2.5. Specification of the core CIF dictionary definition language (DDL1)

By S. R. Hall and A. P. F. Cook

2.5.1. Introduction

The CIF approach to data representation described in Chapter 2.2 is based on the STAR File universal data language (Hall, 1991; Hall & Spadaccini, 1994) detailed in Chapter 2.1. An important advantage of the CIF approach is the self-identification of data items through the use of tag–value tuples. This syntax removes the need for preordained data ordering in a file or stream of data and enables appropriate parsing tools to automate access independently of the data source. In this chapter, we will show that the CIF syntax also provides a higher level of abstraction for managing data storage and exchange – that of defining the meaning of data items (*i.e.* their properties and characteristics) as attribute descriptors linked to the identifying data tag.

Each attribute descriptor specifies a particular characteristic of a data item, and a collection of these attributes can be used to provide a unique definition of the data item. Moreover, placing the definitions of a selected set of data items into a CIF-like file provides an electronic dictionary for a particular subject area. In the modern parlance of knowledge management and the semantic web, such a data dictionary represents a *domain ontology*.

In most respects, data dictionaries serve a role similar to spoken-word dictionaries and as such are an important adjunct to the CIF data-management approach by providing semantic information that is necessary for automatic validation and compliance procedures. That is, prior lexical knowledge of the nature of individual items ensures that each item in a CIF can be read and interpreted correctly *via* the unique tag that is the link to descriptions in a data dictionary file. Because the descriptions in the dictionary are machine-parsable, the semantic information they contain forms an integral part of a data-handling process. In other words, machine-interpretable semantic knowledge embedded in data dictionaries leads directly to the automatic validation and manipulation of the relevant items stored in any CIF.

2.5.1.1. The concept of a dictionary definition language (DDL)

The structure or arrangement of data in a CIF is well understood and predictable because the CIF syntax may be specified succinctly (see Chapter 2.2 for CIF syntax expressed using extended Backus–Naur form). In contrast, the meaning of individual data values in a file is only known if the nature of these items is understood. For CIF data this critical link between the value and the meaning of an item is achieved using an electronic 'data dictionary' in which the definitions of relevant data items are catalogued according to their data name and expressed as attribute values, one set of attributes per item.

The dictionary definitions describe the characteristics of each item, such as data type, enumeration states and relationships between data. The more precise the definitions, the higher the level of semantic knowledge of defined items and the better the efficiency achievable in their exchange. To a large degree, the precision achievable hinges on the attributes selected for use in dictionary definitions, these being the semantic vocabulary of a dictionary. Within this context, attribute descriptors constitute a dictionary definition language (DDL). Definitions of the attributes described in this chapter are provided in Chapter 4.9.

The main purpose of this chapter is to describe the DDL attributes used to construct the core, powder, modulated structures and electron density CIF dictionaries detailed in Chapters 3.2–3.5 and 4.1–4.4. We shall see that each item definition in these dictionaries is constructed separately by appropriate choice from the attribute descriptors available, and that a sequence of definition blocks (one block per item) constitutes a CIF dictionary file. The organization of attributes and definition blocks in a dictionary file need not be related to the syntax of a data CIF, but in practice there are significant advantages if they are. Firstly, a common syntax for data and dictionary files enables the same software to be used to parse both. Secondly, and of equal importance, data descriptions and dictionaries need continual updating and additions, and the CIF syntax provides a high level of extensibility and flexibility, whereas most other formats do not. Finally, the use of a consistent syntax permits the dictionary attribute descriptors themselves to be described in their own DDL dictionary file.

While the basic functionality and flexibility of a dictionary is governed by the CIF syntax, the precision of the data definitions contained within it is determined entirely by the scope and number of the attribute descriptors representing the DDL. Indeed, the ability of a DDL to permit the simple and seamless evolution of data definitions and the scope of the DDL to precisely define data items both play absolutely pivotal roles *via* the supporting dictionaries in determining the power of the CIF data-exchange process.

2.5.2. The organization of a CIF dictionary

The precision and efficiency of a data definition language are directly related to the scope of the attribute vocabulary. In other words, the lexical richness of the DDL depends on the number and the specificity of the available language attributes. The breadth of these attributes, in terms of the number of separate data characteristics that can be specified, largely controls the precision of data definitions. However, it is the functionality of attributes that determines the information richness and enables higher-level definition complexity. For example, the attributes that define child–parent relationships between data and key pointers to items in list packets are essential to understanding the data hierarchy and to its validation. Attributes provide the semantic tools of a dictionary.

The choice and scope of attributes in the DDL are governed by both semantic and technical considerations. Attributes need to have a clear purpose to facilitate easy definition and comprehension, and their routine application in automatic validation processes. A CIF dictionary is much more than a list of unrelated data definitions. Each definition conforms to the CIF syntax, which requires each data block in the dictionary to be unique. However, the functionality of a dictionary involves elements of both relational and object-oriented processes. For example, attributes in one definition may refer to another definition *via* `_list_link_parent` or `_list_link_child` attributes, so as to indicate the dependency

Affiliations: Sydney R. Hall, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia; Anthony P. F. Cook, BCI Ltd, 46 Uppergate Road, Stannington, Sheffield S6 6BX, England.

of data items in lists. In this way the DDL, and consequently the dictionaries constructed from the DDL, invoke aspects of relational and object-oriented database paradigms. It is therefore useful to summarize these briefly here.

A relational database model (Kim, 1990) presents data as tables with a hierarchy of links that define the dependencies among tables. These explicit relationships enable certain properties of data to be shared, and, for related data values, to be derived. The structure of data links in a relational database is usually defined separately from the component data. This is an important strength of this approach. However, when data types and dependencies change continually, static relationships are inappropriate, and there is a need for non-relational extensions.

The object-oriented database model (Gray *et al.*, 1992) allows data items and tables to be defined without static data dependencies. A data item may be considered as a self-contained 'object' and its relationships to other objects handled by 'methods' or 'actions' defined within the objects. A database may have a base of statically defined explicit relationships with a dynamic layer of relationships provided by presenting some (or all) items as objects. Objects have well defined attributes, some of which may involve relationships with other data items, but objects need not have pre-ordained links imposed by the static database structure.

The attributes and the functionality of CIF dictionaries incorporate aspects of both the basic relational and the object-oriented model. These provide the flexibility and extensibility associated with object-oriented data, as well as the relational links important to data validation and, ultimately, data derivation.

2.5.3. Definition attributes

Efficient data exchange depends implicitly on the prior knowledge of the data. For CIF data, this knowledge is specified in a data dictionary using definition attributes. A unique set of attribute values exists for each kind of data item, be it numerical, textual or symbolic, because these characteristics represent its identity and function. This is illustrated below with two simple examples.

2.5.3.1. Example 1: attributes of temperature

Every numerical data item has distinct properties. Consider the number 20 as a measure of temperature in degrees. To understand this number it is essential to know its measurement units. If these are degrees Celsius, one knows the item is in the *temperature* class, *degrees Celsius* sub-class, and that a lower enumeration boundary of any value can be specified at -273.15. Such a constraint can be used in data validation. More to the point, without any knowledge of both the class and subclass, a numerical value has no meaning. The number 100 is unusable unless one knows what it is a measure of (*e.g.* temperature or intensity) and, equally, unless one knows what the units are (*e.g.* degrees Celsius, Kelvin or Fahrenheit; or electrons or volts).

2.5.3.2. Example 2: attributes of Miller indices

Knowing the inter-dependency of one data item on another plays a major role in the understanding and validation of data. If a triplet of numbers $5, 3, 0$ is identified as Miller indices h, k, l, one immediately appreciates the significance of the index triplet as a vector in reciprocal crystal space. This stipulates that the three numbers form a single non-scalar data item in which the indices are non-associative (*e.g.* $3, 5, 0$ is not equivalent to $5, 3, 0$) and irreducible (*e.g.* the index 3 alone has no meaning). As a reciprocal-space vector, the triplet has other properties if is part of a list of other reciprocal-space data items; namely, its value represents a key or list pointer (*i.e.* a unique key to a row of items in a list table) to other data items in the list associated with this vector. This means that data forming a 'reflection' list are inaccessible if these indices are absent, or invalid if there is more than one occurrence of the same triplet in the list. Such interdependency and relational information is very important to the application of data, and needs to be specified in a dictionary to enable unambiguous access and validation. Other types of data dependencies will be described in Section 2.5.6.

Table 2.5.4.1. *Comparison of DDL1 and DDL2 variants*

DDL1	DDL2
Data names identify the category of data as `_<category>_<detail>`	Data names identify the category as `_<category>.<detail>`
Definitions are declared as data blocks with `data_<dataname>`	Definitions are declared starting with `save_<dataname>` and ending with `save_`
An irreducible set of items is declared within one definition *e.g.* Miller indices h, k, l	All items are defined in separate frames related by `_item_dependent.name`
Items that appear in lists are identified with the attribute `list_`	List and non-list data items are not distinguished
List dependencies are declared within each definition *e.g.* `_list_reference`	Dependencies are declared in a category definition *e.g.* `_category_key.name`
	Identifies subcategories of data within category groupings *e.g.* matrix
	Provides aliases to equivalent names, including those in DDL1 dictionaries

2.5.4. DDL versions

The capacity of a DDL to precisely define data items depends implicitly on the scope of the available attributes. It is quite possible, therefore, that a completely new data property cannot be specified using an existing DDL. In a field where data types evolve rapidly, the currently used dictionary language may be inadequate for the precise specification of an item. It is inevitable that future data items will exceed the capacity of existing dictionary attributes and methods to describe new data properties, and the dictionaries must evolve much in the same way that spoken languages continually adapt to changing modes of expression.

The first DDL used in crystallography (in 1990) was developed to compile a dictionary of 'core structural' crystallographic data items (Hall *et al.*, 1991). These data items were intended for use when submitting a manuscript to the journal *Acta Crystallographica* (and still are). The 'core' DDL is known commonly as DDL1 (Hall & Cook, 1995). Several years later, the definition of macromolecular crystallographic data items needed hierarchical descriptors for the different levels of structural entities, and an 'mmCIF' DDL, known as DDL2 (Westbrook & Hall, 1995), was developed. DDL2 was used to build the mmCIF dictionary (Bourne *et al.*, 1997). DDL1 is described in this chapter and DDL2 is described in Chapter 2.6. The DDL1 attributes have been used to construct the crystallographic core (fundamental structural), pd (powder diffraction), ms (modulated and composite structures) and rho (electron density) dictionaries. These dictionaries are discussed in Chapters

```
data_atom_site_Cartn_
    loop_ _name                '_atom_site_Cartn_x'
                               '_atom_site_Cartn_y'
                               '_atom_site_Cartn_z'
    _category                   atom_site
    _type                       numb
    _type_conditions            esd
    _list                       yes
    _list_reference            '_atom_site_label'
    _units                      A
    _units_detail              'angstroms'
    _definition
;   The atom-site coordinates in angstroms specified
    according to a set of orthogonal Cartesian axes
    related to the cell axes as specified by the
    _atom_sites_Cartn_transform_axes description.
;
```

(*a*)

```
save__atom_site.Cartn_x
    _item_description.description
;   The x atom-site coordinate in angstroms
    specified according to a set of orthogonal
    Cartesian axes related to the cell axes as
    specified by the description given in
    _atom_sites.Cartn_transform_axes.
;
    _item.name                 '_atom_site.Cartn_x'
    _item.category_id           atom_site
    _item.mandatory_code        no
    _item_sub_category.id       cartesian_coordinate
    _item_aliases.alias_name   '_atom_site_Cartn_x'
    _item_aliases.dictionary    cif_core.dic
    _item_aliases.version       2.0.1
     loop_
    _item_dependent.dependent_name
                               '_atom_site.Cartn_y'
                               '_atom_site.Cartn_z'
    _item_related.related_name
                              '_atom_site.Cartn_x_esd'
    _item_related.function_code associated_esd
    _item_type.code             float
    _item_type_conditions.code  esd
    _item_units.code            angstroms
     save_
```

(*b*)

Fig. 2.5.4.1. Comparison of DDL versions: (*a*) DDL1, (*b*) DDL2.

3.2–3.5 and presented in full in Chapters 4.1–4.4, respectively, of this volume.

It is helpful to give a brief comparison of the definition capabilities of DDL1 and DDL2. The differences may be best illustrated by simply comparing their application to the definition of the same data item. The two separate definitions of `_atom_site_Cartn_x` are shown in Fig. 2.5.4.1.

Although both DDL versions conform to the syntax of the STAR File (DDL1 conforms to the CIF syntax but DDL2 uses STAR File save frames not permitted in CIF data files) and are composed of similar attributes, there are clear differences. These are summarized in Table 2.5.4.1.

The similarity in the two DDL versions is such that software exists for parsing and validating CIFs against the DDL1 or DDL2 dictionaries interchangeably (*e.g.* see the CIF toolbox software *CIFtbx* described in Chapter 5.4).

2.5.5. The structure of DDL1 definitions

The organization of definitions in a CIF dictionary is straightforward. We are all familiar with how words are described in a spoken-language dictionary. Each defined word is followed by a sequence of attribute descriptions, such as phonetic annotation, grammatical context, word origins, meanings, examples of use and so on. The definition of data items in a CIF dictionary is organized in the same way. Each item description is preceded by the item's unique identifying tag as a datablock code, and is composed of a sequence of attribute items specifying the item's characteristics. A full description of the DDL1 attributes is given in Chapter 4.9.

A series of example definitions will be used to introduce the basic structure of the dictionary and definitions. The definitions are of familiar crystallographic data items, extracted from the core CIF dictionary in Chapter 4.1. Note that, in some cases, the text part of the definitions has been abbreviated for conciseness.

```
data_cell_formula_units_Z
    _name                       '_cell_formula_units_Z'
    _category                    cell
    _type                        numb
    _enumeration_range           1:
    _definition
; The number of the formula units in the unit cell
  as specified by _chemical_formula_structural,
  _chemical_formula_moiety or _chemical_formula_sum.
;
```

Fig. 2.5.5.1. DDL1 definition with a few attributes.

```
data_on_this_dictionary
    _dictionary_name            cif_example.dic
    _dictionary_version         0.0
    _dictionary_update          1999-03-15
    _dictionary_history
;  1999-03-11  Created as a dictionary example
   1999-03-15  Further simplifications
;
data_parameter_ABC
    _name                       '_parameter_ABC'
#        <<<<<<<<<< other data attributes here
data_factor_XYZ
    _name                       '_factor_XYZ'
#        <<<<<<<<<< other data attributes here
data_and_so_on
    _name                       '_and_so_on'
#        <<<<<<<<<< other data attributes here
```

Fig. 2.5.5.2. DDL1 definition showing basic dictionary organization.

2.5.5.1. Definition example 1: formula units per cell (Z)

A definition for formula units in a crystal unit cell (Z) is shown in Fig. 2.5.5.1. The definition includes only the attributes `_name`, `_category`, `_type`, `_enumeration_range` and `_definition`. These identify the item's unique properties and enable its validation. The `_category` attribute indicates the class of the defined data item. Since lists may contain only items of one category type, this attribute is critical to data items appearing in lists (see Section 2.5.5.5). The attribute `_enumeration_range` has the value '`1:`' which stipulates that the item must be 'one or a higher number'.

2.5.5.2. Definition example 2: dictionary audit information

A CIF dictionary file contains, in addition to the data definitions, information about the nature of the dictionary. Fig. 2.5.5.2 shows the typical organization of definitions in a dictionary. Each dictionary starts with the audit information giving its version and creation history. This is followed by definitions in separate data

```
data_cell_length_
    loop_ _name               '_cell_length_a'
                              '_cell_length_b'
                              '_cell_length_c'
    _category                  cell
    _type                      numb
    _type_conditions           esd
    _enumeration_range         0.0:
    _units                     A
    _units_detail             'angstroms'
    _definition
;   Unit-cell lengths in angstroms corresponding to
    the structure reported.
;
```

Fig. 2.5.5.3. DDL1 definition of irreducible data items.

```
data_atom_site_attached_hydrogens
_name            '_atom_site_attached_hydrogens'
    _category              atom_site
    _type                  numb
    _list                  yes
    _list_reference        '_atom_site_label'
    _enumeration_range     0:8
    _enumeration_default   0
    _definition
;   The number of hydrogen atoms attached to the atom
    at this site excluding any hydrogen atoms for
    which coordinates (measured or calculated) are
    given.
;
```

Fig. 2.5.5.4. DDL1 definition of a 'list' data item.

blocks. The name of a data block matches the defined data name or the initial portion thereof. Each definition data block contains a sequence of data declarations, one for each attribute. The attributes within a data block constitute the total definition information, and only those attributes appropriate to a given defined item need be specified.

2.5.5.3. Definition example 3: irreducible data items

Some data items are closely related to other data items. Fig. 2.5.5.3 shows the definition of the length components `_cell_length_a`, `*_b` and `*_c` of the crystal unit-cell metric tensor. Because of their different enumeration and units attributes, the definition blocks for cell lengths and angles are specified separately. This definition shows how data items closely related in function may be defined in the same data block. The `loop_` command is used to list the multiple `_name` values. The attribute `_type_conditions` is used to indicate that these values are measurements and can have a standard uncertainty ('s.u.') value appended in parentheses *e.g.* 7.254(2). (The label 'esd' reflects the historical but inaccurate terminology 'estimated standard deviation'.) The `_units` and `_units_detail` attributes stipulate that the measurement unit of length is ångströms.

2.5.5.4. Definition example 4: list data

The definition of the number of hydrogen atoms attached to an atom site is shown in Fig. 2.5.5.4. The attributes `_list` and `_list_reference` signal whether the item is used in a list and what key item is in that list. As described in Chapters 2.1 and 2.2, a CIF list is a two-dimensional table where data items are in the columns (headed by the data names) and the values are the rows. The attribute `_list` is `yes` if a data item may be used in a list. The attribute `_list_reference` specifies the key item as `_atom_site_label`. Each value of a key item must be unique so that the row of associated items can be accessed unambiguously. The attribute `_enumeration_default` specifies a default value for the defined item if it is absent from the data file.

2.5.5.5. Definition example 5: category information

Each CIF dictionary also contains example applications of defined data items, grouped according to category. Fig. 2.5.5.5 shows these for the category ATOM_SITE, as defined in the core dictionary (see Chapter 3.2). A `_type` attribute with a value of `null` specifies that a data block contains no definition information. The particular examples shown here illustrate that the reference key for the ATOM_SITE category is `_atom_site_label`. This data item is present in both examples shown, and in each packet the label is unique.

2.5.5.6. Definition example 6: mandatory and linked items

The example definition in Fig. 2.5.5.6 shows the attributes for the item `_atom_site_label`. As illustrated in Section 2.5.5.5, this item is the reference key to a list of items belonging to the category ATOM_SITE. The attribute `_list_mandatory`, which is set to a value of `yes`, specifies that this item is a mandatory item to a list of category ATOM_SITE items. The attribute `_list_link_child` identifies items in other categories that are 'linked' by derivation to this item and therefore share the same data values. This dependency is known as a child dependency. The definition shows that data items describing the labels of atom sites in a list of angular geometry, *i.e.* `_geom_angle_atom_site_label_1` and `_geom_angle_atom_site_label_2`, are the same labels described by `_atom_site_label` in the atom-site list. This is because the geometry is derived directly from the atom-site data. Note that this dependency requires that the ATOM_SITE list be present in the same CIF as the GEOM_ANGLE list, otherwise the molecular geometry information cannot be linked to the three-dimensional structural information.

2.5.5.7. Definition example 7: joinable lists

In the example definition shown in Fig. 2.5.5.7, the key item `_atom_site_aniso_label` is shown to have a special relationship with the key item `_atom_site_label`. This is because the data items in the list category ATOM_SITE_ANISO may be merged with (*i.e.* joined to) items in the list category ATOM_SITE. If this happens, the item `_atom_site_label` assumes the role as the key to the merged packets and this is signalled using the attribute `_list_link_parent`. When these two categories appear in separate lists, the category ATOM_SITE_ANISO data require the category ATOM_SITE data to be present in the data instantiation, but not *vice versa*. Note that the parent relationship, unlike that in the example of Section 2.5.5.6 arising from a derivation dependency, is because ATOM_SITE_ANISO is a subcategory of ATOM_SITE.

2.5.5.8. Definition example 8: equivalent items

As with the preceding example definition, this example concerns anisotropic atomic displacement parameters. Fig. 2.5.5.8 shows the definition of the component items `_atom_site_aniso_U_` that comprise an irreducible set of matrix elements. The key item in a list of ATOM_SITE_ANISO items is `_atom_site_aniso_label`. This definition illustrates how the attributes `_related_item` and `_related_function` are used to identify the similar data items `_atom_site_aniso_B_` that are atomic displacement parameters related by a simple conversion factor. Other attributes used to specify data relationships are described in Section 2.5.6 below.

```
###############
## ATOM_SITE ##
###############
data_atom_site_[]
    _name                       '_atom_site_[]'
    _category                     category_overview
    _type                         null
    loop_ _example
          _example_detail
# - - - - - - - - - - - - - - - - - - - - - - - - -
;   Example 1 - based on data set TOZ of Willis,
    Beckwith & Tozer [Acta Cryst. (1991), C47,
    2276-2277].
;
;   loop_
    _atom_site_label
    _atom_site_fract_x
    _atom_site_fract_y
    _atom_site_fract_z
    _atom_site_U_iso_or_equiv
    _atom_site_adp_type
    _atom_site_calc_flag
    _atom_site_calc_attached_atom
O1    .4154(4)  .5699(1)  .3026(0)  .060(1)  Uani  ? ?
C2    .5630(5)  .5087(2)  .3246(1)  .060(2)  Uani  ? ?
C3    .5350(5)  .4920(2)  .3997(1)  .048(1)  Uani  ? ?
N4    .3570(3)  .5558(1)  .4167(0)  .039(1)  Uani  ? ?
# - - - - data truncated for brevity - - - -
H321C  .04(1)  .318(3)   .320(2)   .14000   Uiso  ? ?
H322A  .25(1)  .272(4)   .475(3)   .19000   Uiso  ? ?
H322B  .34976  .22118    .40954    .19000   Uiso
                                            calc C322
;
# - - - - - - - - - - - - - - - - - - - - - - - - -
;   Example 2 - based on data set DPTD of Yamin,
    Suwandi, Fun, Sivakumar & bin Shawkataly
    [Acta Cryst. (1996), C52, 951-953].
;
;   loop_
    _atom_site_label
    _atom_site_chemical_conn_number
    _atom_site_fract_x
    _atom_site_fract_y
    _atom_site_fract_z
    _atom_site_U_iso_or_equiv
S1  1  0.74799(9)   -0.12482(11) 0.27574(9) 0.0742(3)
S2  2  1.08535(10)   0.16131(9)  0.34061(9) 0.0741(3)
N1  3  1.0650(2)    -0.1390(2)   0.2918(2)  0.0500(5)
C1  4  0.9619(3)    -0.0522(3)   0.3009(2)  0.0509(6)
# - - - - data truncated for brevity - - - -
;
    _definition
;    Data items in the ATOM_SITE category record
     details about the atom sites in a crystal
     structure, such as the positional
     coordinates, atomic displacement parameters,
     and magnetic moments and directions.
;
```

Fig. 2.5.5.5. DDL1 overview of a category of items.

```
data_atom_site_label
    _name                       '_atom_site_label'
    _category                     atom_site
    _type                         char
    _list                         yes
    _list_mandatory               yes
    loop_ _list_link_child
                '_atom_site_aniso_label'
                '_geom_angle_atom_site_label_1'
                '_geom_angle_atom_site_label_2'
                '_geom_angle_atom_site_label_3'
                '_geom_bond_atom_site_label_1'
                '_geom_bond_atom_site_label_2'

    loop_ _example    C12     Ca3g28     Fe3+17
                      H*251  boron2a  C_a_phe_83_a_0
                      Zn_Zn_301_A_0
    _definition
;    The _atom_site_label is a unique identifier for
     a particular site in the crystal.
;
```

Fig. 2.5.5.6. DDL1 definition of a ‘mandatory’ data item.

```
data_atom_site_aniso_label
    _name                      '_atom_site_aniso_label'
    _category                    atom_site
    _type                        char
    _list                        yes
    _list_link_parent          '_atom_site_label'
    _definition
;   Anisotropic atomic displacement parameters are
    usually looped in a separate list. If this is the
    case, this code must match the _atom_site_label
    of the associated atom in the atom coordinate
    list and conform with the same rules described
    in _atom_site_label.
;
```

Fig. 2.5.5.7. DDL1 definition of a ‘parent’ data item.

```
data_atom_site_aniso_U_
    loop_ _name                 '_atom_site_aniso_U_11'
                                '_atom_site_aniso_U_12'
                                '_atom_site_aniso_U_13'
                                '_atom_site_aniso_U_22'
                                '_atom_site_aniso_U_23'
                                '_atom_site_aniso_U_33'
    _category                    atom_site
    _type                        numb
    _type_conditions             su
    _list                        yes
    _list_reference            '_atom_site_aniso_label'
    _related_item              '_atom_site_aniso_B_'
    _related_function            conversion
    _units                      A^2^
    _units_detail              'angstroms squared'
    _definition
;   These are the standard anisotropic atomic
    displacement components in angstroms squared.
;
```

Fig. 2.5.5.8. DDL1 definition showing ‘related’ data items.

2.5.5.9. Definition example 9: enumeration states

The last example, in Fig. 2.5.5.9, shows the definition of an item whose value is restricted to a predictable set of values known as enumeration states. The attributes `_enumeration` and `_enumeration_detail` are used to specify which enumeration states are permitted for the defined data item. Only one of these states may appear as the value for the defined item in a CIF. The attribute `_enumeration_default` specifies the state value that is used if an item is not instantiated.

2.5.6. DDL1 attribute descriptions

This section provides an overview of the different attributes that make up the core data dictionary language DDL1. A more detailed description of attributes is given in the DDL1 dictionary in Chapter 4.9. In this dictionary the attributes are used to define themselves!

```
data_atom_site_adp_type
    _name                      '_atom_site_adp_type'
    _category                    atom_site
    _type                        char
    _list                        yes
    _list_reference            '_atom_site_label'
    loop_ _enumeration
          _enumeration_detail
                         Uani   'anisotropic Uij'
                         Uiso   'isotropic U'
                         Uovl   'overall U'
                         Umpe   'multipole expansion U'
                         Bani   'anisotropic Bij'
                         Biso   'isotropic B'
                         Bovl   'overall B'
    _definition
;   A standard code used to describe the type of
    atomic displacement parameters used for the site.
;
```

Fig. 2.5.5.9. DDL1 definition showing enumeration states.

DDL1 attributes are most easily understood when considered in groups with common descriptive functions. There are five basic functional groups of attributes, which perform the definition tasks of identifying, describing, typing, relating and registering data items.

2.5.6.1. Identification attributes

Establishing the identity of a data item is a primary function of a dictionary. In a data instantiation each item is recognized by a unique code, known as its data name or tag. This tag provides the most fundamental level of data validation.

The 'identification' attribute in DDL1 dictionaries is `_name`, and appears in a definition as

```
_name  '<dataname>'
```

The tag of the defined item is the value of `_name` and, because it starts with an underscore, it must always be bounded by quotes to prevent it from being interpreted as the start of another tag–value pair. If there is more than one data item in a definition, as in the case of an irreducible set of items, the data names are entered as a list starting with the statement

```
loop_ _name
```

(see the example in Fig. 2.5.5.8).

The presence of the identification attribute `_name` in a definition is mandatory. Its value, the name of the defined items, provides for spelling validation. Note that if a data item in a CIF is not defined in a dictionary the CIF is still valid, although CIF parsers that employ dictionaries as part of the scanning process usually flag undefined items. The accepted practice to date is that undefined data are largely ignored by a dictionary validation process. In contrast, the need for certain defined items to be present in CIF data can be crucial because of list dependencies. The attributes that specify data relationships and links are considered in Section 2.5.6.4.

2.5.6.2. Descriptive attributes

Three DDL1 attributes are used to provide text descriptions of a defined data item. These are present in a dictionary for human readability, browser software or for the production of text dictionaries such as those in Part 4. This group of attributes is not machine interpretable.

```
_definition
_example
_example_detail
```

The `_definition` attribute provides a text description and as such is the primary semantic content in a defined data item. It may also be used to provide supplementary information about other machine-parsable attributes in the definition (see the definition in Fig. 2.5.5.3). The attributes `_example` and `_example_detail` are used to show typical instantiations of the defined item, as also shown in Fig. 2.5.5.6.

2.5.6.3. Typing attributes

This class of attributes is used to specify the fundamental characteristics of data items and how they may be instantiated. These attributes are machine-parsable and of particular importance in the validation of CIF data. They are

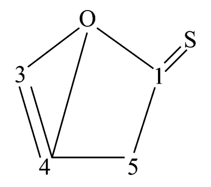
```
_enumeration
_enumeration_default
_enumeration_detail
_enumeration_range
_list
_list_level
_type
_type_conditions
_type_construct
_units
_units_detail
```

Enumeration attributes are used to specify restricted values, or 'states', of a data item (see the example in Fig. 2.5.5.9). They are not applicable in a definition of an item with unrestricted values. The attributes `_enumeration` and `_enumeration_detail` are used in definitions to specify permitted states and their descriptions. For instance, in a definition of atomic element symbols these attributes would be used to list the IUPAC 'element symbols' and 'element names'. When a data item is restricted to an ordinal set of values, the attribute `_enumeration_range` is used to define the minimum and maximum values of the logical sequence in the format `<min>:<max>` The attribute `_enumeration_default` may be used to specify the default value if an item is not present in a data instantiation.

List attributes specify how, and when, data items are used in a (looped) list. The attribute `_list` has a value of `yes` if a defined item must be in a list, and `no` if it must not. A value of `both` allows for both uses. The attribute `_list_level` specifies the nesting level of a list defined item. Only level 1 data are allowed in a CIF (see Chapter 2.2). Higher-nested list levels are permitted for MIF data (Allen *et al.*, 1995; see Chapter 2.4 for a description of the MIF format). Fig. 2.5.6.1 shows MIF data specifying a 2D chemical molecular diagram. The first four items in the category DISPLAY are in a level 1 list and the next two items in the category DISPLAY_CONN are in a level 2 list. Note that, according to the STAR File syntax, the nest level is automatically incremented after the first four values, and only decremented when a `stop_` signal is encountered.

Additional `_list` attributes for specifying relational dependencies between data items are described in the next attribute group.

Type attributes are used to specify the form of data. The attribute `_type` is restricted to the states `numb`, `char` and `null` that code for a number, a text string and a dictionary descriptor, respectively. The implications of this typing, in terms of how numbers or character strings are interpreted, is not specified by the CIF syntax. That is, a 'number' is simply a string of characters having one of several possible representations (*e.g.* as an integer, a floating point or in scientific notation) and it is up to the parsing software to interpret these strings appropriately. The same applies in the treatment of text strings defined as 'character'. Such strings may be bounded



```
loop_
    _display_id
    _display_symbol
    _display_coord_x
    _display_coord_y
    loop_
        _display_conn_id
        _display_conn_symbol
            1  1    180  110    2 1b    stop_
            2  O    140  140    3 1b    4 1b    stop_
            3  3    100  110    4 2b    stop_
            4  4    120   60    5 1b    stop_
            5  5    160   60    1 1b    stop_
            6  S    205  140    1 2b    stop_
```

(*a*)

```
data_display_coord_
    loop_ _name              '_display_coord_x'
                             '_display_coord_y'
    _type                    numb
    _list                    yes
    _list_level              1
    _definition
;   The projected coordinates of the display
    object identified in _display_object and
    _display_symbol.
;

data_display_conn_symbol
    _name                   '_display_conn_symbol'
    _type                    char
    _list                    yes
    _list_level              2
    _definition
;   The symbol code for the bond connection to the
    displayed objects identified by _display_id and
    _display_conn_id. The symbol codes and
    descriptions are stored in the validation file
    'mif_core_bonds.val'.
;
```

(*b*)

Fig. 2.5.6.1. (*a*) Hypothetical example and (*b*) the typical definition of nested items.

by white space, quotation characters or, in the case of multi-line text, by semicolon characters in column 1 of a line.

The attribute `_type_conditions` is used to identify application-specific conditions that apply to the typing of data items. This attribute is used to alert parsing software of string constructions that may disrupt the normal identification and validation of data types. The permitted attribute states are `none`, `esd` (and its preferred synonym `su`) and `seq`. In the definition example in Fig. 2.5.5.3 the code `esd` signals that cell-length values are measurements and may have a standard uncertainty value appended in parentheses. Fig 2.5.5.8 indicates the use of the code `su` for the same purpose. A full description of `_type_conditions` states is given in the DDL1 dictionary in Chapter 4.9.

If a data item consists of a concatenation of values, the attribute `_type_construct` may be used to specify the encoding algorithm for the component values. The language used for this algorithm is POSIX regex (IEEE, 1991). The specification of `_type_construct` can include the data names of other defined items. Validation software interprets this attribute as format specifications and constraints, and expands the embedded data items according to their

```
data_publ_date
    _name                  '_publ_date'
    _type                  char
    _type_construct
        (_publ_year)/(_publ_month)/(_publ_day)
    _definition
;   Specifies the syntax for declaring the
    publication date.
;

data_publ_year
    _name                  '_publ_year'
    _type                  char
    _type_construct        19|20[0-9][0-9]
    _definition
;   Specifies how the publication year will
    appear. Syntax is valid only for publication
    in the 20th and 21st centuries!
;
```

Fig. 2.5.6.2. Definition example: type constructions.

separate definition. For example, the chronological date is a composite of day, month and year. These may be represented in many different ways and the `_type_construct` attribute enables a specific date representation to be defined. The two definitions shown in Fig. 2.5.6.2 illustrate how a date value of the construction '1995/03/25' may be embedded into the dictionary definitions (the month and day definitions have been omitted here for brevity).

Units attributes specify the measurement units permitted for a numerical data item. The attribute `_units` specifies a code that uniquely identifies the measurement units. A description of the units identified by the code is given by the attribute `_units_detail`. Typical uses of this attribute are shown in Fig. 2.5.5.8.

2.5.6.4. Relational attributes

This class of attributes is used to describe special relationships and dependencies between data items. These attributes are machine parsable and are formally defined in the DDL1 dictionary in Chapter 4.9. They are

```
_category
_list_link_child
_list_link_parent
_list_mandatory
_list_reference
_list_uniqueness
_related_item
_related_function
```

The attribute `_category` is used to specify to which group, or basis set, a data item belongs. The value of this attribute is a name string that identifies the group and it is usually the leading part of the tags for all items in this group. Data items in a list must have the same category value. Data items in the same category may, however, be divided into different lists, provided each list contains an appropriate key data item (see `_list_reference` below).

List-link attributes are used to specify dependencies between data items in different lists. The attribute `_list_link_parent` identifies an item in another list from which the defined item was derived. The parent data item, or items in the case of irreducible sets, must have a unique value within its own list. The `_list_link_child` attribute is declared in the definition of a parent item and identifies items in other lists that depend implicitly on the defined data item being present in the same data instantiation. The functionalities of these two attributes mirror each other. The parent item must be present in any data instantiation containing the child items, but not the converse. The definition examples in Figs 2.5.5.6 and 2.5.5.7 illustrate the use of these attributes.

The attributes **`_list_mandatory`** and **`_list_reference`** are also connected to each other. The former signals (with values of `yes` or `no`) whether the presence of a data item is essential to preserving the validity of a list of items. The latter attribute identifies the data item in the list that provides the unique key value to each packet or row of items in a list. A packet is made up of listed values, one for each item in the name list (*i.e.* the packet size matches the number of data names at the head of the list). The **`_list_reference`** attribute identifies items that are the keys to specific packets in the list. In Example 2 of Fig. 2.5.5.5 the key item is **`_atom_site_label`** and the listed labels S1, S2, N1 and C1 must be unique.

The attribute **`_list_uniqueness`** is used to identify items that must be unique for a list to be valid and accessible. This attribute is similar to **`_list_reference`** except that it appears only in a definition in which **`_list_mandatory`** is set to `yes`. This simplifies validation because it may be used as the placeholder for all items that jointly identify the uniqueness of a list packet. This is in contrast to the attribute **`_list_reference`**, which appears in the definition of every item dependent on this item.

Relational attributes are used to link equivalent data. The **`_related_item`** attribute identifies items related to the defined data item. The nature of this relationship is specified with the **`_related_function`** attribute according to the restricted value states of `alternate`, `convention`, `conversion` and `replace`. The definition of these states is detailed in Chapter 4.9. Relational attributes are used to provide equivalent data items, to replace definitions when definitions are superseded or to change access pathways. These facilities are for archives, because they enable old data to be accessed and the associated definitions to remain in a dictionary even when superseded by new definitions. The old and new definitions are linked by these attributes so that all related data items can be validated and accessed.

2.5.6.5. Dictionary registration attributes

This class of attributes is used to register the dictionary version and audit information. They are

```
_dictionary_history
_dictionary_name
_dictionary_update
_dictionary_version
```

Dictionary attributes are used to record the creation and update history of a dictionary. The attribute **`_dictionary_history`** specifies the entry and update information of the dictionary, and **`_dictionary_name`** specifies the generic name of the electronic file containing the dictionary (the actual name of the file can vary from site to site). The attributes **`_dictionary_version`** and **`_dictionary_update`** specify the version number and the date of the last change in the dictionary. Both items represent important external reference information. An example application of these attributes is shown in Fig. 2.5.5.2.

References

Allen, F. H., Barnard, J. M., Cook, A. F. P. & Hall, S. R. (1995). *The Molecular Information File (MIF): core specifications of a new standard format for chemical data. J. Chem. Inf. Comput. Sci.* **35**, 412–427.

Bourne, P. E., Berman, H. M., McMahon, B., Watenpaugh, K. D., Westbrook, J. D. & Fitzgerald, P. M. D. (1997). *Macromolecular Crystallographic Information File. Methods Enzymol.* **277**, 571–590.

Gray, P. M. D., Kulkarni, K. G. & Paton, N. W. (1992). *Object oriented databases*. New York: Prentice Hall.

Hall, S. R. (1991). *The STAR File: a new format for electronic data transfer and archiving. J. Chem. Inf. Comput. Sci.* **31**, 326–333.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

Hall, S. R. & Cook, A. P. F. (1995). *STAR dictionary definition language: initial specification. J. Chem. Inf. Comput. Sci.* **35**, 819–825.

Hall, S. R. & Spadaccini, N. (1994). *The STAR File: detailed specifications. J. Chem. Inf. Comput. Sci.* **34**, 505–508.

IEEE (1991). *IEEE Standard for Information Technology – Portable Operating System Interface (POSIX) – Part 2: Shell and utilities*, Vol. 1, IEEE Standard 1003.2-1992. New York: The Institute of Electrical and Electronic Engineers, Inc.

Kim, W. (1990). *Introduction to object oriented databases*. Boston: MIT Press.

Westbrook, J. D. & Hall, S. R. (1995). *A dictionary description language for macromolecular structure.* http://ndbserver.rutgers.edu/mmcif/ddl/.