[*Proceedings of the National Symposium on Machine Translation*, UCLA February 1960]

THE COMIT SYSTEM[1]

Victor H. Yngve
Massachusetts Institute of Technology

About two years ago a need was felt at M.I.T. for a programming system specially tailored to the needs of mechanical translation. Our idea was to provide an automatic programming system that would make it easy for the linguist to write his own programs. We conceive of the computer as having a place in a research effort as a tool for increasing the productivity of the people associated with it, not as a hard master requiring research workers to understand and control countless details that are of little direct significance to their work. Consequently, we aimed at relieving the linguist of many details like packing text into 36-bit registers; shifting letters into certain chosen locations in registers; allocating storage in an economical way; lining everything up and squeezing it through an accumulator that was designed for arithmetic; calculating addresses when they matter to no-one, but are required for proper computer operation; or spending time worrying about achieving the fastest program or the most economical use of high-speed memory. In other words, we hope that the linguist can concentrate on problems central to mechanical translation and be able to write programs about as fast as he works out linguistic solutions.

Our decision to invest a substantial amount of time in designing and writing an automatic programming system was based on a very practical consideration. We believed at that time, and still believe, that we are in a phase of mechanical translation that will be characterized by the production of many and diverse approaches to the problem. We consider it to be desirable to try out as many of these approaches as possible and to subject them to the most searching tests of quality and fidelity that can be devised. With an adequate automatic programming system, the cost of making such experiments

---

[1] This work was supported in part by the National Science Foundation; and in part by the U.S. Army (Signal Corps); Air Force Research Division, Air Research and Development Command; and the Office of Naval Research.

is reduced and, more important, the time lag between conception of a scheme and a final running program is considerably reduced.

G.H. Matthews and I worked up a set of specifications for what we thought a programming system should offer the linguist in the way of facilities. We started from a notation very much like the notation used by A.N. Chomsky in his transformational grammars, but we added many features for convenience and very carefully defined the meaning of each aspect of the notation in terms of the computer operations that were to be carried out. We found that we had to add a method of addressing the grammar rules, a method of making program branches, a powerful set of subscript conventions, and many other features. We tried to design the notation to be as natural and simple as possible so that it would be easy to learn and easy to use. At the same time, we tried to foresee as many types of programming situations as possible and tried to provide for them in a clear and obvious way.

We took our ideas to the M.I.T. Computation Center, and there we found Sheldon Best, Arnold Siegal and Frank Helwig, who became interested in providing our notation with an automatic programming system. Together we spent several months going over the specifications and improving and clarifying them. Some valuable new features were added. Finally, we had the notation in a satisfactory form and programming began. Altogether about twelve people have contributed to the COMIT system, as it became called, and perhaps eight man-years have been spent on it. This work has been divided about equally between the two groups. Now, about two years later, the programming and coding are finished, and the system is undergoing final checkout. Actually the system consists of two separate programs, a compiler and an interpreter, each occupying over 8, 000 registers in the IBM 704 computer.

From the user's point of view, however, there is only one system. He writes his program in the convenient COMIT notation, and submits it along with the text or raw material that his program is to work on. He gets back his results in the form that he has specified in his program. If he wants to run the same program again, with perhaps a different text, he merely saves the binary version of his program produced by the compiler, and resubmits it with his new

text for running with the interpreter only. The advantage of the compiler-interpreter split is that the compiler can translate the problem-oriented COMIT programming language into a machine-oriented binary version which will run much faster with an interpretive program and make much more efficient use of memory space. Thus, although the whole system is designed with the primary objective of conserving the linguist's time by providing him with a powerful research tool that is easy to use, the programs will in general be fairly efficient and fast.

One of the easiest programs to write in COMIT is a dictionary routine. One merely has to list the words on punch cards and submit them to the compiler. Alphabetization is taken care of automatically. There would be room for about 4, 000 words in core storage. A binary search is provided so the program would run quite fast. One could write a dictionary program in machine code or in other programming languages that would beat a COMIT dictionary both in vocabulary size and speed of running, but probably in no other system could a dictionary of this size be compiled with as little expenditure of effort on the part of the linguist.

In anticipation of translation programs, one will probably want to provide the dictionary entries with grammatical codes. This can easily be done by using the subscript facility of COMIT. Subscripts can be represented by nearly any mnemonic abbreviation, number, or word, that suits the fancy of the linguist. Subscripts are then automatically encoded by the compiler in a very efficient manner.

I shall not go into details about how easy it is in COMIT to replace, rearrange, delete, or add linguistic material; how to make use of the built-in random element in the program in linguistic research; how easy it is for a COMIT program to write another COMIT program; or indeed another program in any notation. Material has been published that goes into these details.

A number of programs have already been written in COMIT in spite of the fact that they can not yet be run on the computer. Many of these programs have been written as exercises to aid us in learning how to make the best use of our new tool. Considerable research in programming methods in COMIT has been carried out by G.H. Matthews and me, and we have presented several short courses in

COMIT. One of the exercises consisted in programming in COMIT a large portion of the General Problem Solving Program (GPS) of Newell, Shaw, and Simon, discussed by them in a recent paper.[2] On the basis of this exercise, it is estimated that the entire GPS routine would require only about 200 COMIT rules, whereas it requires about 1,000 instructions in International Programming Language (IPL), the programming language used by them. Whether COMIT is any more convenient for such programs remains an open question, but many people around M.I.T. are finding it convenient for a variety of unusual uses. It has been used in a program for the automatic control of milling machines. It is being used for a theorem proving routine. It is being used in programs to do algebra and calculus. Several game-playing programs have been written or are under consideration. One of them, a Scrabble program, written by Bill Cooper, looks as if it would play a very good game. Kenneth Knowlton of M.I.T. wrote an information retrieval program in COMIT during a summer spent at IBM at San Jose.

The availability of the COMIT notation has already had a profound effect on mechanical translation research. It has been possible for us to write down in an unambiguous fashion our ideas on translation. This has aided greatly in clarifying our own thoughts and in communicating them to each other. There is nothing like a clearly written program to overcome terminological barriers to communication. We have come to realize the very great importance of an adequate notational system.

The availability of the COMIT notation has also provided us with a frame of reference within which to work. G.H. Matthews has written several sentence-recognizing programs in COMIT. The work of David Dinneen on French has been within the framework of COMIT, and Anthony Phillips has written a German-compound-splitting routine in COMIT.

---

[2] A. Newell, J. C. Shaw, and H. A. Simon, "Report on a General Problem-Solving Program", Preprints for the International Conference on Information Processing, UNESCO, Paris, June 15-20, 1959.

For those of you that want to use COMIT, I am happy to announce that it will be ready soon. It can be used on any IBM 704 with a 32, 000-word core memory. We will either distribute it through the SHARE organization, or will undertake the distribution ourselves. In any case, if you want it, you should write us. We will be happy to send you, when it is ready, a copy of the new programming manual, being prepared by John Viertel. There is also a new reference manual being prepared by Frank Helwig and Kenneth Knowlton.