B. PROJECT SCOPE

Our objective is to produce an extended Forth language, "patternForth," which includes facilities for pattern matching and string manipulation.

The basic language structure will be defined by Forth. Since many anticipated uses for patternForth will involve its integration into "conventional" Forth applications, it is essential to remain compatible with the basic Forth language, and retain access to all of Forth's features. PatternForth shall follow the Forth syntax for program definition, data declarations, and control structures.

SNOBOL4 is the archetype for patternForth's the pattern processing [EMM87]. It embodies the necessary pattern matching functions. PatternForth shall employ "SNOBOL-like" syntax when this does not conflict with Forth.

Icon's newer programming methodology introduces some novel and quite powerful constructs, most notably the "generator" concept. PatternForth will not mimic Icon, but we will explore Icon constructs as (and if) they can be derived from the patternForth implementation.

We will now identify which elements of this pantheon are essential to patternForth, and which are the unessential "bells and whistles."

1. Essential features

a) Dynamic memory management for string storage

Most of the processing to be performed will involve variable-length text strings. As strings are input, manipulated, and output, some dynamic method of allocating and releasing storage for them will be required.

Many dynamic memory packages are costly in terms of execution time; others may run efficiently for a time but require occasional long interruptions to consolidate storage. Our manager must embody a reasonable compromise between speed and the efficient use of memory.

b) Associative reference

The ability to use text strings as subscripts of an array is an extremely powerful feature of SNOBOL4 and other text processing languages such as Awk [AHO78]. Storage locations can be given string- valued "addresses," by which they can be directly referenced. This "access by associative reference" solves many text processing problems, and will be a necessary adjunct to the language.

This implies a fast mechanism for converting text strings to unique memory addresses, and some way to associate some other data with each string. Put another way, we will require a storage location to be "attached" to every content-addressible string.

c) Pattern matching

This is the prime objective of the project.

The language needs the ability to match ambiguous patterns, i.e., patterns containing elements which may be matched by more than one string. This requires the pattern scanner to support complex backtracking [KER76], so as to attempt all possible matches to an ambiguous pattern.

d) Pattern definition

The following constructs will be needed to build patterns:

* concatenation of pattern elements;

* alternation of pattern elements, in which either of two or more pattern elements can satisfy the pattern;

* grouping, to allow subpatterns to be employed as pattern elements in concatenation or alternation;

* literal strings, to exactly match given text or binary data;

* an assortment of pattern functions, to specify ambiguous matches of various kinds; and

* conditional functions, to allow other calculations to affect the outcome of a pattern match.

e) Predefined pattern functions

Both SNOBOL4 and Icon are rich in predefined functions and operators. Most of these could be derived, albeit with some loss of efficiency, from a small set of functions. We will provide only this minimal function set, in an extensible framework that allows the addition of new functions and operators.

This minimal set of SNOBOL4 functions is:

* ARB, to match an arbitrary number of characters;

* LEN(n), to match a string of length n;

* POS(n), to match a given position within the string;

* ANY(s), to match any single character from string s.

f) Other pattern operations

Often it is desirable to have access to the "inner workings" or "internal results" of the pattern scanner. We will need two functions:

* capture match results, to store the text which matched an ambiguous pattern for later evaluation; and

* obtain cursor position, to make the scan position in the subject string available during the matching process.

g) String operators

Several other functions will be needed to create, manipulate, and analyze string data:

* copy a string to another (string assignment)

* return the length of a string

* concatenate two strings

* extract a substring

* create a filled string of length n

* convert strings to numbers, and vice versa.

h) String variables

We will need to define a "string variable" whose contents can be changed but whose identifier is fixed, i.e., a type of string storage which is <u>not addressed by its contents</u>. An example would be a variable to hold input.

i) Pattern variables

We will also need to define a "pattern variable" which is not content-addressed, and whose contents are a pattern.

j) Text arrays

We need to be able to work with groups of strings, identifying a particular string by a numeric subscript. This is the SNOBOL4 ARRAY construct.

k) TABLES (multiple hash tables)

We also need to be able to work with groups of strings, or other data, in which each item is identified by a string-valued "subscript." This is the SNOBOL4 TABLE.

A TABLE differs from the associative reference previously described in that it is a named subset of the memory space. In essence, it creates a new "context" for associative reference, so that the same string "address" may be used within different named TABLEs to reference different data.

2. Optional features

These features are not essential, but could increase the power of patternForth, or make it more closely resemble its model. They are not among the goals of this project.

The descriptions of these features, and the prospects for continued work in these areas, will be discussed later in this paper.

a) Memory compaction

b) Garbage collection

c) EMS Memory

d) Generators (as in ICON)