[ f[y;x_1 \ldots x_n] = (\text{if } \text{atom } [y] \text{ then } h[y;x_1 \ldots x_n] \text{ else } \text{g}[y;x_1 \ldots x_n; f[\text{car}[y];x_1 \ldots x_n]; f[\text{cdr}[y]; x_1 \ldots x_n]]) \]

The introduction of recursive function definitions into a deductive system is a difficult problem. We want to be able to do so because recursive definitions are a powerful way of describing algorithms.

However, the introduction of recursive definitions indiscriminately will result in contradictions. The reason for this is that unlike proper definitions, recursive definitions are not eliminable. They actually introduce new premises into the system. The specific premise introduced by a recursive definition is that there is a function that satisfies the defining equation. If there is none, then the equation makes the system inconsistent.

Since we shall discuss recursion induction later, it might be well to
point out that a given equation need not have a unique solution. However, if it is convergent, then there is a unique solution.

For example, consider the equations

(1) \( \text{fact}(x) = (\text{if } x = 0 \text{ then } 1 \text{ else } x \cdot \text{fact}(x-1)) \)

(2) \( \text{fact1}(x) = \text{fact}(1)(x+1)/x+1 \)

Equation 1 converges and defines the factorial of a natural integer.

Equation 2 does not converge. It is satisfied by the entire family of functions \( \text{fact1}(x) = k \cdot x! \) where \( k \) is any constant.

Recursive definitions can be introduced under certain restrictions that guarantee their convergence. However, this requires that a certain portion of the arithmetization of metamathematics be accomplished first. This is difficult to do without using recursive definitions.

A convenient way to avoid this circularity is to allow the introduction of primitive recursive definitions as an axiom schema. This is consistent because the condition of total convergence is always met by primitive recursive functions.

The final topic of this paper is to obtain a general primitive recursion schema for a variety of domains. These domains will be built from primitive domains using the \( \oplus \) and \( \times \) operations described by McCarthy in "A Basis for a Mathematical Theory of Computation".

Consider a domain \( S \) defined from primitive domains \( A, B, C, \ldots \) \( S \) is defined by setting it equal to some form in \( S, A, B, C, \ldots \) which is composed using only \( (, ), \oplus \), \( \times \) as syntactic constants. We shall give a mechanical translation from the equation defining a domain into the primitive recursive
schema for that domain.

1. The left side of the equation is "S =". We translate this as

   "f(y,x_1 \ldots x_n) ="

2. The form "((A \& B))" is translated as

   "((if \( p_{A,B}(x) \) then \( h(x,y_1 \ldots y_n) \) else \( \beta \))"

   where \( \beta \) is the translation of \( \beta \).

3. The form "((A \times B))" is translated

   "\( g(y,x_1 \ldots x_n, f(\pi_{A,B}(x),y_1 \ldots y_n), f(\rho_{A,B}(x),y_1 \ldots y_n)) \)"

The canonical functions and predicates \( p, \lambda, \rho \) etc. are defined on pp. 49-50 of McCarthy's paper.

This translation has been properly defined only for certain domains.

As an example, we translate the definition of integers into primitive recursion schema.

**Integers**

\[
x = (\{0\} \; + \; (\{0\} \times I))
\]

\[
f(x,y_1 \ldots y_n) = \text{if } x \epsilon \{0\} \text{ then } h(x,y_1 \ldots y_n) \text{ else }
\]

\[
g(y,x_1 \ldots x_n, f(\pi_{A,B}(x),y_1 \ldots y_n), f(\rho_{A,B}(x),y_1 \ldots y_n))
\]

Note that \( x \epsilon \{0\} \) means \( x=0 \)

\( \pi_{A,B}(x) \) means \( \lambda((x) 0) \)

\( \rho_{A,B}(x) \) means \( \lambda((x) x-1) \) when \( x \neq 0 \).
CS-TR Scanning Project
Document Control Form

Report # A1M-55

Each of the following should be identified by a checkmark:
- Artificial Intelligence Laboratory (AI)
- Laboratory for Computer Science (LCS)

Document Type:
- Technical Memo (TM)

Date: 11/30/95

Number of pages: 5

Not to include DOD forms, printer instructions, etc... original pages only.

Intended to be printed as:
- Single-sided or

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

Print type:
- Inkjet Printer
- Laser Print

Check each if included with document:
- DOD Form
- Spine
- Cover Page
- Other:

Page Data:

Blank Pages: 1

Photographs/Tonal Material: 1

Other (note description/page number):
- IMAGE MAP! (1-5) UN# 40 TITLE PAGE, 1-4
- (6-9) SCAN CONTROL, TRGT'S (3)

Scanning Agent Signoff:

Date Received: 11/30/95
Date Scanned: 12/5/95
Date Returned: 1/7/95

Scanning Agent Signature: Michael W. Cook

Rev 994 DSL/LCS Document Control Form custform.vsd
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.