The University of Arizona

Department of Computer Science
Tucson, Arizona 85721

Transporting the SIL Version of SNOBOL4

An Overview

Ralph E. Griswold

1. Introduction

The macro implementation of SNOBOL4 is written in SIL, an assembly-like language for an *abstract machine* (AM) that does not correspond to the architecture of any actual computer. SNOBOL4 may be implemented on a particular *target machine* (TM) by implementing SIL on that machine. This is usually done by treating SIL operations as macros in the assembly language of the TM. Consequently, SIL is implemented by providing suitable TM macro definitions for the AM operations.

This paper is an overview of the process involved in installing the macro implementation of SNOBOL4 on a TM. A list of material available to support such installations is included for reference.

2. Suitable TMs

SNOBOL4 as implemented in SIL is designed for large-scale digital computers with substantial amounts of memory. The SNOBOL4 system is organized as one large program (rather than as a number of modules) in order to avoid addressing and binding problems. As a consequence, it cannot be segmented easily.

To give an idea of the memory resources required, the SIL implementation of SNOBOL4 requires a user region of at least 165K bytes on the IBM 360/370, and 185-200K bytes are needed to run large production programs satisfactorily. SIL implementations on other computers require similar memory resources.

Although the macro implementation of SNOBOL4 has been installed satisfactorily on machines with virtual-memory environments, performance of these implementations is usually poor unless relatively large amounts of physical memory are available. Installations on machines with small amounts of memory have generally been unsuccessful.

SIL makes few specific demands on TM architecture. It does not, for example, even assume the existence of any hardware registers. The suitability of a particular TM architecture and its instruction set affects the size and efficiency of the result to some extent, and may make the implementation of SIL more or less difficult. TM architecture itself does not generally determine the feasibility of implementing SIL, however.

TM word size (or equivalent addressing constraints) may have a significant impact on the size of an implementation, and in extreme cases may be a major factor in determining feasibility. This aspect of architecture is important because SIL is based on the concept of a ‘‘descriptor’’, which is the basic ‘‘word’’ of the AM. If the TM word fits the AM descriptor well, implementation is relatively easier and more economical of space. If the fit is poor, the amount of TM memory needed may nearly double. Experience has shown that a word size of 60 bits is close to ideal. A 45-bit word is the smallest ever successfully used to represent an AM descriptor. Below this word size, at least two TM words are needed to represent each AM descriptor. The significance of this matter is illustrated by the fact that the SIL version of SNOBOL4 requires about 10,000 descriptors of AM memory for dynamic allocation during SNOBOL4 program execution.

3. The Transporting Process

The conceptually simple technique for transporting SNOBOL4 to a specific TM involves a number of steps, which require the installer to:

(1) review the documentation

(2) translate the source material into TM form

(3) design data layouts for the TM

(4) provide TM macro definitions for SIL operations

(5) assemble the translated SNOBOL4 system

(6) test and debug the result

Rather substantial documentation about the macro implementation of SNOBOL4 is available; see Section 5.

The macro implementation of SNOBOL4 is distributed to would-be installers as two machine-readable files, one containing the SIL program and the other containing a description of syntax tables used by the compiler. Since character sets and assembly-language formats vary from one system to system, one of the installer's first tasks is to detect such problems and to translate the SIL program as distributed into a form suitable for the TM. One of the best tools for performing this translation is SNOBOL4 itself. The availability of SNOBOL4 on a machine other than the TM permits cross translation.

Data layout involves adaptation of the TM architecture to the AM descriptor. Descriptors consist of fields used to represent various kinds of values manipulated by SIL. Field layout depends on size requirements of SIL and positioning that is influenced by TM architecture and SIL use of the fields.

In addition, the installer must construct syntax tables that are used in lexical analysis by the SNOBOL4 compiler and in pattern matching. Syntax tables may be constructed using descriptors or they may be tailored to the architecture of the TM.

There are 130 different SIL macros. Many operate on SIL descriptors or their fields. Others move strings, perform lexical analysis, do input/output and so forth.

4. Difficulties with the Implementation

Although the method of implementing SIL appears to be simple on the surface, the amount of work required is significant and there are a number of difficulties and potential pitfalls.

Some SIL operations are complicated. Examples are string concatenation, real exponentiation, and string-to-numerical conversions. Other potentially troublesome SIL operations include input, output, and interfacing the operating system.

The representation of syntax tables is left largely to the installer, since character set sizes and collating sequences vary substantially from one TM to another. This aspect of the installation therefore requires more installer knowledge and skill than most other aspects.

Another major stumbling block is the lack of test programs for either SIL or SNOBOL4. While the latter can be supplied with relative ease by any experienced SNOBOL4 programmer, the lack of test routines for SIL is a much more serious problem. In fact, the only existing SIL program is SNOBOL4 itself. Consequently, unless the installer is willing to make a substantial commitment to developing SIL tests locally, SIL must be debugged by running SNOBOL4 in its entirety.

In spite of these problems, the macro implementation of SNOBOL4 has been successfully implemented on over 30 different computers. The time required for these implementations has varied considerably, and depends on such factors as installer competence, quality of local computer facilities, and TM architecture. The range of effort required typically varies from three to six man-months.

5. Available Material

There is a substantial amount of material available to the would-be installer of the SIL implementation of SNOBOL4. Much of the basic documentation is given in a book that is available through book suppliers. The rest of the material is available from the University of Arizona:

Ralph E. Griswold
Department of Computer Science
Gould-Simpson Building
The University of Arizona
Tucson, Arizona 85721
U.S.A.

telephone: (602) 621-6613

Documents with identifying numbers should be requested by number.

1. Version 3.11 SIL source code and syntax table descriptions in machine-readable form. This material is available in a variety of tape formats. The standard distribution is 9-track, 1600 bpi, unlabeled fixed-blocked, EBCDIC. The distribution also is available on MS-DOS 5.25″ diskettes. Enquire for prices.
2. S4D58: *Implementing SNOBOL4 in SIL; Version 3.11*. Detailed instructions for designing data layouts and implementing the individual macro operations; the ‘‘cookbook’’.
3. *The Macro Implementation of SNOBOL4; A Case Study of Machine-Independent Software Development*. (author: Ralph E. Griswold, publisher: W. H. Freeman & Co.) A description of the SIL version of SNOBOL4 that describes data structures, algorithms, the SIL macros, and gives examples from the IBM 360 and CDC 6000 implementations. This book is available from book sellers. The price is approximately $25.00. The terminology used in this book is different from that used in the actual SIL source. See S4D59 below.
4. *Corrigenda for The Macro Implementation of SNOBOL4*. Corrections to the Freeman book listed above.
5. S4D59: *Comparison of Terminologies for the SIL Implementation of SNOBOL4*. Explains the differences between terminology of the Freeman book and that actually used in the machine-readable SIL program.
6. S4D26c: *Source and Cross-Reference Listings for the SIL Implementation of SNOBOL4; Version 3.11*. Listing of SNOBOL4 written in SIL. This document is primarily useful for its cross reference to program symbols.
7. S4D20a: *IBM 360 Macro Definitions for Version 3 of SNOBOL4*. Listing of the IBM 360 macro definitions for SIL operations; primarily useful as an example of an existing implementation. The macro definitions are also available in machine-readable form.
8. S4D19a: *IBM 360 Subroutines for Version 3 of SNOBOL4*. Listing of the IBM 360 subroutines that support SIL operations; primarily useful as an example of an existing implementation. The subroutines are also available in machine-readable form.
9. S4D57: *Implementations of SNOBOL4*. Compilation of SNOBOL4 implementations, including those done in SIL; primarily useful as a source of contacts with other SIL implementors.