Bold taggles in Algol 68

GNU68-2025-002 (draft)

by Jose E. Marchesi

Foreword

The following specification has been released under the auspices of the GNU Algol 68 Working Group, and has been scrutinized to ensure that

a. it is strictly upwards-compatible with Algol 68,
b. it is consistent with the philosophy and orthogonal framework of the language, and
c. it fills a clearly discernible gap in the expressive power of that language.

The source of this document can be found at `https://git.sr.ht/~jemarch/gnu68`.

The informal description of this proposal introduces the proposed new language features, providing a rationale and usage examples.

The formal definition of this proposal uses the existing formalism and conventions of the Standard Hardware Representation for Algol 68, and it is expressed as modifications to the Standard Hardware Representation.

Finally, the implementation notes of this proposal describes a way in which the features added by this specification can be implemented. No implementer should feel committed to do things as described there; the same language facilities may well be implementable in other ways, more suitable to specific implementations.

1 Informal Description

Bold words and tags

The Standard Hardware Representation for Algol 68 defines a *bold word* as the hardware representation of tokens whose representation in the reference language is written using bold letters and digits, such as for example `bold begin symbol`, `completion symbol`, and `bold-letter-f-letter-o-letter-o`. These sort of tokens are used in the strict language to denote syntactic marks, standard modes, user-defined mode indicators and both predefined and user-defined operator indicators.

The Standard Hardware Representation for Algol 68 defines a *tag* as the hardware representation of tokens whose representation in the reference language is written using non-bold letters and digits, such as for example `letter-f-letter-o-letter-o`. These sort of tokens are used in the strict language to denote identifiers, be them predefined or defined by the user via an identity declaration for example.

Stropping regimes

The way of writing bold words and tags using worthy characters depends on the particular stropping regime in use. The Standard Hardware Representation describes three standard stropping regimes that result in three different ways to write bold words.

In POINT stropping bold words are written writing a point character (`.`) followed by the worthy letters or digits corresponding to the bold-faced letters or digits in the word. For example, `bold begin symbol` can be represented as `.begin` or `.BEGIN` or `.bEgIn`. In this stropping regime it is possible to represent bold-faced digits, like in `.algol68`.

In the UPPER regime bold words are written like in POINT, but the leading point character can be omitted if the bold word is not immediately preceded by another bold word, and only upper-case letters are to be used in bold words. For example, `bold begin symbol` is represented as `BEGIN`. In this stropping regime it is not possible to omit the leading point character when a bold word includes a bold-faced digit, as there is not such a thing as an upper-case digit.

The RES regime handles bold words in exactly the same way than POINT, and only differs from it in its handling of tags.

Readability of indicators

Bold words are therefore composed by letters and digits, typed as bold letters and digits in the reference language. It is very common, however, for user-defined mode indicators to contain several natural words. Consider for example the following mode declaration:

```
mode treenode = struct (treenodepayload data, ref treenode next);
```

The mode indicators **treenode** and **treenodepayload** can be a little difficult to read, as the different natural words implicated in the indicators are not easily distinguishable at first sight.

This problem can be alleviated by using the written form of bold words with a leading point character (which is theoretically supported in all standard stropping regimes) that supports mixing case. This leads to mode indicators such as `.TreeNodePayload` which is much more readable.

However, modern Algol 68 implementations tend to implement a slightly non-conforming UPPER stropping where writing bold words leaded by point characters is no longer supported[1]. In these cases it is no longer possible to use mixing case.

The above discussion also applies to operator indicators, even though these are seldom composed by more than one natural word, usually a verb.

Bold taggles

This GNU extension allows to use underscore characters (_) in bold words in all standard stropping regimes. This is done by specifying that a bold word is written as a sequence of *taggles* rather than as a sequence of worthy letters and digits. Each taggle itself is written as a sequence of one or more worthy letter and digits, optionally followed by a trailing underscore character.

It is now possible to write the mode definition above using UPPER stropping as:

```
MODE TREE_NODE = STRUCT (TREE_NODE_PAYLOAD data, REF TREE_NODE next);
```

Note that from the definition of taggle it follows that a trailing underscore is allowed, like in `PRIVATE_MODE_`, but no prefixing underscores are allowed, and indicators such as `FOO__BAR` and `FOO_BAR__` are not legal.

Tags are also defined in terms of taggles in all the standard stropping regimes. However, unlike in tags, no typographic display features can appear between the taggles in a written form of a bold word. This means that the mode indicator **treenodepayload** cannot be written as `TREE NODE PAYLOAD` in UPPER stropping.

Comparison to similar extensions

This GNU extension is not to be confused with the support provided by the Algol 68 Genie interpreter for using underscore characters in tags and bold words. There are two important and rather fundamental differences.

First, the underscore characters in Algol 68 Genie's tags and bold words are integral part of the represented identifiers, mode indicators and operator indicators, and not just a convenient way to write more readable written forms of these. Therefore, in Algol 68 Genie `TREE_NODE` denotes the mode indicator **tree_node**, not **treenode** as mandated by the Report.

Second, Algol 68 Genie doesn't use the concept of taggle, and therefore a set of arbitrary rules has to be used for inserting underscore characters in both tags and bold words. Leading underscore characters are not allowed. Therefore `_code` and `__TREE_NODE` are not valid. Zero or more trailing underscore characters are allowed, but these are not part of the represented identifiers or indicators. Therefore `code__` and `TREE_NODE__` are both allowed and represent the same

[1] This is the case of both Algol 68 Genie and the GCC Algol 68 front-end

than `code` and `TREE_NODE` respectively. Any number of sequences of one or more underscore characters are allowed anywhere else in the tags or bold words. Therefore both `foo___bar` and `TREE__PAYLOAD_FACTOR` are both valid. In this case the underscore characters become integral part of the denoted identifier and mode indicator, respectively.

2 Formal Description

The description of the stropping of bold words in POINT stropping in the Standard Hardware Representation is changed from:

```
3.5.1

Bold words.

  - A bold word is written as a point (".") followed, in order, by the
    worthy letters or digits corresponding to the bold-faced letters or
    digits in the word.

  - A bold word must be followed by a disjunctor.
```

to

```
3.5.1

Bold words.

  - A bold word is written as a point (".") followed by a sequence of
    {one or more} taggles whose worthy letters and digits correspond, in
    order, to the bold-faced letters and digits in the word.
```

When the extension is enabled, the new rule for writing bold words applies to all the three standard stropping regimes described in the Standard Hardware Representation.

3 Implementation Notes

Stropping is to be implemented purely at the lexical level and it must not have any impact on the letters and digits that constitute bold words and tags. Therefore, in principle, of all the components of a compiler or interpreter only the lexical analyzer (or tokenizer) shall concern itself with the stropping regime.

A programming editor can highlight any sequence of two or more consecutive underscore characters as an error: these are not syntactically valid in any standard stropping regime, out of string denotations and pragments.

Implementations are advised to always gate the usage of this extension through a command-line option. Appropriate `PORTCHECK` diagnostics should be emitted whenever appropriate.