2.2.7.3.1. *CIF grammar*

(58) A CIF may be an empty file, or it may contain only comments or white space, or it may contain one or more data blocks. Comments before the first block are acceptable, and there must be white space between blocks.

```
<CIF> ::= <Comments>? <WhiteSpace>?
          { <DataBlock>
             { <WhiteSpace> <DataBlock> }*
             { <WhiteSpace> }?
          }?
```

(59) For a data block, there must be a data heading and zero or more data items or save frames.

```
<DataBlock> ::= <DataBlockHeading>
                  { <WhiteSpace>
                    { <DataItems> | <SaveFrame> }
                  }*
```

(60) A data-block heading consists of the five characters **data_** (case-insensitive) immediately followed by at least one non-blank character selected from the set of ordinary characters or the non-quote-mark, non-blank printable characters.

```
<DataBlockHeading> ::= <DATA_> { <NonBlankChar> }+
```

(61) For a save frame, there must be a save-frame heading, some data items and then the reserved word **save_**.

```
<SaveFrame> ::= <SaveFrameHeading>
                {<WhiteSpace> <DataItems>}+
                <WhiteSpace> <SAVE_>
```

(62) A save-frame heading consists of the five characters **save_** (case-insensitive) immediately followed by at least one non-blank character selected from the set of ordinary characters or the non-quote-mark, non-blank printable characters.

```
<SaveFrameHeading> ::= <SAVE_> { <NonBlankChar> }+
```

(63) Data come in two forms:

(i) A data-name tag separated from its associated value by a `<WhiteSpace>`.

(ii) Looped data. The number of values in the body must be a multiple of the number of tags in the header.

```
<DataItems>  ::= <Tag> <WhiteSpace> <Value> |
                 <LoopHeader> <LoopBody>
<LoopHeader> ::= <LOOP_> { <WhiteSpace> <Tag> }+
<LoopBody>   ::= <Value> { <WhiteSpace> <Value> }*
```

2.2.7.4. Common semantic features

2.2.7.4.1. *Introduction*

(1) The Crystallographic Information File (CIF) standard is an extensible mechanism for the archival and interchange of information in crystallography and related structural sciences. Ultimately CIF seeks to establish an ontology for machine-readable crystallographic information – that is, a collection of statements providing the relations between concepts and the logical rules for reasoning about them.

Essential components in the development of such an ontology are:

(*a*) the basic rules of grammar and syntax, described in Sections 2.2.7.1 to 2.2.7.3;

(*b*) a vocabulary of the tags or data names specifying particular objects;

(*c*) a taxonomy, or classification scheme relating the specified objects;

(*d*) descriptions of the attributes and relationships of individual and related objects.

In the CIF framework, the objects of discourse are described in so-called data dictionary files that provide the vocabulary and taxonomic elements. The dictionaries also contain information about the relationships and attributes of data items, and thus encapsulate most of the semantic content that is accessible to software. In practice, different dictionaries exist to service different domains of crystallography and a CIF that conforms to a specific dictionary must be interpreted in terms of the semantic information conveyed in that dictionary.

However, some common semantic features apply across all CIF applications, and the current document outlines the foundations upon which other dictionaries may build more elaborate taxonomies or informational models.

2.2.7.4.2. *Definition of terms*

(2) The definitions of Section 2.2.7.1.2 also hold for this part of the specification.

2.2.7.4.3. *Semantics of data items*

(3) While the STAR File syntax allows the identification and extraction of tags and associated values, the interpretation of the data thus extracted is application-dependent. In CIF applications, formal catalogues of standard data names and their associated attributes are maintained as external reference files called data dictionaries. These dictionary files share the same structure and syntax rules as data CIFs.

(4) At the current revision, two conventions (known as dictionary definition languages or DDLs) are supported for detailing the meaning and associated attributes of data names. These are known as DDL1 (Hall & Cook, 1995) and DDL2 (Westbrook & Hall, 1995), and they differ in the amount of detail they carry about data types, the relationships between specific data items and the large-scale classification of data items.

(5) While it may be formally possible to define the semantics of the data items in a given data file in both DDL1 and DDL2 data dictionaries, in practice different dictionaries are constructed to define the data names appropriate for particular crystallographic applications, and each such dictionary is written in DDL1 or DDL2 formalism according to which appears better able to describe the data model employed. There is thus in practice a bifurcation of CIF into two dialects according to the DDL used in composing the relevant dictionary file. However, the use of aliases may permit applications tuned to one dialect to import data constructed according to the other.

2.2.7.4.4. *Data-name semantics*

(6) Strictly, data names should be considered as void of semantic content – they are tags for locating associated values, and all information concerning the meaning of that value should be sought in an associated dictionary.

(7) However, it is customary to construct data names as a sequence of components elaborating the classification of the item within the logical structure of its associated dictionary. Hence a data name such as **_atom_site_fract_x** displays a hierarchical arrangement of components corresponding to membership of nested groupings of data elements. The choice of components readily indicates to a human reader that this data item refers to the fractional *x* coordinate of an atomic site within a crystal unit

cell, but it should be emphasized from a computer-programming viewpoint that this is coincidental; the attributes that constrain the value of this data item (and its relationship to others such as `_atom_site_fract_y` and `_atom_site_fract_z`) must be obtained from the dictionary and not otherwise inferred.

(8) *Comment:* In practice data names described in a DDL2 dictionary are constructed with a period character separating their specific function from the name of the category to which they have been assigned. In the absence of a dictionary file, this convention permits the inference that the data item with name `_atom_site.fract_x` will appear in the same looped list as other items with names beginning `_atom_site.`, and that all such items belong to the same category.

2.2.7.4.5. *Name space*

(9) The intention of the maintainers of public CIF dictionaries is to formulate a single authoritative set of data names for each CIF dialect (*i.e.* DDL1 and DDL2), thus facilitating the reliable archive and interchange of crystallographic data. However, it is also permissible for users to introduce local data names into a CIF. Two mechanisms exist to reduce the danger of collision of data names that are not incorporated into public dictionaries.

(10) The character string `[local]` (including the literal bracket characters) is *reserved* for local use. That is, no public dictionary will define a data name that includes this string. This allows experimentation with data items in a strictly local context, *i.e.* in cases where the CIF is not intended for interchange with any other user.

(11) Where CIFs including local data items are expected to enjoy a public circulation, authors may register a *reserved prefix* for their sole use. The registry is available on the web at http://www.iucr.org/iucr-top/cif/spec/reserved.html.

A reserved prefix, *e.g. foo*, must be used in the following ways:

(i) If the data file contains items defined in a DDL1 dictionary, the local data names assigned under the reserved prefix must contain it as their first component, *e.g.* `_foo_atom_site_my_item`.

(ii) If the data file contains items defined in a DDL2 dictionary, then the reserved prefix must be:

(*a*) the first component of data names in a category defined for local use, *e.g.* `_foo_my_category.my_item`.

(*b*) the first component following the period character in a data name describing a new item in a category already defined in a public dictionary, *e.g.* `_atom_site.foo_my_item`.

(12) There is no syntactic property identifying such a reserved prefix, so that software validating or otherwise handling such local data names must scan the entire registry and match registered prefixes against the indicated components of data names. Note that reserved prefixes may not themselves contain underscore characters.

2.2.7.4.6. *Note on handling of units*

(13) The published specification for CIF version 1.0 permitted data values expressed in different units to be tagged by variant data names (Hall *et al.*, 1991, p. 657):

> ... Many numeric fields contain data for which the units must be known. Each CIF data item has a default units code which is stated in the CIF Dictionary. If a data item is not stored in the default units, the units code is appended to the data name. For example, the default units for a crystal cell dimension are ångströms. If it is necessary to include this data item in a CIF with the units of picometres, the data name of `_cell_length_a` is replaced by `_cell_length_a_pm`. Only those units defined in the CIF Dictionary are acceptable. The default units, except for the ångström, conform to the SI Standard adopted by the IUCr.

This approach is deprecated and has not been supported by any official CIF dictionary published subsequent to version 1.0 of the core. All data values must be expressed in the single unit assigned in the associated dictionary.

A small number of archived CIFs exist with variant data names as permitted by the above clause. If it is necessary to validate them against versions of the core dictionary subsequent to version 1.0, the formal compatibility dictionary cif_compat.dic (ftp://ftp.iucr.org/cifdics/cif_compat.dic) may be used for the purpose. *No other use should be made of this dictionary.*

2.2.7.4.7. *Data-value semantics*

(14) The STAR syntax permits retrieval of data by simply requesting a specific data name within a specific data block. Prior knowledge about data type (*e.g.* text or numbers), whether the item is looped or whether the item exists in the file at all is unnecessary. However, applications in general need to know data type, valid ranges of values and relationships between data items, and a program designer needs to know the purpose of the data item (*i.e.* what physical quantity or internal book-keeping function it represents). While such semantic information may be defined informally for local data items (ones not intended for exchange between different users or software applications), formal descriptions of the semantics associated with data values are catalogued in data dictionary files. Currently two formalisms (dictionary definition languages) for describing data-value attributes are supported; full specifications of these formalisms (known as DDL1 and DDL2) are provided in Chapters 2.5 and 2.6.

2.2.7.4.7.1. *Data typing*

(15) Four base data types are supported in CIF. These are:

(i) **numb**: a value interpretable as a decimal base number and supplied as an integer, a floating-point number or in scientific notation;

(ii) **char**: a value to be interpreted as character or text data (where the value contains white-space characters, it must be quoted);

(iii) **uchar**: a value to be interpreted as character or text data but in a case-insensitive manner (*i.e.* the values `FOO` and `foo` are to be taken as identical);

(iv) **null**: a special data type associated with items for which no definite value may be stored in computer memory. It is the type associated with the special character literal values `?` (query mark) and `.` (full point), which may appear as values for any data item within a data file (see Section 2.2.7.4.8 below). It is also the type assigned to items defined in dictionary files that may not occur in data files.

(16) *Comment:* Many applications distinguish between multi-line text fields and character-string values that fit within a single line of text. While this is a convenient practical distinction for coding purposes, formally both manifestations should be regarded as having the same base type, which might be 'char' or 'uchar'. Applications are at liberty to choose whether to define specific multi-line text subtypes, and whether to permit casting between subtypes of a base type. The examples of character-string delimiters in Section 2.2.7.1.4(20) are predicated on an approach that handles all subtypes of character or text data equivalently.

(17) Where the attributes of a data value are not available in a dictionary listing, it may be assumed that a character string inter-