ALGOL, more than just ALGOL

HT de Beer*

Abstract. *The ALGOL 60 report is regarded as 'highly influential'. In this article the history of the development of ALGOL 60 and its report is told to find out why this document was so influential.*

Introduction

On May 20th, 2006, Peter Naur received the 2005 ACM Turing Award. This award is often regarded as the "Nobel Prize" in Computer Science. Naur was rewarded for his 'pioneering work on defining the Algol 60 programming language. (...) [He] was editor in 1960 of the hugely influential "Report on the Algorithmic Language Algol 60." He is recognised for the report's elegance, uniformity and coherence, and credited as an important contributor to the language's power and simplicity.'[1]

More than forty-five years after the publication of the ALGOL 60 report it is still regarded as 'hugely influential'. At the same time the ALGOL 60 programming language defined in the report is almost forgotten. Where other languages defined around 1960 are still in use today, like FORTRAN, LISP and COBOL, only ALGOL's legacy is still with us, the language itself is long gone.

In my Masters thesis on *The History of the ALGOL Effort*[2] (2006) I argue that the ALGOL effort was a catalyst for the transformation of the field of compiler writing and programming languages into a scientific field. The ALGOL 60 report was the key to this transformation and this explains the influence of the ALGOL 60 report: its scope was much wider than yet another algorithmic programming language, it was about the definition of programming languages in general.

This generality of the ALGOL report was caused by the use of a new notation to define the syntax of programming languages: the BNF. The BNF was invented by John Backus in 1959 to define the syntax of the International Algebraic Language (IAL) and was used by Peter Naur for the definition of ALGOL 60. Apparently, using the BNF was the natural thing to do. Before, during, and after the ALGOL 60 meeting almost anything about ALGOL 60 was discussed in the ALGOL effort. The use of the BNF, however, was not a topic of discussion.

Although the generality of the ALGOL 60 report explains the influence of the ALGOL 60 report, it does not explain where it came from. Why was the ALGOL 60 report about more than just ALGOL 60? To answer this question the origin and the development of ALGOL 60 is described from the start

*Instituut voor Informatica, Universiteit van Amsterdam, The Netherlands, deBeer@science.uva.nl

[1] ACM Pressroom, 'Software Pioneer Peter Naur Wins ACM's Turing Award. Dane's Creative Genius Revolutionized Computer Language Design' (2006), `http://campus.acm.org/public/pressroom/press_releases/3_2006/turing_3_01_2006.cfm`

[2] HT de Beer, 'The History of the ALGOL Effort', Master's thesis, Technische Universiteit Eindhoven (2006), `http://www.heerdebeer.org/ALGOL`

in the middle of the 1950s till the publication of the ALGOL 60 report in March 1960.

The start of the ALGOL effort

In October 1955, an international symposium on automatic computing was held in Darmstadt, Germany. In the 1950s, the term "automatic computing" referred to almost anything related to computing with a computer. During this symposium 'several speakers 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.'[3]

To meet this need the Gesellschaft für Angewandte Mathematik und Mechanik (GAMM; association for applied mathematics and mechanics) set up a subcommittee for programming languages. This committee consisted of eight members: Bauer, Bottenbruch, Graeff, Läuchli, Paul, Penzlin, Rutishauser, and Samelson.[4] In 1957, it had almost completed its task; instead of creating yet another algorithmic language, however, it was decided to 'make an effort towards worldwide unification.'[5]

The need for a universal algorithmic language was not felt only in Europe. Some computer user organisations in the USA, like SHARE, USE, and DUO also wanted one standard programming language for describing algorithms. In 1957, they asked the Association of Computing Machinery (ACM) to form a subcommittee to study such a language.[6] In June 1957, the committee was formed consisting of fifteen members from the industry, the universities, the users, and the federal government: Arden, Backus, Desilets, Evans, Goodman, Gorn, Huskey, Katz, McCarthy, Orden, Perlis, Rich, Rosen, Turanski, and Wegstein.[7]

Before this committee had held any meeting at all, a letter of the GAMM subcommittee was send to the president of the ACM to propose a joint meeting to create one international algebraic language[8] instead of two different but similar languages. The ACM agreed and three preparatory meetings were held in the USA to create a proposal for the language. On the third meeting, held at Philadelphia, 18 April 1958, Bauer presented the GAMM proposal to the ACM subcommittee.[9] Both proposals shared many features but the American proposal was more practical than the European counterpart that was more universal.[10]

The IAL meeting at Zürich

From May 27 to June 1, 1958, the proposed joint meeting was held at Zürich, and was attended by F.L. Bauer, H. Bottenbruch, H. Rutishauser and K. Samelson from the GAMM subcommittee and by J. Backus, C. Katz, J. Perlis and J.H. Wegstein from the ACM counterpart. Although Bauer stated in his letter to the ACM that the GAMM subcommittee hoped 'to expand the circle through representa-

[3]Heinz Rutishauser, *Description of ALGOL 60*, volume 1, edited by F. L. Bauer et al. (Springer-Verlag, 1967), p. 5.

[4]Peter Naur, 'Transcripts of Presentations', in: *HOPL-1: The first ACM SIGPLAN conference on History of programming languages* (New York, NY, USA: ACM Press, 1978), p. 148, Frame 3.

[5]5 Rutishauser, *Description of ALGOL 60* , p. 5.

[6]R. W. Bemer, 'A Politico-Social History of Algol', in: Mark I. Halpern and Christopher J. Shaw, editors, *Annual review in automatic programming*, volume 5 (London: Pergamon, 1969), p. 160.

[7]Alan J. Perlis, 'The American side of the development of Algol', in: *HOPL-1: The first ACM SIGPLAN conference on History of programming languages* (New York, NY, USA: ACM Press, 1978), pp. 4–5.

[8]Bemer, 'A Politico-Social History of Algol', p. 160.

[9]Perlis, 'The American side of the development of Algol', p. 5.

[10]Rutishauser, *Description of ALGOL 60* , p. 5.

tives from England, Holland and Sweden'[11], the joint meeting was attended by Americans, Germans and Swiss only. Nonetheless, international interest was growing, especially after the publication of the preliminary report on the new language in 1958.[12]

The discussions at the Zürich meeting were based on the two proposals for the new language and it was decided that:

1. 'The new language should be as close as possible to standard mathematical notation and be readable with little further explanation.
2. It should be possible to use it for the description of computing processes in publications.
3. The new language should be mechanically translatable into machine programs.'[13]

In addition, the language was to be machine independent: it was supposed to be designed with no particular machine in mind.

Creating a language with these goals seemed a bit problematic: character sets available in hardware were much more limited than character sets needed in publications. In addition, there was an international disagreement on the symbols to use. For example, the decimal point was problematic: the Americans were used to a period and the Europeans to a comma.[14] To solve all these representational issues Wegstein proposed to define the language on three different levels of representation: reference, hardware and publication. With this 'master stroke'[15] the joint meeting ended successfully with the publication of the *Preliminary Report: International Algebraic Language*,[16] IAL was born.

Compared to other early algorithmic programming languages, like FORTRAN, IT, and MATH-MATIC, IAL did have some distinctive characteristics: it was designed by two committees at a joint meeting, it was the result of an international effort, and it was designed to be machine independent. On the other hand, IAL shared many features with these early algorithmic languages. There was a general agreement on what features an algorithmic language should contain: variables, assignment, expressions, selection, iteration, and maybe some sort of procedure concept.

Despite different backgrounds, the proposals from the GAMM subcommittee and the ACM subcommittee were similar. Nevertheless, because of these different backgrounds, the two subcommittees did have different goals with the new language. The Europeans wanted a language for numerical work that could be implemented and run on European computing machines.[17] Creating IAL was their first effort to create an higher level programming language and they tried to create a universal language from scratch.

[11]Bauer cited in Bemer, 'A Politico-Social History of Algol', p. 161.

[12]Rutishauser, *Description of ALGOL 60*, p. 6; Friedrich L. Bauer, 'Die Algol-Verschwörung', in: Hans Dieter Hellige, editor, *Geschichten der Informatik. Visionen, Paradigmen, Leitmotive* (Berlin: Springer, 2004), p. 246.

[13]A. J. Perlis and K. Samelson, 'Preliminary Report: International Algebraic Language', *Commun. ACM* 1:12 (1958), p. 9.

[14]Perlis, 'The American side of the development of Algol', p. 6.

[15]Ibid.

[16]Perlis and Samelson, 'Preliminary report: IAL'

[17]Alan J. Perlis, 'Transcripts of Presentations', in: *HOPL-1: The first ACM SIGPLAN conference on History of programming languages* (New York, NY, USA: ACM Press, 1978), p. 141.

The Americans, on the other hand, did have experience with the development and use of algorithmic programming languages. In fact, the existence of many similar but different algorithmic programming languages in the USA was the reason the ACM subcommittee was set up in the first place: they wanted to create a standard capturing all these different languages. Furthermore, they hoped that by developing this standard language they would get new insights in programming languages in general.[18]

The combination of these different perspectives resulted in a language where 'the central ideas [of earlier languages] were captured in general and elegant constructions – types, compound statements, conditionals, loops, switches and procedures.'[19] In other words IAL 58 became a standard denoting the state of algebraic programming languages of the late 1950s.

The development of ALGOL 60

The preliminary report on the international algebraic language spawned some new language efforts. In the USA, the report was taken and used as a set of guidelines to create new algebraic languages like NELIAC, MAD, and JOVIAL.[20] Furthermore, the development of IAL was continued in both the subcommittee of the ACM and the subcommittee of the GAMM. Meanwhile, the name of the language was changed from the '"unspeakable" and pompous acronym, IAL'[21] to ALGOL.

During the further development of ALGOL the focus of the Americans was mainly on practical aspects of the language. They wanted to improve the language by extending it, by adding more types, and by adding input and output facilities. Another suggestion to improve the language was to tidy up the syntax a bit.[22] This practical attitude to the ALGOL effort was a result of the state of programming in the USA: programming was becoming a professional field and the experience gained with existing programming languages provided a good feedback to the ALGOL effort.

In Europe, the GAMM subcommittee continued with the development of the ALGOL language. Soon, however, after two meetings where the implementation of ALGOL was discussed, in Mainz, November 1958, and in Copenhagen, February 1959, the ALGOL effort became truly international. Interested people from different backgrounds and different countries in Europe were invited to take part in the discussions on ALGOL.[23]

The discussions on ALGOL were documented in the newly founded *ALGOL Bulletin* edited by Peter Naur. This formal discussion channel was important to synchronise and to focus the discussions. During the development of ALGOL the number of different participants in the European part of the ALGOL effort had grown to include almost 50 centres and 74 persons;[24] management of the discussions was needed to enable meaningful participation, hence the influence of the ALGOL Bulletin.

In Europe, the procedure concept and the scopes of variables were the most discussed topics.[25] The

[18]Ibid., pp. 140–141.

[19]Idem, 'The American side of the development of Algol', p. 7.

[20]Ibid., p. 8.

[21]Ibid., p. 6.

[22]Idem, 'Transcripts of Presentations', p. 144.

[23]Peter Naur, 'The European side of the last phase of the development of ALGOL 60', in: *HOPL-1: The first ACM SIGPLAN conference on History of programming languages* (New York, NY, USA: ACM Press, 1978), pp. 19–20.

[24]Ibid., pp. 34–35.

[25]Perlis, 'The American side of the development of Algol', p. 9.

Europeans aimed to improve the language fundamentally and the main target was the complex procedure concept in ALGOL 58. The discussions on the procedure concept focused mainly on parameters. During these discussions, both in America and in Europe, all facets of ALGOL were discussed, from simple syntactical improvements to implementation and fundamental changes in the language.

Backus's notation

Meanwhile, at the UNESCO International Conference on Information Processing, held at Paris from 15 till 20 June 1959, J.W. Backus presented *The syntax and semantics of the proposed international algebraic language of the Zürich ACM-GAMM Conference*[26] about a formal description of the syntax of ALGOL 58. To be able to describe the syntax formally he invented a new metalanguage based on Emile Post's production system.[27] This notation became known as the Backus Normal Form and later as the Backus Naur Form,[28] it is, however, best known by its abbreviation BNF.

Backus started working on this notation because 'there must exist a precise description of those sequences of symbols which constitute legal [IAL] programs (...) [and] heretofore there has existed no formal description of a machine-independent language'.[29] Because of the nature of the ALGOL effort a formal notation was needed to properly discuss and define the language: everyone involved in ALGOL should interpret the language the same way for ALGOL to be universal.

Arithmetic expressions E are defined as follows:

a A number, a variable (other than Boolean), or a function is an expression.
Form: $E \sim N \quad \sim V \quad \sim F$

b If E_1 and E_2 are expressions, the first symbols are neither "$+$" nor "$-$", then the following are expressions:

$$\begin{array}{llll} E & \sim + E_1 & \sim & E_1 \times E_2 \\ & \sim - E_1 & \sim & E_1 / E_2 \\ & \sim E_1 + E_2 & \sim & E_1 \uparrow E_2 \downarrow \\ & \sim E_1 - E_2 & \sim & (E_1) \end{array}$$

The operators $+$, $-$, $\times$, $/$ appearing above have the conventional meaning. The parentheses $\uparrow\downarrow$ denote exponentiation, where the leading expression is the base and the expression enclosed in parentheses is the exponent.

Figure 1. The definition of arithmetic expressions in the IAL report (From: Perlis and Samelson, *Preliminary Report – International Algebraic Language*, pp. 12–13.)

The notation used to describe early programming languages like FORTRAN and IAL was natural language combined with some patterns denoting the form of the various language elements. In Figure 1, the definition of arithmetic expressions in the IAL report is given. In this definition the symbol $\sim$ is undefined but the meaning of this symbol is clear: on the right-hand side of the $\sim$ symbol all possible forms the left-hand side element can take are listed. In this example, all forms an arithmetic

[26]John W. Backus, 'The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference', in: *IFIP Congress* (1959), pp. 125-131.

[27]John Backus, 'Programming in America in the 1950s – Some Personal Impressions', in: N. Metropolis, J. Howlett and Gian-Carlo Rota, editors, *A History of Computing in the twentieth century* (Academic Press, 1980), p. 132.

[28]Donald E. Knuth, 'backus normal form vs. Backus Naur form', *Commun. ACM* 7:12 (1964), p. 736.

[29]Backus, 'The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference.', p. 129.

expression can take are listed, from a simple number, N, to a combination of subexpressions combined with an operator, like $\mathsf{E}_1 \times \mathsf{E}_2$.

The disadvantage of this notation was that it resulted in ambiguous descriptions. Even for simple language elements, like numbers, expressions, and simple control structures, this was problematic. For definition of complex structures, like the procedure statement and declarations, the notation was totally insufficient.

Using Backus's notation, however, the syntax of a language could be described by "production rules". Each rule was of the shape $<\mathsf{metalinguistic\ variable}> :\equiv \mathsf{pattern}$. A pattern was built up from metalinguistic variables and symbols of the language. All possible patterns for a metalinguistic variable were connected with the *or* symbol, denoting a choice between the different patterns for the metalinguistic variable.

$$\begin{aligned}
\langle\mathsf{factor}\rangle &:\equiv \langle\mathsf{number}\rangle \mathit{\ or\ } \langle\mathsf{function}\rangle \mathit{\ or\ } \langle\mathsf{variable}\rangle \mathit{\ or\ } \langle\mathsf{subscr\ var}\rangle \mathit{\ or\ } (\ \langle\mathsf{ar\ exp}\rangle\ ) \mathit{\ or\ } \langle\mathsf{factor}\rangle \uparrow \langle\mathsf{ar\ exp}\rangle \downarrow \\
\langle\mathsf{term}\rangle &:\equiv \langle\mathsf{factor}\rangle \mathit{\ or\ } \langle\mathsf{term}\rangle \times \langle\mathsf{factor}\rangle \mathit{\ or\ } \langle\mathsf{term}\rangle\ /\ \langle\mathsf{factor}\rangle \\
\langle\mathsf{ar\ exp}\rangle &:\equiv \langle\mathsf{term}\rangle \mathit{\ or\ } + \langle\mathsf{term}\rangle \mathit{\ or\ } - \langle\mathsf{term}\rangle \mathit{\ or\ } \langle\mathsf{ar\ exp}\rangle + \langle\mathsf{term}\rangle \mathit{\ or\ } \langle\mathsf{ar\ exp}\rangle - \langle\mathsf{term}\rangle \\
\langle\mathsf{ar\ exp\ A}\rangle &:\equiv \langle\mathsf{ar\ exp}\rangle \\
\langle\mathsf{relation}\rangle &:\equiv\ < \mathit{\ or\ } > \mathit{\ or\ } \leq \mathit{\ or\ } \geq \mathit{\ or\ } = \mathit{\ or\ } \neq \\
\langle\mathsf{rel\ exp}\rangle &:\equiv (\ \langle\mathsf{ar\ exp}\rangle\ \langle\mathsf{relation}\rangle\ \langle\mathsf{ar\ exp\ A}\rangle\ )
\end{aligned}$$

Figure 2. The formal definition of arithmetic expressions using Backus's notation. (From: Backus, *The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference*, p. 130.)

As an example of the use of Backus's notation, again the syntax of arithmetic expressions is given (Figure 2). Comparing Figure 1 with Figure 2, it is immediately clear that the latter is a less ambiguous description than the former description. Using his notation Backus was able to denote the operator precedence by splitting up the description of expressions into different parts: factors, terms and expressions.

The new notation was an huge improvement over the one used earlier. Nonetheless, it was improved even further by Peter Naur, replacing *or* by $|$ and $:\equiv$ by $::=$. With these small improvements, and with the use of complete words for metalinguistic variables instead of using abbreviations of the same words, as Backus did, Naur improved the readability of the description.[30] In Figure 3 the definition of the syntax of arithmetic expressions in the ALGOL 60 report using the BNF is given. This notation and description is similar with Backus's earlier description.

Peter Naur's draft

After a last meeting in December at Mainz, Peter Naur wrote a draft of the language discussed at this meeting to prepare himself for the Paris ALGOL 60 meeting. With this highly structured document and by describing the syntax using a slightly changed version of Backus's notation, the BNF, Naur wanted to convince the members of the ALGOL 60 committee to use his notation or a similar formal notation.

Naur's changes to Backus's notation were not fundamental. The important contribution of Peter Naur

[30] Backus, 'Programming in America in the 1950s – Some Personal Impressions', p. 133.

```
<adding operator> ::= + | −
<multiplying operator> ::= × | / | ↑
<primary> ::= <unsigned number> | <variable> | <function designator> |
(<arithmetic expression>)
<factor> ::= <primary> | <factor> ↑ <primary>
<simple arithmetic expression> ::= <term> | <adding operator><term> |
<simple arithmetic expression><adding operator><term>
<if clause> ::= if <Boolean expression> then
<arithmetic expression> ::= <simple arithmetic expression> |
<if clause><simple arithmetic expression> else <arithmetic expression>
```

Figure 3. The definition of arithmetic expressions in the ALGOL 60 report using the BNF. (From: Naur, ed., *Report on the Algorithmic Language ALGOL 60*, p. 17.)

to Backus's notation was his use of it in the ALGOL 60 report.[31] Only after the publication of that report the BNF became more widely known. Before the publication of the ALGOL 60 report Backus's notation 'was received with a silence that made it seem that precise syntax description was an idea whose time had not yet come'.[32] Naur 'thus proved the usefulness of the idea in a widely read paper and it was accepted'.[33]

Although Naur thought that there would be a problem with 'making the ALGOL committee actually use this [Backus] notation and the precise prose formulations that would have to go along with it'[34] the use of a better notation to describe programming languages was not a controversial topic. According to Bauer 'there was no question that we [Bauer, Rutishauser, and Samelson] would like for the outcome of the Paris Conference a form similar to the one Backus had used for [his] ICIP paper (some of us had been quite familiar with equivalent formulations in mathematical logic even before and did see the notational advantages.)'[35]

On the fourth day of the Paris meeting, Peter Naur's draft[36] was "chosen" as the basis for the discussions and hence it was "chosen" as the basis for the new language. As a result, Peter Naur was also "invited" to be the editor of the final report.[37] Bauer, on the other hand, is less positive about the choice made at the meeting to use Naur's draft and appoint him as the editor: 'Peter Naur had not been commissioned to do so, it was a fait accompli. It therefore sounds poetic (...) that his draft was "chosen" as the basis of the discussions; the Committee was simply forced to do so.'[38]

Notwithstanding these issues, Naur's draft was used as the basis of the final discussions at the ALGOL 60 meeting and Naur became the editor of the final report. Concequently, Naur's draft and the ALGOL 60 report were similar of structure and content. After an introduction of the ALGOL effort and the BNF, groups of different programming language elements were described using a general pattern:

[31]Knuth, 'backus normal form vs. Backus Naur form', p. 736.

[32]Backus, 'Programming in America in the 1950s – Some Personal Impressions', p. 133.

[33]Ibid.

[34]Naur, 'European side of development of ALGOL 60', p. 20.

[35]Bauer cited in Naur, 1978, p. 41.

[36]Peter Naur, 'ALGOL 60 Draft Report, 1960 January 9 [Regnecentralen, Copenhagen]', in: Peter Naur, editor, *Computing, a Human Activity* (ACM Press, 1992), pp. 67–88.

[37]Idem, 'European side of development of ALGOL 60', p. 21.

[38]Bauer cited in Naur, 1978, p. 41.

first, the syntax is defined using the BNF, then some examples are given, and finally, the semantics are treated.

This structure was adopted in the ALGOL 60 report. Actually, most of the prose and definitions in the draft were copied almost exactly into the final report. Of course, during the ALGOL 60 meeting in Paris the language and the description were changed here and there to reflect the discussions. Although the draft was copied for a large part, the final report contained much more than included in the draft: in the draft the more controversial aspects of the language were not defined, would be changed, or even removed.

Naur did not define the if-statement, the for-statement, the alternate statement, and the do-statement. He did define input and output statements which would not appear in the final report. Furthermore, the switch declaration was not included and the procedure declaration was incomplete. Finally, the procedure statement became a highly discussed topic at the ALGOL 60 meeting and the differences between Naur's version and the final version are large.

Naur's draft set the example for the structure, the use of the BNF, and the definition of simple language elements. The draft was a sound foundation for the ALGOL 60 meeting: many minor issues were covered by the draft and the participants of the meeting could focus on the complex and controversial programming language elements. Of course, by having such a structured and complete foundation, the discussions on those complex elements were influenced by the structure and the use of the BNF in Naur's draft.

While both Backus and Naur thought that the new notation was something controversial, it did not appear to be a problem at all for those present at the Paris meeting. The new notation was far more suitable for language definition than the one used earlier. Backus's notation or the BNF was the one available, more so because Peter Naur had written a draft using the BNF for the Paris meeting as part of his '"plan" to make an appeal to the members of the ALGOL Committee concerning the style of the language description'.[39] The notation was available, it was meaningful to use, and hence, it was used.

On the other hand, there is no trace at all that using the BNF was a topic of discussion before or during the ALGOL 60 meeting. It was just accepted as if using this new notation for ALGOL 60 was the most natural thing to do. And exactly this use of the BNF had an enormous impact on ALGOL 60 and programming languages in general: through the use of the BNF the ALGOL 60 report was not only about ALGOL 60 any more, although this was not recognised at that time.

In section 1.1 of the ALGOL 60 report, titled *formalism for syntactic description*, the BNF is introduced. The rest of the report, the definition of ALGOL 60, can also be seen as the prime example of the use and power of the BNF. Suddenly the ALGOL 60 report was not about yet another algebraic programming language, it was about a whole set of programming languages; it was about all languages that could be described using the BNF. The notional power of the BNF and its implications for the field of translator writing and programming languages were only to appear after the publication of the ALGOL 60 report.

What would have happened if Backus did not invent his notation or Naur did not prepare his draft for the ALGOL 60 meeting? This question can not be answered, of course. Still, it is unlikely that

[39]Bauer cited in Naur, 1978, p. 41.

ALGOL 60 would have been defined using a notation more formal than the one used in the ALGOL 58 report. The most influential aspects of ALGOL 60, the use of the BNF and the structure of the report, were more or less a coincidence. It is most suprising how the work and idea's of John Backus and Peter Naur were able to have such an enormous impact on the shaping of the field of programming languages.

The ALGOL 60 meeting at Paris

From January 11 till January 16, 1960 the joint ALGOL 60 meeting was held in the Hotel des Londres in Paris, hosted by IBM World Trade Europe.[40] According to Perlis (1978), 'The meetings were exhausting, interminable, and exhilarating. (...) diligence persisted during the entire period, The chemistry of the 13 was excellent. (...) Progress was steady and the output, Algol60, was more racehorse than camel.'[41] Instead of 'just adding a few corrections to ALGOL 58, it was necessary to redesign the language from the bottom up.'[42]During the meeting, several complex issues were discussed in subcommittees which, after completion of their work, reported to the whole committee.

One of the main topics at the Paris meeting was the complex concept of the procedure. Known and new problems of this concept were resolved by various subcommittees. First, the distinction between input and output parameters was removed.[43] This solved a number of problems but not all of them. Eventually, under great time pressure, the distinction between *call-by-name* (enabling the so-called Jensen's device) and *call-by-value* was invented.[44]

Let f, a procedure with one call-by-value parameter x, be called with argument m, a variable with value 3. Before the body of f is executed, the value of m is assigned to x. During the execution of the body of f, x is treated as local to that body. The call-by-value parameter concept in ALGOL 60 is similar with the parameter concept in mathematical functions: sin(m+3) is computed by first computing the value of m + 3 and then applying the function sin on that value.

If procedure f is adapted to have only one call-by-name parameter, y, and it is, again, called with argument m, all occurrences of y in the body of f are replaced with m. That is, not the value of m is substituted into the body, but m itself. Because an argument can be everything from a simple number up to a complex expression, using call-by-name parameters can result in programs which are difficult to understand. Through a call-by-name parameter an variable can be changed outside of its natural scope. As a result, this call-by-name parameter concept was one of the most controversial features of ALGOL 60.

Another issue at the Paris meeting was recursion. Recursive procedures were new in 1960 and their usefulness was not widely recognised. The proposal to add the **recursive** keyword to the language to denote a recursive procedure was rejected.[45] It is unclear, however, if recursion as such was rejected, or that only the use of the keyword **recursive** was rejected.[46]

Nevertheless, in the report resulting the the ALGOL 60 meeting, recursion was not mentioned at all,

[40]Perlis, 'The American side of the development of Algol', p. 11.

[41]Ibid., pp. 11–12.

[42]Rutishauser, *Description of ALGOL 60* , p. 7.

[43]Naur, 'Transcripts of Presentations', p. 147.

[44]Ibid., pp. 157–158.

[45]Perlis, 'Transcripts of Presentations', p. 159.

[46]Naur, 'Transcripts of Presentations', p. 160.

it was not explicitly forbidden, nor described explicitly as a feature in the language. Through the definition of the procedure concept, however, 'recursiveness [was] almost the obvious thing, because you have access from inside the body to anything outside [including] the possibility of [a] recursive call.'[47] After the ALGOL 60 meeting, van Wijngaarden en Dijkstra in Amsterdam did see the implication of this definition of the procedure concept and asked Naur by phone to add to the final report that recursive calls were possible.[48] According to Bauer (1978), this was the result of 'the Amsterdam plot in introducing recursivity.'[49]

As said before, on the fourth day, Naur's draft was chosen as the basis for the final language and Naur decided on how to proceed during the rest of the meeting.[50] The meeting now continued by first reading and discussing complete parts of the language without changing it. Then, in the next phase, every member would write down the changes he proposed in a form similar to the form used in the draft. After collecting these proposals, the whole committee voted on them successively.[51] At the end of the meeting an almost complete report was ready for the final editing by Peter Naur.

However, twelve hours after the meeting ended, Naur thought that he had found some inconsistencies in the procedure concept. To solve these issues, the discussions were continued by mail. It appeared that the topic was truly controversial: accepted proposals were not understood (but by the author) and rejected proposals seemed to be the only coherent alternatives. Fortunately, one of the proposals made by mail solved the problem. In February the last changes were made to enable recursivity and some example programs were added too. Finally, at 2 March 1960, the final report was published.[52]

The *Report on the Algorithmic Language ALGOL 60* 'was a fitting display for the language. Nicely organised, tantalisingly incomplete, slightly ambiguous, difficult to read, consistent in format, and brief, it was a perfect canvas for a language that possessed those same properties. Like the Bible it was meant, not merely to be read, but to be interpreted.'[53] At that time, the formal notation used in the ALGOL 60 report was frightening many people at first. Nevertheless, the way ALGOL 60 was defined in this report would become the standard for defining programming languages.

ALGOL, more than just ALGOL

Now that the story of the development of ALGOL 60 is told, it is time to answer the question why the ALGOL 60 report was about more than just ALGOL 60. The introduction and use of the BNF by Backus and Naur is an important part of the answer. Without it, it is doubtful if the ALGOL 60 report would have had the same impact, or if ALGOL 60 itself would have been as structured and balanced as it was.

The use of the BNF is not the whole answer, however, a good idea alone is not enough. Remember, initially Backus's notation did not gain much attention. The same could have happened to the ALGOL 60 report, but for the nature of the ALGOL effort. ALGOL 60 was not yet another programming language, it was aspired to be the new universal and international algebraic programming language;

[47]Ibid., p. 159.

[48]Ibid.

[49]Bauer cited in Naur, 1978 p. 160.

[50]Naur, 'European side of development of ALGOL 60', p. 21.

[51]Ibid., p. 1.

[52]Ibid., pp. 29–30.

[53]Perlis, 'The American side of the development of Algol', p. 12.

it was born to great expectations.

The other part of the answer why ALGOL was more than ALGOL lies with the nature of the ALGOL effort. It was an international effort, combining scientists of two worlds working together on one common goal: to define a programming language close to mathematics, a language for use in publications, a language that was machine translatable, and a language that was machine independent. The first and the third goal were common goals for other algebraic language efforts, the other two goals were not. The initial result, ALGOL 58 was a standard denoting the state of algebraic programming languages in the late 1950s. It was a perfect start for a new language, it was a perfect start for further discussions on programming languages.

During 1959 ALGOL was discussed among a growing number of interested people and the ALGOL effort became a truly international effort. The result of these discussions was put together at the joint ALGOL meeting in Paris, 1960. Where ALGOL 58 denoted the state of algebraic programming in the late 1950s, ALGOL 60 was something new. With new or improved programming language concepts like the block, recursion, and procedures and with a clean and structured syntax, ALGOL 60 started a new generation of programming languages.

Suddenly ALGOL became a popular topic, and, as a result, the ALGOL 60 report was read by an huge number of interested people from all over the world. The ALGOL effort was the perfect bandwagon for the spread of new ideas like the BNF and a programming language like ALGOL 60. It became the bandwagon of the field of translator writing and programming languages and the ALGOL 60 report became the key to the transformation of that field from a craftsmanship into a scientific field.

References

[1] Backus, John, 'Programming in America in the 1950s – Some Personal Impressions', in: Metropolis, N., Howlett, J. and Rota, Gian-Carlo, editors, *A History of Computing in the twentieth century* (Academic Press, 1980), pp. 125–135.

[2] Backus, John W., 'The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference', in: *IFIP Congress* (1959), pp. 125–131.

[3] Bauer, Friedrich L., 'Die Algol-Verschwörung', in: Hellige, Hans Dieter, editor, *Geschichten der Informatik. Visionen, Paradigmen, Leitmotive* (Berlin: Springer, 2004), pp. 237–254.

[4] Beer, HT de, 'The History of the ALGOL Effort', Master's thesis, Technische Universiteit Eindhoven (2006), `http://www.heerdebeer.org/ALGOL`.

[5] Bemer, R. W., 'A Politico-Social History of Algol', in: Halpern, Mark I. and Shaw, Christopher J., editors, *Annual review in automatic programming*, volume 5 (London: Pergamon, 1969), pp. 151–237.

[6] Knuth, Donald E., 'backus normal form vs. Backus Naur form', *Commun. ACM* 7:12 (1964), pp. 735–736.

[7] Naur, Peter, 'The European side of the last phase of the development of ALGOL 60', in: *HOPL-1: The first ACM SIGPLAN conference on History of programming languages* (New York, NY, USA: ACM Press, 1978), pp. 15–44.

[8] Naur, Peter, ‘Transcripts of Presentations’, in: *HOPL-1: The first ACM SIGPLAN conference on History of programming languages* (New York, NY, USA: ACM Press, 1978), pp. 147–161.

[9] Naur, Peter, ‘ALGOL 60 Draft Report, 1960 January 9 [Regnecentralen, Copenhagen]’, in: Idem, editor, *Computing, a Human Activity* (ACM Press, 1992), pp. 67–88.

[10] Perlis, A. J. and Samelson, K., ‘Preliminary Report: International Algebraic Language’, *Commun. ACM* 1:12 (1958), pp. 8–22.

[11] Perlis, Alan J., ‘The American side of the development of Algol’, in: *HOPL-1: The first ACM SIGPLAN conference on History of programming languages* (New York, NY, USA: ACM Press, 1978), pp. 3–14.

[12] Perlis, Alan J., ‘Transcripts of Presentations’, in: *HOPL-1: The first ACM SIGPLAN conference on History of programming languages* (New York, NY, USA: ACM Press, 1978), pp. 139–147.

[13] Pressroom, ACM, ‘Software Pioneer Peter Naur Wins ACM’s Turing Award. Dane’s Creative Genius Revolutionized Computer Language Design’ (2006), `http://campus.acm.org/public/pressroom/press_releases/3_2006/turing_3_01_2006.cfm`.

[14] Rutishauser, Heinz, *Description of ALGOL 60*, volume 1, edited by Bauer, F. L. (Springer-Verlag, 1967).