AB33.3.8 Comments on suggested improvements to ALGOL 68.

S.R. BOURNE and M.J.T. GUY.

1. Report of subcommittee on maintenance and improvements (AB32)

Items (1) to (4) (which are really all parts of one item) represent a great improvement over the previous situation. We have already implemented these changes and find a substantial simplification of the syntax analysis, as a number of delicate situations (such as (proc a b; ...) which might either be a formula or a declaration until the definition of a has been found) have been removed.

Two possible slight improvements to (4):

a) Use skip as the empty symbol. This saves adding yet another symbol to the language. It requires that R 8.2.7.2 a step 2 be changed as follows:

   * then some mode → then, if it is united from void, void, else some mode *

   in order that union(int,void)uiv := skip have the right effect.

b) Update collaterals (and all that implies) to permit structures with zero fields. One can then define mode void = struct() and the empty symbol is automatically (). There is then no need for void-denotations and void can be relegated to the standard prelude.

(5) also represents a clear rationalisation, although we have not yet considered the problems of implementation; things of shape

(...)(...)(...)...

tend to be a bit delicate syntactically. The change to vacuums is very welcome indeed; we have not yet implemented them, but were proposing to implement () as a vacuum.

(6) is also a relief from a messy problem. We had deferred implementing both formal declarers (we require them to be virtual as RRE do) and go-on symbols in both calls and formal parameters because of the problems in this region.

We would also like to endorse (7) and (9). In place of (8), it would seem preferable to admit that (/ and /) were a mistake (we implement almost the full syntax without insisting on a distinction

between round and square brackets).

2. Report of subcommittee on data processing and transput (AB32)

Since this report covers a more contentious and less well understood area than the previous one, we have more reservations about the proposals it contains. Additionally, we have as yet no experience of implementing 'proper' transput for ALGOL68.

(A) seems to fill a definite gap. It is a pity that decomposed-coercends have the same disagreeable properties as components of flexible arrays and hence must be subject to the same restrictions.

For (D) see suggestion (i) which seems to cover most cases wanted in practice.

We would strongly endorse (F), (G), (H) (see also our suggestion (m)), (I), (J), (strongly supported; the remarks are certainly true of our operating system, and probably of most others), (K), (L) (what happens to boolean patterns in view of (7) of committee on improvements? Are booleans represented as 01 or 01?), (M) and (O).

While supporting (N), we feel this ought to go much further. The I/O in general seems (in effect) much too concerned with card readers and punches, and other devices belonging to the dark ages. When one deals with terminals, it becomes clear that lines filled out with spaces or not are very different objects. Similarly, when preparing output for sending to another machine, whose input conventions do not attain the standards one might wish (e.g. where tabs or suitable numbers of spaces are not equivalent, or where the ordering of overprinted characters is significant).

We would therefore recommend that:

i) a book should be [1:0 flex][1:0 flex][1:0 flex]int ,

2) that operations should be available in the standard prelude to perform the operations "define end of line", "define end of page",
3) that string denotations should contain representations of 'backspace','newline','newpage','tab' (see suggestion (1) on string denotations),
4) alternative input conversion codes should be available to provide 'line reconstruction', where this is appropriate. Default settings should include line reconstruction.

We do not like (E); it seems to be shoring up a fundamentally unsatisfactory situation but we do not have any more constructive a suggestion.

We are strongly opposed to item (C). Its implementation would involve complications in many parts of the language. The definition of MODE would need to be complicated considerably and those parts of a compiler that perform mode-identification (at present not very pretty) would become considerably more complicated.

We feel that the need implied by the examples is somewhat bogus. On the assumption that x takes only one value in a particular program (the most common case), the first problem would be better solved by inserting the text of the procedure very-slow-sort in a range where x is defined.

There is a more fundamental objection to this proposal. It essentially assumes that objects ref a and ref b are interchangeable, i.e. that the store of the underlying machine is addressed in a uniform way for all modes. While this is true of many present machines and implementations, we note that modern developments in segmentation and in storage hierarchies make such assumptions dangerous. The assumption is also firmly contrary to the spirit of the report.

3. Other items

We wish to support C.H.Lindsey's suggestions(AB32.3.2). In connection with the second, it would seem desirable to define the extensions so as to insist that expressions which are written once are only elaborated once. (Lindsey points out that this is one of the possible elaborations, except in his pathological examples.) Continuing in this theme, it would seem desirable that any side effects of a mode-declaration should occur at the time of the declaration, rather than when the actual declarer is used as a generator.

Specifically, replace R 7.2.2 by

"A mode-declaration is elaborated in the following steps:
Step 1: Its constituent actual-declarer D is developed.
Step 2: All constituent boundscripts of D are elaborated collaterally.
Step 3: The "expansion" of the mode declaration is that actual-declarer obtained by replacing each constituent boundscript of D by the corresponding value obtained in step 2."

and alter 7.1.2 c, step 2:

⁎ the protected actual declarer of that mode declaration → the expansion of that mode declaration ⁎
⁎ or identity declaration →, mode declaration or identity declaration ⁎ .

We can now alter R9.2 c so that, for example,

amode τ_1, τ_2

is regarded as an abbreviation of

mode another = amode; another τ_1, another τ_2

avoiding the repeated elaboration of boundscripts. (Note that care is needed in defining the ordering, or lack of ordering, if the declaration is part of a collateral declaration.)