The preferred behaviour of a CIF application is to determine the type of a data value by looking up the corresponding dictionary definition. However, some CIF-reading software may not be designed with the ability to parse dictionaries; and indeed any CIF reader may encounter data names that are not defined in a public or accompanying dictionary. It is therefore appropriate to adopt a strategy of interpreting as a number any data value that looks like one, *i.e.* adopts any of the permitted ways to represent a numeric value. Therefore, in the absence of a specific counter-indication (from a dictionary definition), the data value in the following example may be taken as the numeric (integer) value 1:

```
_unknown_data_name   1
```

On the other hand, if `_unknown_data_name` were explicitly defined in a dictionary with a data type of 'char', then the value should be stored as the literal character `1`.

This is a subtle point, perhaps of interest only to software authors. Nevertheless, the consistent behaviour of CIF applications will depend on correct implementation of this behaviour.

The data type **char** covers single characters or extended character strings. Since CIF tokens are separated by white space, any character string that includes white-space characters (including line-terminating characters) must be delimited by one or other of a set of special characters used for this purpose. The detailed rules for quoting such strings are given in Section 2.2.7.1.4 and comprise the standard CIF syntax rules for this case. No semantic distinction is made in general between short character strings and text strings that extend over several lines, described in the specification document as 'text fields', although again particular CIF applications may choose to impose distinctions. Note that numbers within a quoted string or a text block (bounded by semicolons in column 1) are not interpreted as type 'numb' but as type 'char'.

The data type **uchar** was introduced explicitly at revision 1.1 of the CIF specification, and is intended to formalize the description and automated handling of certain strings in CIFs that are case-insensitive (such as data names and data-block headers).

The data type **null** is a special type that has two uses. It is applied to items for which no definite value may be stored in computer memory. As such it is a formal device for allowing the introduction of data names into dictionary files that do not represent data values permissible within a data file instance. The usual example is that of the special data names introduced in DDL1 dictionaries (such as the core dictionary) to discuss categories.

The more important use of the null data type is its application to the meta characters '`?`' (query) and '`.`' (full point) that may occur as values associated with any data name and therefore have no specific type. (Arguably, for this case 'any' might be a better type descriptor than 'null'.)

The substitution of the query character '`?`' in place of a data value is an explicit signal that an expected value is missing from a CIF. This 'missing-value signal' may be used instead of omitting an item (*i.e.* its tag and value) entirely from the file, and serves as a reminder that the item would normally be present.

The substitution of the full-point character '`.`' in place of a CIF data value serves two similar, but not identical, purposes. If it is used in looped lists of data it is normally a signal that a value in a particular packet (*i.e.* a value in the row of the table) is 'inapplicable' or 'inappropriate'. In some CIF applications involving access to a data dictionary it is used to signal that the default value of the item is defined in its definition in the dictionary. Consequently, the interpretation of this signal is an application-specific matter and its use must be determined according to the application. For example, in a CIF submitted for publication in *Acta Crystallographica* the presence of a '`.`' value for the item `_geom_bond_site_symmetry_1` is predetermined as the default value 1_555 (as per the dictionary definition). Note that, in this instance, it is also equivalent to 'no additional symmetry' or 'inapplicable'.

2.2.5.3. Extended data typing: content type and encoding

The initial implementation of CIF assumed that most character strings would represent identifiers or terse descriptions or comments, and that the correct behaviour of the majority of CIF applications would be simply to store these in computer memory or retrieve them verbatim. Only a few data values were foreseen as having extended content that might need special handling. For example, the complete text of a manuscript was envisaged as being included in the field `_publ_manuscript_processed`. The handling of this field (its extraction and typesetting) would be left to unspecified external agents, although some clue as to the provenance of the contents of that field (and thus their appropriate handling) would be given by `_publ_manuscript_creation`.

However, the evolution of CIF applications has required that some element of typographic markup be permitted in a growing number of data values, and future applications may be envisaged in which graphical images, virtual-reality models, spreadsheet tables or other complex objects are embedded as the values of specific data items. Since it will not be possible to write general-purpose CIF applications capable of handling all such embedded content, techniques will need to be developed for transferring each such field to a specialized but separate content handler. In the meantime, the rather *ad hoc* conventions for introducing typographic markup available at present are described in Sections 2.2.7.4.13–17. It is hoped that in the future different types of such markup may be permitted so long as the data values affected can be tagged with an indication of their content type that allows the appropriate content handlers to be invoked.

It has also been necessary to allow native binary objects to be incorporated as CIF data values. This was done to support the storage of the large arrays of image data obtained from area detectors. Since the CIF character set is based on printable ASCII characters only, encodings including compression have been developed to permit interconversion between ASCII and binary representations of such data (see Chapter 2.3).

Nowadays, arbitrary embedded objects may be transported in web pages *via* the http protocol (Fielding *et al.*, 1999) or as attachments to email messages structured according to the MIME protocols (*e.g.* Freed & Borenstein, 1996). Identification of encoding techniques and hooks to invoke suitable handlers are carried in the relevant Content-Type and Content-Encoding http or MIME headers. It is suggested that this may form the basis of suitable tagging of content types and encoding for future CIF development.

A candidate for a CIF-specific encoding protocol is the special convention introduced with CIF version 1.1 to interconvert long lines of text between the new and old length limits (Section 2.2.7.4.11). This is an encoding in the sense that it is a device designed to retain any semantic content implicit in textual layout, while conforming to slightly different rules of syntax. It is designed to enable CIFs written to the longer line-length specification to be transformed so that they can still be handled by older software. Since the object of the exercise is to manage legacy applications, it is likely that the interconversion will be done through external applications, or filters, designed specifically for the purpose. Such a conversion filter is conceptually the same as a filter to convert a binary file into an ASCII base-64 encoding, for example.