```
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