PAPER: THE EUROPEAN SIDE OF THE LAST PHASE OF THE DEVELOPMENT OF ALGOL 60

Peter Naur

Copenhagen University
Copenhagen, Denmark

Introduction

In preparing this account of some of the developments leading to ALGOL 60, I have primarily sought to present such relevant information that is readily available to myself, but not otherwise accessible or well known. In addition I have tried to answer the specific questions formulated by the organizers of the Conference on the History of Programming Languages. The notes fall in three freely intermixed groups: those that relate to existing documents; those that reflect my own reasoning as a participant in the development, and those that try to answer the organizers' questions. Where the proper support has been lacking, I have left open holes in the presentation. Otherwise I have tried to be specific, giving names, dates, and reasons, as far as I have them available.

While this manner of presentation seems to me the best way to support more penetrating historical research into the period, it is also bound to give rise to controversy. First, it is openly one-sided, in the manner that any autobiography is. Further, by being specific, the presentation will offer sensitive points of attack to those who see the same development from a different angle and with different eyes. When these circumstances are combined with real events that involved differences of opinion and interest, strong reactions are bound to be provoked. This emerged clearly in the remarks from the reviewers of the draft of this paper.

In dealing with the reviewers' remarks I have decided to a large extent to leave my original formulation unchanged and to include the remarks in appendices to the paper. This seems to be the most consistent way to extend the scope of the account so as to include the views of others.

It should be made clear that my account is essentially incomplete in that the important European development that preceded the Zurich report is only briefly sketched in the quotation from H. Rutishauser's book given in Section 1.1 (cf. Appendix 5, Bauer's Point 1.1). My original plan was to include a detailed analysis of the documents from the time. However, when the original documents came into my hand, I had also seen the reactions of my colleagues to the first draft of my account. These latter made me realize that I would be the wrong person to report on the events happening before February 1959 when I joined the effort. Thus, although with regret, I decided to leave the account in its present incomplete state.

I wish to thank the language coordinator, David Gries, and the reviewers, F. L. Bauer, B. A. Galler, J. Sammet, and J. H. Wegstein, for their many useful notes on the draft. I am particularly grateful to Mike Woodger for his generous help in placing information and original documents at my disposal and for his encouragement. I also wish to thank P. Läuchli and K. Samelson for lending me original documents. The notes in Appendices 4, 5, 6, and 7, appear by permission from their authors.

My participation in the History of Programming Languages Conference is supported by Statens teknisk-videnskabelige forskningsråd, Copenhagen.

1. Background

1.1. The Roots and an Overview (Quotation from H. Rutishauser)

In the present account of the development of ALGOL 60, the center of attention will be the last phase in which the present writer took part. What went before that was highly significant, but since I cannot contribute to its description with anything that relates to my firsthand experience I shall cover the early phases with the aid of a quotation of a brief history of the complete development of the language, written by H. Rutishauser, one of the initiators of the language, who died in 1970. The quotation is taken from Rutishauser (1967) (cf. Appendix 5, Bauer's point 2.1). It has been edited to avoid footnotes and to give references to those original documents from the time that are known to me. Rutishauser writes:

§2. Historical Remarks on Algorithmic Languages

The very first attempt to devise an algorithmic language was undertaken in 1948 by K. Zuse (1948–1949). His notation was quite general, but the proposal never attained the consideration it deserved.

In 1951 the present author tried to show that in principle a general purpose computer could translate an algorithmic language into machine code. (Lecture at the GAMM (Gesellschaft für angewandte Mathematik und Mechanik) meeting, Freiburg i Br. March 28–31, 1951. Published in Rutishauser (1952)). However, the algorithmic language proposed in this paper was quite restricted; it allowed only evaluation of simple formulae and automatic loop control (it contained essentially the for-statement of ALGOL 60). Besides that, the translation method was intermixed with the idea of a *stretched program,* which at that time certainly had some merit as a timesaving device (see Rutishauser, 1953) but was not essential for the purpose to be achieved. For these and other reasons this paper did not receive much attention either.

In 1954 Corrado Boehm published a method to translate algebraic formulae into computer notation [Böhm (1954) officially presented at the ETH on July 10, 1952, as a thesis]. He considered neither subscripted variables nor loop control, but his method to break up formulae into machine instructions was at this state a noteworthy step towards the *pushdown methods* described by Samelson and Bauer (1959, 1960). Further early attempts to translate mathematical formulae into machine code were made in 1952 by A. E. Glennie (unpublished report) in England and in 1953 by A. A. Liapunov [cited in the introduction to Ershov (1959)] in Russia.

Thus by 1954 the idea of using the computer for assisting the programmer had been seriously considered in Europe, but apparently none of these early algorithmic languages was ever put to actual use.

The situation was quite different in the USA, where an assembly language epoch preceded the introduction of algorithmic languages. To some extent this may have diverted attention and energy from the latter, but on the other hand it helped to make automatic programming popular in the USA. Thus, when in 1954 Laning and Zierler (Adams and Laning, 1954) presented their algorithmic language—the first one ever actually used—and shortly thereafter the IBM FORTRAN System (IBM, 1954) was announced, the scientific world was prepared for this new concept.

Meanwhile at Darmstadt an international symposium on automatic computing was held in Oct., 1955 [the proceedings of this meeting are collected in Hoffmann and Walther (1956)], where, among other things, algorithmic languages and their translation into machine code were also discussed. Several speakers [mentioned in the proceedings: H. Rutishauser, A. Walther, and J. Heinhold] stressed the need for focusing attention on unification, that is, on *one universal, machine-*

independent algorithmic language to be used by all, rather than to devise several such languages in competition. This became the guiding idea of a working group called the *GAMM Subcommittee for Programming Languages,* which was set up after the Darmstadt meeting in order to design such a universal algorithmic language. [GAMM stands for Gesellschaft für angewandte Mathematik und Mechanik, i.e., Society for applied mathematics and mechanics. The proceedings of the Darmstadt meeting mention the formation of a GAMM committee for programming, chairman J. Heinhold, members H. Rutishauser and H. Unger].

This subcommittee had nearly completed its detailed work in the autumn of 1957, when its members, aware of the many algorithmic languages already in existence, concluded that, rather than present still another such language, they should make an effort towards worldwide unification. [cf. Appendix 5, Bauer's point 2.2]. Consequently, they suggested to Prof. J. W. Carr, then president of the ACM (Association for Computing Machinery), that a joint conference of representatives of the ACM and the GAMM be held in order to fix upon a common algorithmic language. This proposal received vivid interest by the ACM. Indeed, at a conference attended by representatives of the USE, SHARE, and DUO organisations and of the ACM, the conferees had likewise felt that a universal algorithmic language would be very desirable. As a result of this conference, the ACM formed a committee which also worked out a proposal for such a language.

§3. The ALGOL Conferences of 1958, 1960, 1962

At that point, direct contact between the GAMM subcommittee was established through F. L. Bauer in April, 1958, when he presented the GAMM proposal [presumably based on Bauer *et al.* (1958a)] at a Philadelphia meeting of the ACM group. A comparison of the proposals of the ACM and the GAMM indicated many common features. The ACM proposal was based on experience with several successful algorithmic languages. On the other hand, the GAMM subcommittee had worked for a much longer time at their proposal and had from the very beginning the universality of the language in mind.

3.1. ALGOL 58

Both the GAMM and ACM representatives felt that, because of the similarities of their proposals, there was an excellent opportunity for arriving at a unified language. They felt that a joint working session would be very profitable and accordingly arranged for a conference to be attended by four members of the ACM committee and four members of the GAMM subcommittee.

The meeting was held at Zurich, Switzerland, from May 27 until June 2, 1958, and was attended by F. L. Bauer, H. Bottenbruch, H. Rutishauser, and K. Samelson of the GAMM subcommittee, and by J. Backus, C. Katz, A. J. Perlis, and J. H. Wegstein of the ACM committee. (In addition to the members of the conference, the following persons participated in the preparatory work of the committees: GAMM: P. Graeff, P. Läuchli, M. Paul, F. Penzlin; ACM: D. Arden, J. McCarthy, R. Rich, R. Goodman, W. Turanski, S. Rosen, P. Desilets, S. Gorn, H. Huskey, A. Orden, D. C. Evans). It was agreed that the contents of the two proposals (Backus *et al.*, 1958; Bauer *et al.*, 1958b) should form the agenda of the meeting and the following objectives were agreed upon:

(a) The new language should be as close as possible to standard mathematical notation and be readable with little further explanation.

(b) It should be possible to use it for the description of numerical processes in publications.

(c) The new language should be readily translatable into machine code by the machine itself.

At this conference [unpublished proceedings: Zurich Conference (1958)] it was soon felt that the discrepancies between the notations used in publications on the one hand and the characters available on input/output mechanisms for computers on the other hand were a serious hindrance and might virtually prevent agreement upon a universal algorithmic language. It was therefore decided [cf. Appendix 4, Wegstein's point 1] to disregard printing usage and properties of input/output mechanisms and to focus attention upon an abstract representation (in the sense of a defining standard), called a *reference language,* from which appropriate *publication and hardware* languages might be derived later as isomorphic descendants of the reference language. (For the relation between reference, publication and hardware language, see Backus *et al.* (1959). It should be recognized that experience has shown that ALGOL programs may well be published in the reference language, and therefore extra publication languages are in fact unnecessary. On the other hand, hardware languages have proved necessary to such an extent that it was decided to standardize a few carefully selected hardware representation of the ALGOL symbols.) The notion was, therefore, that reference, publication and hardware languages should be three levels of one and the same language; the conference, however, would then discuss only the reference language. Accordingly, the algorithmic language ALGOL [on the origin of this name see Appendix 4, Wegstein's point 2 and Appendix 5, Bauer's point 2.2a], which was agreed upon at this conference and published in the ALGOL report (Backus *et al.,* 1959), is defined only on the reference level.

After publication of the ALGOL report (Backus *et al.,* 1959) much interest in the language ALGOL developed. At the initiative of P. Naur an ALGOL Bulletin (1959) was issued which served as a forum for discussing properties of the language and for propagating its use. The Communications of the ACM introduced an algorithm-section, in which numerical processes are described in terms of ALGOL. Elsewhere ALGOL was also used more and more for describing computing processes.

On the other hand it was soon found that certain definitions given in the ALGOL-58 report were either incomplete or even contradictory or otherwise unsatisfactory for the description of numerical processes. As a consequence many proposals were made to remove these defects (most of these proposals have been published in the *Communications of the ACM,* Vol. 2 (1959) and/or in the *ALGOL Bulletin,* No. 7).

3.2. ALGOL 60

In view of the constructive criticism that evolved and the proposals made, it was decided that another international ALGOL conference should take place. Accordingly, the GAMM subcommittee organized a preliminary meeting at Paris in Nov. of 1959, attended by about 50 participants from Western Europe, from which 7 delegates for the final ALGOL conference were selected. The ACM committee likewise selected 7 delegates at a preparatory meeting held in Washington D.C. at the same time. Both the European and the USA delegation made proposals for removing the inconsistencies from the ALGOL report and also for making changes in the language. These proposals took the criticisms as much as possible into consideration.

The conference, held at Paris, Jan. 11–16, 1960, was attended by J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, P. Naur, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, M. Woodger (W. Turanski of the American delegation was fatally injured in an automobile accident just prior to the Paris conference). The proposals worked out prior to the conference again formed the agenda of the meeting, but in addition the conferees had at their disposal a completely new draft report prepared by P. Naur (ALGOL 60 document 5), which served as a basis for discussion during the conference.

From the beginning it was obvious that rather than just adding a few corrections to ALGOL 58, it was necessary to redesign the language from the bottom up. This was done, and accordingly ALGOL 60, as the language emerging from the Paris conference is officially called, was in many respects entirely different from ALGOL 58. It is defined on the reference level in the ALGOL-60 report (Backus *et al.,* 1960) edited by P. Naur. Since publication of the report, ALGOL 58 has become obsolete, but many of its features have been carried over into other algorithmic languages.

Here ends the quotation from Rutishauser.

In accordance with this description ALGOL 60 is the name of a notation for expressing computational processes, irrespective of any particular uses or computer implementations. This meaning of the name will be retained throughout the present account.

1.2. Centers and Individuals

ALGOL 60 was developed as an international, noncommercial, mainly informal, scientific activity of a unique kind, involving a number of computation centers or individuals associated with such centers. The central planning and control included mainly the organization of meetings and some publication. The necessary support was obtained locally and individually from the supporting computation centers or from research grant organizations. Most of the concepts of conventional projects, such as cost estimates and budget control, were never applied to the development activity as a whole. For this reason no meaningful estimate of the total cost can be made.

The organizational nature of the participating centers differed widely. Appendix 1, Table 1, gives the identity and a few notes about the European centers, including the names of the associated individuals. Appendix 1, Table 2, lists all individuals whose names appear somewhere in the available records, and their organization where known.

1.3. The Committees

Organizationally the development leading to ALGOL 60 had four phases:

1. 1955–1958 April: Preparatory work in several committees. In Europe: GAMM committee.
2. 1958 April–1959 January: The combined GAMM–ACM committee develops ALGOL 58, described in the Zurich report (Backus *et al.*, 1959).
3. 1959 February–1959 November: Open discussion of the Zurich report.
4. 1959 December–1960 March: The combined European–American ALGOL 60 committee develops ALGOL 60.

The work was started by some of the individuals who had originated the idea of it, notably H. Rutishauser, F. L. Bauer, and K. Samelson. This led to the formation of the GAMM subcommittee, having F. L. Bauer as chairman. This subcommittee made a deliberate effort to widen the circle of interest and support to suitable organizations in various European countries. This effort became effective from the later part of 1958. Thus a discussion of implementation problems taking place in Mainz, 1958 November 21, was attended by: B. Vauquois, Grenoble; P. J. Landin, London; W. Heise, Copenhagen; M. Woodger, Teddington; R. Basten, Stuttgart; K. Leipold, Munich; M. Poyen, Paris; W. Händler, Backnang; H. Rutishauser, Zurich; E. W. Dijkstra, Amsterdam; G. Ehrling, Stockholm; H. Bottenbruch, Darmstadt; K. Samelson, Mainz; F. L. Bauer, Mainz; Th. Fromme, Bad Hersfeld; M. Paul, Mainz. (Concerning ALGOL 58 implementation in Europe, see appendix 5, Bauer's point 2.2a.)

During the period of open discussion, 1959 February to November, any volunteer had the opportunity to contribute to the discussion channeled through the *ALGOL Bulletin*.

The selection of the European participants in the final ALGOL 60 committee was made in November 1959, during the open meeting taking place in Paris. The selection was proposed by the European participants at the Zurich meeting and endorsed by the full meeting. The idea behind the selection clearly was to make such additions to the earlier committee that would extend the geographical basis, as far as possible. The total number was limited to seven, by agreement with the ACM committee. Since H. Bottenbruch of the earlier committee wished to withdraw from the work the original members H. Rutishauser (Switzerland), F. L. Bauer (Germany), and K. Samelson (Germany), could be supplemented by the following four: P. Naur (Denmark), B. Vauquois (France), A. van Wijngaarden (Netherlands), and M. Woodger (Great Britain). This committee of seven met 1959 December 14–16 in Mainz and joined the ACM committee in the 1960 January 11–16 meeting in Paris. During these meetings there were no observers and the members acted essentially as equal individuals.

1.4. Contributors' Attitudes

The European participants in the work on ALGOL 60 shared, as the common idea, the belief in, and willingness to contribute to, a common programming language. Apart from that the attitudes differed widely from one participant to another. Some participants had worked on the principles of compiler construction, as described in the quotation from Rutishauser given in Section 1.1 above. (See also Appendix 5, Bauer's point 2.3.) There was hardly any clear distinction between language designers, implementors, and users; most participants in the work saw themselves potentially in all three roles, or at least in two of them. This also meant that the design of the language had regard to the problems of implementation. The concrete effects of this are difficult to identify, however. It is probably true to say that there was a general conviction that by limiting the number of mechanisms of the language and by adhering to a clear, systematic structure the problems of implementation would tend to stay within reasonable bounds.

1.5. Time Constraints

The final design of ALGOL 60 was scheduled during the 1959 June UNESCO conference in Paris. It was decided that proposals for the final language were to be collected until November 1959 and that a final design conference should be held in early 1960. The time limits thus imposed probably acted more as a spur than as a constraint on the design work. There is no reason to assume that a less tight schedule would have helped to remove the differences of the designers' opinions concerning the final language. Again, the reasons for the deliberate omissions from the final language, in particular details of input and output mechanisms (cf. Section 2.8 below), were probably only partly the tightness of schedule.

1.6. Discussion Techniques

In the present section on the manner in which the European discussions that led to ALGOL 60 were conducted, I shall give myself free rein in accounting for my motivations and feelings. This mode of presentation may give offense to my own modesty and the reader's sense of propriety. However, it appears to me necessary as a clue to several of the events in the development of ALGOL 60 with which I was mostly concerned. (See also Appendix 6, M. Woodger's note 1.)

As to the objective significance of the discussion techniques, it seems fairly clear that the strong emphasis on rapid exchanges of written formulations had a substantial influence on the development of the language itself. Besides, an account of the forms of discussion will make it clear how the later stages of the shaping of language ideas remains readily accessible from original documents.

Until the Mainz gathering November 1958 and the European ALGOL implementation conference 1959 February in Copenhagen, the European discussions had been concentrated in the closed GAMM committee. At the meeting in February 1959 the subject had been made open for discussion by the appearance of the Zurich report on ALGOL 58 in *Numerische Mathematik.* Much of the discussion at that meeting took the form of questions to the authors of the Zurich report from a wider audience, who had had the report available for a period of a month or two. During this meeting I developed an acute feeling of frustration because of the discussion form. I felt strongly that a verbal discussion was inadequate in settling the complicated questions, and in particular that the answers given to concrete queries were unsatisfactory because the time for their preparation was too short. In response to this feeling I developed the idea of a series of discussion letters, to be the medium for a more satisfactory form of discussion. This idea of an *ALGOL Bulletin,* to be circulated from Regnecentralen, was received positively, first by Niels Ivar Bech, the head of Regnecentralen where I was employed, and then by the group of implementors centered around F. L. Bauer and K. Samelson.

With this support I quickly formulated the rules of operation of the *ALGOL Bulletin,* which were stated in the first issue, of 1959 March 16. Among these rules, which run to two full pages, the most noteworthy are those that stress the clear division of contributions into separate parts, according to subject, each with an appropriate heading and separately numbered. I was much concerned with achieving a division of the debate into manageable units, so that lengthy repetitions of statements of a problem could mostly be replaced by brief reference to the "Zurich report" and to previous discussion.

The *ALGOL Bulletin* met with an active, positive response. A subcommittee on ALGOL at the UNESCO conference in Paris, members E. W. Dijkstra, W. Heise, A. J. Perlis, and K. Samelson, adopted the *ALGOL Bulletin* as the official collector in the eastern hemisphere of suggestions concerning the language (AB 4.1, 1959, June 15–20). Its contents during the period leading up to the period of meetings starting in November 1959 can be summarized as follows:

The ALGOL Bulletin

No.	Date of issue (1959)	Number of pages	Number of separate items of technical discussion
1	March 16	6	3
2	May 5	8	15
3	June 8	6	9
4	August 13	7	12
5	September 28	9	22
6	October 17	2	0
7	November 3	21	48

A major influence on the way ALGOL 60 was eventually described came from the paper given at the 1959 June UNESCO conference in Paris by John Backus (1959). To me personally this paper was in the first instance a disappointing surprise because of the lack of agreement between the language described in it and that of the Zurich report. This indicated that the common understanding among the members of the Zurich committee did not by far include everything that seemed clear enough in the Zurich report. As another disappointment, Backus's report did not contribute much to resolving the more subtle questions already raised by the Zurich report.

All this indicated to me the crying need for integrating the formulation of a precise and complete language description into the actual language development process. I was led to the conviction that the formulation of a clear and complete description was more important than any particular characteristic of the language.

Only in October 1959, during a second phase of studying Backus's Paris report, did I penetrate through his formal, syntactic description. This made me realize that his notation would be a highly valuable tool for the kind of description that I had in mind. However, this still left me with the problem of how to go about making the ALGOL committee actually use this notation and the precise prose formulations that would have to go along with it.

In this situation I seriously considered making an appeal to the members of the ALGOL committee concerning the style of the language description. However, before getting that far I concluded that the matter was too difficult to be solved just by an appeal and an explanation. The only hope was for me to produce samples of what was needed in a form that by its inherent quality would carry the day. In particular, I realized that it would be crucial to my plan that a substantial part of the new language description be written before the final meeting of the ALGOL committee in January 1960.

My situation was decisively improved when, during the November 1959 meeting in Paris, I was made a member of the ALGOL committee. In this way I took part in the meeting of the European part of the committee in Mainz in December 1959. This meeting was almost exclusively concerned with problems that led to the structure of blocks and procedures (cf. Section 2.10 below). The use of Backus's notation could only be mentioned in passing; my recommendation to that effect appears as one of 55 brief notes on revisions of the Zurich report as follows: "15) Change the syntactical description" (ALGOL 60 document 4A); Backus's notation was not used in the proposals from the meeting (ALGOL 60 document 2). (For another view of this see Appendix 5, Bauer's point 2.4.) The decisive action concerning the development of the new style of description was taken during the weeks following the Mainz meeting, when I worked out the results of this meeting according to the notions of language description that I had formed. In order to press the matter forward as much as possible, on January 2 I sent all other committee members a document of seven tightly typed pages, including samples of my proposed descriptions (ALGOL 60 document 3). This document contains the first appearance of my slightly revised form of Backus's notation (cf. Appendix 2; also Appendix 5, Bauer's point 2.6). The main result of this work was the 18-page draft report that I finished on January 9 and took along to the Paris meeting (ALGOL 60 document 5). This was structured in the manner known from the final report on ALGOL 60, and thus includes the partitioning into subsections giving syntax, examples, and semantics.

As was to be expected the meeting of the ALGOL committee in Paris, 1960 January 11–16, was dominated by intense and often heated discussions of the various characteristics of the language. Meetings of the full committee were chaired by Joe Wegstein, with Mike Woodger as secretary. Quite a number of questions were delegated to subcommittees, however. The results of the subcommittees were expressed in reports (ALGOL 60 documents 9, 11, 12, 15–18, 26, 27), while the minutes of the meetings were never made available to the members (see, however, Woodger, 1960).

As to the choice of notation and style of the description of the new language, my preparations were met with a response that was positive beyond my expectations. In fact, on the fourth day of the meeting the committee decided to take over my draft report (ALGOL 60 document 5) as the basis of the discussion of the new language, in preference to the Zurich report; moreover, I was invited to be the editor of the report of the meeting (for another view of this, see Appendix 5, Bauer's point 2.5). In order to support me in this capacity, I was given the option to decide on the most suitable mode of operation of the committee in discussing those parts of the language that were described in reasonably finished drafts. The procedure that I chose for this purpose was as follows. In a first phase the full meeting would read a suitable section of the documentation and have a brief discussion for clarification and insertion of corrections, but not for changes. In a second phase each member, working individually, would develop his proposals for changes in the form of written, concrete rewordings of the existing document. These proposals were collected by me, sorted and numbered according to subject, and typed in a copy for each member (ALGOL 60 document 31). In a third phase the full meeting would go through these proposals and would decide on each of them, usually by voting. This procedure was quite effective. Through the insistence on formulations in the form of wordings to the final report each member was urged to clarify his proposals to himself before proposing them to the full meeting, thus saving much time. A total of 80 proposals, many of them quite short, were formulated and decided upon, and this made the final editing of the bulk of the report of the meeting an easy matter. For myself as editor the procedure had the great advantage that in most matters I could proceed with confidence that the wishes of the committee were expressed clearly and positively.

1.7. Influence from Other Languages

It is probably true to say that the collective knowledge of the total group of participants in the development of ALGOL 60 included all there was to be known about programming languages at the time. However, the knowledge of each individual was mostly far more limited. My own experience stemmed primarily from my work with the EDSAC, Cambridge, England, in 1951 and 1953. This had given me a strong impression of the importance of closed subroutines, including the use of what in retrospect may be described as parameters called by name (Naur, 1977). Of later developments I had a reading acquaintance with R. A. Brooker's Autocode for the Manchester Mark I machine and with the first version of FORTRAN, but I had not tried to use them.

Undoubtedly, for the many participants who became active in the work after late 1958, the language ALGOL 58 as described in the Zurich report was the major influence, since this was the declared basis of the discussion.

1.8. Intended Purpose of ALGOL 60

The declared purpose of the development work on ALGOL has been stated in the quotation from Rutishauser given in Section 1.1. To this two remarks will be made.

First, the intended application area of the language is to some extent implicit in the historical situation. When talking about "standard mathematical notation" and "numerical processes," it is understood that the problem area is what later came to be known as "processes of numerical analysis" or even "scientific computing." The point is that at the time other application fields were only just emerging and not well understood. Scientific computing, on the other hand, could be based on an old tradition of computational methods derived as a part of mathematical analysis. The influence from this area on ALGOL 60 is clear partly from the fact that several of the most active contributors were also active in numerical analysis, and partly from the choice of subject of the sample programs given with the description of the language.

Second, the declared purpose of ALGOL includes potentials for contradictions of a fairly deep kind. These may be seen to lie at the root of at least some of the more persistent conflicts in the development of the language. The conflict arises between some view of what constitutes "standard mathematical notation" and some view of what is required of a language that is to be "readily translatable into machine code." An expression of this conflict is the difference between viewing a function as a closed transformation of a given input to yield a certain output and viewing it as a more or less open computational process that conceivably may have side effects. This conflict had a significant influence on the discussion leading to the procedure concept of ALGOL 60.

The users were often, but by no means exclusively, a direct concern in the design of the language. The declared objectives include that the language should be "readable with little further explanation." In the discussions of the details a recurrent theme is whether a feature or notation will enhance the readability.

2. Rationale of Content of Language

2.1. From Zurich Report to ALGOL 60: Topics and Treatment

The present account will be concerned almost exclusively with the development leading from the language described in the Zurich report (Backus *et al.,* 1959) to the language ALGOL 60 (Backus *et al.,* 1960). One reason for concentrating on this second phase in the development is that, at least in its European part, it includes the development of some of the most striking and original parts of the language, namely, block structure, with dynamic arrays and localization of the scope of entities, and procedures, with name-call of parameters and recursive activations. A further reason is that the development, which depends on many individual contributions, can be traced fairly accurately in the existing documentation. This means that the rationale behind the language can be displayed very concretely.

A subdivision of the description according to subject cannot be carried very far, because of the close interconnection between several of the most important issues. In the following sections several minor questions that can be treated separately are first discussed. This is followed in Section 2.10 by a discussion of the complex of questions related to the large-scale structure of programs.

The account will be based directly on the documents from the time, in particular the *ALGOL Bulletin* (1959). References to this series of documents will be given in the form "AB <section number>, <date>", where the date gives the time of writing.

2.2. Declaration of Named Constants

The discussion of declarations of named constants (or initializations) developed in a curious way. The SAAB group (AB 2.6, 2.7, 1959, Apr. 22) suggested that certain numbers, such as the mathematicians' π and e, should be made a fixed part of the language, and further that the language should give means for initialization of both simple and subscripted variables. The first of these proposals was rejected by a large majority after a brief discussion (AB 4.4, 1959, June).

Declarations of named constants were again recommended by E. W. Dijkstra, W. Heise, A. J. Perlis, and K. Samelson, forming a subcommittee at the UNESCO conference in Paris (AB 4.7, 1959, June 15–20). A question about the meaning of this proposal in case of an attempted assignment of a new value to the constant was posed by the group at Regnecentralen (AB 5.3.2, 1959, Sept. 10). This question was given two different answers by Rutishauser (AB 7.7, 1959, Oct. 21) and by the SAAB group (AB 7.48, 1959, Nov. 2). Otherwise the suggestion to have named constants met with approval. It also appears in the proposals from the American participants in the ALGOL 60 conference (ALGOL 60 document 6).

In spite of the general support of the idea the proposal to include it in the language was rejected on the last day of the meeting of the ALGOL 60 committee.

2.3 Arithmetic Expressions

The question of the meaning of an assignment of a real value to a variable declared integer was brought up by Naur (AB 4.15, 1959, July 1). In reply, Rutishauser stated that implicit rounding is performed (AB 5.10, 1959, Sept. 10). During the final ALGOL 60 conference the question of arithmetic expressions and assignments was discussed in a subcommittee, members: P. Naur, A. J. Perlis (ALGOL 60 document 16). The report of this subcommittee describes all combinations of arithmetic types and operators and identifies four alternatives for the assignment of a real value to an integer declared variable. In the absence of Rutishauser the meeting voted to leave this assignment undefined. Later the same day the issue was reopened by an appeal from Rutishauser: "Since I believe it is a grave error, I propose to accept possibility 2a [implicit rounding] of 2.5 in place of possibility 1 [undefined]." (ALGOL 60 document 31, item $\infty - 1$). This change of an earlier decision was then adopted by the full meeting.

2.4. Inclusion of Additional Types

A proposal to go beyond the basic types, Boolean, integer, and real, was made by Garwick (AB 4.16, 1959, Aug. 6). He suggested that the use of the Boolean operators with operands of type integer should be understood as manipulation of bit patterns. The suggestion was opposed by the Mailüfterl group (AB 5.11.1, 1959, Sept. 7) and by Rutishauser (AB 5.11.2, 1959, Sept. 10), whose objection was that with these operations the machine independence of ALGOL would be lost.

A more ambitious proposal for inclusion of additional types was made by A. van Wijngaarden and E. W. Dijkstra (AB 7.35, 1959, Oct.). They mentioned, "e.g., complex numbers, vectors, matrices, lists (sets) of quantities." The proposal is supported by a few, brief remarks about how the operators could be applicable "in the conventional meaning." The proposal does not seem to have stirred much attention during the discussions of the following months; the necessary understanding of user needs and implementation methods was lacking.

At the final ALGOL 60 conference J. Backus thought it was important to include string handling capabilities but "didn't try very hard to get together a good string proposal" (Backus, 1979). As a result string facilities were included in the final language only in the form of string constants used as parameters of procedure calls.

2.5. Simple Control Statements

A significant simplification in the area of simple execution control statements was achieved mostly through the medium of the *ALGOL Bulletin*. The start was made by Bring and Ehrling who suggested that the STOP of the Zurich report should be interpreted as a pause waiting for a restart signal, and that there was a need for DUMMY statement similar to FORTRAN's CONTINUE (AB 2.3 and 2.4., 1959, Apr.).

The new interpretation of the STOP statement met with some support but was opposed by Bauer and Samelson (AB 3.2.2, 1959, June 1), who suggested that the function of the proposed statement was fully covered by normal input functions.

As the next step in the discussion of STOP, Samelson suggested that STOP has essentially the same function as RETURN (AB 7.27, 1959, Oct.). Finally, during the ALGOL 60 conference Rutishauser suggested that RETURN is unnecessary (ALGOL 60 document 20; see also section 2.10 item 43 below). Thus both STOP and RETURN were found to be superfluous.

In response to the suggestion for a DUMMY statement Rutishauser, opposing the form of the original suggestion, suggested that the desired effect could be achieved by admitting the blank between two semicolons to act as a statement (AB 3.4, 1959, May 21). This suggestion was met with positive response from many sides.

2.6. The For Statement

The Zurich report described the controlling part of repetitive statements as "'For' statements." M. Woodger, P. Landin, E. W. Dijkstra, and A. van Wijngaarden during the meeting in Copenhagen, 1959 February 27, suggested that these and similar parts of the language should belong to a separate category of the syntactic skeleton, quantifier statements. In the same vein K. Samelson suggested that the proper syntactic class of these parts is that of a clause (AB 7.26, 1959, Oct.).

The Zurich report permitted the repetitions of a for statement to be controlled either by a list of expressions or by a list of step-until elements. E. W. Dijkstra, W. Heise, A. J. Perlis, and K. Samelson, forming a subcommittee at the UNESCO conference in Paris, suggested that control by a list having an aribitrary mixture of the two kinds of elements should be permitted (AB 4.11, 1959, June 15–20).

J. Garwick and H. Rutishauser independently suggested that when the list defined by a step-until element is empty no executions of the controlled statement should take place (AB 7.14 and 7.15, 1959, Oct.).

In the Zurich report the three expressions of a step-until element were separated by parenthesis, with *A* **step** *B* **until** *C* being written *A*(*B*)*C*. This use of parentheses clearly may become confused with their use in function designators. A proposal to change the notation by using apostrophes instead of parentheses was made by the Siemens group (AB 7.29, 1959, Oct. 27). The final form of the for statement was based mostly on American proposals (ALGOL 60 documents 7, 25).

2.7. Conditional Statements

The Zurich report had an if statement and an alternative statement. A proposal to combine these and to admit also conditional expressions was made by K. Samelson (AB 7.25, 1959, Oct.). This agreed well with American proposals (ALGOL 60 document 6) and was adopted in the final language.

2.8. Machine-Dependent Auxiliary Information

The suggestion that programs might include auxiliary information that might serve certain machine-dependent purposes was made by E. W. Dijkstra, W. Heise, A. J. Perlis, and K. Samelson, forming a subcommittee during the UNESCO conference in Paris (AB 4.13, 1959, June 15–20). Proposals about how to understand this suggestion in more detail were made by the group at Regnecentralen (AB 5.8, 1959, Sept. 1, and 7.38, 1959, Oct. 15). Some of these proposals met with positive response but did not find their way to the final report on the language.

2.9. Input and Output

The Zurich report states that input and output operations will be taken care of by means of procedure declarations that are partly expressed in a different language than ALGOL itself. A suggestion for a manner in which input and output might be expressed in the language itself was made by E. W. Dijkstra, W. Heise, A. J. Perlis, and K. Samelson forming a subcommittee at the UNESCO conference in Paris (AB 4.6, 1959, June 15–20). This suggestion was clearly modeled on FORTRAN, and used format statements. This suggestion was incorporated in the draft report used as the basis for the ALGOL 60 conference (ALGOL 60 document 5). However, that conference decided to reject the proposal after hardly any discussion.

2.10. Blocks and Procedures

The present section, which accounts for the development of the intricate problems of blocks and procedures, is divided into 52 subsections, each corresponding to a separate contribution to the events.

1. The Zurich report (Backus *et al.*, 1959). For an understanding of the development during 1959 a fairly detailed knowledge of the relevant parts of the Zurich report is indispensable. In Appendix 3, full quotations from the Zurich report are given. These quotations are briefly and informally summarized below. The quotations and the summaries are similarly divided into sections numbered 1.1 to 1.7.

1.1. A *function* with parameters, e.g., $F(a,b,c)$, yields a single value, obtained through an application of the rules of a corresponding function or procedure declaration to the parameters. Certain functions have fixed meanings, e.g. abs, sign, entier, sqrt, and sin.

1.2. Any sequence of statements may be used to form a *compound statement*, e.g., **begin** A; B; C **end**.

1.3. A *procedure statement*, e.g., $P(a,b) =: (c,d)$, calls for the activation of a corresponding process, defined by a procedure declaration, using a and b as input parameters and c and d as output parameters. Very flexible parameter correspondences are permitted. As an example, an actual parameter of the form $A(p, \quad , \quad)$ may correspond with a formal parameter $F(\quad , \quad)$ since both have two empty positions. Certain procedures may be available without declaration and may perform input and output operations. During the activation of the procedure the formal parameters are replaced by the corresponding actual parameters, replacement being understood as symbol substitution. Return statements are replaced by suitable goto statements.

1.4. *Declarations* may be written anywhere and pertain to the entire program or procedure in which they occur. Their effect is independent of the state of execution of the program.

1.5. In *array declarations* the subscript bounds must be given as integer constants.

1.6. A *function declaration,* e.g, $G(u) := u + 3/h$, defines the rules for obtaining the values of the corresponding functions. The rule must be of the form of a single expression. This may contain the formal parameters and other variables, sometimes known as hidden parameters of the function.

1.7. A *procedure declaration* defines the actions activated by the corresponding procedure statements and functions by means of a series of statements. The heading of the declaration defines the forms of corresponding procedure statements and functions, each having certain inputs, outputs, and exits as parameters. One procedure declaration may define several different procedures and functions, having different names, entry points, and parameter lists. The procedure heading contains declarations concerning parameters. An array declaration concerning a parameter may contain expressions in its subscript bounds. All identifiers used within the statements of the procedure declaration have identity only within the procedure. The statements of the procedure declaration may be expressed in a language other than ALGOL.

Here end the summaries of the parts of the Zurich report.

The Zurich report is dated 1958 October 20 and became widely available in Europe around 1959 February 1. During the meeting in Copenhagen, 1959 February 26–28, requests for clarifications and proposals for modifications were discussed. No official proceedings of this meeting were produced, but the main points of the discussions are identified in a set of notes taken by M. Woodger (1959b). Apart from a question from A. van Wijngaarden concerning the possibility of jumping to a label inside a compound statement, thus avoiding entry through the **begin,** these notes indicate little that relates to the later discussions of blocks and procedures.

The following points 2 to 32 are abstracts of the contributions to the *ALGOL Bulletin* concerning procedures and what later became known as blocks, made before 1959 November 1.

2. A. Bring and G. Ehrling, AB 2.5.1–2.5.2.2, 1959 April. Discussion of array declarations, their possibly variable bounds [*dynamic arrays*], and procedures. Suggests that it is often unnecessary to give array declarations for input and output parameters in the procedure heading. Also that such array declarations may be used to produce a procedure working under more restrictive conditions. Further that declarations of internal arrays in a procedure have their natural place within the procedure itself. Discusses several ways of allocating storage for arrays of varying size within procedures. Suggests the need for a way to identify the open subscript positions of arrays given as actual parameters [*dummy variables*].

3. P. Naur and P. Mondrup, AB 2.5.3 1959 May 1. Refers to 1.5 and shows how the example discussed in 2 may be solved with the aid of arrays declared in the program, avoiding dynamic storage allocation.

AB 2 was distributed on 1959 May 5.

4. G. Ehrling, AB 3.5–3.6, 1959 May 10. Amplifies 2 by a discussion of ways to handle dynamic arrays. Suggests that the actual subscript bounds of an array given as parameter to a procedure may most conveniently be communicated as a hidden parameter. Suggests the need for indicating the maximum storage requirement of the internal arrays of a procedure by means of a special declaration.

5. P. Naur, AB 3.7, 1959 June 8. Discussion of the meaning of the distinction between input and output parameters of procedures. Asks about the effect of supplying the same variable as both input and output parameter in the same call, and suggests ways to avoid undesirable effects of doing so.

AB 3 was distributed on 1959 June 8.

6. E. W. Dijkstra, W. Heise, A. J. Perlis, and K. Samelson forming a subcommittee at the UNESCO conference in Paris, 1959 June 15–20, AB 4.9–4.10. Proposal that in the procedure heading a declaration of a particular form may be included, to indicate that the functions named in the declaration may be used within the procedure without being specified as procedure parameters [*require*]. The proposal is extended with a facility for renaming functions at the same time as they are declared to be accessible within a procedure, *equivalence*.

7. P. Naur, AB 4.14, 1959 July 1. Suggests that the identifier alone should suffice to identify an object. This would imply that differences in the empty parenthesis structures attached to identifiers would not suffice to differentiate objects.

AB 4 was distributed on 1959 August 13.

8. Regnecentralen, AB 5.6.2, 1959 Sept. 10. Describes an ambiguity in the meaning of the renaming mechanism of 6 in the case that the procedure referred to belongs to a set of several procedures defined by the same declaration.

9. Jan V. Garwick, AB 5.9.1, 1959 Aug. 21. Referring to 7, recommends that the same identifier could be used for several kinds of object.

10. H. Rutishauser, AB 5.9.3, 1959 Sept. 10. Referring to 7, stresses the power of the notation of empty subscript positions in referring to vectors extracted from an array. Stresses also the contribution of empty brackets to readability.

11. Regnecentralen, AB 5.9.4, 1959 Sept. 15. In response to 9 and 10, suggests a specific grouping of program objects as the basis for multiple uses of the same identifier.

12. Regnecentralen, AB 5.12, 1959 Sept. 5. Suggests that in array declarations in procedure headings (1.7) the bound expressions may contain only simple formal input variables, not arrays and functions. Reason: to remove unnecessary complications from translators, as illustrated by an example.

13. G. Ehrling, AB 5.13, 1959 Sept. 15. Suggests that procedure declarations may be nested indefinitely and that a procedure declared at a certain level of nesting should be accessible at all interior levels of nesting from that level unless the name of it is redefined. Thus any program may be made into a procedure by enclosing it between **begin** and **end** and adding a procedure heading. Library procedures will be referred to as if declared in a

level surrounding the main program. A special declaration is suggested to make it possible to get access to a library procedure even from a level where the name of it has been used for another function.

14. Regnecentralen, AB 5.15, 1959 Sept. 1. Question: why are switches not permitted as output parameters in procedure statements?

15. Regnecentralen, AB 5.16, 1959 Sept. 1. Proposal that labels and switches appearing as parameters in procedure headings and statements should be distinguished by being followed by colon.

16. Regnecentralen, AB 5.18, 1959 Sept. 26. Referring to 2 and 4, it is again stated that array declarations in procedure headings are superfluous. The question is asked why they were put there in the first place.

AB 5 was distributed on 1959 September 28.

17. H. Rutishauser, AB 7.4.1, 1959 Oct. 21. Referring to 12 it is claimed that the example given to support the suggestion cannot occur.

18. H. Rutishauser, AB 7.6, 1959 Oct. 21. Referring to the renaming proposal of 6 it is shown how the desired effect may be achieved with the aid of the procedure parameter mechanism.

19. H. Rutishauser, AB 7.10, 1959 Oct. 21. The proposal of 6 is reformulated as a "require" declaration that may be put in the procedure heading, for listing names of functions and procedures that are used, but not declared, within the procedure. Names of standard procedures are exempt.

20. Regnecentralen, AB 7.11, 1959 Nov. 1. Question: what should be the form of a procedure entered as input parameter to another procedure? Should a complete, empty bracket structure be included?

21. H. Rutishauser, AB 7.16, 1959 Oct. A discussion of the need for some sort of dynamic arrays, leading to a proposal for a fully dynamic array declaration, to be written just after the **begin** of the compound statement for which it is valid. Some suggestions are made about how the transition to this radical solution may be achieved by way of intermediate steps. The position of other declarations in the program structure is left as an open question.

22. H. Rutishauser, AB 7.17, 1959 Oct. 21. Answer to the query of 16: it is correct that array declarations in procedure headings are mostly superfluous. However, they serve to improve readability for humans.

23. K. Samelson, AB 7.22, 1959 Oct. Proposal: "A declaration is a prefix to a statement. It is valid for, and part of, the statement following it."

24. K. Samelson, AB 7.23, 1959 Oct. Proposal: "In all array declarations, subscript bounds may be arbitrary integer valued expressions."

25. Siemens, AB 7.39, 1959 Oct. 27. Discuss the need for means of structuring programs into blocks that do not need special mechanisms, such as procedure parameters, for communication with the surrounding program. No specific form is proposed.

26. A. van Wijngaarden and E. W. Dijkstra, AB 7.31, 1959 Oct. Proposal: "Declarations . . . pertain to that part of the text which follows the declaration and which may be ended by a contradictory declaration."

27. A. van Wijngaarden and E. W. Dijkstra, AB 7.33, 1959 Oct. " . . . We suggest that the level declaration **new** (I,I, . . .) has the effect that, the named entities have no relationship to identically named entities before in the following text, until the level decla-

ration **old** (I,I, . . .) which attributes to the entities named herein the meaning that they had before. These level declarations may be nested and form the only way to introduce a new meaning to a name . . . ."

28. A. van Wijngaarden and E. W. Dijkstra, AB 7.34, 1959 Oct. Suggest the introduction of the type declaration **dummy** for use with formal variables of function declarations and elsewhere.

29. H. Bottenbruch, AB 7.34, 1959 Oct. 26. Suggests that functions and procedures should be available inside procedure declarations without being communicated as parameters. Further that a procedure defined within another one should have direct access to the identifiers of that other procedure, thus making the "hidden parameters" of functions (cf. 1.6) also available for procedures.

30. H. Bottenbruch, AB 7.37, 1959 Oct. 26. Suggests that declarations should have a dynamic meaning, i.e., become valid when encountered during program execution.

31. Regnecentralen, AB 7.39, 1959 Oct. With reference to 13, the place of library procedures within the hierarchy of procedures is discussed. Several different possibilities are described. In conclusion it is recommended that library procedures are understood to be implicitly declared in a level surrounding the program and that they should be quoted in the heading of each interior level where they are used.

32. Regnecentralen, AB 7.40, 1959 Nov. 2. Referring to 21, the proposal for fully dynamic arrays is opposed on the basis of the claim that the handling of the limited storage capacity will remain with the programmer in any case.

AB 7 was distributed on 1959 November 3.

During the period following 1959 November 1 the discussions took place in meetings: 1959 November 12–14 in Paris, 49 participants; 1959 December 14–16 in Mainz, seven European representatives to the final ALGOL 60 conference; 1960 January 11–16 in Paris, the final ALGOL 60 conference. The following points summarize the reports of these meetings and some related contributions.

33. Subcommittee of Paris meeting 1959 Nov. 12–14, members P. V. Ellis, H. Bottenbruch, E. W. Dijkstra, P. Naur, K. Samelson, B. Vauquois, AB 8.1.2. Concerning dynamic arrays the subcommittee envisaged that these would only be allowed within procedures. It was recommended that all declarations will have to be written at the head of the program or of a procedure. Concerning the range within which a declaration should be valid, the subcommittee recognized the two extremes, by write up and by time succession, and in addition considered permitting dynamic declarations only when the two extremes are coincident. No unanimous recommendation could be given.

34. Subcommittee of Paris meeting 1959 Nov. 12–14, members E. W. Dijkstra, P. J. Landin, P. Naur, AB 8.1.2.2. Referring to 27, the notion of declarations **new** and **old** was recommended for serious consideration. Referring to 6 and 29 it was recommended that the notions of **require** and **equivalence** be extended to all kinds of entities, not only functions and procedures. Detailed proposals for a notation in the form of examples were given.

35. Subcommittee of Paris meeting 1959 Nov. 12–14, members H. Rutishauser, G. Ehrling, M. Woodger, M. Paul, AB 8.1.4. Recommends that procedures given as actual parameters may include n ($\geq k$) parameter positions of which k are empty, where the corresponding formal procedure has k empty parameter positions. Further that such parameters should be furnished with the appropriate bracket-comma-colon structure. It is noted

that this automatically prevents recursive activation through a parameter. Also that the empty parameter positions might be filled by suitably defined dummy variables.

36. Subcommittee of Paris meeting 1959 Nov. 12–14, members H. Bottenbruch, F. L. Bauer, W. L. van der Poel, J. Jensen, M. Woodger, AB 8.1.5. Dealing with the meaning of input and output parameters of procedures, cf. 5, the responsibility of avoiding unintended effects in special cases is generally left to the authors and users of procedures. As an exception, it is stated that a variable used purely as an input parameter will not be changed by the procedure statement.

37. European representatives to the ALGOL 60 conference, meeting in Mainz, 1959 December 14–16 (ALGOL 60 document 2).

37.1. Proposal: Any declaration is a prefix to a possibly compound statement and only valid for that statement. The declarations for a statement and any values of variables attached to them cease to be valid after exit from the statement. Identifiers that are thus declared to be local for a statement may be used independently outside that statement. Identifiers that are not declared to be local for a statement retain any significance that they have in the surrounding level. Labels and switches are local without needing a declaration.

37.2. Proposal: A procedure or function is formed from any statement by adding the heading giving the name, the list of formal input and output parameters, information concerning parameters, optionally the list of global identifiers, and optional information about global identifiers. The information concerning formal parameters and global identifiers is expressed in a notation that gives the complete structure of parameters and subscripts, to any level of nesting, with the aid of dummy identifiers. The dynamic entry to the procedure or function is the first statement following **begin.** At least one operational end must be indicated by a return statement. The value of a function must be assigned to the identifier naming the function.

37.3. Proposal: In a procedure call the actual parameters must have exactly the same nature as the corresponding formal parameters. Any expressions involved are to be evaluated when the procedure call is encountered.

As a personal comment, I believe that at the end of the meeting in Mainz the European representatives felt strongly that the proposal concerning declarations, 37.1, and the structure of procedures, 37.2, was a significant achievement. It managed to combine a variety of proposals into a scheme of great simplicity. There was also a clear feeling that the proposal would appear radical to the American representatives and would need the support of considerable persuasion in order to be accepted.

On the other hand, I felt that an important area had been neglected: the handling of procedure parameters. Neither discussion 36 nor proposal 37.3 contributed to an understanding of how the notions of input and output parameters on the one hand, and the replacement rule of the Zurich report, 1.3, on the other, should be reconciled.

These concerns, together with my feeling of an urgent need for a more precise and complete style of description, are the background of the following documents.

38. Memorandum to the international ALGOL committee from P. Naur, 1960 January 2 (ALGOL 60 document 3).

38.1. The reasoning behind the syntactical structure proposed by the European members.

38.2. The reasoning behind the following characteristics of the European proposal for procedures: only one procedure is defined in any one procedure declaration; partly empty parameter positions in procedure statements are disallowed; the necessity for empty bracket skeletons has been removed; a procedure heading will give complete information concerning the formal parameters; functions and procedures without parameters are possible; all functions are defined through declarations starting with the delimiter **function;** procedure declarations become similar in structure to other compound statements.

38.3. Draft formulation of the section on procedure statements for the final report on the language. The draft has syntactic rules expressed in my slightly modified form of Backus's notation (cf. Appendix 2; also Appendix 5, Bauer's point 2.6) appearing here for the first time, followed by examples and semantic rules. As the dominating part, the semantic rules include detailed explanations of the effects of parameter substitutions. The explanations are expressed in terms of cases, depending on the following descriptors: input or output parameter; kind of parameter (simple, array, function, procedure, label, switch); types of the formal and the actual parameter. A total of 15 different cases are described in terms of evaluations, replacements, and unique, internal variables. [In retrospect these rules may be described as an attempt to supply a complete definition of call by reference.]

39. Draft report on ALGOL 60 by P. Naur, 18 pages, 1960 January 9 (ALGOL 60 document 5). This describes the complete language according to both European and American proposals, as they were known at the time, with minor modifications, except for leaving open the description of if statement, for statement, alternative statement, do statement, and the semantics of procedure declarations. Throughout the description is based on subsections giving syntax, examples, and semantics. Following a proposal from A. Perlis, the international ALGOL 60 committee adopted the report as the basis for the discussions at the meeting in Paris, on 1960 January 14. This implied the adoption of the modified Backus notation for describing the syntax, a question that was never brought up for separate discussion.

During the ALGOL 60 meeting in Paris, 1960 January 11–16, the substance of the European proposals on program structure and the local significance of declarations was adopted with only one significant modification: the introduction of the notion of **own** variables proposed by J. Backus (ALGOL 60 documents 11, 31 item 163 by J. McCarthy). Procedures and their parameters caused more discussion, as clear from the documents summarized below.

40. Subcommittee of the ALGOL 60 meeting, members H. Rutishauser and F. L. Bauer, 1960 Jan. 13 (ALGOL 60 document 12). Proposal for reducing the number of parameter substitution cases given in 38.3. Not adopted.

41. Subcommittee of the ALGOL 60 meeting, members C. Katz, A. van Wijngaarden, M. Woodger, 1960 Jan. 14 (ALGOL 60 document 17). Proposal to abolish the distinction between input and output parameters. Replacement rules corresponding to call by reference are sketched. Adopted.

42. Subcommittee of the ALGOL 60 meeting, members J. Green, A. Perlis, K. Samelson, 1960 Jan. 14 (ALGOL 60 document 18). Proposal for procedure statement having input and output parameters, described in terms of a detailed replacement scheme. Not adopted.

43. H. Rutishauser, 1960 Jan. 14 (ALGOL 60 document 20). Proposal to abolish the return statement, thus making the function declaration of 1.6 unnecessary. Adopted.

44. H. Rutishauser, 1960 Jan. 14 (ALGOL 60 document 22). Proposal for "array functions", thus having expressions of the form $T[k](x)$. Not adopted.

45. Subcommittee of the ALGOL 60 meeting, membership not recorded, but believed to include A. Perlis and A. van Wijngaarden, 1960 Jan. 16 (ALGOL 60 documents 26, 27). Proposal for procedure statements and declarations. Functions are defined by procedure declarations that include assignment of a value to the procedure name. Both procedures and functions without parameters are admitted. No descriptions of the formal parameters are included in the heading. The treatment of each parameter in the call is controlled by a name part in the heading that lists those parameters whose names should be substituted. For all other parameters the value of the actual parameter is used as an initialization. No details of the meaning of name substitution is given. Adopted during the last hour of the meeting, at about 9 p.m.

When the meeting ended it remained my task to make a coherent report out of its agreements. Less than 12 hours after the adoption of 45 I became convinced that it introduced essential ambiguities into the language. This led to a quick exchange of letters among the representatives, as summarized below.

46. P. Naur to the members of the ALGOL 60 committee 1960 Jan. 17 (ALGOL 60 document 201). Claims that the scheme 45 introduces an ambiguity in a case like A (f) where f is declared to be a procedure (functionlike) without parameters; seemingly it is not clear whether one value or the functional procedure will be supplied through the call. As a cure for this trouble it is proposed to reintroduce a distinction between declarations of procedures and functions and to disallow functions without parameters.

47. H. Rutishauser to the members of the ALGOL 60 committee, 1960 Jan. 20 (ALGOL 60 document 204). Points out that 45 also lacks all information about the types of formal parameters. Proposes to reintroduce a limited amount of description about the formal parameters into the procedure heading.

48. A. Perlis to the members of the ALGOL 60 committee, 1960 January 20 (ALGOL 60 document 205). Opposes the proposal of 46. Demonstrates that the ambiguity of 46 does not in fact exist, on the basis of a table showing the effect of call by name and value for 5 kinds of actual parameters: variable, label, array, expression, and procedure. Of particular interest is case 7: the actual parameter X is an expression, corresponding to the formal parameter U called as name:

Action induced by the call:

a) The auxiliary procedure

procedure *$n;* **begin** *$n: $n* := *X* **end** *$n;*

is defined. (See note 2.)

b) Substitute the name *$n* for *X* in the call.

c) same as case 9 [for which the actual parameter *X* is a procedure].

Note 2. *$n* is an internal identifier. [. . .] It might be argued that case 7 not be defined. However, it is treated in such a simple way and the operational intent is so transparent that it seems silly to leave it undefined.

49. M. Woodger to the members of the ALGOL 60 committee, 1960 January 25 (ALGOL 60 document 208). Detailed description of the parameter treatment "as I understand it, based on my notes of 16th January." The rules distinguish between three forms of actual parameters: identifiers, subscripted variables, and other forms of expressions.

Identifiers are substituted, subscripted variables have their subscript expressions evaluated and the subscripted variables with constant subscripts so obtained are substituted, and other forms of expression are evaluated as an initialization of the formal variable. This scheme makes no use of a name list in the procedure heading.

To appreciate my situation as editor of the committee's report at this stage it should be clear that the full committee agreed on the urgency of the task. By now everyone was ready to see even his most cherished ideas abandoned if necessary in order to arrive at a publishable conclusion. It was quite clear that on the question of procedure parameters there was no agreement, and not even a common understanding. The last proposal adopted, 45, seemingly was understood in a coherent manner only by one member, Alan Perlis, as presented in 48. On the other hand, the only coherent alternative, 49, was virtually identical with proposals 38.3 and 40 that had been rejected, and moreover was inconsistent with the last proposal adopted, 45. When I saw that the clarification of 48 did in fact make my argument in 46 invalid, and moreover that the notion of 48 was clear and simple, I was pleased to go along with it. In addition, the proposal of 47 could be incorporated without difficulty. Thus the path was clear for the next documents. (For another view of these matters see Appendix 5, Bauer's point 2.7, the note on this in Appendix 6, Woodger's point 2, and the note in Appendix 8.2).

50. Report on the algorithmic language ALGOL 60, draft for the authors, edited by P. Naur, 1960 February 4 (ALGOL 60 document 213). This is essentially a carefully tidied-up collection of previous agreements. A few matters of detail have been filled in and changed. Thus a syntax for strings has been added; the delimiter **procedure** has been replaced by ⟨type⟩ **procedure** where appropriate; the listing of **name** in the procedure heading has been replaced by a listing of the complementary set, **value.** (See also Appendix 5, Bauer's point 2.8.)

51. P. Naur to the members of the ALGOL 60 committee, 1960 February 4 (ALGOL 60 document 212). Notes on the draft report, 50. Accounts briefly for the origin of the procedure concept, in particular how the proposal of 46 has been dropped. It continues: "In passing I would like to mention a fact which has been pointed out to me by Mr. Jensen: As a direct consequence of our new rules the partly empty arrays and procedures [cf. 1.3 and 35] are, in fact, automatically available. How this comes about in practice you will be able to see from the examples in sections 4.7.2 and 5.4.2 on the procedure Innerproduct."

52. The last substantial change of language concept was the admission of recursive procedure activations. This took place as follows. One of the proposals of the American representatives to the ALGOL 60 Conference was to add the delimiter **recursive** to the language, to be used in the context **recursive function** or **recursive procedure** at the start of declarations (ALGOL 60 document 6). This proposal was rejected by a narrow margin. Then on about 1960 February 10, while the draft 50 was being studied by the members of the committee, I had a telephone call from A. van Wijngaarden, speaking also for E. W. Dijkstra. They pointed to an important lack of definition in the draft report, namely, the meaning, if any, of an occurrence of a procedure identifier inside the body of the declaration other than in the left part of an assignment. They also made it clear that preventing recursive activations through rules of the description would be complicated because of the possibilities of indirect activations through other procedures and their parameters. They proposed to clarify the matter by adding a sentence to section 5.4.4: "Any other occurrence of the procedure identifier within the procedure body denotes activation of the pro-

cedure." I got charmed with the boldness and simplicity of this suggestion and decided to follow it in spite of the risk of subsequent trouble over the question (cf. Appendix 5, Bauer's point 2.8 and the oral presentation).

The reactions from the members of the ALGOL 60 committee arrived in a steady stream during the month of February 1960. Examples of the use of the language in realistic problems, to be included at the end of the report, were produced, procedure Euler by A. van Wijngaarden (ALGOL 60 document 214) and procedure RK by H. Rutishauser (ALGOL 60 document 221). The remarks and questions from H. Rutishauser led to several letters going back and forth between us (ALGOL 60 documents 217, 219–221), in which he continued to make new suggestions for improvement, while finally expressing his satisfaction with the result. Long lists of suggestions and corrections of detail were produced by most members of the committee. The tone of the letters was friendly and helpful but the attitude sceptical. With one exception: at the bottom of a five-page list of typed comments I found the following handwritten addition: "Otherwise we think you did a magnificent job and we'll go along with what you deem the best compromise of the various criticisms. A. J. Perlis (for A. Holt, J. McCarthy)" (ALGOL 60 document 215).

The suggestions from the members of the committee were incorporated in the final version of Report on the Algorithmic Language ALGOL 60, distributed to the approximately 100 addresses of the *ALGOL Bulletin* mailing list on 1960 March 2, and soon thereafter published in several journals (Backus *et al.*, 1960). This attracted much attention and gave rise to discussion and controversy. This, however, is beyond the present story.

3. A Posteriori Evaluation

3.1. Declared Objectives

Curiously, the ALGOL 60 Report (Backus *et al.*, 1960) includes no statement on the objective of the language. However, from its introduction it emerges clearly that it is to be regarded as the completion of the project described in the introduction to the Zurich report (Backus *et al.*, 1959). We are thus justified in taking over the objectives given in that document.

I. The new language should be as close as possible to standard mathematical notation and be readable with little further explanation.
II. It should be possible to use it for the description of computing processes in publications.
III. The new language should be mechanically translatable into machine programs."

Taking these three points in order, it would, I feel, be impossible to claim that ALGOL 60 satisfies I. The part of the notation of ALGOL 60 that is similar to standard mathematical notation is confined to simple arithmetic and Boolean expressions. And even in this respect ALGOL 60 is deficient in so far as the notation is strictly linear and so does not admit the standard, two-dimensional form of mathematical expressions, a fact that has been pointed out by critics of the language all along.

While programming languages have been constructed that match standard mathematical notation of expressions to a much higher degree than does ALGOL 60, what seems to me of more interest is the question of how to handle the many other aspects of computational processes. Considering such aspects one may be led to the view that I cannot be satisfied and thus must be regarded as unreasonable. Such a view was expressed as early as January 1959 by M. Woodger (1959a):

> 3.3 A principal objective was to make the language as close as possible to standard mathematical notation and readable with little further explanation. In this I feel it has failed, and that indeed one cannot expect as much as this. Certainly the algebraic equations occurring as components of the expressions of the language are close to ordinary mathematical notation, but the logical structure of the processes intended, the control of the sequence of operations and of the use of storage requires a great deal of explanation and study before it is properly understood. There is of course no existing standard mathematical notation for this logical control.

To me this is a valid comment as far as objective I is concerned.

Point II of the declared objectives, on the other hand, was fully achieved. ALGOL 60 remains to this day a much used publication language. Within the application area for which it was designed, computation processes developed by numerical analysis, it has proved to be perfectly adequate, and it has even been found useful over a much wider field of problems.

Point III of the declared objectives likewise was fully achieved. The criticism that among the many ALGOL 60 compilers that have been made very few will handle the full ALGOL 60 language without exception to me carries little weight in this context. The parts of the language omitted in implementations have mostly been those that are of doubtful usefulness or are ambiguous in their meaning, or both. Their omission has had very slight influence on the overall usefulness of the language.

In my opinion those declared objectives of the language that were reasonable were achieved to a very high degree.

3.2. Implicit and Derived Objectives

In the present section we shall consider some motives in the work on ALGOL 60 that were not expressly included among the objectives, but that may have had a similar influence on the actual work.

A motive that was directly associated with the ACM ALGOL committee, as described in the introduction to the Zurich report (Backus *et al.*, 1959), was to establish "a single universal computer language". There was quite clearly a hope that ALGOL would become established instead of certain other languages, in particular FORTRAN. This hope was clearly disappointed, and one may ask why. A clarification of this question requires an account of the deliberations and decisions made within certain American organizations. This lies outside the scope of the present account. An analysis of the question has been published by Bemer (1969). Suffice it to say that in retrospect the attempt to establish ALGOL as a replacement for FORTRAN appears as a valiant effort that certainly deserved being tried, but that had only very slight chance of succeeding. Perhaps the simplest demonstration of the validity of this statement is the fact that PL/I, which also was charged with driving out FORTRAN, and had a much greater initial chance of success, likewise failed. For a further discussion of this aspect of programming languages, see Naur (1975).

Apart from the question of the success, in a political sense, of ALGOL 60 regarded as a "universal computer language", there is the question of the universality of the language in a technical sense. I believe it is true to say that although the attitudes of the participants on this question varied considerably there was a quite strong opinion that the scope of the language should remain limited to "computing processes." This we find expressed quite clearly by H. Rutishauser, in a response to a suggestion by Garwick that the language should include operations on bit patterns. Rutishauser writes (AB 5.11.2, 1959, Sept. 10):

> With the proposal [AB] 4.16 we would lose machine-independency of ALGOL, moreover it would mean a generalization of ALGOL towards a data-processing language. This however is—at least in my opinion—beyond the scope of the present ALGOL-group. Of course this does not mean that we cannot work on a universal data-processing language, but such activity should not interfere with our efforts to settle the definitions of ALGOL.

Thus the answer to the charge that ALGOL 60 failed to quality as a universal language is that it was not designed to be one.

Another set of implicit objectives may be said to have been derived from the basic idea of the whole development, that the language should be suitable to be used commonly, across divisions according to computer type, etc. From this idea it follows directly that the language must be described in a machine-independent way, and further that the description must meet the highest standards of clarity, unambiguity, and completeness.

As described already in Section 1.6 above, the problem of description was recognized as a separate issue during the language development. Moreover, the form of description chosen, based on subsections consisting of mutually supporting parts giving a formal syntax and semantical rules expressed concisely in prose, was subsequently recognized as a separate contribution of the language. In this context there has been a tendency to stress one-sidedly the use of a formal syntax language. It should be emphasized that the description of ALGOL 60 is based on three indispensable elements: (1) the formal syntax language; (2) the use in the syntax of full names for nonterminal symbols, providing immediate link between syntax and semantics; and (3) semantics expressed in concise, low-redundancy prose, consciously phrased to approach completeness while reducing inconsistency. In a posteriori evaluation of objectives one may ask to what extent this form of description meets the implied objective of a complete, unambiguous description. In comments concerning this question there has been a tendency to stress the fact that a number of deficiencies were found in the *Report on the ALGOL 60 Language* and that it was deemed necessary to publish the revised report (Backus *et al.*, 1962). Further, that even after this phase deficiencies in the description have been found. In evaluating these facts one may perhaps refer to the independent evidence of D. E. Knuth, who, in a very careful description of the known problems related to the description of the language states (Knuth, 1967): "The following list [of known trouble spots in ALGOL 60] is actually more remarkable for its shortness than its length. A complete critique which goes to the same level of detail would be almost out of the question for other languages comparable to ALGOL 60, since the list would probably be a full order of magnitude longer." Thus one may perhaps be allowed to claim that the objective of completeness and unambiguity in the description was achieved, as far as practically and humanly possible.

As an issue of some importance in the later influence from the language ALGOL 60, one may say that the implicit objective of simplicity and generality of concepts, closely related to clarity and completeness in the description of the language, to a significant extent entailed the need for innovation of language concepts as well. Thus one may regard the concepts of block structure and locality of quantities to be the result of a search for a simple and general solution for dynamic arrays and the need to give access to certain global quantities within procedures. A still more striking example of the influence from the form of description is the manner in which the procedure concept of the Zurich report was rejected in favor of the novel concept of the ALGOL 60 procedures. As described already in item 38.3 of Section 2.10 above, a detailed description of the procedures mechanism was produced in advance of the final meeting on ALGOL 60. As further described in item 40 of the same section, an attempt was made during the meeting to avoid the complexity of that description, without avail. This was a decisive incident in the change of outlook that led to the abolishing of the distinction between input and output parameters of procedures. It was brought about by an indirect attack on that concept, taking the form of a clear description of its meaning. The very length of that description, with the need for a number of subdivisions, became the evidence that the underlying concept was unacceptable.

3.3. Contributions of Language

ALGOL 60 contributed significantly to the field of programming language design in several ways that, although to some extent related, may be described in terms of five issues, as follows:

Contribution 1: A language suitable to be used as a computer programming language was defined independently of any implementation.

Contribution 2: A language suitable for the expression of algorithms in pure man to man communication was established.

Contribution 3: A new style of language description suitable for supporting contributions 1 and 2 was demonstrated.

Contribution 4: The language demonstrates what can be achieved if generality and economy of concept is pursued and combined with precision in description.

Contribution 5: The language included novel ideas, in particular in the areas of blocks and procedures.

Of these five contributions, items 1, 2, and 4 correspond rather directly to the objectives of the language. Item 5 was the concrete result of item 4.

Contribution 3 does not correspond to an objective stated in the Zurich report or the ALGOL 60 report. However, the importance of it was emphasized in the American proposal for the Zurich meeting (Backus *et al.*, 1958). This covers what I personally consider the best point of the language: the brevity, precision, and relative completeness of its description.

Each of the five contributions of the language has had an influence on the subsequent development. Tracing this in detail would be a major undertaking. Here only the barest outlines will be given. Thus as far as implementations of the language are concerned, suffice it to say that large numbers of implementations have been made, by many different implementors and for many different types of computers. The problems of implementation of the language have given rise to a substantial literature of its own.

Again, the use of the language in publications has been substantial. Several journals, including *Communications of the ACM*, *The Computer Journal*, and *BIT*, have published programs expressed in ALGOL 60 as a regular feature. It is probably true to say that for regular publication of general-purpose algorithms ALGOL 60 is the most used language.

As far as the influence from the style of description of ALGOL 60 on later developments is concerned, the most unexpected effect was the suggestion by Irons (1961) that the formal syntactic description of the language could be used to control the compiler for the language directly. This suggestion stimulated a large number of studies of the formal structure of computer languages and of compilation processes controlled directly by language descriptions. Although some of the claims made this approach appear to have been exaggerated, it has undoubtedly contributed substantially to the understanding and methodology of compiler construction.

In an outline of the influence from ALGOL 60 on later languages at least three different directions can be identified, corresponding to three different areas of main emphasis of the development. In the first direction of development the stress is on extensions of the language to cover further data structures and operations, while ALGOL 60 is retained as a subset of the new language. The outstanding example of this line of development is SIMULA (Dahl and Nygaard, 1966). In the second direction of development the stress is on the contributions 3 and 4 of ALGOL 60, leading to a higher degree of formalization and generalization of the language and its description. This is the direction taken in ALGOL 68 (van Wijngaarden *et al.,* 1969). In the third direction of development the size and style of description of ALGOL 60 are retained, but the mechanisms of the language are replaced by more recently developed concepts that to a still higher degree than ALGOL 60 itself combine simplicity and generality. An example of such a development is PASCAL (Wirth, 1971).

ALGOL 60 gave rise to many dialects. In fact, many of the implementations of the language include minor deviations from ALGOL 60 itself and thus are dialects. It is significant that the preliminary description, ALGOL 58 as described in the Zurich Report, inspired several dialects, such as MAD and JOVIAL, that deviate quite strongly from their origin and that early in their development became independent of it. By contrast, only a few of the close dialects of ALGOL 60 became significant as language developments. Thus one may claim that the striving for unification, that was part of the initial idea of the ALGOL development (cf. the introduction to the Zurich report), was to some extent successful.

Influence from ALGOL 60 on computer hardware design appears in a rather pure form in the Burroughs B5000 series of computers. In other computer designs a possible influence may be found in the provision of instructions for handling stacks and procedure calls, but in these cases the influence appears more as a general trend in which both other programming languages, such as LISP, and other application requirements have been active as influences.

3.4. Design Weaknesses

As described in Section 2.2, named constants were omitted from the language although it seemed fairly clear that they would be useful.

Other weak points of the design became the subject of much later discussion. One outstanding case is the concept of own with its lack of an initialization facility and its poor match with the procedure concept. Another case is the meaning of the for clause. This is related to questions of the side-effects and the order of evaluation of expressions. Several areas of difficulty are not just the result of oversights during the design of the language, but reflect differences of opinion of the design group. Thus the discussion of side effects of expressions and function calls reflects disagreement whether expressions should conform more to a mathematical transformation concept or more to an algorithmic processing concept.

The question of weaknesses of the language was taken up by many people soon after the publication of the ALGOL 60 Report, and an ALGOL Maintenance Committee was formed in the USA. The imminent risk of confusion arising from a high frequency of changes of the official specifications was soon realized by the authors, however, and it was decided to exercise severe constraints on changes. As a consequence, only one revision of the language was made, in 1962, touching only some obvious ambiguities while leaving the language virtually unchanged. The following five areas of difficulty were identified, but left for resolution at a later stage: (1) side effects of functions; (2) the call-by-name concept; (3) **own**: static or dynamic; (4) for statement: static or dynamic; and (5) conflict between specification and declaration. Such a resolution was never made within the frame of ALGOL 60.

A thorough discussion of the weaknesses of ALGOL 60 was eventually published by Knuth (1967).

3.5. Main Problems Related to ALGOL 60

Just as in the later phases of the language design, the main problems for the implementers were created by the blocks and procedures. However, as a result of a lively development effort soon after the publication of the language, these problems were understood and solved. In their solution they contributed substantially to a wider understanding of techniques of dynamic storage allocation.

For the users the biggest problems directly related to the language probably have been, partly the proper use of procedure parameters, partly the proper use of semicolon as a statement delimiter.

4. Implications for Current and Future Languages

As already sketched above, ALGOL 60 has had a substantial influence on later language developments, including SIMULA, ALGOL 68, and PASCAL. The reasons for this have been discussed in Section 3.3, "Contributions of Language."

Whether any further influence from the development of ALGOL 60 may still be expected is doubtful, since that development was very closely tied to its historical situation. However, if current and future language designers are looking for so far unattended aspects of the ALGOL 60 development that might again be found effective, the attention should be turned to the use of writing during the development process. Indeed, as described in Section 1.6, Discussion Techniques, the insistence on written contributions, both during the period of open debate and during the final design meeting, appears to me to have been of paramount importance for the intellectual quality of the language.

As for ALGOL 60 itself, it is still being used both for publications and for programming of computers to a substantial extent and will probably retain its influence in this manner for many years to come.

Appendix 1. Centers and Individuals

Cf. Section 1.2.

TABLE 1

European Centers Participating in the Development of ALGOL 60

The centers are ordered alphabetically by city name. Other identifications are included, with references to the city name. The list includes all European centers that appear in the mailing list of the ALGOL Bulletin until early 1960.

Amsterdam, Netherlands. Mathematisch Centrum (public service laboratory): Edsger W. Dijkstra, Aad van Wijngaarden, J. A. Zonneveld.
ASEA, see Västerås.
Backnang, Germany. Telefunken Gmbh (manufacturer). G. Beyer, W. Händler.
Bad Hersfeld, Germany. Zuse KG (manufacturer). Th. Fromme.
Bonn, Germany. Institut für Angewandte Mathematik, Universität Bonn. C. A. Petri.
Borehamwood, England. Elliott Brothers (London) Ltd. P. Shackleton.
Bull, Compagnie, see Paris 1.
Cambridge, England. The University Mathematical Laboratory. D. F. Hartley.
Chalmers Tekniska Högskola, see Göteborg.
Copenhagen, See København.
Darmstadt, Germany. Institut für Praktische Mathematik, Technische Hochschule. Hermann Bottenbruch, Heinz Schappert, Alwin Walther.
Didcot, England. Atomic Energy Research Establishment Harwell, Theoretical Physics Division. Ian C. Pyle.
Dresden, Germany. Funkwerk Dresden. G. Buchholz.
Eidgenössische Technische Hochschule, see Zürich.
Elliott Brothers Ltd., see Borehamwood.
English Electric, see London 1 and Kidsgrove.
Facit, see Stockholm 2.
Göteborg (Gothenburg), Sweden. Chalmers Tekniske Högskola (technical university). Ingmar Dahlstrand, H. Fuhrer.
Grenoble, France. Institut Fourier, Faculté des Sciences (university). Bernard Vauquois.
Hannover, Germany. Institut für Praktische Mathematik und Darstellende Geometrie, Technische Hochschule. Heinz Unger.
Harrow, England. Computer Development Ltd. J. H. Wensley.
Harwell, see Didcot.
Helsinki, Finland. Matematiikkakonekomitea. O. Varho.
Hersfeld, see Bad Hersfeld.
International Computers and Tabulators Ltd., see London 3.
Jena, Germany. Zeiss Werk (manufacturer). Dr. Kammerer.
Kidsgrove, England. English Electric Co. Ltd. F. Wetherfield.
Kjeller, Norway. Norwegian Defense Research Establishment. S. A. Övergaard.
København, Denmark. Regnecentralen (nonprofit computer development laboratory). Willy Heise, Jørn Jensen, Per Mondrup, Peter Naur.
Leidschendam, Netherlands. Dr. Neher Laboratorium. W. L. van der Poel.
Linköping, Sweden. Svenska Aeroplan Aktiebolaget (manufacturer). Börje Langefors, Sven Yngvell.
London 1, England. The English Electric Company Ltd., London Computing Service. Peter Landin.
London 2, England. Ferranti Ltd. George E. Felton, Stanley Gill.
London 3, England. International Computers and Tabulators Ltd. Peter V. Ellis.
London 4, England. University of London Computer Unit. A. R. Edmonds.
Lund, Sweden. Avdelningen för Numerisk Analys, Lunds Universitet. Carl-Erik Fröberg.
Mailüfterl Group, see Wien.
Mainz, Germany. Institut für Angewandte Mathematik, Johannes Gutenberg Universität. Friedrich L. Bauer, Manfred Paul, Klaus Samelson.

TABLE 1 *(Continued)*

Matematikmaskinnämnden, see Stockholm 1.
Mathematisch Centrum, see Amsterdam.
München 1, Germany. Max-Planck-Institut für Astrophysik. K. Hain.
München 2, Germany. Siemens und Halske AG. H. Gumin, Willy Heise, K. Leipold.
München 3, Germany. Technische Hochschule München. F. L. Bauer, P. Graeff, J. Heinhold, M. Paul, F. Penzlin, K. Samelson, Gerhard Seegmüller.
Munich, see München.
Napoli, Italy. Centro Di Calcolo Elettronico.
Norsk Regnesentral, see Oslo.
Oslo, Norway. Norsk Regnesentral. Ole-Johan Dahl, P. Gotaas.
Paris 1, France. Compagnie des Machines Bull. M. Bourgain, P. H. Dreyfus, Mme. Poyen.
Paris 2, France. College de France. M. Poyen.
Paris 3, France. Groupe Drouot. J. M. Barroux.
Paris 4, France. IBM France. Francois Genuys.
Regnecentralen, see København.
SAAB group, see Linköping.
Siemens, see München 2.
Standard Elektrik Lorenz AG, see Stuttgart.
Stockholm 1, Sweden. Matematikmaskinnämnden (public computer development laboratory). A. Bring, Stig Comét, Gunnar Ehrling.
Stockholm 2, Sweden. AB Åtvidabergs Industrier (Facit). A. Olsson, B. Westling.
Stuttgart-Zuffenhausen, Germany. Standard Elektrik Lorenz AG. R. Basten, W. Heydenreich.
Svenska Aeroplan Aktiebolaget (SAAB), see Linköping.
Teddington, England. National Physical Laboratory. Mike Woodger.
Telefunken Gmbh, see Backnang.
Tyresö, Sweden. Bo Nyman ABN AB. Bertile Owe.Uppsala, Sweden.
Uppsala Universitet, Klaus Appel.
Västerås, Sweden. ASEA (manufacturer). P. Gjerløv.
Vienna, see Wien.
Wien, Austria, Mailüfterl Group, Institut für Niederfrequenztechnik, Technische Hochschule. H. Zemanek.
Zürich, Switzerland. Institut für Angewandte Mathematik, Eidgenössische Technische Hochschule. P. Läuchli, Heinz Rutishauser, R. Schwarz.
Zuse KG, see Bad Hersfeld.

TABLE 2

European Individuals Participating in the Development of ALGOL 60

For persons attached to a center described in Table 1 the appropriate city identification is given. Other persons are indicated as "private." The list includes all persons who appear in the European part of the mailing list of the *ALGOL Bulletin* until early 1960. It is known to be incomplete as far as participants in meetings are concerned.

Appel, Klaus: Uppsala
Barroux, J. M.: Paris 3
Basten, R: Stuttgart
Bauer, F. L.: München 3, Mainz
Beyer, G.: Backnang
Böhm, Corrado: Italy (private)
Bottenbruch, H.: Darmstadt
Bourgain, M: Paris 1
Bring, A.: Stockholm 1
Brokate, K. G.: Böblingen, Germany (private)
Buchholz, G.: Dresden
Comét, Stig: Stockholm 1
Dahl, Ole-Johan: Oslo
Dahlstrand, Ingmar: Göteborg
Dijkstra, Edsger W.: Amsterdam
Dreyfus, P. H.: Paris 1
Edmonds, A. R.: London 4
Ehrling, Gunnar: Stockholm 1
Ellis, P. V.: London 3
Felton, George E.: London 2
Fröberg, Carl-Erik: Lund
Fromme, Theodor: Bad Hersfeld
Fuhrer, H.: Göteborg

TABLE 2 *(Continued)*

Garwick, Jan V.: Haag, Holland (private)	Petri, C. A.: Bonn
Genuys, Francois: Paris 4	Poel, W. L. van der: Leidschendam
Gill, Stanley: London 2	Poyen, Mme J.: Paris 1
Gjerløv, P.: Västerås	Poyen, M.: Paris 2
Gotaas, P.: Oslo	Pyle, Ian C.: Didcot
Graeff, P.: München 3	Rutishauser, Heinz: Zürich
Gumin, H.: München 2	Samelson, Klaus: Mainz, München 3
Hain, K.: München 1	Schappert, Heinz: Darmstadt
Händler, W.: Backnang	Schwarz, R.: Zürich
Hartley, D. F.: Cambridge	Seegmüller, Gerhard: München 3
Heinhold, J.: München 3	Shackleton, P.: Borehamwood
Heise, Willy: København, from 1960: München 2	Unger, Heinz: Hannover
Heydenreich, W.: Stuttgart	Varho, O.: Helsinki
Jensen, Jørn: København	Vauquois, Bernard: Grenoble
Kammerer, Dr.: Jena	Ville, J. A.: Paris (private)
Kudielka: Wien (private)	Walther, Alwin, Darmstadt
Landin, Peter: London 1	Wensley, J. H.: Harrow
Langefors, Börje: Linköping	Westling, B.: Stockholm 2
Läuchli, P.: Zürich	Wetherfield, F.: Kidsgrove
Leipold, K.: München 2	Wijngaarden, Aad van: Amsterdam
Mondrup, Per: København	Woodger, Mike: Teddington
Naur, Peter: København	Wynn, Peter: München (private)
Olsson, A.: Stockholm 2	Yngvell, Sven: Linköping
Owe, Bertil: Tyresö	Zemanek, Heinz: Wien
Paul, Manfred: München 3, Mainz	Zonneveld, J. A.: Amsterdam
Penzlin, F.: München 3	Øvergaard, S. A.: Kjeller

Appendix 2. Backus' Syntax Notation and Its Modification

Cf. Section 1.6

The difference between the syntax notation introduced in Backus (1959) and that used in the ALGOL 60 report (Backus *et al.*, 1960) is demonstrated most simply by means of an example. Thus Backus (1959) has the following formula, which shows how a label may be attached to a statement:

```
<basic stmt> : ≡ <b stmt> o̅r̅
            <label> : <b stmt>
```

The corresponding rule in the ALGOL 60 report is as follows:

```
<basic statement> : : =
    <unlabeled basic statement> |
    <label> : <basic statement>
```

Thus the modified notation uses ::= instead of :≡ and | instead of o̅r̅. In addition, in the modified form the designations of the syntactic constituents, such as "basic statement," are chosen to be exactly the same as those used for the same items in describing the semantics, without abbreviation.

The significance of the modification has been discussed by Knuth (1964).

Appendix 3. Quotations from the Zurich Report[†] Relevant to the Development of Blocks and Procedures

Cf. Section 2.10

1.1. From Part II, 3. Expressions

(iv) Functions F represent single numbers (function values), which result through the application of given sets of rules to fixed sets of parameters.

Form: $F \sim I(P, P, \ldots, P)$

where I is an identifier, and $P, P, \ldots, P$ is the ordered list of actual parameters specifying the parameter values for which the function is to be evaluated. A syntactic definition of parameters is given in the sections on *function declarations* and *procedure declarations*. If the function is defined by a *function declaration,* the parameters employed in any use of the function are expressions compatible with the type of variables contained in the corresponding parameter positions in the function declaration heading (cf. *function declaration*). Admissible parameters for functions defined by *procedure declarations* are the same as admissible input parameters of procedures as listed in the section on *procedure statements.*

Identifiers designating functions, just as in the case of variables, may be chosen according to taste. However, certain identifiers should be reserved for the standard functions of analysis. This reserved list should contain

abs (E)	for the modulus (absolute value) of the value of the expression E
sign (E)	for the sign of the value of E
entier (E)	for the largest integer not greater than the value of E
sqrt (E)	for the square root of the value of E
sin (E)	for the sine of the value of E

and so on according to common mathematical notation.

1.2. From Part II, 4. Statements Σ

(b) Strings of one or more statements (see note below) may be combined into a single (compound) statement by enclosing them within the "statement parentheses" **begin** and **end.** Single statements are separated by the statement separator ";".

Form: $\Sigma \sim$ **begin** $\Sigma; \Sigma; \ldots ; \Sigma$ **end**

Note: *Declarations* which may be interspersed between statements have no operational (dynamic) meaning. Therefore, they have no significance in the definition of compound statements.

1.3. From Part II, 4. Statements Σ

(ix) Procedure statements. A procedure serves to initiate (call for) the execution of a *procedure,* that is, a closed, selfcontained process with a fixed ordered set of input and

[†]Backus *et al.* (1959).

output parameters, permanently defined by a *procedure declaration*. (cf. *procedure declaration*).

Form: $\Sigma \sim I(P_i, P_i, \ldots, P_i) =: (P_0, P_0, \ldots, P_0)$

Here I is an identifier which is the name of some procedure, i.e., it appears in the heading of some procedure declaration (cf. *procedure declaration*), $P_i, P_i, \ldots, P_i$ is the ordered list of actual input parameters specifying the input quantities to be processed by the procedure.

The list of actual output parameters $P_0, P_0, \ldots, P_0$, specifies the variables to which the results of the procedure will be assigned, and alternate exits if any. The procedure declaration defining the procedure called contains in its heading a string of symbols identical in form to the procedure statement, and the formal parameters occupying input and output parameter positions there give complete information concerning the admissibility of parameters employed in any procedure call shown by the following replacement rules:

Formal parameters in procedure declaration	Admissible parameters in procedure statement
Input Parameters	
Single identifier (formal variable)	Any expression (compatible with the type of the formal variable)
Array, i.e., subscripted variable with $k(\geq 1)$ empty parameter positions	Array with $n(\geq k)$ parameter positions k of which are empty
Function with k empty parameter positions	Function with $n(\geq k)$ parameter positions k of which are empty
Procedure with k empty parameter positions	Procedure with k empty parameter positions
Parameter occurring in a procedure (added as a primitive to the language, see note below)	Every string of symbols S, which does not contain the symbol "," (comma)
Output parameters	
Single identifier (formal variable)	Simple or subscripted variable
Array (as above for input parameters)	Array (as above for input parameters)
(Formal) label	Label

Note: Within a program certain procedures may be called which are themselves not defined by procedure declarations in the program, e.g., input–output procedures. These procedures may require as parameters quantities *outside* the language, e.g., a string of characters providing input–output format information.

If a parameter is at the same time an input and output parameter, this parameter must obviously meet the requirements of both input and output parameters.

Within a program, a procedure statement causes execution of the procedure called by the statement. The execution, however, is effected as though all formal parameters listed in the procedure declaration heading were replaced, throughout the procedure, by the actual parameters listed, in the corresponding position, in the procedure statement.

This replacement may be considered to be a replacement of every occurence within the procedure of the symbols, or sets of symbols, listed as formal parameters, by the symbols, or sets of symbols, listed as actual parameters in the corresponding positions of the procedure statement, after enclosing in parentheses every expression not enclosed completely in parentheses already.

Furthermore, any return statement is to be replaced by a goto statement referring, by its label, to the statement following the procedure statement, which, if originally unlabeled, is treated as having been assigned a (unique) label during the replacement process.

The values assignable to, or computable by, the actual input parameters must be compatible with type declarations concerning the corresponding formal parameters which appear in the procedure.

For actual output parameters, only type declarations duplicating given type declarations for the corresponding formal parameters may be made.

Array declarations concerning actual parameters must duplicate, in corresponding subscript positions, array declarations referring to the corresponding formal parameters.

1.4. From Part II.

5. Declarations Δ

Declarations serve to state certain facts about entities referred to within the program. They have no operational meaning and within a given program their order of appearance is immaterial. They pertain to the entire program (or procedure) in which they occur, and their effect is not alterable by the running history of the program.

1.5. From Part II, 5 Declarations Δ

(*ii*) *Array declarations* Δ. Array declarations give the dimensions of multidimensional arrays of quantities.

Form: $\mathbf{\Delta} \sim$ **array** $(I, I, \ldots I[l:l'],$
$I, I, \ldots, I[l:l'], \ldots)$

where **array** is the array declarator, the I are identifiers, and the l, and l' are lists of integers separated by commas.

1.6. From Part II, 5 Declarations Δ

(*iv*) *Function declarations* Δ. A function declaration declares a given expression to be a function of certain of its variables. Thereby, the declaration gives (for certain simple functions) the computing rule for assigning values to the function (cf. *functions*) whenever this function appears in an expression.

Form: $\mathbf{\Delta} \sim I_N(I,I, \ldots, I) := E$

where the I are identifiers and E is an expression which, among its constituents, may contain simple variables named by identifiers appearing in the parentheses.

The identifier I_N is the function name. The identifiers in parentheses designate the formal parameters of the function.

Whenever the function $I_N(P, P, \ldots, P)$ appears in an expression (a *function call*) the value assigned to the function in actual computation is the computed value of the defining expression E. For the evaluation, every variable V which is listed as a parameter I in the *function declaration,* is assigned the current value of the actual parameter P in the corresponding position of the parameter list of the function in the function call. The (formal) variables V and E which are listed as parameters in the declaration bear no relationship to variables possessing the same identifier, but appearing elsewhere in the program. All variables other than parameters appearing in E have values as currently assigned in the program.

Example: $I(Z) := Z + 3 \times y$

....

$alpha := q + I(h + 9 \times mu)$

In the statement assigning a value to alpha the computation is:

$alpha := q + ((h + 9 \times mu) + 3 \times y)$

1.7. From Part II, 5 Declarations Δ

(*vi*) *Procedure declarations* Δ. A procedure declaration declares a program to be a closed unit (a procedure) which may be regarded as a single compound operation (in the sense of a generalized function) depending on a certain fixed set of input parameters, yielding a fixed set of results designated by output parameters, and having a fixed set of possible exits defining possible successors.

Execution of the procedure operation is initiated by a *procedure statement* which furnishes values for the input parameters, assigns the results to certain variables as output parameters, and assigns labels to the exits.

Form: $\Delta \sim$ **procedure** $I(P_i) =: (P_0), I(P_i) =: (P_0), \ldots, I(P_i) =: (P_0)\ \Delta; \Delta; \ldots; \Delta;$
begin $\Sigma; \Sigma; \ldots; \Delta; \Delta; \ldots; \Sigma; \Sigma$
end

Here, the I are identifiers giving the names of different procedures contained in the procedure declaration. Each P_i represents an ordered list of formal input parameters, each P_0 a list of formal output parameters which include any exits required by the corresponding procedures.

Some of the strings "$=: (P_0)$" defining outputs and exits may be missing, in which case corresponding symbols "$I(P_i)$" define a procedure that may be called within expressions.

The Δ in front of the delimiter **begin** are declarations concerning only input and output parameters. The entire string of symbols from the declarator **procedure** (inclusive) up to the delimiter **begin** (exclusive) is the *procedure heading*. Among the statements enclosed by the parentheses **begin** and **end** there must be, for each identifier I listed in the heading as a procedure name, exactly one statement labeled with this identifier, which then serves as the entry to the procedure. For each "single output" procedure $I(P_i)$ listed in the heading, a value must be assigned within the procedure by an assignment statement "$I := E$", where I is the identifier naming that procedure.

To each procedure listed in the heading, at least one **return** statement must correspond within the procedure. Some of these **return** statements may however be identical for different procedures listed in the heading.

Since a procedure is a self-contained program (except for parameters), the defining rules for statements and declarations within procedures are those already given. A formal input parameter may be

(a) a single identifier *I* (formal variable)
(b) an array *I*[, , . . . ,] with *k*(*k* = 1, 2, . . .) empty parameter positions
(c) a function *F*(, , . . . ,) with *k*(*k* = 1, 2, . . .) empty parameter positions
(d) a procedure *P*(, , . . . ,) with *k*(*k* = 1, 2, . . .) empty parameter positions
(e) an identifier occurring in a procedure which is added as a primitive to the language.

A formal output parameter may be

(a) a single identifier (formal variable)
(b) an array with *k*(*k* = 1, 2, . . .) empty subscript positions.

A formal (exit) label may only be a label.

A label is an admissible formal exit label if, within the procedure, it appears in **goto** statements or **switch** declarations.

An array declaration contained in the heading of the procedure declaration, and referring to a formal parameter, may contain expressions in its lists defining subscript ranges. These expressions may contain

1. numbers
2. formal input variables, arrays, and functions.

All identifiers and all labels contained in the procedure have identity only within the procedure, and have no relationship to identical identifiers or labels outside the procedure, with the exception of the labels identical to the different procedure names contained in the heading.

A procedure declaration, once made, is permanent, and the only identifiable constituents of the declaration are the procedure declaration heading, and the entrance labels. All rules of operations and declarations contained within the procedure may be considered to be in a language different from the algorithmic language. For this reason, a procedure may even initially be composed of statements given in a language other than the algorithmic language, e.g., a machine language may be required for expressing input–output procedures.

A tagging system may be required to identify the language form in which procedures are expressed. The specific nature of such a system is not in the scope of this report.

Thus by using procedure declarations, new primitive elements may be added to the algorithmic language at will.

Appendix 4. Notes by J. H. Wegstein

1. To Section 1.1

A bit of the human side of this history would give it a little more life. For example, the 1958 meeting was held at the Eidgenossische Technische Hochschule and the emotional welcoming address by the director along with being surrounded by the walls of this famous institution were a real inspiration to the participants. However, after two full days of sparring the meeting came to a complete deadlock with one European member pounding on the table and declaring: "No!, I will never use a period for a decimal point." That evening

Wegstein visited the opposing camps and proposed defining the three levels of language. The proposal was immediately adopted the following morning and progress resumed.

2. To Section 1.1

The name ALGOL was not adopted in 1958 but emerged later and was applied retrospectively to the 1958 results. (At one point, Prof. Bauer [or Alan Perlis, cf. appendix 7,1] wanted to call it "agricola" pointing out that it would someday be as common as cola but he couldn't defend the prefix.)

Appendix 5. Notes by F. L. Bauer

1. General Remarks

1.1. To Introduction

The time between publication of the Zurich report and the Paris conference is viewed with the eye of a Copenhagen and possibly an Amsterdam observer. Much discussions went on between the four Zurich participants of the European side, quite naturally. This discussion was initially not at all and later only sporadically reflected in the *ALGOL Bulletin,* an undertaking set up by Peter Naur on his own. Although the four Zurich participants supported the *ALGOL Bulletin,* they could not publish there the technical details of the compiler design that was under way, and many observations and reflections did come directly from this experience.

The detailed, however very interesting, discussion of the development of even minor points of the language between Zurich and ALGOL 60 is justified if a similarly thorough treatment is given to the development from the GAMM proposal to the Zurich report. Otherwise, the presentation is an axiomatic approach, with the Zurich report (who does still remember its details?) as an axiom.

Hopefully, Alan Perlis will explain some of the deliberations that led to some of the interesting points of the Zurich Report. The development on the European side, however, is not in his scope.

1.2

Some of the "improvements" of the Paris conference over the Zurich report are doubtful. This can be seen from the fact that some of the U.S. ALGOL dialects, like NELIAC, based on ALGOL 58, turned out to be competitors to ALGOL 60. Some of the obscurities and ambiguities that made the Rome Revision necessary have been side effects of changes between Zurich and Paris. This should be pointed out by Peter Naur more clearly.

1.3

I recommend that the material that Samelson has made available to Peter Naur, in particular the two GAMM proposals from October 1957 and March 1958 (translation May 1958) and of the Zurich notes, be carefully studied and evaluated by Peter Naur and be given equal consideration as the ALGOL 60 development. [Note: "Formelgruppen" stands for "procedure."]

The "sectional character" of function definitions (p. 7) and the "section postulates" of

p. 10 are first steps towards the block structure. Unfortunately, these attempts were not fully appreciated by the American members of the Zurich conference. Other proposals were rejected on grounds of "experience with FORTRAN" (Backus); for example the notation $a + 3 \rightarrow s$ for an assignment, which expresses the natural flow of computation and supports optically optimization, was finally reverted.

2. Details

2.1. To Section 1.1

Rutishauser's Historical Remarks are from the introduction to his book, which is not just "his book on ALGOL 60," but is the Volume I, Part a, of a series planned under the title "Handbook for Automatic Computation." As Editors of this series in the famous Springer Grundlehren are listed F. L. Bauer, A. S. Householder, F. W. J. Olver, H. Rutishauser, K. Samelson, E. Stiefel. This project (so far, Vol. I, Part b, by Gries, Hill and Langmaack and Vol. II by Wilkinson and Reinsch have appeared, too) has a deep connection with the development of ALGOL, however. It was started in 1957 by the initiative of F. K. Schmidt, a mathematics professor in Heidelberg, who was consulting Springer. The handbook project called for the collection of experienced and tested (numerical) algorithms, thus making both a universal language for describing them and a compiler for operating them necessary. The handbook project was in particular an incentive for developing compilers in Munich, Zurich, and Oak Ridge (see below).

2.2. To Section 1.1, Rutishauser's § 2

The GAMM Subcommittee met at Lugano, October 16–22, 1957, to finalize the draft. At that meeting Bottenbruch and Bauer had just returned from a trip to the U.S., where they had been in contact with J. W. Carr, A. J. Perlis, and some others listed in Rutishauser's footnote. They reported that they had felt that an attempt of unification would be welcomed. Correspondingly, a letter was written to J. W. Carr, the President of the ACM, inviting cooperation. This led to the activity on the U.S. side that brought the establishment of the ACM Committee.

A similar invitation was at that time addressed to learned societies in other countries, but no reply was obtained.

2.2a. To Section 1.3

ALGOL 58 implementation started in Europe from a nucleus formed by the interested institutions in Zurich, Munich, and Darmstadt and, since April 1958, Mainz; a group which called itself ZMMD-Group. It was based on the work described in the German Patent Application "Verfahren zur automatischen Verarbeitung von kodierten Daten und Rechenmaschinen zur Ausübung des Verfahrens," Nr. B 44 122 IX 42m, submitted March 30, 1957.

By the way, the name ALGOL came up during one of the ZMMD-meetings in summer 1958, at an evening party. Somebody said "Algorithmic Language—ALGOL". Those among us who had studied astronomy did see the pun, and the name did stick on. In fact, the European side of the Zurich conference participants in 1958 had not liked the IAL (International Algebraic Language) favored by the American side; it went, however, into JOVIAL, etc.

2.3. To Section 1.4

At the November 1958 Mainz meeting of persons interested in ALGOL, the ALGOL 58 compiler programmed by M. Paul from Mainz was presented; it came into practical use for the Z22 in March 1959. In Zurich and Munich, similar progress was made.

2.4. To Section 1.6

It is amusing to see how Peter Naur looks at the use of the Backus notation from his personal point of view. Among Rutishauser, Samelson, and myself there was no question that we would like for the outcome of the Paris conference a form similar to the one Backus had used for its ICIP paper (some of us had been quite familiar with equivalent formulations in mathematical logic even before and did see the notational advantages.) Thus, there was no need to do more than mention the use of Backus's notation in passing. If Peter Naur had seen this a result of his "plan" to make an appeal to the members of the ALGOL Committee concerning the style of the language description, he was running into open doors.

2.5. To Section 1.6

Assuming that Peter Naur did not know this explains one of the strange incidents of the Paris ALGOL 60 conference. While Rutishauser, Samelson, and myself (and possibly others) had concentrated on preparing themselves for the conference by working through and sorting material of the European preparatory meetings, the last one being held in Mainz in December 1959, they were surprised by Peter Naur handing them out at the beginning of the Paris conference his 18-page draft report. Peter Naur had not been commissioned to do so; it was a fait accompli. It therefore sounds poetic if he has written that his draft Report was "chosen" as the basis of the discussion; the committee was simply forced to do so, after Peter Naur had gained this advantage. Likewise, he became automatically the editor; it was only a question of politeness to invite him. Since there was some concern that he would use this position to exert some influence of his own on the language (what has happened indeed, as he indicates himself), this development was not considered to be very healthy by some committee members. (I gave Peter Naur the warning, he would now be "the pope"—a role in which I would have preferred to see Rutishauser—but he seemingly did not appreciate the remark). On the other hand, this situation has certainly helped to get a draft of the report finished in spite of the very tight time schedule of the six day Paris Meeting.

2.6. To Section 1.6

In Section 1.6 and again in Section 2.10 item 38.3, Peter Naur speaks of "my slightly revised form of Backus's notation" and "my slightly modified form of Backus's notation." I think the minor notational difference is not worth mentioning. If some people speak of Backus–Naur form instead of the original Backus Normal Form, then they indicate that Peter Naur, as the editor of the ALGOL 60 report, brought this notation to a wide attention. Backus–ALGOL Form would be more appropriate anyhow.

2.7. To Section 2.10 item 49†

A particularly sad experience is described with the events after the conference‡ had ended. Samelson and others had unsuccessfully tried to have procedures as parameters

† See also Appendix 6, Woodger's point 2. ‡ Paris, Jan. 11–16, 1960.

included in ALGOL 60 in a straightforward manner (This seemingly is not recorded by Peter Naur, showing the arbitrariness of working along the written notes only). Alan Perlis then with proposal 48 of Jan. 20 *admitted* that he understood the "call by name" to be a hidden introduction of procedure parameters (later to be known as Jensen's trick). The remark "seemingly was only understood in a coherent manner by one member" is therefore cynical and should be omitted. It was under great pressure that the conference had ended and a majority had voted down a minority in the controversial point 45. Straightforward introduction of procedure parameters (and of the lambda calculus), as it was advocated by the minority, would have outflanked some of the ambiguities that made the Rome revision necessary.

2.8. To Section 2.10 Items 50 and 52

Events described by Peter Naur in item 50 (replacing *name* listings by *value* listings!) and item 52 (the Amsterdam plot on introducing recursivity) show clearly that Peter Naur had absorbed the Holy Ghost after the Paris meeting. It should be mentioned, however, that there was not only scepticism among the committee members, but also resignation that there was nothing one could do when the editor did arbitrarily change the outcome of the conference: it was to be swallowed for the sake of loyalty. These feelings, however, seemingly have not been sensed by the editor. Otherwise, I think too, he did a magnificent job.

Appendix 6. Notes by M. Woodger

1. To Section 1.6†

I am very glad that you have described your technique for control over the discussions which led to the final report. This is a contribution still not recognized for what it was really worth, and possibly even unique—I know of no other comparable case.

2. To Appendix 5, Bauer's Point 2.7‡

I have puzzled long over Bauer's point 2.7 and over my notes made at the time. I have no idea what he refers to, and I believe that my notes taken at the time are fairly representative of what was discussed in open session, although I did not understand all the details of each argument. The copies enclosed cover the discussions on 15 and 16 January 1960. My marginal note to Perlis's proposal 1 of 4:45 P.M. on 15th January shows clearly that he was proposing to treat expressions given as parameters in just the manner of his letter of 20th January [cf. Section 2.10 item 40]. My note of proposal 2 by Samelson, and the voting on it (only Bauer supporting it) suggests that Bauer may be thinking of this one. My notes on the evening discussion of 15th January support this. But this is purely conjecture, and Fritz [Bauer] must be more specific if he is to be taken seriously.

† From letter to P. Naur, 1977 October 3.
‡ From letter to P. Naur, 1978 January 27.

Appendix 7. Comments by K. Samelson, 1978 December 1

For remarks on this see Appendix 8

1. To Section 1.1

The report starts with the background, viz. ALGOL 58, which Naur starts by quoting Rutishauser's book. To this and to the respective reviewers notes, two remarks should be added. The first is that initiative and drive in setting up contacts in particular to the ACM was primarily F. L. Bauer's, who thus might be called the initiator of ALGOL. Secondly, to the best of my recollection, Joe Wegstein, in his notes [Appendix 4,2], is in error ascribing the name proposal Agricola to Bauer (from whom it would have been in questionable taste). It was Alan Perlis who, in one of the lighter moments of the meeting, made the suggestion.

2. To Section 1.6

Next, some comments seem appropriate with regard to Section 1.6., headed "Discussion Techniques," essentially a presentation of Naur's own view of his part in preparing ALGOL 60, his way to editorship of the ALGOL 60 report and his part in the ALGOL 60 committee meeting in Paris, January 1960.

Well known facts are

that Naur was the initiator and editor of the *ALGOL Bulletin,*

that he was active in contributing there,

that he was an active participant in the preparatory conferences on the European side, viz. Copenhagen February 1959, Paris November 1959, Mainz December 1959,

that he surprised the entire ALGOL 60 committee with a draft compiling and consolidating the proposals made for the Paris meeting (*not* giving a new language design) which was accepted as a guide for the meeting, and

that on account of this work he was asked to be editor of the final report.

The narrative around these facts, however, to me seems to give more insight into certain facets of the author's personality, his energy and capacity for work on the one side, his style of cooperation on the other side, than into the history of ALGOL 60.

3. Concerning BNF

On the subject of BNF (which to me has always meant Backus Normal Form, since John Backus quite alone introduced it in his Paris '59 paper) for the ALGOL 60 report, I can only contradict him. As far as I, or even the entire GAMM group, is concerned no persuasion whatsoever was necessary to use it. Rather, BNF just needed to be presented by John to be accepted as a superior means of syntax description. When Naur [Section 1.6, tenth paragraph] says of the December '59 Mainz meeting "The use of Backus's notation could only be mentioned in passing . . . ," I would say this was so because it hardly needed mentioning, being so obvious, and there were much more problematical matters at hand.

4. Concerning the ALGOL 60 Meeting in Paris

To the account of the way the ALGOL 60 meeting in Paris was conducted, I would add that the mode of operation to get the material for the report in shape proposed by Naur was effective as long as there was sufficient time for discussing the matters to be voted upon, and the will to cooperate could overcome material controversies, until, in the end, time was running out on us, and on the most controversial subject, procedures (cf. Section 2.10 item 40ff.). The foreseeable result was that the final votes, after insufficient deliberation, produced a procedure concept that even Naur, as editor, found inacceptable, whereupon he on his own made changes in the direction of previous concepts, repairing part (although far from all, in my opinion) of the damage, but threw in a few unnecessary changes, too.

5. Concerning the Central Controversy

This topic, procedure concepts, carries me right into the primary objection I have against the paper. There was considerable controversy in the development. Naur mentions it rather briefly in the next but last paragraph of 1.8, but in his discussion of the content of the language, Section 2, only treats the procedure case at length. Furthermore, in 1.8, he (to my mind and recollection) wrongly labels the opposing views. Thus he covers up rather than reports on the central controversy.

This conflict was not one between "standard mathematical notation" and language "readily translatable into machine code." These two, I found then as today, go very well together. The conflict really was between the wish for extreme "flexibility" and "power" of the language, giving freedom to the programmer to express his desires in as compressed (and puzzling) a way as he wished, and, at the other side, the wish to restrict syntax in order to secure transparency, in particular local readability, comprehensibility (today I would say verifiability), and translatability of programs.

That the liberalists (the "trickologists") were the stronger party shows up all over the ALGOL 60 report. Cases in point are: the dynamic for clause (allowing all kinds of manipulations with controlled variable, step, boundary values), functions (type procedures) with no restriction on side effects, procedure headings (disappearance of the separation of arguments, results and exits, of obligatory parameter specifications and specifications of parameter structure; preponderance of call-by-name concept), general designational expressions, even the introduction of blocks as distinct from statements (a major reason being that everybody agreed upon forbidding jumping into a block, but some people found it important to be able to jump into conditional statements, right over the if clause, which then required restrictions against jumping into for statements).

The restrictionists (the "safetishists") were essentially the GAMM group: Bauer, Rutishauser, Samelson (where I always have seen myself as the strictest one; others may see it differently), with occasional support by some other like Green or McCarthy. The hard core of the liberalists consisted of Naur, Perlis, and v. Wijngaarden, with most of the others tending to support them. The outcome was as sketched out above, with the largest, and most conspicuous object, procedures, in a particularly "flexible" (for trick programmers) and cumbersome (for translators and human readers) form partly due to the time pressure on the committee.

The wide divergence of opinions on this topic is illustrated by the last paragraph of section 3.2 of the paper. Here, the author presents the unification of procedure parameters as a decisive step forward, giving a clear description of meaning, where I always have seen a procrustean generalization "unifying" conceptually different classes of objects. This unification burdens the normal case with the problems arising from the exceptional case, where the normal case is the procedure with well-defined sets of input and output parameters, and the exceptional case is the procedure with parameters with parameter-dependent input–output characteristics (which in these days of structured programming no self-respecting programmer would touch). The closing paragraph of 3.5 (proper use of procedure parameters one of the biggest problems for users) seems to support this view.

I felt then, and today, that all these things were very much to the detriment of the language which was definitely not an exercise in language design but an attempt to create a universally acceptable (and, hopefully, accepted) programming language for numerical computations. Since safe (structured) programming has become the fashion in later years, many other people today may feel the same (if thinking of ALGOL 60 at all).

6. Concerning Recursive Procedures

There are some other points in Naur's account which I have to take up. The first is his remark on recursive procedures, (Section 2.10, item 52). The U.S. proposal was to introduce a particular delimiter **recursive** in order to tag recursive procedures, not the admissibility of recursive procedures proper which at the meeting was hardly questioned. The proposal was rejected because it was essentially useless. A delimiter as proposed would have made sense only in case recursive procedures were to receive special treatment by the translator (which alone would be concerned) but could not obviate a check for actual recursivity, and could only generate additional "errors" (of erroneous presence, or absence although needed), and it would probably be insufficient to mark recursivity via actual (for formal) procedure parameters. Saying nothing about recursivity automatically introduced it in full, and the meaning of procedure (or type procedure) identifiers in procedure bodies was always clear by syntactic position. Thus the "bold addition" "Any other occurrence of the procedure identifier within the procedure body denotes activation of the procedure" was superfluous. It only served to prohibit otherwise syntactically legal misuses of the procedure identifier, e.g., accumulating intermediate values computed iteratively under the final function identifier.

7. To Section 3.1

I have always been ignorant, and wondered very much about how the statement of objectives given in the Zurich report came to be left out of the ALGOL 60 report. Considering precision and thoroughness of both the editor and the report, simple oversight seems unlikely.

A possible reason is paragraph 1. "The new language should be as close as possible to standard mathematical notation, and be readable with little further explanation."

In Section 3.1. Naur takes a very "literal" view of this paragraph. Even there I think that he (and Woodger) judge far too severely disregarding the formulation "as close as possible" which takes into account both requirements presented by reading–writing devices (strict linearity of text) and by analyzing (parsing) techniques (reduction of context dependency).

However, in Section 1.8. Naur himself indicates that the term "mathematical notation" could be, and was, interpreted in a far wider sense than purely typographically by taking into account the intent, essentially the mathematical concept of a function as a mapping from domain to range, and e.g., an expression as expressing the mapping rule which absolutely excludes any kind of so-called side effects. This Naur himself contrasts with the concept of the more or less (I would say much more than less) open computational process that conceivably may have side effects which later was brought to "perfection" in ALGOL 68. This is again the basic conflict between linguistic discipline and unrestricted liberty mentioned above.

8. To Section 3.4

Section 3.4, Design Weaknesses, lists the five problem areas identified in the revision of 1962. Of these, four are connected with the procedure concept, which supports the view that the procedure concept was far from mature due—in part at least—to time restraints posed by the strict time schedule of the Paris meeting. Apart from this I feel that the most serious (and curious) lack in ALGOL 60 was that of a simple general repetitive clause of the **while** type. The ALGOL 60 **while** rather perversely couples a "controlled variable" to the **while** via the **for** making them into a kind of siamese twins whereas the **for** can be considered to be an important special case of the general **while.**

9. Concluding Remark

As a concluding remark I should add that I had grave doubts about the wisdom of the decision to treat the "history" of ALGOL 60 on a congress and in a publication since this could only raise old, long forgotten controversies. But since the decision was made, and the publication was presented, I felt compelled to add my admittedly personal (and, as such, possibly biased), view, attempting to balance the official report by the minority report I should have written—but abstained from doing so for unity's sake—in 1960.

Appendix 8. Remarks to Samelson's Comments in Appendix 7 by P. Naur

1. The Background of Samelson's Comments

For an understanding of some of my present remarks the sequence of events leading to Samelson's comments in Appendix 7 has to be clarified. Samelson first wrote his comments in response to my preliminary unpublished notes and report dating from 1977 August to October. These comments of Samelson's, which I shall refer to as version A, were dated 1977 December. He did not send them to me until 1978 February 14. At that time it was too late for me to take them into account in my report for the preprints of the History of Programming Languages Conference. However, in a letter to Samelson on 1978 February 21 I commented on them and in particular asked him to clarify the claims made by Bauer in Appendix 5, point 2.7, and the remarks to this made by Woodger in Appendix 6, point 2, and suggested that he bring his comments in accordance with the final version of my report. Samelson did not respond to this at the time. Meanwhile I made use of Samelson's version A in preparing my oral presentation at the conference in June (see transciption p. 147ff). The document brought along by Bauer to the conference (see the Question and Answer session) also is the unrevised version A. After two invitations from me, in letters of 1978 June 8 and October 5, Samelson finally produced version B of his comments, as given in Appendix 7, and sent them to me with a letter on 1978 December 11.

2. Concerning Bauer's Claim in Appendix 5, Point 2.7

On this background it is curious that Samelson's revised set of comments, Appendix 7, is silent on the question of Bauer's claim in Appendix 5, point 2.7, as was his version A. The matter is, however, clarified in the following remark by Samelson: "Fritz [Bauer] showed me the notes [Appendix 5], and as I had no clear recollection except that I knew I had at some time or other proposed a λ-like notation for function or procedure parameters, I thought that Fritz remembered better than I did, and let it pass. For your, and Mike [Woodger]'s remarks [Appendix 6, 2] I would now conclude, that Fritz mixed up times and events" (Samelson, 1978).

3. On Samelson's Comment on BNF

As concerns Samelson's remarks on the subject of BNF (Appendix 7, 3) I do not see the contradiction, cf. Section 2.10, item 39.

4. Concerning Samelson's Two Parties

Concerning Samelson's central section, Appendix 7, item 5, that presents the picture of two opposing parties, the "liberalists" and the "restrictionists," one of course must accept that Samelson views his own role and the events in the manner he describes. However, talking only about matters of which I have firsthand experience and evidence, Samelson misrepresents my motives and actions, and his picture is contradicted by the discussions as they emerge from the original documents. To a considerable extent it seems that Samelson's presentation is colored by events that took place after March 1960 and that thus lie beyond the period covered by my report.

The relevant questions have mostly been dealt with in my main report. Thus it should be clear that my own primary interest was in the area of clear, unambiguous language description, an interest that ought to accord well with what Samelson calls a "restrictionist" attitude, in particular in dealing with a language designed for wide use.

It may also be noted that many of the language points that Samelson claims were the result of the "liberalist" influence were already in ALGOL 58, established before two of the three "hard core liberalists" had joined the effort. This holds for the dynamic **for** clause, designational expressions, and procedure parameters called by name. In fact, the presence of procedure parameters called by name in ALGOL 58 was due to Samelson himself, who in referring to the first draft of the ALGOL 58 report says: " . . . where I (to my later deep regret) filled a gap left open by the Zurich conference inserting the strict text replacement rule for procedure parameters (pre-"call by name")" (Samelson, 1977). Up to the time of the ALGOL 60 report no opposition to these features of ALGOL 58 was voiced by the "restrictionists."

Further, the outstanding new dynamic feature of ALGOL 60, the dynamic array declaration, after having been advocated by Ehrling (Section 2.10, items 2 and 4), was strongly supported by Rutishauser, claimed to be a "restrictionist" (see Section 2.10, item 21), while it was initially opposed by me, claimed to be a "liberalist" (Section 2.10, item 32).

On the question of procedures, I, far from proposing new radical ideas, spent much effort in trying to save a procedure concept that retained the "restrictionist" separation of input and output parameters (ALGOL 60 documents 5, 10, and 14). In the final round my main influence was that I followed Rutishauser's "restrictionist" suggestion for parameter specifications (Section 2.10, item 47), while also insisting that the proposal already adopted by the full committee should be understood and described clearly. How these actions can be reconciled with a membership of Samelson's "hard core liberalists" I fail to comprehend.

Altogether Samelson's claim that many of the problematic features of ALGOL 60 were due to the influence of a "stronger party of liberalists" has no support in the recorded facts. Concerning several of the features, for example, the undesirable dynamic possibilities of the **for** statement, there was simply no discussion. If in the period leading up to ALGOL 60 the members of the "restrictionist party," Bauer, Rutishauser, and Samelson, had specific proposals for avoiding these possibilities, they failed to communicate them to the ALGOL community.

More generally I feel that any attempt to describe the discussions leading to ALGOL 60 in terms of parties having clearly identifiable views and goals will be misleading. Neither the problems, nor people's views, were that simple. The description of conflicting views in Section 1.8 is not meant to imply that a corresponding grouping of the ALGOL 60 authors into parties existed.

5. On the Admission of Recursive Procedures

My answer to Samelson's remarks on the admission of recursive activations in ALGOL 60 (Appendix 7, 6) is given in my oral presentation, the transcript of which appears below.

6. The Omission of Objectives

The answer to Samelson's question (Appendix 7, 7) about the omission of the statement of objectives in the ALGOL 60 report is simple. The introduction to the report was drafted by the chairman of the meeting, Wegstein (ALGOL 60 document 28), and adopted by the full meeting with a few corrections. In the draft ALGOL 60 report this introduction was taken over unchanged. The omission of the statement of objectives was an oversight, committed by the full committee of which Samelson himself was a member, first during the meeting in Paris, and again during the weeks of scrutiny of the draft. It is quite unnecessary to look for sinister actions by "hard core liberal party members" behind this omission.

7. On the Conflicts of the Present Discussion

Finally, in response to Samelson's final remarks (Appendix 7,9), it appears from the above discussion that to Samelson the old facts are as forgotten as the old controversies. Most of the conflict of the present discussion would disappear if Samelson would consider the facts as they are recorded in the original documents, instead of blaming the imperfections of ALGOL 60, to which he has contributed equally with the rest of the authors, on a "liberalist party" existing only in his own imagination.

REFERENCES

AB, see *ALGOL Bulletin.*

Adams, C. W., and Laning, Jr., J. H. (1954). The MIT systems of automatic coding. In *Proceedings of a Symposium on Automatic Programming for Digital Computers,* pp. 40–68. Washington, D.C.

ALGOL 60 documents (1959–1960). Unpublished technical memoranda prepared in connection with the ALGOL 60 conference in Paris, 1960, Jan. 11–16. During the conference the available documents were numbered 1 to 30. Here the numbers 31 and 201 to 221 have been used for further related documents.

2: European representatives to the ALGOL 60 conference (1959) Dec. 14–16. Meeting of the European representatives to the ALGOL conference. Mainz. 4 pp.

3: Naur, P. (1960) Jan. 2. Material related to the coming international ALGOL conference in Paris. Copenhagen. 1 + 7 pp.

4A: European ALGOL committee (1959) Dec. 19. Aide-memoir on revisions of the Zürich report. Mainz. 2 pp.

5: Naur, P. (1960) Jan. 9. ALGOL 60—draft report. Regnecentralen, Copenhagen. 18 pp.

6: American representatives to the ALGOL 60 conference (1960) Jan. Proposals for the conference. 12 pp.

7: Turanski, Wm. (1960) Jan. 11. Report on ACM proposal for for-clauses. 2 pp.

8: Katz, C. (1960) Jan. 11. Report on switch declaration and do statement for ACM. 1 page.

9: Perlis, A. (1960) Jan. 12. A proposal on the global-equivalence controversy. 2 pp.

10: Naur, P. (1960) Jan. 11. Procedure declaration (supplement to document 5). 3 pp.

11: Backus, J., Green, J., Samelson, K., and van Wijngaarden, A. (1960) Jan. 13. Report of the committee on local, etc. . . . 1 page.

12: Bauer, F. L., and Rutishauser, H. (1960) Jan. 13. Semantics of procedure statements. 2 pp.

13: Vauquois, B. (1960) Jan. 13. Proposal about subscripted variables. 1 page.

14: Naur, P. (1960) Jan. 13. Summary of parameter replacement rules of document 5. 1 page.

15: Naur, P., and Perlis, A. Meaning of types and assignments. 1 page.

16: Naur, P., and Perlis, A. (1960) Jan. 13. Types of expressions—assignments. 3 pp.

17: Katz, C., van Wijngaarden, A., and Woodger, M. (1960) Jan. 14. Report of the committee on procedure declarations and procedure calls with only one list of parameters. 1 page.

18: Green, J., Perlis A., and Samelson, K. (1960) Jan. 14. Procedure statements. 3 pp.

19: Naur, P. (1960) Jan. 14. Addition to document 5—conditional statements. 2 pp.

20: Rutishauser, H. (1960) Jan. 14. Remark concerning function declaration. 1 page.

21: McCarthy, J. (1960) Jan. 14. Namechange. 1 page.

22: Rutishauser, H. (1960) Jan. 14. Array-functions. 1 page.

23: Naur, P. (1960) Jan. 15. Compound statements and blocks (correction to document 5). 1 page.

24: McCarthy, J. Scopes of identifiers. 1 page.

25: For statement. 2 pages.

26: Procedure statements (1960) Jan. 16. 2 pp.

27: Procedure declarations (1960) Jan. 16. 2 pp.

28: Wegstein, J. H. Report on the algorithmic language ALGOL 60 (draft for the introduction). 3 pp.

29: Wegstein, J. H. At the end of the introduction add. 1 page.

30: Samelson, K., and Woodger, M. Syntax of expressions and statements. 2 pp.

31: ALGOL 60 committee (1960) Jan. 13–16. First and second list of suggested changes in document 5. Each item is authored by a member of the committee. The items are numbered 101 to 175, followed by 173–175 (used again) and $\infty - 1$. 33pp.

201: Naur, P. (1960) Jan. 17. To the members of the ALGOL 60 committee. Paris. 2 pp.

202: van Wijngaarden, A. (1960) Jan. 19. Letter to P. Naur. Amsterdam. 1 page.

203: McCarthy, J. (1960) Jan. 20. Letter to P. Naur. Cambridge, Massachusetts. 1 page.

204: Rutishauser, H. (1960) Jan. 20. Letter to the members of the ALGOL 60 committee. Zurich. 2 pp.

205: Perlis, A. (1960) Jan. 20. Letter to the members of the ALGOL 60 committee. Pittsburgh, Pennsylvania. 1 page.

206: Wegstein, J. H. (1960) Jan. 21. Letter to P. Naur. Washington, D.C. 1 page.
207: Bauer, F. L., and Samelson, K. (1960) Jan. 22. Letter to P. Naur. Mainz. 1 page.
208: Woodger, M. (1960) Jan. 25. Letter to the members of the ALGOL 60 committee. Teddington. 3 + 5 pp.
209: Katz, C. (1960) Jan. 25. Letter to P. Naur. 2 pp.
210: Rutishauser, H. (1960) Jan. 25. Letter to the members of the ALGOL 60 committee. Zurich. 1 page.
211: Green, J., and Backus, J. (1960) Jan. 26. Telegram to P. Naur. White Plains, New York. 1 page.
212: Naur, P. (1960) Febr. 4. Letter to the members of the ALGOL 60 committee. Copenhagen-Valby. 1 page.
213: Backus, J., *et al.* (1960) Febr. 4. Report on the algorithmic language ALGOL 60 (draft). Copenhagen-Valby. 26 pp.
214: van Wijngaarden, A. (1960) Febr. 4. Letter to the members of the ALGOL 60 committee. Amsterdam. 4 pp.
215: Perlis, A. J. (for Holt, A., and McCarthy, J.) (1960) Febr. Comments on the report draft. Pittsburgh, Pennsylvania. 5 pp.
216: Bauer, F. L., and Samelson, K. (1960) Febr. 12. Letter to P. Naur. Mainz. 2 pp.
217: Rutishauser, H. (1960) Febr. 14. Letter to P. Naur. Zurich. 5 pp.
218: Woodger, M. (1960) Febr. 16. Letter to P. Naur. Teddington. 2 pp.
219: Rutishauser, H. (1960) Febr. 17. Letter to P. Naur. Zurich. 2 pp.
220: Naur, P. (1960) Febr. 18. Letter to H. Rutishauser. Copenhagen-Valby. 1 page.
221: Rutishauser, H. (1960) Febr. 26. Letter to P. Naur. Zurich. 4 pp.

ALGOL Bulletin (1959). Mimeographed discussion letters (P. Naur, ed.). No. (AB) 1, 1959 March 16, 6 pp.; No. 2, 1959 May 5, 8 pp.; No. 3, 1959 June 8, 6 pp.; No. 4, 1959 Aug. 13, 7 pp.; No. 5, 1959 Sept. 28, 9 pp.; No. 6, 1959 Oct. 17, 2 pp.; No. 7, 1959 Nov. 3, 21 pp.; No. 8, 1959 Dec. 12, 15 pp. Regnecentralen, Copenhagen.

Backus, J. (1959). The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference. *Proc. International Conf. on Information Processing,* pp. 125–132, UNESCO.

Backus, J. (1979). Jan. 16. Letter to P. Naur. 3 pp.

Backus, J. W., Desilets, P. H., Evans, D. D., Goodman, R., Huskey, H., Katz, C., McCarthy, J., Orden, A., Perlis, A. J., Rich, R., Rosen, S., Turanski, W., Wegstein, J. (1958). Proposal for a programming language. Unpublished memorandum. 20 pp.

Backus, J. W., Bauer, F. L., Bottenbruch, H., Katz, C., Perlis, A. J. (ed.), Rutishauser, H., Samelson, K. (ed.), Wegstein, J. H. (1959). Report on the algorithmic language ALGOL. *Num. Math.* **1**: 41–60. Also (1958). Preliminary report—international algebraic language. *Comm. ACM* **1**(12): 8–22.

Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Naur, P. (ed.), Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., van Wijngaarden, A., and Woodger, M. (1960). Report on the algorithmic language ALGOL 60. *Num. Math.* **2**: 106–136. Also *Comm. ACM* **3**(5): 299–314.

Backus, J. W., *et al.* (1962). Revised report on the algorithmic language ALGOL 60. *Num. Math.* **4**: 420–453. Also *Comm. ACM* **6**(1): 1–17. Also *Computer J.* **5**: 349–367.

Bauer, F. L., Bottenbruch, H., Graeff, P., Läuchli, P., Paul, M., Penzlin, F., Rutishauser, H., and Samelson, K. (1958a). Formelübersetzungsprojekt Zürich-München-Darmstadt—Projektstufe 1: Festlegung der Formelsprache—Interner Bericht No. 2c. Unpublished memorandum. 21 + 23 pp.

Bauer, F. L., Bottenbruch, H., Rutishauser, H., Samelson, K., under cooperation of Graeff, P., Läuchli, P., and Paul, M.(1958b). Proposal for a universal language for the description of computing processes. Unpublished memorandum, Zurich. 1 + 21 pp.

Bemer, R. W. (1969). A politico-social history of ALGOL. In *Annual review in automatic programming.* (M. Halpern and C. Shaw, eds.), Vol. 5, pp. 151–237. Oxford: Pergamon.

Böhm, C. (1954). Calculatrices digitales du dechiffrage de formules logico-mathematiques par la machine même. Thesis ETH Zurich.

Dahl, O., and Nygaard, K. (1966). SIMULA—an ALGOL-based simulation language. *Comm. ACM* **9**(9): 671–82.

Ershov, A. P. (1959). *Programming programme for the BESM computer* (translated from Russian by M. Nadler). Oxford: Pergamon.

Hoffmann, W., and Walther, A. (1956). *Elektronische Rechenmaschinen und Informationsverarbeitung.* Nachrichtentechnische Fachberichte, Bd. 4. Braunschweig: Vieweg.

IBM (1954). *Specifications for the IBM mathematical formula translating system FORTRAN.* Preliminary report. Applied Science Division, Internat. Business Machines Corp., New York.

Irons, E. T. (1961). A syntax directed compiler for ALGOL 60. *Comm. ACM* **4**(1): 51–55.
Knuth, D. E. (1964). Backus Normal Form vs. Backus Naur Form. *Comm. ACM* **7**(12): 735–736.
Knuth, D. E. (1967). The remaining trouble spots in ALGOL 60. *Comm. ACM* **10**(10): 611–618.
Naur, P. (1975). Programming languages, natural languages, and mathematics. *Comm. ACM* **18**(12): 676–683.
Naur, P. (1977). A vector-handling subroutine using call by name written for EDSAC in 1951. Unpublished.
Rutishauser, H. (1952). Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen. Mitt. No. 3 aus dem Inst. für angewandte Math. der ETH. 45 pp. Basel: Birkhaeuser.
Rutishauser, H. (1953). Bemerkungen zum programmgesteuerten Rechnen. Vorträge über Rechenanlagen. pp. 34–37. Max-Planck-Institut für Physik, Göttingen.
Rutishauser, H. (1967). *Description of ALGOL 60.* Handbook for Automatic Computation, Vol. I, part a. Berlin and New York: Springer-Verlag.
Samelson, K. (1977). Nov. 21. Letter to P. Naur. 1 page.
Samelson, K. (1978). Dec. 11. Letter to P. Naur. 2 pp.
Samelson, K., and Bauer, F. L. (1959). Sequentielle Formelübersetzung. *Elektronische Rechenanlagen* **1**: 176–182.
Samelson, K., and Bauer, F. L. (1960). Sequential formula translation. *Comm. ACM* **3**(2): 76–83.
van Wijngaarden, A. (ed.), Koster, C. H. A., Mailloux, B. J., and Peck, J. E. L. (1969). Report on the algorithmic language ALGOL 68. *Num. Math.* **14**(2): 79–218.
Wirth, N. (1971). The programming language PASCAL. *Acta informatica* **1**: 35–63.
Woodger, M. (1959a). Report on a visit to Germany and Switzerland in November 1958. Unpublished memorandum. Teddington, National Physical Laboratory. 5 pp.
Woodger, M. (1959b). Copenhagen ALGOL conference. Unpublished, handwritten notes 6 pp.
Woodger, M. (1960) Jan. 10–16. ALGOL 60 conference in Paris. Unpublished, handwritten notes. 17 pp.
Zurich conference on "universal language" (1958) May 27–30. Unpublished summaries of the agreements of the conference. Zurich. 20 pp.
Zuse, K. (1948–1949). Über den allgemeinen Plankalkül als Mittel zur Formulierung schematisch-kombinativer Aufgaben. *Arch. Math.* **1**: 441–449.

TRANSCRIPTS OF PRESENTATIONS

THOMAS CHEATHAM: We're now to talk about that great compromise, ALGOL. We have two representatives: Alan Perlis, who, when ALGOL was born, was one of the American representatives; and Peter Naur, who was one of the European representatives. And we have proposed that what we shall do is have these gentlemen talk for a half hour apiece, and we will then field all the questions in the half hour remaining at the end, rather than have them after each speaker.

By the time of ALGOL, Alan was head of the Computer Science Department at Carnegie Institute of Technology, which of course has since been renamed. He received his Ph.D. in Applied Mathematics from MIT, and then got into computing very very quickly, working at BRL, Purdue, and Carnegie, with a strong interest in programming languages. We heard about the IT system this morning. John Backus remarked on that, one of the earlier programming languages on the 650. And there were other systems as well.

At the present time, Al is Professor of Computer Science at Yale University, where he continues his interest in programming languages, and is perhaps the most noteworthy APL freak on the East Coast.

ALAN J. PERLIS: ALGOL came into existence in several stages. Its first stage might very aptly be called the FORTRAN stage, because it owes a great deal to FORTRAN and, to a lesser degree, the other languages that were in existence prior to the first ALGOL meeting in Zurich in 1958. The various theological periods of ALGOL's history can be classified as