SNOBOL Bulletin

Paul

Number 6 June, 1969

An interesting exercise for students in symbol manipulation techniques has been reported by Professor Peter Wegner of Cornell University. Technical Report No. 68-27 of the Department of Computer Science at Cornell describes a LISP interpreter which has been written in SNOBOL4. The data definition facility of SNOBOL4 was used to create the data type "CONS", with two fields CAR and CDR. Such a data definition creates the constructor function CONS and the two selectors CAR and CDR. The predicates ATOM and EQ were defined as SNOBOL functions. The remaining functions required by the LISP interpreter were programmed in SNOBOL using the five basic primitives and essentially transliterating the LISP definitions given by McCarthy and others. The interpreter implements only the "pure" form of LISP discussed in McCarthy's April, 1960 paper in C. ACM. It does not allow for the definition of a function by means of DEFINE, nor does it implement the numeric capabilities or the PROG feature of LISP1.5. Input/output is passed over briefly and it appears that a list must be entered as a sequence of CONS statements. For further details, contact Professor Wegner.

Mr. L. D. Yarbrough of the Arcon Corporation in Wakefield, Massachusetts provides the following technique for evaluating Boolean functions of string variables. This technique is easily extended to arithmetic modulo k, cryptographic substitutions, and the evaluation of a variety of other functions on strings:

> "Assume that the strings P, Q, ... contain sequences of binary digits, e.g. P = '01101011010010', all of the same length, How can we efficiently compute the functions "P and Q", "P or Q", etc?
>
> We observe that the following string,
>
> '11100010'
>
> has the property that the leftmost occurence of two particular bits adjacent in the string is immediately followed by the 'and' function of those two bits. That is, the leftmost occurence of '00' is followed by '0', '01' by '0', '10' by '0', and '11' by '1'. Thus the above 8-digit string is a SNOBOL representation of "and". We can define the "AND" function as follows:

```
        DEFINE('AND(R1,R2)', 'AND.1')/(...)
AND.1   AND =
AND.2   R1 *C1/'1'* = /F(RETURN)
        R2 *C2/'1'* =
        '11100010' C1 C2*C3/'1'*
        AND = AND C3                    /(AND.2)
```

We further observe that '10111000' is a similar representation for 'or', '1100010' represents 'exclusive or', and that similar sequences of eight or fewer bits can be found for each of the 16 boolean functions of two variables. Even simpler is the functional representation of 'not', for which either '010' or '101' will work. A little further study of these eight-bit sequences will reveal that if one such sequence represents a particular boolean function, another can be obtained by simply reversing the order of the eight bits. Thus, '01000111' will also represent 'and'. This may be important in an application of the technique in which it is known a priori that the data consists mostly of ones (in which case the first representation of 'and' will be more efficient) or of zeroes (in which case the latter is better).

A similar technique can be used to perform, say, modulo-three arithmetic: the sequence

'000112022101',

if scanned as in the program above, will yield the desired sum of two digit-strings, modulo three, and the sequence

'111221200020100'

will yield the produce of two digit-strings, modulo three.

A simple substitution cipher, in which each letter is uniformly represented by another (unique) letter in the enciphered message, is easily represented by the method we have outlined above, and encoding and decoding programs will be quite simple. The actual representations, including handling of special characters like blanks, etc., as well as the translation programs, can be safely

left to the reader. In this case, of course, we are dealing with functions of a single variable which can take on many values. Also, the encoding-decoding programs will be inverses: one will deal with the variable and the preceding character, the other with the variable and the following character.

A sophisticated reader will by now be able to extend the method of encoding and the analytic technique to include all of Group Theory, etc. The number of ways in which this highly compact and speedy function evaluation technique can be used are endless."

Although most of the readers whom I hear from these days are using or considering SNOBOL4, SNOBOL3 is not yet dead. Mr. John M. Chambers of the University of Wisconsin writes:

"First, a comment that may be of interest to the readers of the SNOBOL bulletin. A version of SNOBOL3 for the Burroughs B5500 has been developed here at the University of Wisconsin. It is basically the Bell Labs version, with a few minor extensions (mostly in the nature of added intrinsic functions). The I/O, of course, is rather different, and includes easily-used facilities for using a teletype as an interactive I/O device. Anyone interested can get a copy of the system by sending me a tape and specifying desired recording density. (The system is 8000 cards of ALGOL code.) A few manuals will also be sent with the system."

Most criticisms of SNOBOL implementations (see SNOBOL Bulletin #5) are fundamental ones. Mr. D. T. Chai of The Bunker-Ramo Corporation has quite a simple one:

"We have mostly used the IBM/360 implementation from Bell Laboratories but have also used Michigan's implementation through a teletype for a small debugging program. Then recently we have used the GE-635 implementation from the Bell Laboratory in Murray Hill. We will probably use other implementations as they become available.

For ease of printout on the teletype, I would like to propose that the statement number appear on the left side rather than on the right side. This will greatly speed up the listing of SNOBOL4 programs."

The SNOBOL group at Bell Laboratories recently sent out a questionnaire to determine the range of uses and users of SNOBOL4. The results (from the information bulletin released by Dr. Griswold) are as follows:

> "338 questionnaires were mailed. To date 266 (78.7%) have been returned by the addressee or someone who has taken over his position. 6 (1.8%) were returned as undeliverable, leaving 66 (19.5%) delivered and not returned, or lost in the mails. Of the returned questionnaires, 9 (3.4%) requested removal of their name from our mailing list.
>
> 225 respondents (84.5%) indicated they were using some form of SNOBOL. Of these 145 (64.5%) are in academic institutions, 51 (22.6%) are in industry, 9 (2.5%) are in government and 20 (11.2%) fall into other categories.
>
> Many versions of SNOBOL are in use and our questionnarie turned up a number of implementations that were new to us. 93 (41.3%) were using SNOBOL4 on the IBM System/360. 20 (10.7%) were using SNOBOL3 on the IBM 7094, 9 (4.5%) COM SHARE SNOBOL on the SDS 940, and 8 (3.5%) SNOBOL3 on the CDC 3600. 26 (8.7%) were scattered over 12 other implementations. The remainder did not specify what system they were using.
>
> Among the most interesting results of our survey are the various applications of SNOBOL. Some general categories were mentioned frequently: education, text processing, and computer-assisted instruction. In some cases we were not certain, from the limited information provided, whether applications were sufficiently similar to be classified together. An idea of the range of applications can be obtained from the following list of responses (some are paraphrased):
>
> instruction in computer science
> macro processing
> syntactic analysis
> linguistics
> grammar testing
> symbolic algebra manipulation

encoding English into Braille
translation of programming languages
artificial intelligence research
music analysis
computer assisted instruction
information retrieval
graphic arts
text editing
experimental compilers
text analysis
translation of FORTRAN source
generation of FORTRAN cross-reference tables
resequencing of FORTRAN programs
natural language processing
automatic indexing
list processing
preprocessing of FORTRAN programs
text formatting
hardware simulation
language interpreters
simulation
pattern recognition
information structure processing
data laundry
FORTRAN macro generator
music synthesis
theorem proving
management information systems
programming language experimentation
stylistic analysis
source deck processing
systems programming development
cryptanalysis
concordances
symbol manipulation
compilation of bibliographies
medical history taking
engineering applications
parsing algorithms
symbolic mathematics
metacompilers
transformational grammar testing
phonological rule testing
source language modification
job control language modification
program conversion

file manipulation
nonnumeric data structures
text processing
language development
compiler writing
literary research
simulation of concept learning
logic
text searching
syntactic transformations
psychological simulation
modelling psychological processes
programming research
information science
heuristic programming
data shaping
content analysis
code generation

Two SNOBOL implementors replied to the comments by users in SNOBOL Bulletin #5. The first letter was from Mr. L. Peter Deutsch at the University of California, who commented on my desire to write various parts of a program in different languages:

> "...I agree with you wholeheartedly in principle. However, this linkability is generally very difficult to achieve without some careful advance planning and standardization of some internal details of all the programming systems involved. We have been doing a little work on this here and suggest that the following runtime features must be common to, and identical in, any set of linkable programming systems:
>
> 1) A storage allocator, both for code and data.
>
> 2) A symbol table mechanism.
>
> 3) At least some of the machinery for calling procedures.
>
> 4) The lowest level of I/O.
>
> "In addition, there are serious problems of data representation to be solved. We recently encountered an application where it would have been extremely desirable to line a SNOBOL4 program to one written in machine language. However, in addition to lack of

compatibility in all four of the areas listed above, we were hindered by the fact that our SNOBOL4 uses a representation for strings which, though vital to its efficient operation, differs radically from our "standard". We are still trying to decide whether to rewrite the SNOBOL program in machine language or to try to bridge the gap. It may well be that the expense of conversion between data formats cancels out a large part of the advantage obtained. Of course, if we had realized these problems when we were designing our programming systems in the first place, we would have been quite willing to sacrifice a certain amount of efficiency for linkability."

In response to several questions I had after reading this letter, Mr. Deutsch expanded his description as follows:

"A dynamic storage allocator at run time is clearly necessary for SNOBOL, and to a lesser extent for ALGOL. However, to ensure a consistent storage management policy, I believe such an allocator should also be used at load time for those languages where separate compilation or static allocation of data storage is useful. This is just one aspect of the well-known general problem of just when and how names become bound to locations, of course. In SNOBOL4, something along these lines is already present in the CODE feature, which is a kind of loading of additional program.

"The symbol table mechanism was not intended for dynamic declaration of variables. It is intended to cover those cases where the names of objects are relevant at run time. I am thinking of SNOBOL and LISP in particular, since the ability to go from a value (data object) to the thing denoted by it is present in both languages. Such a facility would also be useful for a linking loader, an interactive debugger, a symbolic dump, or as a library routine in languages where a small amount of symbol manipulation might be embedded in other kinds of computation. With respect to passing of data addresses from one language to another, most languages allow passing of data "by value" with no interpretation by the procedure call machinery. The

critical question is conversion between differing representations of the same data object. Something like Standish's data description language is probably the best at this time.

"We intend to pursue these questions at a fairly low level of effort over the next year. I feel this interchange has been of some value and I will inform you of any significant progress we may achieve. As a first step, we hope to produce a FORTRAN IV with enough generality to allow linking with other languages we expect to implement later on (particularly SNOBOL); we are investigating possible hardware extensions to facilitate this."

A report from Mr. H. J. Strauss of Bell Laboratories presents some other work along the lines discussed above. Entitled "External Functions for SNOBOL4", it describes an external function facility which is implemented in version 1 release 3 of SNOBOL4 for the IBM System/360. The external functions give the user access to previously compiled programs in other languages, the use of Fortran library subroutines, and increased speed in performing operations which cannot be done efficiently in SNOBOL4 Currently the only programs which can be accommodated are those written in Fortran IV and System/360 assembly language. The author says that he is investigating the addition of other languages such as PL/I. External functions can be used in an overlay structure if it is not possible to load them into core with the SNOBOL4 system.

In order to obtain the linking capability Mr. Strauss defined two additional SNOBOL4 primitive functions, LOAD and UNLOAD. (The latter is only necessary when the space used by an overlay must be freed.) The programmer who writes external functions must be cognizant of the structure of the SNOBOL4 processor to some extent and must also have a more than usually comprehensive knowledge of the implementation of the language in which he is writing.

A letter from Mr. Paul McJones of the Computer Center, University of California at Berkeley describes a hand coded version of SNOBOL4 for CDC 6000 series machines:

"Although it is an interpretive system, the SNOBOL4 implementation for the CDC 6000 series which was designed at the University of California, Berkeley campus, Computer Center (CAL) may be of interest in light of your SNOBOL Bulletin #5. All features of SNOBOL as described in Preliminary Report on the SNOBOL4 Programming Language are implemented (except the dump and trace capabilities, which are not yet finished), and the performance seems quite good in comparison with the Institute for Defense Analyses (IDA) SNOBOL implementation for the CDC 6000 series. CAL SNOBOL will load in a 10000_8 field length of 60 bit words, compared with 70000_8 for the IDA system. Timing figures indicate that execution speed is roughly 4 times faster for CAL SNOBOL, with a similar ratio for compilation speed. A sample program to do symbolic differentiation required about 2 milliseconds per statement using the IDA version on our 6400. CAL SNOBOL required about 1/2 millisecond per statement. This is certainly not a tenfold difference, but coupled with the much smaller space requirement it makes SNOBOL an attractive language in view of our multiprogramming operating system and job mix containing a high proportion of small student jobs. Several classes here are using SNOBOL as an aid in learning the fundamentals of compiler techniques. There are certainly few languages other than SNOBOL which would enable a LISP compiler, for example, to be written with fewer than 300 statements.

CAL SNOBOL was designed by Charles Simonyi of our Computer Center. The compiler employs a finite-state machine algorithm to analyze the input text. The compilation process is one pass, and produces interpretive "micro instructions" which are essentially reverse Polish. The compiler and other primitive functions which are not referred to by the object program are purged from memory. The compiler and interpreter are hand coded, which accounts for the relative efficiency of CAL SNOBOL. The IDA implementation consists almost entirely of macro calls. (This is reported to be the Bell Labs version, with the macro definitions written to suit the 6000 series.)

A manual describing our implementation is in the works. Persons interested in further information about CAL SNOBOL should contact:

Paul McJones
Computer Center
University of California
Berkeley, California 94720"

As Mr. McJones points out, the speed up is not up to the factor of 10 which was requested by several users. The saving in space is more impressive, however, since the large amount of core required for the Bell Laboratories version is a serious drawback for many users. If the claim that all features except DUMP and TRACE are implemented is true, then this points to some serious inefficiencies in the distributed version. The distributed version is made up entirely of macro calls, and is hence machine independent to a large degree. In any such endeavor, one realizes that a certain amount of inefficiency will undoubtedly arise. However, a factor of 7 in space utilization seems an excessive price to pay for the machine independence. We must be very careful in evaluating the machine independent approach on the basis of this example, because there are two places where the inefficiency could have arisen: The macros might not have been written to provide sufficient optimization of the generated code, or the macro operations themselves could have been poorly chosen. In the former case, the method is still valid although the particular implementation would leave something to be desired. In the latter case one could argue that a better choice of operations exists or that SNOBOL4 is not amenable to the machine independent approach.

The Bell Laboratories group announces the availability of version 2 of SNOBOL4 which corrects errors in previous releases and includes a number of new language features:

Version 2.0 of SNOBOL4 is now available on the IBM System/360. This version corrects errors in previous releases and includes a number of new language features. Version 2.0 is approximately 12,000 bytes smller than its predecessor (Version 1, Release 3) and runs from 5% to 25% faster.

Version 2.0 has been sent to 360 installations that have requested SNOBOL4 systems since November 1, 1968 as well as to all installations which had a tape deposited with us at that time. Installations using earlier versions are urged to update to Version 2.0 which may be obtained by sending a DTR or mini-reel to:

Mr. R. E. Griswold
Room 2E-414
Bell Telephone Laboratories, Incorporated
Holmdel, New Jersey 07733

Installations converting to MFT-II or MVT must obtain Version 2.0 since earlier versions do not run under these systems.

Other recently announced implementations of SNOBOL4 are:

CDC 3600: Version 2.0 running. Contact:

Mr. Alfred I. Towell
c/o Research Computing Center
Hper Building
Indiana University
Bloomington, Indiana 47401

phone: (812)-337-1911

CDC 6000 Series: Version 2.0 running. Available soon from:

Mr. A. O. Hiebert
Manager, Distribution Department
Software Development Division
Control Data Corporation
3145 Porter Drive
Stanford Industrial Park
Palo Alto, California 94304

phone: (415)-321-8920

GE635: Version 2.0 running. Contact:

Miss P. A. Hamilton
Room 2C-561
Bell Telephone Laboratories, Incorporated
Murray Hill, New Jersey 07479

phone: (201)-582-2008

IBM System/360: Version 2.0 running. Contact:

Mr. R. E. Griswold
Room 2E-414
Bell Telephone Laboratories, Incorporated
Holmdel, New Jersey 07733

phone: (201)-949-5653

RCA Spectra 70 Series: Version 1, Release 3 running. Contact:

Mr. M. C. Paull
Radio Corporation of America
RCA Laboratories
David Sarnoff Research Center
Princeton, New Jersey 08540

phone: (609)-452-2700 x2176

UNIVAC 1108: Preliminary version running, Version 2.0 in final stages of development. Contact

Mr. Leon Osterweil
University of Maryland
Computer Science Center
College Park, Maryland 20704

phone: (301)-454-2946

Implementations of Version 2.0 are under way for the SDS Sigma 7, Burroughs B5500/6500, and Atlas 2.

In addition to these "standard" versions of SNOBOL4, there are independently implemented versions containing most SNOBOL4 features for the SDS 940 and CDC 6400. For more information, contact:

Mr. Bulter Lampson
Department of Electrical Engineering
University of California
Berkeley, California 94720

phone: (415)-845-6000 x4622

A book entitled "The SNOBOL4 Programming Language" has recently been published by Prentice-Hall at a price of $6.50. It is a correction and revision of the original SNOBOL manual written by the group at Bell Laboratories. It describes version 2.0 of SNOBOL4, while the previously circulated version of the manual described version 1, release 3. Orders may be placed with:

Order Department
Prentice-Hall, Incorporated
Englewood Cliffs, New Jersey 07632

I would like to encourage all SNOBOL users to submit short descriptions of their projects for inclusion in the bulletin. The questionnaire cited above shows a diversity of applications, and it would be interesting to hear more about them. Of particular importance are problems which are suited to students just learning the language. My experience has been that students react more favorably to problems which are obviously from the real world than to those cooked up by their professor. Please forward all material of interest to:

Professor W. M. Waite
Department of Electrical Engineering
University of Colorado
Boulder, Colorado 80302