DOCUMENTATION PROBLEMS: ALGOL 60

PETER NAUR, Regmecentralen, Copenhagen, Denmark

In presenting the following notes I wish to make it clear that they express only my private views. Thus although these views naturally have been greatly influenced by the experience I have gained through my participation in the international ALGOL effort they do not represent the agreed opinion of any group and I alone should be held responsible for any misrepresentations.

Administrative Aspects of ALGOL 60

The ALGOL effort was initiated in 1957 when the central European association GAMM (Gesellschaft für Angewandte Mathematik und Mechanik) suggested to ACM that a common effort should be made to establish a machine-independent algorithmic language. The first outcome of this suggestion was the publication of the preliminary Report on the Algorithmic Language ALGOL by the end of 1958 [1]. The language described in this report is sometimes referred to as "ALGOL 58". A more definitive version of ALGOL was described in the Report on the Algorithmic Language ALGOL 60, published in 1960 [2]. The authors of this report represented the following organizations: Association for Computing Machinery, Association Française de Calcul, British Computer Society, Gesellschaft für Angewandte Mathematik und Mechanik, and Nederlands Rekenmachine Genootschap. A supplement of corrections to the ALGOL 60 report was issued during 1962 [3] and the International Federation for Information Processing (IFIP) adopted the revised report to be an official IFIP publication in August 1962. About the same time the newly formed IFIP Working Group, which is now the official maintanance body for ALGOL, held its first meeting.

No official roster of existing implementations of ALGOL 60 exists. The following list is based mainly on information collected during February 1962 [3] and is almost certainly incomplete. Included are such implementations which are claimed by their authors to be based on true subsets (as distinct from dialects) of ALGOL 60.

Implementations of Subsets of ALGOL 60

Quoted are: name of machine, date of completion, name of institution, of reporter. The list is arranged in order of completion.

Electrologica X1. August 1960. Stichting Mathematisch Centrum, Amsterdam, Holland. E. W. Dijkstra.

ORACLE. January 1961. Oak Ridge National Laboratory, Oak Ridge, Tennessee. M. Feliciano and A. A. Grau.

DERA. January 1961. Institut für Praktische Mathematik, Darmstadt, Germany. W. Börsch-Supan.

ZUSE Z22. July 1961. Institut für Angewandte Mathematik, Mainz, Germany. F. L. Bauer.

PERM. September 1961. Rechenzentrum der Technischen Hochschule, Munich, Germany. G. Seegmüller.

DASK, October 1961. Regnecentralen, Copenhagen, Denmark. P. Naur.

CDC 1604. November 1961. Institute for Defense Analysis, Princeton, New Jersey. E. T. Irons.

FACIT EDB 2. November 1961. Facit Electronics AB, Gothenburg, Sweden. I. Dahlstrand.

DEUCE Mk 2A. November 1961. The English Electric Co. Ltd., Whetstone, England. B. Randell.

Siemens 2002. December 1961. Mathematisches Institut, Mainz, Germany. K. Samelson.

IBM 7070. December 1961. Duke University, Durham, North Carolina. T. Gallie and B. Balch.

ER 56. February 1962. Standard Electric Lorenz AG, Stuttgart-Zuffenhausen, Germany. A. Wilhelmy.

SMIL. March 1962. Lunds University, Lund, Sweden, T. Ekman.

National Elliott 803. March 1962. Elliot Bros., London, England. C. A. R. Hoare.

UNIVAC 1105. March 1962. Armour Research Foundation, Chicago, Illinois. R. W. Floyd.

GIER. August 1962. Regnecentralen, Copenhagen, Denmark. P. Naur.

Technical Aspects of ALGOL 60

The language was designed to be a vehicle for expressing the processes of scientific and engineering calculations and of numerical analysis. As evidenced by the contributions to the Algorithms section of the present journal, the actual use of it covers a rather wider field, including administrative, combinatorial and general data processing problems. In designing the language, equal stress was placed on man-to-man and man-to-machine communication.

The features of the language may be briefly described as follows: It is an algebraic language including full arithmetic and Boolean (logical) expressions. The operands may be declared to be of either real, integer or Boolean type. Subscripted variables with any number of subscripts are included. The size of arrays of subscripted variables may vary dynamically during the run of the program. Expressions may include nesting of parentheses to any depth and there are no restrictions on the expressions written as subscripts. Also, a free mixture of integer and real type is permitted. Names of variables and other entities in programs may be chosen freely by the programmer and may have any number of characters. The language includes facilities for dividing the program into a hierarchy of nested blocks, the extent to which names are shared by enclosing blocks being under the control of the programer. Powerful process-defining facilities, so-called procedures, are included. These will enable the programmer to define new functions and operators (summation, quadrature, etc.). Procedures may be defined and used recursively.

Defining Description of ALGOL 60

The fact that the official description of ALGOL 60 [2] is the defining description of the language, and not a description of some independent entity (such as an existing system on a machine or ideas in someone's mind), has had a profound influence on the choice of the form of this document. Clearly, to work as a defining description it must be complete and unambiguous. However, one additional quality becomes almost indispensable: brevity. The chief argument for insisting on a brief document, apart from economy of sheer physical effort, is that trying to say the same thing twice is almost bound to lead to a contradiction. Thus in order to avoid ambiguities we must be brief. This desire for brevity is also entirely consistent with the desire, strong among the authors of the ALGOL 60, for producing a language based on few, general rules.

The task of writing the ALGOL 60 report was facilitated greatly by the preparations made during the preceding two years. In particular, the invention by Backus [4] of a formal notation for describing most of the syntax of the language proved to be a great help in preparing a complete description. The successful use of this formal notation for describing a language designed for actual communication (and not just as an object of study) must be regarded the most significant contribution of the ALGOL effort to the problem of documentation.

In view of the above primary considerations it should be clear that a number of secondary considerations had to be neglected. Thus it is obvious that a defining description attempting to meet the above requirements cannot be expected to serve as a well-balanced introduction to the language. There will be a need for introductory texts and further illustrations of the use of the language. However, it was felt that provided the defining report would indeed meet its primary objectives the preparation of such texts could be left to any individual initiative. Also, the readability of the defining report could only be considered where it was clear that no trading of primary qualities was involved. As one such concession to readability, the sparing use of abbreviations in the ALGOL 60 report can be mentioned.

The successes and failures of the documentation effort embodied in the ALGOL 60 report may be assessed quite accurately from an analysis of the corrections to the report which were published in the supplement [3]. The fact that these corrections, which are based on points for discussion raised by workers all over the world over a period of two years, run to less than 4 pages (to be compared with the 30 pages which constitute the actual documentation part of the report) shows that the general form of the report has served its purpose well. Looking at the corrections themselves we find that the largest group, by number, consists of trivialities such as misprints and unhappy turns of phrase. A group of about the same number of corrections is concerned with incomplete specifications. It is interesting to note that both incomplete textual explanations and incomplete formal syntax are represented. Thus although the Backus notation certainly is a tremendous help in documentation it provides no safety against errors. The remaining corrections are concerned with the removal of ambiguities which are caused by contradictions within the report. Again we find that the syntax is responsible for about as many errors as the descriptive text.

Secondary Documentation of ALGOL 60

It is to be expected that every group which wishes to make serious use of ALGOL 60, in particular compiler-writing groups, will need additional documentation. This will be of two kinds: (a) additional defining descriptions specifying local restrictions and standard procedures, in particular input and output procedures; (b) introductory texts.

This activity has been going on for several years in many centers. However, the experience gained in this way can be described only by those involved and I record here some of the experiences only of my own group.

When the ALGOL 60 report became available our first problem was to teach the language to a group of people already experienced in programming. We felt that this group ought to be familiar with the defining report and I therefore produced a Course of ALGOL 60 Programming [5], which is really only a guide to the reading of the ALGOL 60 report with problems and notes. The idea was that once the students had worked through the ALGOL 60 report in the manner suggested in this course they would be in a position to use the report as their standard reference, forgetting about the course itself. This approach was used at several occasions during 1960 with good success. It turned out that with the help of the extra guidance given in the course even quite inexperienced students could be made familiar with the approach taken in the ALGOL 60 report within a few days.

This development was parallelled with the writing of a report defining that subset of ALGOL 60 which can be processed in our computers ("A Manual of the DASK ALGOL Language" [6]). Since this report is again a defining one, we chose the same style as the one used in the ALGOL 60 report, with extensive use of the Backus notation.

When the teaching of the language was taken over by other user groups having more specialized interests the method of direct reference to the defining reports was abandoned, however, and in fact three different self-contained elementary introductions to ALGOL 60 in Danish have been prepared by such groups. This has raised no compatibility problems because all the introductory texts have been based on the specifications given in the defining reports.

Conclusions

In trying to draw conclusions from the experience I have so far had in documenting programming languages I would like, first of all, to stress the fact that describing a language is not essentially different from any other kind of report preparation, which again is not essentially different from preparing any kind of written communication. To succeed in this task requires a great deal of effort and practise. However, like any other skill, that of writing may be developed systematically and I would therefore strongly urge the reporter on programming languages to consult some of the literature on the subject. As a guide to style I would recommend Strunk and White, *The Elements of Style* [Macmillan, 1959]. Advice on technical writing may be found in, e.g. Nelson, *Writing the Technical Report* [McGraw Hill, 3rd ed. 1952]. A more comprehensive list of references on writing may be found in Grier Parke, *Guide to the Literature of Mathematics and Physics* [Dover, 1958]. In stressing this aspect I would like to point out that notwithstanding the progress in constructing mechanical metalanguages the natural languages remain our ultimate metalanguage. They should therefore receive at least as much attention as any auxiliary notation.

As my second point, I believe that the distinction between defining reports and secondary reports is a sound one which should be observed in language documentation. This idea was forced on those who designed ALGOL 60 and will force itself on anybody who really wishes to try to define his language.

As my third point (in this order), I recommend the use of suitable special metalanguages as a part of the defining reports. The Backus notation is probably as good as anything we have at present, but it still leaves a great deal to be desired even when used with ALGOL 60 (many syntactic rules cannot be expressed in it). The language reporter should keep abreast of the development in this area and should be prepared to invent and use such tools wherever the special metalanguage is easier to work with than natural language.

REFERENCES

1. Report on the Algorithmic Language ALGOL, by the ACM Committee on Programming Languages and the GAMM Committee on Programming (A. J. PERLIS and K. SAMELSON, Eds.). *Numer. Math. 1* (1959), 41–60; *Comm. ACM 1*, 12 (1958), 8–22.

2. BACKUS, J. W., ET AL. Report on the Algorithmic Language ALGOL 60 (P. NAUR, Ed.). *Numer. Math. 2* (1960), 106–136; *Acta Polytech. Scandinavica: Math. Comput. Mach.* Ser. no. 5 (1960); *Comm. ACM 3*, 5 (1960), 299–314.
3. ALGOL Bulletin no. 15, June 1962. Available on request from Regnecentralen, gl. Carlsbergvej 2, Copenhagen Valby, Denmark.
4. BACKUS, J. W. The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference. In *Proceedings of the International Conference on Information Processing*, Unesco, Paris, June 15–20, 1959. R. Oldenbourg, München; Butterworths, London; 1960.
5. NAUR, P. A course of ALGOL 60 programming. Available from Regnecentralen (see [3] above).
6. JENSEN, J., JENSEN, T., MONDRUP, P., NAUR, P. A manual of the DASK ALGOL language, 2d ed. 1961. Available from Regnecentralen (see [3] above).

COBOL

JOSEPH F. CUNNINGHAM,* Vice Chairman, Executive Committee, Conference on Data Systems Languages

Origin of Language Development

Prodded in part by a recognition of the tremendous programming investment within the Defense Department and in part by the suggestion that a common language would result only if an active sponsor supported it, the Defense Department played host in May, 1959, to a group of approximately 70 people representing all phases of automatic data processing to discuss the possibilities of adopting a common language.

This meeting was the formation of the Conference on Data Systems Languages. After two days of discussion it summarized the conclusions as follows: (a) A common language bridging all equipment was needed. (b) The time was now appropriate for development or adoption of such a language. (c) A plan should be developed for proceeding.

A Short Range Task Force Committee was organized for the express purpose of evaluating all existing languages and nominating one, or in the event that nontechnical consideration might so dictate, to select the best features from all languages and form them into a language which would be *"problem-oriented and machine-independent."* This charter, or more appropriately, charge, would result in a source program language which could be accepted by the users and manufacturers, and when accepted, solve the continuing problem of lack of compatibility across equipments.

The Short Range Task Force completed specifications for a language called COBOL (Common Business Oriented Language) by January, 1960. It was edited by an independent group under the chairmanship of Charles Phillips, and it was forwarded to the Government Printing Office as a Report to the Department of Defense.

The Executive Committee and the Short Range Committee recognized that the practical utility of the language and its definitions would be determined in phases, as follows:

a. *When Developing Compilers.* As each manufacturer or compiler developer established the mechanism whereby object programs would be created, the ambiguities, omissions and technical deficiencies would become apparent. To assure compatibility among source COBOL programs and the various object programs necessitated the standardization of interpretation and elimination of ambiguities. During this period the ability to develop a compiler which would permit translation of the language would become apparent.

b. *When Compiling Object Programs.* At this stage, as the user compiles object programs, additional interpretations, clarifications and ambiguities would be uncovered. Since this was the "proving ground," the added factor of compiler efficiency, both in compilation and of object program, would add a level of complexity. There is valid reasoning to accept the premise that object program efficiency and compiling efficiency are indicative of efficiency of compiler development and not necessarily the base language.

c. *When Running.* As the user, employing different equipment on the same source program, attempts performance on more than one equipment, the consistency of interpretation of the source language through the compiler into object program will be determined by comparison of the end results. At this point comparison is possible, proving not only the efficiency of the compiler and the particular hardware complex, but also the interpretation of the specifications for the source language. COBOL 60 was implemented by two manufacturers—RCA and UNIVAC. Both manufacturers on the same day in Philadelphia and Camden, using programs written by United States Steel and the General Services Administration, demonstrated the compiling of these programs using COBOL 60.

Since the Short Range Task Force concluded its activity with the development of COBOL it was then dissolved and the Executive Committee established the Maintenance Committee, consisting of a "Users Group" and a "Manufacturers Group." It was proposed that any changes resulting from the aforementioned conditions would be brought to the attention of the Maintenance Committee through either group, and thereby consistent interpretations would, hopefully, be achieved. The experience of the first implementers and the increasing knowledge of languages and compiler development resulted in the updating of the COBOL language and the publication, in 1961, of the second report, which has become known as COBOL 61.[1]

COBOL is a subset of normal English suitable for expressing the solution to business data processing problems. It is divided into 4 divisions:

(1) PROCEDURE DIVISION, which contains the procedures specifying how the data is to be processed;

(2) DATA DIVISION, which contains a description of the data being processed;

(3) ENVIRONMENT DIVISION, which contains a description of the equipment being used in the processing;

(4) IDENTIFICATION DIVISION, which contains the program identification.

---

* Department of the Air Force, Washington, D. C.

[1] COBOL-1961—Revised Specifications for a Common Business Oriented Language. U. S. Government Printing Office, Washington, D. C., 1961 (0-598941).