Again it should be emphasized that the rows and columns of the table are identified by parsing each value and referring back to the sequence in the header of its identifying tag, and not by a conventional layout of items. It is usual for CIF writers to lay out two-dimensional data arrays as well formatted tables for ease of inspection in text editors or other viewers, but CIF readers must never rely on layout alone to identify a tabulated value.

A corollary of this is that the number of values in a looped list must be an exact integer multiple of the number of data names declared in the loop header.

Within a single data block the same data name may not be repeated. Thus if a data item may have multiple values, these items must be collected together within a looped list, the data name itself being given once only in the loop header.

A data value is a string of characters. CIF distinguishes between numerical and character data in a broad sense (see Section 2.2.5.2 below). Numerical values may not contain white space (and indeed are constrained to a limited character set and ordering, essentially encompassing a small range of ways in which numbers are characteristically represented in printed form). Because tokens are separated by white space, character data that include white-space characters must be quoted. If the data value does not extend beyond the end of a line of text, it may be quoted by matching single-quote (apostrophe, `'`) or double-quote (quotation mark, `"`) characters. If the data value does extend beyond the end of a line of text, then paired semicolon characters `;` *as the first character of a line* may be used as delimiters. The closing semicolon is the first character of a line immediately following the close of the data-value string. Fig. 2.2.3.1 shows examples of all three types of delimited character values that include white space.

Character strings that begin with certain other characters must also be quoted. These leading characters are those which introduce tokens with special roles in a STAR File (such as underscore _ at the start of a data name, hash # at the start of a comment and dollar $ identifying a save-frame reference pointer). Likewise, the STAR File reserved words `loop_`, `stop_` and `global_` must be quoted if they represent data values, as must any character string beginning with `data_` or `save_`.

Lines of text are restricted to 2048 characters in length and data names are restricted to 75 characters in length. These are increases over the original values of 80 and 32 characters, respectively.

2.2.4. Portability and archival issues

The CIF format is designed to be independent of operating system (OS) and programming language. Nevertheless, variations in the way that each OS specifies and handles character sets mean that care must be taken to ensure that CIF software is portable across different computer platforms. There are also constraints on the application of these specifications in order to maintain compatibility between archival systems. These issues are discussed briefly here. More details are given in the formal CIF specification (see Section 2.2.7). In general, compatibility and portability considerations for different OSs are of little importance to users of CIFs, but they need to be well understood by software developers.

2.2.4.1. Character set

The characters permitted in a CIF are in effect the printable characters in the ASCII character set. However, a CIF may also be constructed and manipulated using alternative single-character byte mappings such as EBCDIC, and multi-byte or wide character encodings such as Unicode, provided there is a direct mapping to the permitted ASCII characters. Accented characters, characters in non-Latin alphabets and mathematical or special typographic symbols may not appear as single characters in a CIF, even if a host OS permits such representations.

White space (used to separate CIF tokens and within comments or quoted character-string values) is most portably represented by the printable space character (decimal value 32 in the ASCII character set). In an ASCII environment, white space may also be indicated by the control characters denoted HT (horizontal tab, ASCII decimal 9), LF (line feed, ASCII decimal 10) and CR (carriage return, ASCII decimal 13). To ease problems of translation between character encodings, the characters VT (vertical tab, ASCII decimal 11) and FF (form feed, ASCII decimal 12) are explicitly excluded from the CIF character set; this is a restriction that is not in the general STAR File specification (Chapter 2.1).

2.2.4.2. Line terminators

Given that the STAR File is built on the premise of a line-oriented text file, it is difficult in practice to provide a complete and portable description of how to identify the start or end of a line of text. The difficulty arises for two reasons.

First, some OSs or programming languages are record-oriented; that is, the OS is able to keep track of a region of memory associated with a specific record. It is usually appropriate to associate each such record with a line of text, but the 'boundaries' between records are managed by low-level OS utilities and are not amenable to a character-oriented discussion. Such systems, where records of fixed length are maintained, may also give rise to ambiguities in the interpretation of padding to the record boundary following a last printable character – is such padding to be discarded or treated as white space? It is for this reason that the elision of trailing white space in a line is permitted (but not encouraged) in the full CIF syntax specification [Section 2.2.7.1.4(17)].

The second complication arises because current popular OSs support several different character-based line terminators. Historically, applications developed under a specific OS have made general use of system libraries to handle text files, so that the conventions built into the system libraries have in effect become standard representations of line terminators for all applications built on that OS. For a long time this created no great problem, since files were transferred between OSs through applications software that could be tuned to perform the necessary line-terminator translations in transit. The best-known such application is undoubtedly the 'text' or 'ascii' transmission mode of the typical ftp (file transfer protocol) client.

Increasingly, however, a common network-mounted file system may be shared between applications running under different OSs and the same file may present itself as 'valid' to one user but not to another because of differences in what the applications consider a line terminator.

The problem of handling OS-dependent line termination is by no means unique to STAR File or CIF applications; any application that manipulates line-oriented text files must accommodate this difficulty. The specification notes the practice of designing applications that treat equivalently as a line terminator the characters LF (line feed or newline), CR (carriage return) or the combination CR followed by LF, since these are the dominant conventions under the prevailing Unix, MacOS and DOS/Windows OSs of the present day. While this will be a sensible design decision for many CIF-reading applications, software authors must be aware that the CIF specification aims for portability and archivability through a more general understanding of what constitutes a line of text.

Copyright © 2006 International Union of Crystallography

2.2.4.3. Line lengths

The STAR File does not restrict the lengths of text lines. The original CIF specification introduced an 80-character limit to facilitate programming in Fortran and transmission of CIFs by email. Contemporary practices have enabled the line-length limit in the version 1.1 specification to be extended to 2048 characters. While the new limit in effect mandates the use of a 2048-character input buffer in compliant CIF-reading software, there is no obligation on CIF writers to generate output lines of this length.

2.2.4.4. Lengths of data names and block codes

CIF imposes another length restriction that is not integral to the STAR File syntax: data names, data-block codes and frame codes may not exceed 75 characters in length. This is an increase over the 32-character limit of the original specification, although this extension had already been approved by COMCIFS to coincide with the release of version 2 of the core dictionary (see Chapter 3.2).

There is no fundamental technical reason underlying this restriction; it permits assignment of a fixed-length buffer for recording such data names within a software application, but perhaps more significantly, it encourages a measure of conciseness in the creation of data names based on hierarchical component terms.

2.2.4.5. Case sensitivity

Following the general STAR File approach, the special tokens `loop_`, `save_`, the reserved word `global_`, data-block codes and save-frame codes, and data names are all case-insensitive. The case of any characters within data values must, however, be respected.

2.2.5. Common semantic features

As mentioned in Section 2.2.1, the STAR File structure allows retrieval of indexed data values without prior knowledge of the location of a data item within the file. Consequently there is no significance to the order of data items within a data block. However, molecular-structure applications will typically need to do more than retrieve an arbitrary string value. There is a need to identify the nature of individual data items (achieved portably through the definitions of standard data names in dictionary files), but also to be able to process the extracted data according to whether it is numerical or textual in nature, and possibly also to parse and extract more granular information from the entire data field that has been retrieved.

Different CIF applications have a measure of freedom to define many of the details of the content of data fields and the ways in which they may be processed – in effect, to define their semantic content. However, there are a number of conventions that are common to all CIF applications and this should be recognized in software applicable to a range of dictionaries.

These are discussed in some detail in the formal specification document in Section 2.2.7.4; in this section some introductory comments and additional explanations are given.

2.2.5.1. Data-name semantics

It is a fundamental principle of the STAR File approach that a data name is simply an arbitrary string acting as an index to a required value or set of values. It is equally legitimate to store the value of a crystal cell volume either as

```
_bdouiGFG78=z  1085.3(3)
```

or as

```
_cell_volume   1085.3(3)
```

provided that the users of the file have some way of discovering that the cell volume is indeed indexed by the tag `_bdouiGFG78=z` or `_cell_volume`, as appropriate.

However, it is conventional in CIF applications to define (in public dictionary files) data names that imply by their construction the meaning of the data that they index. Chapter 3.1 discusses the principles that are recommended for constructing data names and defining them in public dictionaries, and for utilizing private data names that will not conflict with those in the public domain.

Careful construction of data names according to the principles of Chapter 3.1 results in a text file that is intelligible to a scientist browsing it in a text editor without access to the associated dictionary definition files. In many ways this is useful; it allows the CIF to be viewed and understood without specialized software tools, and it safeguards some understanding of the content if the associated dictionaries cannot be found. On the other hand, there is a danger that well intentioned users may gratuitously invent data names that are similar to those in public use. It is therefore important for determining the correct semantic content of the values tagged by individual data names to make maximum possible disciplined use of the registry of public dictionaries, the registry of private dataname prefixes, and the facilities for constructing and disseminating private dictionaries discussed in Chapter 3.1.

2.2.5.2. Data typing

In the STAR File grammar, all data values are represented as character strings. CIF applications may define data types, and in the macromolecular (mmCIF) dictionary (see Chapter 3.6) a range of types has been assigned corresponding to certain contemporary computer data-storage practices (*e.g.* single characters, case-insensitive single characters, integers, floating-point numbers and even dates). This dynamic type assignment is supported by the relational dictionary definition language (DDL2; see Chapter 2.6) used for the mmCIF dictionary and is not available for all CIF applications.

However, a more restricted set of four primary or base data types is common to all CIF applications.

The type **numb** encompasses all data values that are interpretable as numeric values. It includes without distinction integers and non-integer reals, and the values may be expressed if desired in scientific notation. At this revision of the specification it does not include imaginary numbers. All numeric representations are understood to be in the number base 10.

It is, however, a complex type in that the standard uncertainty in a measured physical value may be carried along as part of the value. This is denoted by a trailing integer in parentheses, representing the integer multiple of the uncertainty in the last place of decimals in the numeric representation. That is, a value of '1085.3(3)' corresponds to a measurement of 1085.3 with a standard uncertainty of 0.3. Likewise, the value 34.5(12) indicates a standard uncertainty of 1.2 in the measured value.

Care should be taken in the placement of the parentheses when a number is expressed in scientific notation. The second example above may also be presented as 3.45E1(12); that is, the standard uncertainty is applied to the mantissa and not the exponent of the value.

Note that existing DDL2 applications itemize standard uncertainties as separate data items. Nevertheless, since the DDL2 dictionary includes the attribute `_item_type_conditions.code` with an allowed value of 'esd', future conformant DDL2 parsers might be expected to handle the parenthesized standard uncertainty representation.