AB52.4.3

A Browse through some Early Bulletins.

by C. H. Lindsey

(University of Manchester)

After IFIP WG2.1 had been formed (initially from the original authors of ALGOL 60) a decision was taken in March 1964 to revive the ALGOL Bulletin, which had lain dormant since the publication of the Revised Report on the Algorithmic Language ALGOL 60 in 1962. Fraser Duncan was appointed as Editor, and AB16 duly appeared in May 1964. As present editor of the AB, I have no access to any issues prior to AB16, but I have managed to piece together a complete set since that date, and they form a fascinating account of what was going on in those years. The following article surveys some of the material published between 1964 and 1972.

ALGOL 60

Of course, ALGOL 60 was not frozen with the publication of the Revised Report in 1962. Much remained to be done as regards subsets, I-O, problems in the Report, and in just trying to understand the beast that had been created. It should be realised that many features in ALGOL 60 seem to have "just happened" and their ramifications (even their implementations) only became apparent as time went on. As John McCarthy said on one occasion, the authors of the original Report were all gentlemen, and did not propose any feature for inclusion that they did not see how to implement sensibly. It was the interactions between the various features which were not so well understood at the time.

Block Structure and Environments.

One of the novel features of ALGOL 60 was, of course, block structure, and the idea that procedures lived in definite environments. This was reasonably well understood by some people by 1964, but not always by implementors. Knuth's famous test case "Man or boy?" appeared in [AB17.2.4] (July 1964 - note how close together issues were in those days). Originally, of course, it was written in ALGOL 60 and used call-by-name, but here it is in ALGOL 68.

```
BEGIN
    PROC a = (INT k, PROC INT x1, x2, x3, x4, x5) INT:
      BEGIN
          LOC INT kk := k;
          PROC b = INT:
            BEGIN
              kk -:= 1;
              a(kk, b, x1, x2, x3, x4)
            END;
          IF kk<=0
          THEN x4 + x5
          ELSE b
          FI
      END;
    print( a(10, INT: 1, INT: -1, INT: -1, INT: 1, INT: 0)
END
```

The point about this program, of course, is that many incarnations of *b* are created in many environments, each of which is able to decrement the particular *kk* in the environment where is was created. Readers are advised NOT to try computing the result by hand. Knuth tried and obtained the result *-121*, which is wrong (at the time, he had broken his right wrist so, as he said [AB19.2.3.4], the calculations were done left handed - but he did then give a formula from which the corect result of any example could be calculated). No wonder he got it wrong! This case recurses

to a depth of 512 in *a* and 511 in *b*. The correct results for various values of *k*, computed on the Electrologica X1 at the Mathematisch Centrum appeared in [AB18.2.5].

k	0	1	2	3	4	5	6	7	8	9	(10	11)
A	1	0	-2	0	1	0	1	-1	-10	-30	(-67	-138)

The X1 actually ran out of its memory (12K of 27-bit words) on *k=10* (memory doubles for each increment of *k*) so the result for *k=10* was from a machine at Kiel and for *k=11* on a KDF9 using the Kidsgrove compiler, which took 12 seconds [AB19.2.3.1]. Other times reported were 1.75 seconds on the ICL Atlas (*k=11*) and times of 20 and 80 minutes on two other machines whose anonymity the Editor agreed to respect [AB20.2.4]. By way of comparison I just ran a PASCAL version of *k=11* on a SUN3, and it took 0.027 seconds.

As a postcript to this whole episode, it may be noted that a couple of years later Bekic was able to use this example to persuade the designers of PL/1 (who really did not understand environments) to mend their ways.

As an aid to understanding those environments, a piece of Science Fiction appeared in [AB17.2.5], written by W. H. Burge, Manager of Systems Programming Research at the Univac Division of Sperry Rand, New York. I think it is worth reproducing it in full.

THE ALGOL MEN

They didn't know where they came from when they were born, they called it "Outside". They did not know when a new person would appear, but noticed that they only appeared when space was available. All people were born with the same amount of experience and property which they called an "environment". When they died, they disappeared, and so did their environment.

There was a ritual called "enter block" which they could perform in order to use a new body. They stored what was left of the old body and its environment secretly, so that no one could interfere with it, and lived their life in a new body. The new body came with a certain amount of experience and property called its "locals". This was added to the environment of the old body to produce an environment for life in the new body.

When an incarnate body died, there was another ritual called "block exit" in which the old body was disinterred, and life continued in this body where it had left off. The incarnated body disappeared together with its locals.

The process of incarnation was difficult and required much concentration. There were many people who were born and died without achieving it. These were called "FORTRANS" or "COBOLS". Some of these were compensated for this by having other abilities, for instance, the FORTRANS could run fast and the COBOLS could describe data by a carefully guarded technique only known to them.

Some people who were lucky found that among their possessions in their environment were bodies called "procedure bodies". They could live in these at will by using the "enter block" ritual. It was possible to pass property on to be used in their lives as procedure bodies. This property was called an "argument".

It required considerable training to keep track of all the interred bodies, and to choose the correct one to reincarnate by the block exit ritual. This was made even more difficult by the existence of so-called "recursive" procedure bodies or Doppelgängers. These bodies found themselves on their own environment. It was clear that when a body incarnated itself, it could not both bury and use its body (the technique of copying bodies was known but its use was deprecated because it wasted space) and so instead of burying the body, they buried information which said where it was.

When a person got tired of his present life and hankered after one of his previous incarnations, he could invoke a "go to" ritual which would take him back to one of his previously stored bodies. This would disinter and reactivate a body which was not necessarily the last one used. This process invariably lost all the intervening bodies. This go to ritual occasionally surprised people. They would enter a procedure body in order to gain something of value, and find themselves projected by a go to into a previous incarnation, instead.

The ALGOL Men were not only able to change their immediate environment or locals but could also change the environments of their interred bodies. Some people thought it was sacrilege to interfere with the possessions of their interred bodies (or non locals), especially when living in procedure bodies. Others thought this practice useful, and used it.

It seemed that the course of a person's life was mapped out for him in advance, although they couldn't tell for sure. Some said that they must do the things they did in a fixed order, others said that the order was not fixed. Occasionally some of the latter found that changing the order made them do things they didn't intend to. Others argued that they had free will because they could make decisions. Although they tried hard, they could never manufacture a new body. The bodies and environments which appeared seemed to have been lying dormant in their genes, waiting to be activated.

The ALGOL Men had a great book which contained guidance called the "[Revised] Report on the Algorithmic Language ALGOL 60". This contained rules and parables which stated how they should behave, and commandments which prohibited certain behaviour. The interpretation of certain sections of this Report was the subject of much scholarly and theological argument. Part of the Report was written in a peculiar language called "Backus normal form". No one had ever deciphered this part, but it seemed to be explaining how their bodies were constructed. The Report prescribed that if certain rules of behaviour were disobeyed then the offender would be put into a state called "undefined" or "hell". There were legends about certain adventurous spirits who, because they possessed some defect of character, had tried these prescribed acts and as a result had vanished together with all their interred bodies and possessions.

People who were deep in incarnations used a lot of space to hold their bodies and possessions and as a consequence people were taxed on how deep they were. Sometimes people would wish to incarnate but find they could not because there was no space left on the world. They had to go into a state of "suspended animation" waiting for space to become available.

There were some people who lived the same parts of their life over and over again. Some of these were harmless and were called "ghosts". There seemed to be no way of finding out whether they would ever break out of their loop or not. Others were dangerous to the community and were called "space thieves". For these each new life cycle produced new possessions and these possessions threatened to overrun the world. A rule was introduced to detect space thieves and throw them Outside. This was called a "debugging rule". There was a similar rule against "body snatching". This rule said that if any person interfered with the interred bodies or property of another then both would be thrown Outside. A person was said to be "running wild" when he did this. This was unfortunate for the victim but his removal was a safety precaution. Since he had been interfered with, there was the chance that he too might run wild. Later, special locks were provided for the graves called "lockouts". These locks could only be opened by the owner. This prevented the whole population from running wild.

A movement sprang up to hoard property in a body's life so that it should be available if that life were to be re-entered. This hoarded property was called the "own" property of the body. There was some confusion about the precise procedure to be follwed when hoarding and there was little guidance in the Report about how to do this. Some people, because they used incorrect methods, produced hoards which they never used again.

There were tales that certain people, called "mystics", were able to obtain information from Outside to guide their lives, and that there were poeple who could transmit information to the Outside by prayer. The Report, however, does not mention this. Some people found themselves incarnated into another form when they used what are called "non-ALGOL code" procedure bodies. Although this is mentioned in the Report, no details are given, and people were constantly surprised by the forms they took.

It is a shame that this brief account cannot include descriptions of other tribes similar to the ALGOL Men such as the Macro People who constructed bodies (or Frankensteins) and activated them, the Ipulvees who listed their property and had an oracle called the Interpreter, and the Lisps who constructed magnificent structures and took them to pieces, and had a strong garbage collectors' union.

The tragedy of the ALGOL Men was that they could not communicate with one another nor could they store things which would be useful to future generations. All men were born equal but did different things with their lives. When a man died he vanished, left nothing, and released the space he occupied. It is clear that what these people needed was the ability to store things produced by people in their lives in a library so that other people could make use of them. It is said that the FORTRANS and COBOLS have a tradition of this sort.

ALGOL 60 Developments.

The two great issues with regard to ALGOL 60 were Subsets, and I-O.

The "Report on SUBSET ALGOL 60 (IFIP)" appeared in AB16.3.1.1, complete with approval from the Coucil of IFIP and all the usual trimmings including permission to reproduce, but only in full. The Subset was in fact a dramatic piece of surgery, and a less drastic subset was produced by ECMA, the European Computer Manufacturers' Association (historically, the ECMA subset actually came before the IFIP one, however IFIP < ECMA < full ALGOL 60). For the ECMA subset, and a discussion of the whole issue, see [AB20.2.5] (July 1965).

Firstly, here are the restrictions common to both subsets:

- disappearance of own
- disappearance of integer labels
- <formal parameter>s to have their types specified
- types of all expressions to be compile-time determinable
- no lower-case letters
- go to an undefined <switch designator> to be undefined, not a dummy

Those all seem quite innocuous, but the next two will be surprising to modern readers, and reflect the fact that many compilers written up to that time had taken these shortcuts.

- no recursion
- only the first 6 characters of an identifier to be significant

Secondly, here are the additional restrictions in the IFIP subset. Some of these were made for reasons of tidiness rather than difficulty of implementation.

- an identifier may not appear twice in a <formal parameter list>
- no for *a[i]* := ... do ...
- no go to if *B* then *L1* else *L2* (and similarly in <switch-list>s)
- no switch *s* := *L1*, *t[i]*, ... (where *t* is another switch)
- no raising of integers to -ve powers
- no integer division
- actual-parameters called by name to be variables (but, not only does this prevent *jensen(i, i↑2)*, it also prevents *read(a[i])*)
- functions to have no side effects

Note the passion people had in those days for efficient implementation of extraordinary *go to*-like constructs which would never even be admitted into a modern language. Note also that many of these restrictions were rejected by ECMA because they eliminated more than they strictly needed to.

The "Report on Input-Output Procedures for ALGOL 60" appeared at the same time [AB16.3.1.2]. Its provisions were extremely primitive, but were only intended as a basis on top of which libraries of more useful facilities could be constructed. The Report blithely concluded by saying that "WG2.1 does not propose any further means for input-output operations". In the meantime, a separate proposal for Input-Output Conventions in ALGOL 60 had been prepared by an ACM committee chaired by Donald Knuth and published in CACM 7 (1964). This was at the opposite extreme, being format-based with every possible bell and every possible whistle (anyone wanting to know whence the formatted transput in ALGOL 68 originated need look no further). Of course this lead to complaints and counter proposals, including pleas by Naur [AB19.3.11.2, AB20.3.2.1] and Garwick [AB19.3.8] to separate number conversion from the actual I-O and a full-blown 15-page alternative scheme from the IBM Vienna Lab [AB20.3.5], and a mere 11 pages from ECMA [AB27.3.1]. A common factor of most of these proposals, judging by some of the comments made, seems to have been a lack of provision for recovery from input errors.

The ALGOL 60 Standard.

There now commenced that struggle to get ALGOL 60 adopted as an ISO Standard. It all started in fine style in May 1964 [AB17.1.1] with a report that ISO/TC97/SC5 had decided to "proceed immediately with ... an ISO Draft Proposal". This was to include the Report, the IFIP Subset, both the IFIP and the Knuth I-O, and a hardware representation. All very straightforward and timely. Of course, we now know with hindsight that this process was not finally completed until 20 years later. By October 1964 a First Draft was being circulated [AB18.1.3] – and the ECMA subset had crept in. After further meetings in September and October 1965 [AB22.2.1], it was all ready for final approval – bar the hardware representation. After that – silence! What happened? Apparently, there was a further draft in April 1967 which was finally approved in April 1968. Next, the text got lost in the ISO system. And then the text got mangled by ISO bureaucrats who didn't understand what they were doing, and it was finally published, complete with omissions and errors, in March 1972. There followed much argument between WG2.1 and ISO which culminated in the withdrawal of the document in 1976. After that, a mere 8 years to get another Standard (now based on the Modified ALGOL 60 Report) through the system is neither here nor there.

ALGOL 60 Trouble Spots.

Of course there were always worries about what the ALGOL 60 Report really meant. In January 1965 [AB19.3.7], Knuth published his famous "List of the remaining trouble spots in ALGOL 60" (later published in CACM 10 October '67), which documents the well-known problems with regard to such things as numeric labels. It was not proposed to fix these things at that time (the controversies still raged), but rather to point out to users features of the langauge to steer clear of. Of course, this was not the end of the worries. Further worries were reported by Bekic [AB20.3.6] and Medema [AB20.3.7]. In [AB27.2.1] Bryan Higman pointed out the reason why making the value of the controlled variable undefined on exit from a <for statement> achieved precisely nothing for the implementor, since its value at the end of each <for list element> must be defined in case it is needed in the next one, as in

```
    for i := 1 step 2 until 9, i+1 do print(i)
```

which must clearly print *12* on the last iteration.

Miscellany.

Here, from J. Nievergelt [AB31.3.5] is what I regard as the ALGOL 60 equivalent of the FORTRAN Venus Probe Joke (you know, the one that starts *DO 1 I = 1.2 ...*). The difference between the two following is just one ";" – clearly just a <dummy statement> which will make no difference to the meaning.

```
    begin procedure p(k) ;   integer k ; k := 1 ; end

    begin procedure p(k) ;;  integer k ; k := 1 ; end
```

It ain't so simple, however. You haven't spotted it? Clue: where does the declaration of the *procedure* p finish?

A rather unlikely semantic problem was reported in [AB26.1.3]. In England, the newly installed automatic barriers on railway level crossings were provided with a warning

"Stop while lights are flashing."

But it seems that the local dialect in parts of Northern England, particularly in Yorkshire, ascribes to "while" the meaning which the rest of the world understands by "until", with the obvious possibility of disastrous accidents at level crossings. The wording on the notices had had to be changed. Many AB readers lived in Yorkshire, and there were expatriate Yorkshiremen throughout the world, so perhaps there were problems with the *while* of ALGOL 60? As a Northener myself (but from Lancashire rather than Yorkshire) I can assure you that "while" is indeed often used with an "until" meaning (I even do it myself in the right company), but the ambiguity can in fact always be resolved by the context.

Another nice touch (with hindsight) was a news item [AB20.1.1] entitled "Release of syntactic ALGOL compiler for PASCAL". Mystified? Well, this was 1965, and it seems that Philips manufactured a machine called "PASCAL" in those days.

Another long-standing tradition was established in [AB27.3.2] (see also [AB28.2.5, AB29.2.1]) with a paper by Brian Wichman on "Timing ALGOL Statements". This was, of course, many years before his invention of the "Whetstone", the systems compared here being the two KDF9 compilers, the ICL 1905, the Elliott 4120 and 4130, and the CDC 3600. Timings for $x := y \uparrow z$ ranged from 99 μ-secs to 47700 μ-secs.

ALGOL X and ALGOL Y.

It was always the intention of WG2.1 to proceed to a more advanced language, and the first mention of ALGOL X (which eventually became ALGOL 68) and of the mythical ALGOL Y (originally conceived as a language which could manipulate its own programs, but in fact degenerating into a collection of features rejected for ALGOL X) was in a paper entitled "Cleaning Up ALGOL 60" by Duncan and Van Wijngaarden [AB16.3.3] which proposed a type *string*, which is reasonable enough, but also a type *label* and a removal of the restriction forbidding a *go to* from outside into a block (clearly, such things were not considered in the least bit harmful in those days).

It should be noted that it was always the intention in those early days for ALGOL X to be a strict upwards extension of ALGOL 60. It was only gradually that the folly of this view became plain.

Features Proposed for ALGOL X.

There followed a long series of wish-lists for the new language, of which a long series of important articles by Tony Hoare will be examined in more detail below. A long list by Peter Naur [AB18.3.9] asked for environment enquiries, *short* and *long* modes, operator-declarations, *string*s (but crude), yet more *label*s and *switch*es, non-rectangular arrays (they nearly did make it into ALGOL 68), operators such as *mod*, *abs*, *round*, *odd*, procedures with variable numbers of parameters, a hint of *struct*s, and [AB19.3.11] a type *character* coupled with a separation of conversion from I-O, and a type *bits*, and [AB22.3.7] a suggestion to replace the ALGOL 60

procedure *p(a,b)*; *integer* *a*; *real* *b*; ... by

procedure *p*(*integer* *a*, *real* *b*); ...

(I am amazed that such a, now generally taken for granted, feature should have been so late in appearing). Quite a lot of familiar stuff there! The wish list of the ALCOR group [AB19.1.1] mentioned complex arithmetic, variable precision, strings, "simultaneous statements", simpler loop-statements, restricted call-by-name, collateral assignment of array elements, and much else besides. Rutishauser [AB19.3.10] was asking for elaborate mechanism to denote lists, with associated features in the for-statement. Van de Laarschot and Nederkorn [AB19.3.2.1] wanted to ensure that *string*s would be first class citizens (no length limitations, proper assignments, etc). Samelson [AB20.3.3] wanted anonymous routines (i.e. λ-expressions or routine-texts) chiefly so that they could be in-situ actual-parameters. In [AB21.3.1] Seegmüller proposed *reference* types (principally as an aid to parameter passing) but, because coercion had not been invented yet, he needed a special "undereferencing" operator *ref*. Thus *integer* *reference* *ii*; *integer* *i*; *ref* *ii* := *ref* *i* (to assign a reference to *i*), or *ii* := *i* (to assign the *integer* in *i*). Genuine coercion (or at least the widening coercion as we now know it) came from Dijkstra [AB21.3.3]. David Hill [AB22.3.9] was pressing for the "operate and becomes" operators, such as "+:=" (except that they were spelt "+:"). O-J Dahl [AB24.3.5] and Král [AB25.3.2] made strong pleas for Multi-programming (though I doubt whether the *par* clause of ALGOL 68 was quite what they had in mind).

Tony Hoare's contributions.

A series of articles by Tony Hoare had a great influence on the development of ALGOL X, and they are worth looking at in more detail.

Case expressions (and statements) [AB18.3.7]. These were just like the case-clauses that eventually got into ALGOL 68, except that the alternatives of the list were separated by *else*s and there was no *out* part (the effect of an out-of-range *case* being undefined, even for <case statement>s). The intention was most definitely to provide an alternative to the ALGOL 60 *switch*es.

Record handling [AB21.3.6]. Records were conceived much as the *struct*s in ALGOL 68 to which they gave rise. However, they had three substantial (and deliberate) restrictions which ALGOL 68 does not impose.

1. Records could only be created, on demand, on the heap. There were to be no locally declared record variables. Thus there were to be *reference* variables and *reference* fields and these were the sole methods of accessing records.
3. However, *reference* values could not point to local variables (so that no scope problems could arise).
2. Records could not have other records as fields (although *array* fields were envisaged).

The proposal envisaged both a garbage-collector and a PASCAL-style *destroy*. There was also provision for *reference*s to *union*s of other types, and a special construction equivalent to the ALGOL 68 conformity-clause for taking the *union*s safely apart. There was further discussion of these ideas in [AB23.3.2] which included, by way of an example, the earliest publication known to me of Dijkstra's well-known algorithm for finding the shortest path between two nodes of a graph.

It is interesting to trace the origins of these ideas for records and references to them. Already, in [AB18.3.12], McCarthy had proposed *cartesian*s (records) and *union*s, but no dynamic allocation and no *reference*s. The first language really to provide complex data structures out of records and pointers was Doug Ross' AED-0. SIMULA 67 also provided much input to Hoare's proposal.

It would seem that the concept of *reference* and its relationship to records, variables and procedure parameters caused much discussion within the Working Group much if it based, so far as I can see, on the difficulties of recognising the underlying fundamentals and inventing suitable terminology for them. A letter from Doug Ross [AB26.2.2] to Van Wijngaarden illustrates this point. It discusses various features, most of which I can recognise in ALGOL 68, but I am sure that Doug thought he was proposing something radically different.

A separate proposal, also contained in Hoare's paper, was for enumeration types (as now provided in PASCAL, but originally introduced with the word *set*).

Cleaning up the for statement [AB21.3.4]. This proposed implicit declaration of the control variable by its mere mention after a *for*, it was to behave like a *value* parameter (so it could not be assigned to), and the expressions for the initial, step and final values were to be evaluated only once. This all seems very familiar today, but the story did not end there. In [AB22.3.1] Galler suggested that the for statement should become an expression, returning the final value of the control variable (I think I like this), and in [AB25.3.2] David Hill was complaining that making the control variable an implicit declaration would scupper Jensen's Device.

File processing [AB25.3.3]. *file*s were to be ordered sequeces of *record*s, and in its initial form the proposal provided the effect of the PASCAL *file* *of* *sometype*. However, there was also to be provision for different record types within the one file, and means to locate specific records (and even to store such locations as fields of other records).

Set manipulation [AB27.3.4]. This proposal was the forerunner of the *set*s in PASCAL, it being explicitly envisaged that they would be stored as bit patterns. However, it went somewhat beyond what eventually appeared in PASCAL, for example there was to be an operation to iterate through a set, and a *shift* operator. Hoare's proposal was roundly criticised by Landin [AB27.3.6] because it was too ad hoc – being designed as a clever way of using bitpatterns rather than an attempt to embody the mathematical concept of sets. However, the interesting part of Landin's critique was his (then) novel idea of trying to identify types with the operators that could be applied to them, complete with an axiomatic description of the type *integer*.

Subscript optimisation and subscript checking [AB29.3.6]. The intention was to allow, for some array *A*,

for *a* *in* *A* *do* *sum* := *sum*+*a*;

The motivation was to optimise the efficiency of loops that scanned arrays and, in particular, to make it impossible for subscript bound errors to arise, at the same time eliminating (or much reducing) the necessity for such things to be checked at run time.

Text processing [AB29.3.7]. I do not undersdtand this proposal. It suggests a type *character* and the handling of textual data in *character* *array*s, but these are conceived as of fixed size and their is no discussion of any need for strings of arbitrary length such as the real world is full of. There are proposals for slicing such arrays (just as in ALGOL 68), and also for fixed collections of characters like the ALGOL 68 type *bytes*. All this seems to be so far behind ALGOL 68 as it then stood (MR93 had been published and Tony was well aware of it) that, even though he disliked ALGOL 68, the lack of discussion – critical if necessary – of these issues I find very odd.

The History of ALGOL X.

The Early Days.

Work on ALGOL X started in earnest at the Princeton meeting of WG2.1 in May 1965 [AB21.1.1]. *strings* were seen as important, but a small majority preferred to compose them out of *characters* using existing data structuring tools. Trees were envisaged (but note that this was before Hoare's Records paper). There was considerable influence from Wirth's Euler language, but still confusion about parameter passing. Strong typing was clearly envisaged. Draft proposals for a full language were solicited for the next meeting.

At the next meeting in October 1965 at St Pierre de Chartreuse [see excellent report by Mike Woodger in AB22.3.10], there were three such drafts on the table, by Wirth (with extensive comments by Hoare incorporating his Record Handling), by Seegmüller, and by Van Wijngaarden – the famous "Orthogonal design and description of a formal language" wherein W-grammars first appeared (did you know that metanotions originally consisted of just one capital letter and that there were exactly 26 of them). The four of them (Van Wijngaarden, Wirth, Seegmüller and Hoare) were commissioned "to agree among themselves" and to produce a proposal for "final approval" (sic) at the following meeting. BUT, it had also been decided that Van Wijngaarden's method of description should be used, so that he would get to write the text. Nevertheless, although Van Wijngaarden's proposal was more concerned with method of description than with language content, it seems that the Wirth/Hoare proposal had been effectively rejected and permission was given for it to published independently, which was done in CACM in June 1966, together with the remark that it had been felt that "the report did not represent a sufficient advance on Algol 60, either in its manner of language definition or in the content of the language itself". The CACM version did, however, use just a little bit of 2-level grammar, with acknmowledgment to Van Wijngaarden's document.

Decisions made at this meeting were
- *label* variables were out. *strings* (but maybe with declared maximum length) *complex*, *bits* and various *long* modes were in.
- Hoare's Records were accepted.
- Row-displays of some form were envisaged.
- Collateral elaboration to discourage side effects.
- Parrellel-clauses (but no semaphores as yet).
- Blocks (containing declarations and statements) could stand as expressions, returning their last expression (or the one before a completion-symbol, which was "." in those days rather than the current *exit*).
- Identifiers could represent values (if declared *val*) or locations (*loc*) or variables (*var*).
- Conditional-expressions and Hoare's case-expressions.
- The control variable of a *for* to be implicitly declared and constant, *step* and *until* parts to be elaborated only once, and *while* forms to have no control variable at all.
- I-O transmission and data conversion to be separated.
- Call by value and by *name* (but name calls to be indicated at point of call also, and actuals of kind *loc* to be passed by reference). Confused? Try the following example:

  real val f (*real val x*, *real loc y*, *real var z*) = <expression>;

  This defines a function whose calls may supply:
  - an *integer* or *real* expression for *x* (input),
  - a *real* or *complex* variable for *y* (output),
  - but only a *real* variable for *z*.

  Still confused?

The Wirth/Hoare language, as described in the CACM article, was in due course implemented on an IBM 360 by Wirth, under the name "ALGOL W", during a visit to Stanford in 1966, and in [AB24.3.3] he describes a few refinements of the language

that he found necessary. See also [AB26.3.4] for comments on this language by the Japanese ALGOL Working Group. The Japanese were also active in language design, and produced their own "ALGOL N" [AB30.3.2], a simplified form of ALGOL 68 with a simplified method of description.

In the next issue [AB23.1.1] it was merely announced that the next meeting of the Working Group had been postponed from April 1966 to October 1966 "to allow the sub-committee ... more time for their work". Apparently, some interim document was produced at that meeting, and also a substantial proposal for transput [AB25.1.1]. It was confidently predicted that the main business of the following meeting, in May 1967, would be ALGOL Y. The next news [AB25.0.1] in March 1967 apologised for the fact that the draft report on the new language did not accompany that issue. "The work at Amsterdam, which includes implementation studies, has taken longer than anticipated." It would be distributed direct from Amsterdam shortly and there would be time for readers to comment before the meeting in September.

Came August and AB26, but still no Draft Report. It seems that at the May meeting, despite the confident predictions, "Discussion of ALGOL Y was rather limited", although there was still no hint of the furore that was to come. The September meeting had been postponed until "not less than 3½ months after distribution of the draft". In the event the Draft, the (in)famous MR93, did not appear until February 1968. Comments were invited [AB27.1.1] and were to be considered by the WG at its meeting in June. The possibility was still being envisaged of obtaining final IFIP approval of the document at the IFIP Congress in Edinburgh in August. And at the end of AB27 appeared the first example of a long tradition of errata to the Draft Report – a mere 9 pages of them.

The Troubles.

The appearance of MR93 was the cause of much shock, horror and dissent, even (perhaps especially) amongst the membership of WG2.1. Generally, only those members who had been present at the meetings in October 1966 and May 1967 had seen the earlier drafts, and the absentees included such notable names as Naur, Dijkstra and Garwick (for both meetings) and Hoare and Woodger (May '67 only) [AB25.1.1, AB26.1.2]. It was said that the new notation for the grammar and the excessive size of the document made it unreadable. However, it could be read and understood, as I demonstrated myself by extracting the underlying language from it and serving it up as "ALGOL 68 with Fewer Tears" [AB28.3.1] – however, I must confess that the task occupied me fully for 6 man-weeks. "Fewer Tears" was a paper written in the form of an ALGOL 68 program, and its definitive version can be found in the Computer Journal 15 2 May '72.

A small (and stormy) meeting of the Working Group in June 1968 instructed the authors to undertake considerable revisions to be submitted to the Group for final acceptance or rejection in December.

Examples of the difficulties which people had with the Report can be found in various articles. Turski [AB29.2.4] found inconsistencies in the definition of the underlying "paper computer". Hoare [AB29.3.4] wanted a more primitive core language with assignment, fancy subscripting, and the like being added by means similar to the addition of new operators, and he called for simpler coercions and no multiple *refs*. Many others submitted detailed suggestions, both to the AB and directly to Amsterdam.

In May 1968, the tenth anniversary of ALGOL 58, a colloquium had been held in Zurich [AB28.1.1], where the recently distributed Report came in for much discussion, being "attacked for its alleged obscurity, complexity, inadequacy, and length, and defended by its authors for its alleged clarity, simplicity, generality, and conciseness". Papers given at the colloquium included "Implementing ALGOL 68" by Gerhard Goos, "Successes and failures of the ALGOL effort" by Peter Naur [AB28.3.3], and some "closing remarks" by Niklaus Wirth [AB29.3.2]. Naur's paper contained criticism of MR93, as providing "the ultimate in formality of description, a point where Algol 60 was strong enough", and because "nothing seems to have been learnt from the outstanding failure of the Algol 60 report, its lack of informal introduction and justification". IFIP seemed to be calling for immediate acceptance or rejection (thus precluding further development of MR93), and thus IFIP was the "true villain of this unreasonable situation", being "totally authoritarian" with a committee structure and communication channels entirely without feedback and with important decisions being taken in closed meetings. "By seizing the name of Algol, IFIP has used the effort ... for its own glorification." Strong words indeed! Wirth's contribution was also critical of MR93, and of the method whereby the Working Group, after a week of "disarray and dispute", and then "discouragement and despair" would accept the offer of any "saviour" to work on the problems until next time. Eventually the saviours "worked themselves so deeply into subject matter, that the rest couldn't understand their thoughts and terminology any longer". Both Naur and Wirth resigned from the Group at this time.

Now at that time the ALGOL Bulletin, which was (and still is) an official IFIP publication, was printed by the Mathematisch Centrum in Amsterdam, and when AB28 (originally dated July 1968) containing Naur's paper was sent there for printing, Van Wijngaarden refused to do so. At the next WG meeting in North Berwick in August there was a furore, with words like "mud" being applied to Naur's paper, and Fraser Duncan refusing to compromise his editorial freedom. AB28 did eventually get published without expurgation, but some months late (and, to be fair, there were also floods and mislaid manuscripts which contributed to the delay).

The North Berwick meeting, which was the first that I myself attended, was five days of continuous politics. I have just been re-reading the minutes, and they display a sorry tale. A poll was taken concerning the future work of the group, and the best part of a whole day, both before and after the poll, was taken up with its form, and how its results were to be interpreted. One can see that mutual trust between the two parties had been entirely lost, and jockeying for position was the order of the day, and of every day. Barely half a day was spent in technical discussion of the Report.

The meeting at Munich in December 1968 for making the final decision was also political, but there was much more willingness to reach a compromise. There had now been three further drafts, MR95, MR99 and MR100, and there was much more technical discussion than there had been at North Berwick, resulting in the final document MR101. The meeting devoted much time to drafting a covering letter which would be acceptable to all, and which "did not imply that every member of the Group, including the authors, agreed with every aspect of the undertaking" but decided that "the design had reached the stage to be submitted to the test of implementation and use by the computing community". This was still too strong for some members, however, and on the last afternoon of the meeting a short Minority Report signed by eight members was presented. The Formal Resolution submitted to TC2 clearly implied that the Covering Letter should accompany the final publication of the Report, together presumably with the Minority Report. In January 1969, TC2 duly forwarded the Report and Covering Letter (but not the Minority Report) to the IFIP General Assembly, which in May authorised publication of the Report, but without the Covering Letter. The text of the Minority Report may be found in [AB31.1.1]. It spoke of the effort which had gone into ALGOL 68 as an experiment which had failed, and claimed that the language's "view of the programmer's task" was ten years out of date and that, doing little to help "in the reliable creation of sophisticated programs", the language "must be regarded as obsolete". It is a pity that its production at the very last moment precluded a more constructive minority report, such as Doug Ross had been trying to organise in the months preceding the meeting (as documented in [AB30.2.3]). The Munich meeting had also recommended to TC2 the setting up of a new WG2.3 on Programming Tools, and after the meeting 11 members resigned, to become the nucleus of the new group.

Post Munich.

ALGOL 68 now entered on its "maintenance phase". Of course people found problems, and there were worries about the way infinite modes had been defined in the Report. Fraser Duncan had been warning about the mathematical unsoundness of these for a long time, but articles by Meertens [AB30.3.4] and Pair [AB31.3.2] did not make their points clearly enough to be noticed (it was not until much later that Hendrik Boom showed the lurking ambiguity plainly for all to see, leading to the fixing of the problem in the Revised Report). In [AB30.3.3] Koster introduced his famous algorithm for determining the equivalence of two modes (essentially, they are equivalent if you can show that they match given the hypothesis that they are the same mode).

The original Covering Letter had envisaged that a revision of the Report might be needed in due course, and in 1971 the WG issued a general call for suggestions [AB32.2.8], stressing however that "for the present, the language definition shall remain stable", and that "there should be only one revision". There then followed a variety of reports by subcommittees of the Working Group, which had been charged with collating such material. Examples are reports on Data-processing and Transput [AB32.3.3, AB33.3.4] containing quite elaborate proposals for "record transfer" to mass-storage devices, and others on General Improvements to the language [AB32.3.4, AB33.3.3], on Conversational Languages and on Operating System Interfaces [AB32.3.5, AB32.3.6, AB33.3.6], and on Sublanguages [AB33.3.5]. All these formed the input to the meeting at Novosibirsk in August 1971 at which a timetable for the production of a Revised Report was laid down.

Also at this meeting, Fraser Duncan resigned as Editor of the ALGOL Bulletin, having been responsible for the production of 18 issues spread over 8 years. I was appointed to take his place, and this seems as good a reason as any for terminating this account here.