MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Cambridge, Massachusetts
Project MAC

Artificial Intelligence Project
Memo 71

Memorandum MAC-M-176
July 28, 1964

STRING MANIPULATION IN THE NEW LANGUAGE

by Daniel G. Bobrow

String manipulation can be made convenient within the *** language by implementing two functions:

1) match[workspace; pattern]

and

2) construct[format;pmatch].

In this memo I describe how I think these two functions can be implemented, and how they might be used to express operations now conveniently denoted in string manipulation languages such as COMIT[1], SNOBOL[2], and METEOR[3].

Pattern Matching

The first argument of match, called the "workspace", is the string to be manipulated. This string is an ordered list of elements. An individual element may be a single character, such as "A", "1", ".", etc., or a sequence of characters, e.g., "cat", "dog", "abq.m" or even another string.

The second argument of match is a pattern or prototype of a string. This prototype consists of a sequence of elementary patterns. We say that the workspace matches the prototype if there is a consecutive set of subsegments of the workspace that match in turn each of the elementary patterns in the prototype.

The following types of elementary patterns should be available:

1. A quoted item--will match any occurrence in the workspace of an identical item.
2. A string name--will match any substring of the workspace identical to the string named.
3. An n item marker--will match any n consecutive elements of the workspace.
4. An n item marker with condition--will match n consecutive items which meet a specified condition.
5. An indefinite string marker--will match a substring in the workspace of any length, including 0 (zero). This elementary pattern in equivalent to the COMIT $. It is tentatively matched with an empty substring. If the remainder of the prototype does not match the remainder of the workspace, the $ is tentatively matched to the workspace substring containing 1 element, etc. In other words, this marker will match the smallest substring of the workspace which will allow the remainder of the prototype to match the remainder of the workspace.
6. An indefinite string marker with condition specified--similar to the COMIT $, this marker will match the smallest substring of the workspace which both meets the specified condition and allows the remainder of the prototype to match the remainder of the workspace.
7. End markers--the left end marker will not permit the following elementary pattern to match any substring which is not an initial substring of the workspace; similarly, a right end marker will permit the immediately preceding elementary pattern to match only a terminal substring of the workspace.

Prototypes can consist of any non-empty sequence of these elementary patterns, with the obvious restriction that end markers should appear only

at the ends of the prototype. If more than one match can be found for any prototype, it is understood that the leftmost match will be taken. There should probably be a mode of operation in which all possible matches will be found by storing and returning later to any stage of the search.

The value of match is determined by the results of the matching operation. If no match is made, this value will be the null list, or some other easily identifiable flag. If a match is achieved, and the prototype consists of the sequence of elementary patterns P_1, P_2, ... P_k, then the value will be a symbolic array of k items. The jth element of this array is a copy of the subsegment S_j of the workspace which matched the elementary pattern P_j.

Each matched subsegment can thus be referred to by a number which is its position in the match array. This suggests another elementary pattern element;

8. A previous match name--this will match a substring in the workspace which is identical to a substring matched by an elementary pattern earlier in the prototype.

In addition to these numerical names, provision should be made to allow any matched subsegment of the workspace to be assigned directly as a value of a variable, as is done in SNOBOL. This could of course, be done indirectly by having a separate set statement and assigning to the variable a value which is the appropriate subsegment from the list of matched subsegments.

Proposed Notations for the Prototype

A prototype is a sequence of elementary patterns P_1, P_2, ..., P_k. A prototype is bracketted by parentheses and the elementary patterns separated by plus signs, i.e.,

$$(P_1 + P_2 + P_3 + \ldots + P_k)$$

Only P_1 or P_k (or both) may be an end marker. We will use $0 for these end markers (consistent with the COMIT 2 notation). For the other elementary patterns we will use:

1. Quoted strings--the quoted string of the *** language, e.g., "ABC" or "XY , (Z" etc., where " and ? may be included in a string by preceding them with ? (see section 2.6.1 of MAC-M-158).
2. String name--the string name, e.g., LIST3 as a pattern element will match a subsegment of the workspace which is the same as the string named LIST3.
3. An n item marker--$n (again consistent with COMIT), e.g., $1, $2, etc.
4. An n item marker with a condition--$n/PROC; the condition is specified by naming a procedure, in this case PROC, which has appropriate values and knows how to communicate the match or no-match condition. Arguments for this procedure may follow the procedure name.
5. An indefinite string marker--$ (consistent with COMIT).
6. An indefinite string marker with a condition $/PROC where PROC is an appropriate procedure name.
7. End markers--$0.
8. Previous match name--n where n is a number which is a previous match name, as in COMIT.

To give a permanent name to any matched substring, that is assign this substring as a value for a variable, precede the elementary pattern by this name followed by a "/", e.g., SUB/$3 will assign the three element substring which matches the $3 to be the value of the variable named SUB. SUB/$3 will not be confused with an elementary pattern consisting of the string name SUB, since in the latter case the "/" following SUB cannot appear.

Constructing New Strings

The function construct will construct a new string, given a format and a symbolic array which is an output from a match. The format is a sequence of elementary format elements which specify items to be placed in the new string. An element can be:

1. A quoted string--e.g., "abc" etc., this element will be placed in the new string in the position given.
2. A string name--e.g., LIST3; a copy of the string named LIST3 will be concatenated into the workspace.
3. A reference number--e.g., 3, which refers to the subsegment of the original workspace matched by the third elementary pattern of the prototype.
4. Any *** function of the above items, including those done for effect--as are LISP pseudo-functions.

A format sequence of elementary format elements F_1, F_2, ..., F_k is bracketted by parentheses and separated by plus signs, i.e., $(F_1 + F_2 + F_3 + F_k)$. Each F_i is independent of any other in the format string (and may be identical to another). The newly constructed string is also independent of the original workspace.

Conclusion

The string manipulation features described here are intended for use within the *** language, and thus no special control features have been suggested. The control mechanisms used by COMIT, SNOBOL and METEOR can be built from those provided. Utilizing the two functions described here, very complex string manipulations may be described very concisely. Since this string manipulation feature is still a proposal, critical suggestions on

notation and features are welcome.

References

1. Yngve, V., Comit Programmer's Reference Manual, MIT Press, 1962.

2. Farber, D., R. Griswold and I. Polansky, "SNOBOL: A String Manipulation Language", JACM, 11, No. 1; January 1964.

3. Bobrow, D. G., "METEOR: A LISP Interpreter for String Transformations", in The Programming Language LISP: Its Operation and Applications, E. C. Berkeley and D. Bobrow (eds.), Information International Inc., Cambridge, Mass., 1964.

This self-addressed form can be used to send the author any comments, corrections, and/or suggestions for revisions of the proposed string manipulation features for the *** language.

Thank you.

- -

To: Daniel G. Bobrow
Rm. 815
545 Technology Square
Cambridge, Mass.

From:

Comments:

Daniel G. Bobrow
Rm. 815
545 Technology Square
Cambridge, Mass.

CS-TR Scanning Project
Document Control Form

Date : 11/30/95

Report # AIM-71

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: 7 (11-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: MINIGRAPH

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: (1-7) UNF'KD TITLE PAGE,	2-7
(8-11) SCANNING TARGETS (3)	

Scanning Agent Signoff:
Date Received: 11/30/95 Date Scanned: 12/6/95 Date Returned: 12/7/95

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: 12/5/1995

M.I.T. Libraries
Document Services