~~CONFIDENTIAL~~

Copy 22
S. Lynder
ANEQ 1
888

CONFIDENTIAL

# AN AUTOPROGRAMMING SYSTEM FOR HARVEST (I)

Walter W. Jacobs
REMP-102
1 January 1958
12 pages

The meaning of a "standard problem of autoprogramming" is explained and illustrated, and it is proposed that such a system be prepared for use with HARVEST.

S 95 - 272



~~CONFIDENTIAL~~

CONFIDENTIAL

AN AUTOPROGRAMMING SYSTEM FOR HARVEST (I )

## Introduction

There is general recognition that it will be necessary to have some type of autoprogramming system for use in conjunction with HARVEST, when this very high-speed, general-purpose computer is installed. However, there are many different interpretations of autoprogramming, ranging all the way from techniques for simplifying and speeding up the programmer's task to standardized languages, which would enable the customer to specify his procedure directly to the computer. Even the particular concept of standardized language can be applied in many ways.

The choice among these approaches must be based on a consideration of the Agency's objectives in acquiring HARVEST. The various types of autoprogramming systems will contribute in different degrees to these objectives.

The increase in speed of operation which HARVEST will provide is desired in order to handle work that is outside the range of feasibility on present general-purpose computers. This objective, in other words, involves computer projects which will require an appreciable number of hours even on HARVEST.

Because of the great speed, the new equipment will have a large capacity by comparison with computers now in use. A second important objective is to use this capacity to increase the Agency's productivity: to systematize and automatize its operations so that much more intelligence can be produced with a given staff.

A third objective is sometimes implied: the elimination of many of the slower present-day equipments so as to unify and simplify MPRO's activities. This type of consolidation must be limited, however, to jobs large enough to be worth doing on HARVEST. This will be clear by analogy with supersonic passenger aircraft: they would never be used for travel between Washington and Baltimore, since total elapsed time would hardly be reduced by these aircraft, while costs would be considerably higher. In other words, it may always be necessary to have less powerful computers for small jobs. This does not, of course, exclude the possibility of organizing standard types of small jobs into runs of size which could properly go on HARVEST.

The same consideration shows that effectively to use HARVEST for the second objective (that of greatly increasing Agency productivity) it will be necessary to organize the flow of work to the computer. Equipment of this capability will not be used efficiently in a job-shop type of operation. This term is intended as a reminder that a computer, like a machine tool, has a "make-ready" cost for each new job. If it takes an average of, say, five minutes to get a computer ready to handle a problem which then runs for 30 seconds, there is a serious waste of capacity incurred.

To summarize: the major use of HARVEST will be on jobs which take a reasonable length of machine time to run, or else on problems which can be organized into production-size runs even though they are individually small.

CONFIDENTIAL

## Preparing a Computer Program

In order to look at the different types of autoprogramming systems and to relate them to the way that HARVEST is to be used, we will first consider in some detail what is involved in preparing a computer program.

The typical development of a computer application involves the following steps:

a. Recognition of a particular problem.

b. Selection of a solution procedure.

c. Decision that the procedure is too laborious to be carried out manually.

d. Adaptation of the procedure to conform to computer capabilities and limitations.

e. Expression of the adapted procedure in the form of detailed machine instructions.

f. Elimination of major bugs in the program.

g. Rounding out of the program for operational use (preparation of operating instructions, write-ups, etc.).

h. Application of program, leading to further debugging and improvement.

The first three steps are generally carried out before there is any thought of using the computer. For this reason, they are often overlooked completely. As a result, the fourth step will almost certainly commence with a statement of the procedure developed in step (b), rather than with a statement of the problem itself. A characteristic example (from an area outside the Agency's work) occurs where a design engineer wishes to obtain a numerical solution for a

differential equation with certain boundary conditions. He reduces this to a definite integral which he cannot evaluate. Almost inevitably, he will ask the computer service to evaluate the integral, rather than to solve the original equation. When the service has to prepare a program to evaluate the integral and already has one to solve the differential equation by some other approach, unnecessary loss of time and waste of effort results.

In step (d), a start is made toward a complete and accurate specification of what is to be done. Procedures to be followed manually are almost invariably stated incompletely; no thought is given to large numbers of detailed decisions until the occasions for them arise in actually working out a problem. For a computer procedure, such decisions must be foreseen and provided for in advance. However, much of this takes place in step (e), the preparation of machine instructions.

When a program is to be reused, much more is involved in complete specification, and the task of adaptation becomes correspondingly more difficult. It is hardly necessary to point out how drastically a cryptanalytic procedure may have to be modified because of a change in the length of message available, or in the accuracy of the text, or in the limitations on usage of the cryptosystem. For this reason, if too little is known about the problem to be sure that the procedure is adequate, the program should be viewed as one-time even though the problem will be a recurrent one. As a practical matter, it is rare that a really complex program on a relatively new problem can be successfully reused without major reworking of the initial form.

Considering how sensitive to the parameters of a problem the proper choice of procedure can be, it is worth asking how often adequate attention is given to this choice. It seems clear that few people have enough cryptanalytic and machine experience to be trusted to make this decision without thorough review, in any case where the amount of machine time or programming effort involved is substantial.

Step (e) involves the translation of the procedure, as formulated in the preceding stage, into the detailed language of the computer. However, it is misleading to imply that step (d) has been completed before this translation begins. With all but the simplest programs, the clarification of the problem statement, the choice of a method of solution, and the adaptation of this method, all are subject to continual modification as the detailed coding, the debugging and the actual running proceed.

The amount of effort involved in carrying out the above steps depends on whether the program is to be run once, or used repetitively. The distinction is important for proper use of computers, and no programmer can do an efficient job unless he knows whether or not his program is a "one-shot" effort. In view of the frequency with which programs have to be redone after the first running, however, it seems desirable to undertake every job initially as though the program would be used only once, deferring any polishing until it is clear that the results obtained from the actual run are satisfactory.

CONFIDENTIAL

Types of Autoprogramming Systems

From the point of view of autoprogramming, four tasks may be isolated from the process which has been discussed:

(a) Clarification of the problem to be solved.

(b) Choice of a general method of solution.

(c) Elaboration of this method into a procedure suitable for machine computation.

(d) Conversion of the statement of procedure into machine instructions.

As has been emphasized, these tasks are not in any simple correspondence with the stages described, because they normally go on during the whole process of program preparation.

Autoprogramming systems can be distinguished according to the particular one of the four tasks in which their influence is first felt. The least ambitious systems affect only the task of translating the detailed procedure into computer language. More intricate systems take over much of the job of elaborating the general method of solution into a detailed procedure; we will refer to these as standard language systems. Finally, the most advanced systems undertake to choose among methods of solution that which is best adapted to handle the problem as stated; these will be called standard problem systems.

The more elaborate the system, the less flexibility there is with respect to the format of problem it will accept. Of course, this is characteristic of all automation. However, in eliminating flexibility, two major advantages are gained. Problem formats can use terminology

which is familiar to problem originators. Also, the format associated with the use of this terminology can compel the originator to state an internally consistent, complete and meaningful problem, something which he can ordinarily succeed in doing only after gaining considerable experience with computers. In view of the fact that a very large part of the elapsed time in producing a successful program is traceable to work on specifying what the problem is, there is strong reason to believe that systems which do not impose such discipline on the problem originator may yield a disappointing reduction in elapsed time.

What is required is a standardized format which is not unduly restrictive of the problems that can be handled, even though it completely restricts the terminology in which the problem is stated. In order to achieve this result, a great deal of study of Agency problems will be necessary. However, this effort will directly contribute to one of the major objectives of HARVEST: systematizing Agency operations for greater efficiency.

## Meaning of Problem Standardization

To clarify the sense in which the term "standard problem" is used, several examples will be given. Consider the problem of deciphering a large amount of text which has been enciphered by a generated key, the starting point of each message being known. Despite the frequency with which this problem has to be solved, there is a strong tendency to treat it from scratch at each occurrence. However, under a standard problem system, each new occurrence would be handled by a fixed procedure, with at most the addition of some new subroutines.

CONFIDENTIAL

The cipher text would presumably already exist on punched cards or tape, in a standardized format and with a self-contained description of that format. The deciphering procedure would contain a series of subroutines with the capability to do all of the following as appropriate:

(a) analyze the specifications of the problem to decide on the method of decipherment;

(b) convert the cipher text from any standard format to the form appropriate for the decipherment routine;

(c) generate the key and combine with cipher text to produce deciphered text;

(d) use a "plain text" recognition test to detect going out of phase, incorrect starting point, or other errors;

(e) edit the deciphered text into standard format for printing, statistical analysis or other processing.

The information provided by the user would include:

(a) a statement in standardized form that the problem was to decipher text by a generated key;

(b) an identification of the text, which would be used in locating it in the punched card or tape files and in checking that the correct text was picked up;

(c) a standardized description of the method of key generation (which could be simply a reference to an exciting sub - routine, together with the data it requires);

(d) an indication of whether an error test is needed, and if it is, information about the test to be used;

(e) a choice among available printing formats;

and (f) an indication about additional processing desired.

Some of this information might well come from the output of a preceding stage, also specified as a standard problem. The further analysis, if any, would again be specified in such a form. Of course, it would usually be desirable for a series of problems to be done in a single run, uninterrupted as long as the check points are safely passed.

Note that there is no need to have the system provide one or more subroutines to handle each of the five steps mentioned earlier. In fact, if the procedures for carrying out these steps are expressed in a standard language system, then the subroutines can be compiled as needed out of simpler elements. This remark emphasizes that standard problem and standard language approaches are not alternatives, with only one to be drawn. Both can and should be used with HARVEST, provided that they are designed so as to work properly together.

As a second example, consider the problem of recovering a "pattern" by separating the m frequency counts on n symbols into two categories according to the parent population from which each was drawn. Since statistical tests are rarely final, there will normally be a secondary test which will be used to select the

CONFIDENTIAL

pattern out of several competing possiblities based on the statistical procedure.

Here there are several possible procedures. A crude statistical test may be used, or a lengthier but more refined one may be better, because the result may be better correlated with that expected on the secondary test. As a third possibility, the statistical test may be rejected altogether, and the secondary test run instead on the $2^m$ possible patterns.

Under a standard problem system, the decision about which procedure to use would be made by the computer, based on such information as size of the frequency counts, size of m , and time taken by the subroutine for carrying out the secondary test. The ultimate consumer need not know even after the results are back which procedure was actually used.

An important consequence of using the standard problem approach is that the customer gives up his freedom to decide about procedures, for the classes of problem to which the approach is applicable. Two remarks are pertinent, however. In the first place, this freedom may be largely illusory, since few people are sufficiently well versed in both programming and subject areas to make final decisions about procedures (and even then the computer sometimes compels a revision! ) Besides, it is in relatively routine and familiar problems that standardization will be most common, and there should not be strong insistence on

making procedural decisions for such problems.

Disadvantages and advantages of standard problem systems

There are two costs associated with the use of the standard problem approach. In the first place, there is a considerable capital investment in Guilding such a system, and in modifying or improving it as experience with it grows. Secondly, standard problem systems do not take advantage of the special features associated with the different occurrences of a given type of problem, so that many of these occurrences will not follow the theoretically most efficient procedure.

These disadvantages do not appear serious. Granting that a great deal of effort is necessary in order to produce a general autoprogramming system, it is likely that more effort would be required to program individually the problems that the system will handle. Also, in producing the system, the investment contributes to a consolidation of the technical progress that is being made in cryptanalysis and other phases of the Agency's work.

The objection that general procedures are inefficient is valid primarily for very large problems, and in fact such problems would still be specially programmed even with general procedures available, provided that the gain in running time made this worthwhile. However, such large problems would not be efficiently programmed from scratch, and even here the initial tests of procedure would make use of the general

system in order to save on elapsed time.

The major advantages of the standard problem approach are in greatly cutting the time to get results, and in its contribution to greater overall efficiency in the applications of the computer. As experience with the system grows, more and more problems that come to the computer will be handled without special coding. More important, however, the user in order to take advantage of existing routines will have to provide the information required by these routines. This will cut down on requests for computer work that have not been adequately considered, and will reduce the likelihood that the request should never have been made, or that it should have included more than was asked for, or that the proper procedure to apply was overlooked.

## Conclusion

The principal objectives of this paper have been to propose the use of a standard problem system, and to clarify the relation of such a proposal to other work being done on autoprogramming techniques for HARVEST. It is hoped that discussion of these questions will be stimulated, and that agreement can be reached among the various groups who are concerned with them.