Proposed new features for the DELTA System

14 September 2018

M.J. Dallwitz, T.A. Paine, and E.J. Zurcher

Introduction

This discussion assumes familiarity with the current DELTA system — that is, the DELTA format (Dallwitz 1980), and the DELTA programs developed in the CSIRO Division of Entomology, as described in Edition 4 of the User’s Guide to the DELTA System (Dallwitz, Paine and Zurcher 1993). The User’s Guide describes more than 150 ‘directives’ used by the programs. The DELTA format comprises those directives which define the meaning of the coded descriptions: CHARACTER TYPES, IMPLICIT VALUES, DEPENDENT CHARACTERS, CHARACTER NOTES, CHARACTER LIST, and ITEM DESCRIPTIONS (plus a few redundant directives such as NUMBER OF CHARACTERS which are convenient for programming). The rest of the directives specify how the programs are to process the coded descriptions for various purposes, such as the generation of natural-language descriptions or keys. Most of the new or enhanced directives described below are extensions of the DELTA format, but some indicate how the new DELTA format will (or could) be used by new programs to overcome deficiencies in the capabilities of current programs.

The central program of the new DELTA system will carry out the functions of the current Confor and Delfor, as well as data entry and editing. A name for this program has not yet been decided. For the time being, we will refer to it as ‘Confor’, and add the qualifiers ‘old’ or ‘new’ if necessary to avoid ambiguity.

An enhanced DELTA format will serve as a data-exchange medium. It will necessarily be considerably more complex than the current format, because of the extra features to be supported. We envisage that the old format will be a subset of the new, with the possible exception of some rarely used features such as ‘variant items’, which may be replaced by more general but incompatible constructs. However, at this stage we would not rule out a complete redesign of the format if this seems desirable. In that case, separate provision would need to be made for reading existing data files.

The new Confor will be able to read and write complete DELTA-format files, but this facility will normally be used only for data exchange. Data input and manipulation will normally be interactive. Therefore, in designing most aspects of the format, we can lean towards ease of programming rather than ease of use. Several people have suggested that many directives could be dispensed with, and the information they contain could be inferred or placed elsewhere. For example, the number of characters and numbers of states can be inferred from the character list, and the character types, dependencies, and notes could be embedded in the character list. Some of these proposals would certainly make the current Confor easier for users, but they seem to have no advantage in the context of the new Confor. They would often make programming more difficult, especially for storage allocation and for ignoring unwanted information (e.g. character notes). It might even be better to split some existing directives; for example, the ‘exclusive’ property of characters, now expressed as the character types EUM and EOM, would more logically be expressed in a separate directive (or eliminated completely).

The user interface and the internal workings of the new Confor will be able to shield users from many of the complexities of the current system. However, the ‘attribute’ (cell of the data ‘matrix’ — see Section 2.2 of DELTA User’s Guide for definition) may remain an exception, because there may be no simpler way of expressing the potentially complex information in an attribute (for example, order of states and association of comments with particular states). The user will have the option of entering an attribute directly in DELTA format, or building it up piece by piece via menus. In the latter case, the details of generating the required syntax will be handled by the program. The natural-language interpretation of the attribute will be visible during entry and editing of the attribute. This will reduce the need for the user to view the attribute in DELTA format, but it may sometimes be necessary to do so, particularly while editing an existing attribute. The format of attributes should therefore be kept as simple as possible.

Accepted proposals

Command syntax and usage

Command words will not be case sensitive.

It may be desirable to distinguish between ‘interactive’ Confor commands and ‘batch’ Confor directives. The interactive commands would be similar to Intkey commands (Dallwitz, Paine and Zurcher 1995), and would need be suitable for incorporation in a hierarchical menu structure. The batch directives would be similar to the current Confor directives, and would be executable only from within files (although they would be assembled and edited interactively).

The syntax of Intkey commands was made different from that of the current Confor directives to simplify interactive entry of commands. In Intkey, a single command may specify the required action, a set of taxa, and a set of characters, whereas in Confor at least three directives would be required for this. In most Intkey commands, ‘space’ is the only delimiter required, but parentheses (to separate a set of taxa from a set of characters) and quotes (to enclose a parameter which has internal spaces) are sometimes required

Delimiter symbols

The special delimiter symbols will be user definable and re-definable. The defaults will be as in the current DELTA format, plus the symbol ‘|’ to indicate that the following symbol is to be interpreted literally, and ‘!’ to indicate that the next two symbols are a hexadecimal representation of a byte. The new definitions will not come into force until the end of the redefining directive.

Example

```
*DEFINE DELIMITERS
#1. |\ <literal>
#2. |! <hexadecimal>
#3. $ <start of directive>
#4. |* <start of character, item, etc.>
#5. { <opening bracket>
#6. } <closing bracket>
. . .
```

With these definitions in force, the start of a character list might read

```
$CHARACTER LIST
*1. {synonyms}/
```

The literal marker, applied to an alphabetic character, will protect it from case changes, e.g. |mRNA.

On output, there will be provision for omission of ‘|’ and ‘!’, for passing them through unaltered, and for substituting other symbols. If ‘!’ is omitted, the hexadecimal codes will be replaced by the corresponding byte.

All delimiters will be effective as single symbols. In particular, delimiters will not need to be associated with spaces, as they are in some contexts in the present DELTA format. (This may require changes to existing data. For example, ‘and/or’ will have to be replaced by ‘and|/or’. It should be possible to carry out these changes automatically.)

Spaces and line length

There will be no limit on line length. Lines may be broken or any number of spaces included, at any space or after any delimiter except angle brackets, parentheses, and decimal points.

Alternative languages

The command words, error messages, and help in Confor will be readily translatable into other languages. (This will be implemented as in the current Intkey).

There will be provision for alternative-language versions of all appropriate data elements, such as character lists and comments. The alternative-language versions will be flagged by means of ‘coded comments’ (see below). Data elements that are not flagged as being in a particular language will be assumed to be common to all languages (for example, taxon names). In Confor, it will be possible to restrict the user’s view to any language or set of languages.

Example

```
*ALTERNATIVE LANGUAGES English French Chinese
*DEFAULT LANGUAGE English

*CHARACTER LIST
#1. <longevity of plants>/
    1. annual <or biennial, without remains of old sheaths or culms>/
    2. perennial <with remains of old sheaths and/or culms>/
. . .

*CHARACTER LIST <@French>
#1. plantes < longévité>/
    1. annuelles <sans vestiges de gaines ou de chaumes>/
    2. vivaces <avec vestiges de gaines ou de chaumes>/
. . .
```

Keywords

Character and taxon keywords will be definable to represent groups of characters and taxa, as in the current Intkey.

Indexed lists

There will be provision for defining numbered and alphabetic lists of entities such as references, countries, and taxonomic names. These lists will be used as character states or as ‘coded comments’ (see below).

Example

```
*NUMBERED LIST <@English> continents
#1. Europe
#2. Africa
#3. Asia-Temperate
. . .

*ALPHABETIC LIST references
#1. Abel, D. J., and Williams, W. T. 1985. A re-examination of four classification fusion
    strategies. Comput. J. 28: 439–43.
#2. Brunt, A., Crabtree, K., and Gibbs, A. (eds). 1990. Viruses of tropical plants. Descriptions
    and lists from the VIDE database. (CAB International: Wallingford.)
#3. Burr, E. J. 1970. Cluster sorting with mixed character types. II. Fusion strategies. Austral.
    Comput. J. 2, 98–103.
. . .
```

An alphabetic list need not be in alphabetic order when represented in DELTA format, but will be automatically maintained in alphabetic order within Confor.

In contexts where an element of a particular list is the only possibility (e.g. in attributes of type LIST — see below), a list element will be represented by its number alone. In other contexts, it will by represented by a ‘comment’ of the form <@*list-name number*>, e.g. <@ref 3>.

Lists will be able to contain references to other lists.

Example

```
*ALPHABETIC LIST authorities
. . .
#10. Harms
#11. van Meeuwen
. . .

*ALPHABETIC LIST species_names
#1. Pericopsis elata (<@auth 10>) <@auth 11>
. . .
```

It will be possible to define relations between lists, such that an element of one list ‘belongs to’ an element of another.

Example

```
*NUMBERED LIST regions
#1. Northern Europe
#2. Middle Europe
#3. Southwestern Europe
#4. Southeastern Europe
#5. East Europe
#6. Northern Africa
#7. Macaronesia
#8. West Tropical Africa
. . .

*RELATED LISTS continents regions
1,1–5 2,6–15 . . .
```

New character type: LIST

The states of these characters will be the elements of a numbered or alphabetic list (see above).

Example

```
*CHARACTER TYPE 5,LI

*CHARACTER LIST
. . .
#5. anatomical references/ references/
. . .

*ITEM DESCRIPTIONS
. . . 5,21/55 . . .
```

New character type: CYCLIC

This type would be used for information such as time of year. It would improve natural-language descriptions and calculation of distances.

Example

```
*CHARACTER TYPE 9,CY
```

```
*CHARACTER LIST
. . .
#8. flowers <whether all year>/ 1. all year/ 2. <not all year>/
#9. flowers <months>/ 1. January/ 2. February/ . . . 12. December/
. . .
*ITEM DESCRIPTIONS
. . . 8,2 9,11–1 . . .
```

Character and state identifiers

It will be possible to define alphanumeric identifiers for characters and states. This will simplify the merging of separately maintained databases. These identifiers will be used only in character lists, not in taxon descriptions.

Example

```
#45<awn01>. awns <presence>/
   1<p>. present/
   2<a>. absent/
```

Alternative wordings for characters

Alternative wordings will be able to be incorporated in characters lists, and invoked selectively. The number of states in a character will be allowed to vary to correspond to the key-states currently in force.

Example

```
*CHARACTER LIST
. . .
#38.0. upper glume <of female-fertile spikelets, mid-zone nerve number>/
   nerved/
#38.1. upper glume of female-fertile spikelets/
   1. without nerves or with a single nerve/
   2. with two or more nerves/
. . .
#297. papillae <presence in the abaxial leaf blade epidermis>/
   1. present/
   2. absent/
#297.1. abaxial epidermal papillae <presence in the leaf blade>/
   1. present on the leaf blade/
   2. absent from the leaf blade/
. . .
*KEY STATES 38,~1/2~
*CHARACTER WORDING 38,1 297,1
```

Taxon names

An alternative form for a taxon name will be a list element.

Example

Poa L./

or

<@species 17>/

Selective omission of the authority will be possible only for the list-element form of the name, as in the example under *indexed lists*. (The current mechanism, where the comment in a name is interpreted as the authority, will not be supported.)

Synonyms

Synonyms will normally be represented in a character or characters of type LIST. The characters involved will be indicated in a SYNONYM CHARACTERS directive. The list will be the same as that used for the taxon names. A mechanism will be provided in Confor to simultaneously search all the synonym characters and the taxon names, so that a taxon can be accessed via any of its synonyms.

Coded 'comments' (subsidiary information)

Coded 'comments' will allow the incorporation of subsidiary information that will be interpretable by programs. Coded comments will be embedded in ordinary comments (that is, enclosed in angle brackets), and will comprise the symbol '@', a single-word 'comment identifier', and the coded information. A coded comment will be terminated by the next coded comment, or by the closing angle bracket of the comment. Confor will recognise the following coded comments.

Coded comment	**Meaning**
<@probability *x*>	Probability or frequency of a state value.
<@*x*%>	Alternative form of 'probability' comment.
<@rarely>	A low probability for a state value. The value of this probability will be settable.
<@only a_1 a_2 ...>	Specifies that a character value is for use in applications a_1 a_2 ... only. It would be omitted from other applications.
<@not a_1 a_2 ...>	Specifies that a character value is not for use in applications a_1 a_2 ... It would be omitted from these applications.
<@for a_1 a_2 ...>	If this comment appears in an attribute, only the values so marked would be used in applications a_1 a_2 ... Equivalent to @not a_1 a_2 ...on the other values.
<@about>	Qualifies numeric values. The extent of the uncertainty would be specifiable by generalisations of the current ABSOLUTE/PERCENTAGE ERRORS directives.
<@?>	Indicates a guessed value.
<@possibly>	Indicates that there is no evidence that the taxon does *not* have the value.
<@reliability *x*>	Specifies a reliability for an attribute, to modify the overall reliability of a character. This information would be important for key generation (although a Key-generation program to use it is not currently planned).
<@edit *commands*>	Apply editing commands to the natural-language description before output.
<@up>	Attribute has been generated from information passed up the taxonomic hierarchy.
<@down>	Attribute has been generated from information passed down the taxonomic hierarchy.
<@note *text*>	Uninterpreted comment (replacing the current 'inner comment' mechanism).

In addition, any numbered or alphabetic list name will be recognised as a comment identifier. The omission of coded comments from natural-language descriptions will be controllable independently for each identifier.

Example

```
*ITEM DESCRIPTIONS
. . . 10,1/3<@prob .1>  11,2/<@rarely>4  12,1<@ref 135 322>  13,2/<@English occasionally
@German gelegentlich @5% @ref 54>3  14,1<@note check in fresh specimens>
15,<@about>15–<@about>20  16,2/1<@only keys>  17,7<@only keys>–8.5–9<@for
classification>–10–12<@only keys>  18,2<@for classification>/3  19,1<@English usually
truncate @German gewöhnlich trunkat><@edit d; (;i;, ;d;);>  20,1/2<@not Australia>
```

The editing command for attribute 19 removes the parentheses from the comment, and places a comma before the comment.

A directive *APPLICATIONS would control the inclusion/omission of values marked by @only, @not, and @for. For example,
*APPLICATIONS keys
would use
16,2/1 17,7–12 18,2/3 20,1/2
and
*APPLICATIONS classification Australia
would use
16,2 17,9 18,2 20,1
Ways of handling values marked with several of these comments have not yet been fully considered.

Indefinite values

In numeric attributes, '~' will indicate an indefinitely large number ('many' or 'large'), or, when used as the lower value of a range, an indefinitely small number.

Example

~ = many
5–~ = 5 or more (or 5 to many)
~–5 = up to 5

Specific, settable numbers will be substituted for the indefinite numbers where necessary (for example, for calculating a mean, or in identification).

Delimiters in attributes

The following restrictions will apply to the positioning of the delimiters ',', '/', '&', and '–' in attributes.

- The delimiters are not allowed in text attributes.
- ',' must precede the other delimiters.
- The delimiters must not be adjacent, or be separated only by comments.
- '&' must not be the next delimiter after '–', and vice versa.

Position and scope of comments in attributes

Comments will be able to be positioned anywhere in attributes, except at the start (i.e. not before the character number).

This added flexibility will worsen the ambiguity in determining the state value(s) with which the comment is associated. The resulting ambiguity in generated natural-language descriptions is no worse than in descriptions written directly in natural language, and can presumably be tolerated. However, ambiguity will not be acceptable in the interpretation of some kinds of coded comments, particularly probabilities.

In mathematical notation, similar problems are usually solved by the use of brackets and rules of precedence.

Example

5,1&2/3<...> — comment applies to '3'.
5,1&2<...>/3 or 5,{1&2}<...>/3 — comment applies to '1&2'.
5,1&{2}<...>/3 — comment applies to '2'.
5,{1&2/3}<...> — comment applies to the whole attribute.]

It may be possible to avoid the use of brackets, while obtaining sufficient functionality, by defining the scope of each type of comment in terms of the delimiters in an attribute.

The scope of ordinary comments and ‘list’ coded comments (such as references) can be undefined, because only natural-language descriptions are affected.

It would not normally be meaningful to associate probabilities with individual state values connected by ‘&’ or ‘–’, and it would be undesirable (though perhaps sometimes convenient through lack of information) to associate a single probability with two or more state values separated by ‘/’. Thus, it might be satisfactory to define the scope of probability comments to extend as far as the delimiter ‘/’.

Example

```
5,1&2<@prob .2>/3 — probability applies to ‘1&2’.
5,1&2/3<@80%> — probability applies to ‘3’.
```

The coded comments ‘@only’, ‘@not’, and ‘@for’ would be treated similarly to probabilities.

The coded comments ‘@?’ and ‘@about’ would apply to the adjacent value.

The coded comments ‘@reliability’, ‘@edit’, ‘@up’, and ‘@down’ would apply to the whole attribute.

Dependent states

There will be a directive to indicate, for a set of list characters using the same list, that states of some characters may or may not be coded depending on whether the same state of another character is coded.

Example

```
*CHARACTER LIST
. . .
#40. infects/ hosts/
#41. does not infect/ hosts/
#42. <symptoms> chlorosis/ hosts/
#43. <symptoms> local lesions/ hosts/
#44. <symptoms> leaf curling/ hosts/
. . .

*APPLICABLE STATES 40,42-44

*INAPPLICABLE STATES 40,41 41,40:42-44
```

Mechanisms will be provided for producing appropriate natural-language descriptions, so that, for example, ‘40,35/77 41,52 42,35 43,35’ might produce ‘infects X (symptoms: chlorosis, local lesions), Y; does not infect Z.

Natural-language descriptions — insertion of parentheses

When incorporating attribute comments in natural-language descriptions, parentheses will be added if and only if the comment *follows* a state value. For example, ‘30,1/<occasionally>2<*C. incompletus*>‘ might produce ‘Rachilla prolonged beyond the uppermost female-fertile floret, or occasionally terminated by a female-fertile floret (*C. incompletus*)’.

Natural-language descriptions — punctuation

The LINK CHARACTERS and REPLACE SEMICOLON BY COMMA directives will be replaced by three punctuation directives:

*PUNCTUATE ;. s_1 s_2 . . .
*PUNCTUATE ,. s_1 s_2 . . .
*PUNCTUATE ,; s_1 s_2 . . .

where s_i is a set of characters of the form

c_1:c_2: . . .

where c_j is a character number or range of numbers. The ‘;.’ sets and the ‘,.’ sets, taken together, must be mutually exclusive. The ‘,;’ sets must be mutually exclusive, and each must be contained in one of the ‘;.’ or ‘,.’ sets.

The first punctuation mark in each directive is the ‘internal’ punctuation mark for each of the sets in the directive, and the second is the ‘terminal’ punctuation mark. Each character not mentioned in one of these directives is the sole member of a set with terminal punctuation mark ‘.’.

A description is divided into ‘sub-sentences’, which are the maximal sets of contiguous attributes in the description, such that each set is fully contained within all the punctuation sets to which any of its attributes belong. That is, an attribute terminates a sub-sentence if it belongs to a punctuation set to which the next attribute does not belong.

A sub-sentence is followed by the terminal punctuation mark of one of the punctuation sets in which it is contained; ‘.’ takes precedence over ‘;’. Each attribute except the last in a sub-sentence is followed by the internal punctuation mark of one of the punctuation sets in which the sub-sentence is contained; ‘,’ takes precedence over ‘;’.

Example

Each set of directives is followed by schematic representations of natural-language descriptions, in which each natural-language attribute is represented by its corresponding character number.

```
*PUNCTUATE ;. 13–15
*PUNCTUATE ,. 4–6 7–9

1. 2. 3. 4, 5, 6. 7, 8, 9. 10. 11. 12. 13; 14; 15. 16. 17. 18.
2. 5. 8. 11. 14.

*PUNCTUATE ,. 1–9 10–12:16–18 13–15
*PUNCTUATE ,; 1–3 4–6 7–9

1, 2, 3; 4, 5, 6; 7, 8, 9. 10, 11, 12. 13, 14, 15. 16, 17, 18.
2; 4, 5. 10, 11, 16, 17.

*PUNCTUATE ;. 1–9 10–12:16–18 13–15
*PUNCTUATE ,; 4–6

1; 2; 3; 4, 5, 6; 7; 8; 9. 10; 11; 12. 13; 14; 15. 16; 17; 18.
2; 3; 5, 6. 10; 12; 17.
```

The punctuation generated by the PUNCTUATE directives will be able to be overridden by punctuation in comments in attributes.

Example

```
12,1<,>/2<.>
```

Natural-language descriptions — omission of words

Words at the start of the feature descriptions of the second and subsequent attributes in a sub-sentence (see above) will be omitted if they are the same as words at the start of the first feature description that could have appeared in the sentence or sub-sentence.

Within a variable attribute, repeated words at the start or end of each state description will be omitted provided that the context is sufficiently simple. Detailed rules have not yet been worked out.

Provision may need to be made to mark some words (for example, articles and adjectives which precede different nouns) as not subject to omission by the above rules.

Caution is necessary in proposing rules for the omission of words or punctuation marks. It is easy to think of circumstances where the application of simple rules will produce improvements; it is harder to think of all the circumstances where those rules will produce unacceptable results. For example, consider the character ‘#1. leaves/ 1. with long hairs/ 2. with short hairs/ 3. without hairs/’, and the attribute ‘1,1/2’. This currently produces the description ‘leaves with long hairs, or with short hairs’.

Omitting the repeated words and the comma gives ‘leaves with long or short hairs’, which is clearly preferable. However, applying that procedure to the attribute ‘1,1&2/2/3’ gives ‘leaves with long and short or short or without hairs’. The presence of comments would complicate matters further, and languages other than English need to be considered. Any general reduction in clumsiness of expression should not result in the introduction, even rarely, of wording which is ambiguous, misleading, or nonsensical.

The following example (provided by E. J. Gouda) leads to difficulties even with a simple attribute. When simple omission of repeated words is used with the character ‘#6. plant with/ 1. a few leaves/ 2. many leaves/ 3. very many leaves/’ and the attribute ‘6,2/3’, the resulting natural-language description is ‘plant with or very many leaves’. The word ‘many’ needs to be flagged as not subject to omission. A simpler, and quite common, example is states of the form ‘*x*’ and ‘not *x*’.

Natural-language descriptions — formulas

In natural-language descriptions, it will be possible to combine adjacent numeric attributes into a ‘formula’. The conversion to a formula will only be carried out if all of the attributes are coded.

Example

```
*CHARACTER LIST
. . .
#44. leaf blades <length>/
   cm long/
#45. leaf blades <width>/
   cm wide/
. . .
*FORMULA CHARACTERS  44<X>45<cm>
*ITEM DESCRIPTIONS
. . .
44,(2–)4–6 45,(1–)2–3(–4)
. . .
```

This would produce ‘leaf blades (2–)4–6X(1–)2–3(–4)cm’.

Natural-language descriptions — editing

It will be possible to edit natural-language descriptions during their creation. Specified editing commands will be applied to attributes after they are generated, but before output. A directive will specify the commands to be applied for every attribute generated from a given character, and a coded comment @edit will allow editing of an individual attribute.

Example

```
*EDIT CHARACTERS
#38. d;1 nerved;i;with a single nerve;
```

I.e., delete ‘1 nerved’, insert ‘with a single nerve’.

Proposal still under consideration

State prefixes and suffixes

The state descriptions in a multistate character should have provision for a prefix and suffix, which would be output only once per attribute in a natural-language description.

Example

```
#6. leaves/ with/
   1. sparse/
   2. dense/
   hairs/
```

The attribute 6,1/2 would produce the natural-language description ‘leaves with sparse or dense hairs’.

This proposal does not cope with the situation where only some of the states have the same initial (or final) words.

Example

```
#6. leaves/
   1. with sparse hairs/
   2. with dense hairs/
   3. without hairs/
```

The attribute 6,1/2 should still produce ‘leaves with sparse or dense hairs’.

Advantages of using prefixes and suffixes are: (1) avoidance of problems with repeated words that must not be omitted; (2) potential for placement of comments before or after the suffix. E.J. Gouda has implemented suffixes in his DELTA programs, and gives the following example, which illustrates the placement of attribute comments before and after the suffix. (His version of the character-list syntax includes the character type, and places the state suffix at the end of the feature description.)

Example

```
#3. RN. plant <length, height>/ cm tall/
#6. OM. plant with <density of the leaves>/ leaves/
   1. a few/
   2. many/
   3. very many/
#7. UM. plant forming <form of the rosette>/ rosette/
   1. a tubular/
   2. a narrowly funnelform/
   3. a broadly funnelform/
   4. a subbulbous/
```

The DELTA description

3,<(15–)>35–60 6,1/<rarely>2 7,1/<sometimes>2/<rarely>4<(inflated sheaths)><(often tinged with red)>

produces the natural-language description

Plant (15–)35–60cm tall, with a few or rarely many leaves, forming a tubular or sometimes a narrowly funnelform or rarely a subbulbous (inflated sheaths) rosette (often tinged with red).

There are two comments at the end of the last attribute, and the suffix is positioned between them. This mechanism would not be possible in the proposed new DELTA format, in which there is no limit to the number of comments that may appear at any position.

Rejected proposal

Extreme values for non-numeric characters

Extreme values should be allowed for all character types and delimiters, e.g. 20,1(/2).

The same effect could be achieved by means of the coded comments <@rarely> or <@probability *x*>, and these would be more suitable for generating natural-language descriptions. (For numeric characters, the parentheses denoting the extreme values can be taken through unchanged to the natural-language descriptions, but this is probably unsuitable for other character types.)

References

Dallwitz, M.J. 1980. A general system for coding taxonomic descriptions. *Taxon* 29: 41–6.
Dallwitz, M.J., Paine, T.A., and Zurcher, E.J. 1993 onwards. User’s guide to the DELTA System: a general system for processing taxonomic descriptions. 4th edition. delta-intkey.com
Dallwitz, M.J., Paine, T.A., and Zurcher, E.J. 1995 onwards. User’s guide to Intkey: a program for interactive identification and information retrieval. 1st edition. delta-intkey.com