AB42.4.2 ALGOL 68 Hardware Representation Recommendations

Wilfred J. Hansen

University of Illinois at Urbana-Champaign

During my work on the Report on the Standard Hardware Representation for Revised ALGOL 68, I encountered a number of ideas which--though meritorious--could not be agreed on by the Subcommittee on ALGOL 68 Support. I have collected a few of these here in the hopes that where implementors choose to provide such a feature they will all provide it in the same way.

1. TITLE's

A valuable facility for identifying listings is to have a title at the top of each page. Therefore, the pragmat

PR TITLE string-denotation PR

should be the last line on its page. The string_denotation should appear in the header of succeeding pages. The provision that it start a new page is followed in a number of implementations of other languages and works well in practice. (A number of people suggested this, and I have forgotten who was first.)

2. 48-Characters Representation

At least two popular languages require only 48 characters and some devices provide only those. Usually they are

letters digits space ' $ ( ) * + , - . / =

Of the 23 special characters which are worthy, 10 are available in the 48 set, six can be avoided by using alternate constructs with bold words (lt, at, if, mod, co, etc.), and three can be represented with alternate characters from the 48 set:

" → ' [→ ( ] →)

Colon and semicolon can be replaced with diphthongs as suggested in the original Report:

: → .. ; → ., := → .=

Though only the first is still mentioned in the Revised Report, all three are non-ambiguous. The third is a special abbreviation for the very common assignment operation, but "..=" should also be acceptable. To resolve the ambiguity inside strings, it should be the case that upb "...," yields four. There is the additional problem that "3.,4" might be either an error or a series; especially curious is

(i|3.,7.,2|5.).

Presumably a compiler will have to interpret these as semicolons so the expression is an *if* rather than a *case*.

The remaining two worthy characters, underscore and apostrophe, have no reasonable representation in the 48 set. Thus, there can be no string escapes and reserved words cannot be intimidated to plain in the RES stropping regime.

3. *up* and *down*

For some reason the Report provides both *up* and *shl* for shift-left-operator and both *down* and *shr* for shift-right-operator. But they are not synonymous because only *up* and *down* can operate on semaphores and because *up* can signify exponentiation while *shl* cannot.

I cannot understand the need for this tangle of relationships, especially when *shl*, *shr*, and ** are available and are--to me at least--more mnemonic. Therfore, I suggest that, even if implemented, *up* and *down* should be restricted to their operations on semaphores. That is, their other uses should be down-played to the extent of being mentioned in some lowly place like a single small footnote.

4. String Escapes

The Standard Hardware Representation requires that a string escape--if it is allowed--must begin with an apostrophe, but make no further specification. There were two reasons: firstly, it was felt to be too machine dependent, and secondly, the authors of the Report on the SHR couldn't agree as to whether prefix or circumfix apostrophes should be used. My own feeling is that the Cambridge scheme is excellent and machine independent. It extends to providing two cases if it is modified to the following form:

```
''      → apostrophe
'/      → new line
'%      → new page
'+      → carriage return (no line feed)
',      → tab
'-      → backspace
'␣      → ␣ (where ␣ represents a space)
'int␣   → {the number of spaces specified by "int"}
'(list of character codes separated by commas)
        → {the indicated characters}
```

Note that '/ and '+ have the functions that slash and plus have when used as ASCII carriage control characters. '% seems a logical extension and the rest seem to be pleasantly mnemonic. I would allow the character codes to be specified as integers, bits-denotations, or by mnemonic name (DC1, VT, ...), but this can be left for implementation definition.

By changing the Cambridge convention to use all special characters or digits following the apostrophe, we achieve the desirable goal of a representation for upper and lower case strings. We simply specify that a letter alone is the lower-case version of the letter and

'letter → upper-case of letter.

On one case devices, all output strings will print in a single case, but the system will be ready for two case devices. (Apostrophe for lower case is currently used on at least the DEC-10.)

5. of

I included a long discussion of the of problem in the Proceedings of the 1975 International Conference on ALGOL 58, Oklahoma State University, pp. 335-337. Essentially, there must be a special-character representation of of because the operation binds so much more closely even than monadic operators and yet the bold of is at least four characters long (unless the crowded case disjunction is used as in "reOFz").

As we worked on the Standard, it became apparent that the best alternative for of is the character that is worthy and yet has no meaning outside strings: apostrophe. Part of the function "value of" in the formula manipulation example of R11.10 then becomes.

```
BEGIN REF FUNCTION f = function_name'ef;
    value'bound_var'f := value_of(parameter'f);
    value_of(body'f)
END
```

Since the "operand" to the left of the of must be a selector, and since a selector must be followed by of, there is little purpose to having a more obtrusive symbol.

For stropping regimes that still use apostrophes, the best proposal that has appeared is the "./" that I proposed at Oklahoma State. Only six of the 25 implementors who voted agreed that it was first choice, but no other symobl had as many votes. In the absence of a clearly superior symbol, I suggest that "./" be adopted as the non-bold of:

```
BEGIN REF FUNCTION f = function name./ef;
    value./bound var./f := value of(parameter./f);
    value of(body./f)
END
```

Among other advantages, "./" is entirely in lower case positions on most keyboards, just as are the letters which are the characters usually surrounding it. Please see the Oklahoma proceedings for further discussion of the merits of these two representations.