Example 3.1.9.1. *A standard CIF dictionary definition block.*

```
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
    loop_ _example
          _example_detail  2    'water oxygen'
                           1    'hydroxyl oxygen'
                           4    'ammonium nitrogen'
    _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.
;
```

Example 3.1.9.2. *A modified data attribute for overlaying a public definition.*

```
data_atom_site_attached_hydrogens_restricted
    _name              '_atom_site_attached_hydrogens'
    _enumeration_range   0:4
```

As an example, consider the core CIF dictionary definition mentioned above of the number of hydrogen atoms that might be attached to an atom site (Example 3.1.9.1).

For a particular application, any structures reporting more than four attached hydrogen atoms might be considered as invalid. A validation program to satisfy this requirement might therefore build a composite dictionary from the public cif_core.dic, which contains the definition in Example 3.1.9.1, and the fragment of Example 3.1.9.2, processed in APPEND/OVERLAY modes.

### 3.1.9.2. Protocol implementation

At the time of publication (2005), there is no reference implementation for this protocol, and so the proper treatment of the fine details of merging and overlay operations is not available. The following guidelines outline the first steps in an implementation under DDL1.4.

The description assumes that a composite dictionary is to be assembled from two public dictionaries, a.dic and b.dic, and a local dictionary mod.dic that includes some modifications to the definitions in one or both of the public dictionaries (and is therefore processed in OVERLAY mode). It is assumed that the composite dictionary will be written to disk as a separate file, virtual.dic, although in practice applications may simply construct the image of the composite dictionary in memory.

(1) Each contributing dictionary fragment should have at most one data block containing the data names `_dictionary_name` and `_dictionary_version` (with, optionally, `_dictionary_update` and `_dictionary_history`). The `*_name` and `*_version` together identify the dictionary file uniquely and should match the corresponding entries in the IUCr register if this is a public dictionary. This information is conventionally stored in a data block named `data_on_this_dictionary`.

In DDL1.4, all four of the items `_dictionary_name`, `*_version`, `*_update` and `*_history` are scalars, *i.e.* may not be looped. Hence a new dictionary identifier section in virtual.dic may be constructed as follows.

(i) Create a data block `data_on_this_dictionary` at the beginning of virtual.dic.

(ii) If a name for the composite dictionary is supplied (*via* a command-line switch, for example), write this as the value of `_dictionary_name`; otherwise generate a pseudo-unique string (*e.g.* concatenate the computer identifier string, process number and current date string).

(iii) If a dictionary version number is supplied (*via* a command-line switch, for example), write this as the value of `_dictionary_version`; otherwise supply the value '1.0'.

(iv) Supply the current date in the format *yyyy-mm-dd* as the value of `_dictionary_update`.

(v) Create a composite `_dictionary_history` by concatenation of the individual `_dictionary_history` fragments. The application may add details of the current merge operation to the history field.

(2) There is no significance to the ordering of data blocks containing definitions in dictionaries, although they are conventionally sorted alphabetically. For convenience, data blocks should be written out in the order in which they are encountered in the input primitive dictionary files, except that definitions modified by subsequent entries remain in their initial location.

(3) In STRICT mode, if the same value of `_name` is present in two or more data blocks, the composite dictionary is invalid and the application should raise a fatal error. Otherwise the composite dictionary simply contains the aggregate definitions from multiple input dictionaries.

(4) In REPLACE mode, a stored definition block is discarded and replaced by a new definition of the item referenced by `_name`.

(5) For the OVERLAY mode (assumed in the present discussion), the following procedure is proposed. Load a data block from the first dictionary file. Locate the `_name` tag. (Because `_name` may be looped, a data block may contain definitions for more than one data name. For convenience, we consider only the case of a data block containing a single value of `_name`. In any event, it is possible to separate a set of looped definitions into individual data blocks, each defining only one of the data names in the initial `_name` loop.) Search the next dictionary file for a data block containing the same value of `_name`. Load the contents of that data block.

(i) If the new data block contains only data items that do not appear in the first data block, they are simply concatenated with those already present.

(ii) If the new data block contains a scalar data item already present in the first data block (*i.e.* with `_list no`), discard the stored attributes.

(iii) If the new data block contains data items that may be looped and that occur in the first data block, build a new composite table of values in the following way: (*a*) construct a valid loop header if necessary; (*b*) do not repeat identical sets of values (*i.e.* collapse identical table rows); (*c*) if it is possible to identify the category key, then raise a fatal error if there are identical instances of a key value [after the normalization of step (*b*) has occurred]; (*d*) else append new rows to the table.

When the new composite data block has been built according to these principles, search the next dictionary file specified and repeat.

### 3.1.10. Public CIF dictionaries

So far, seven CIF dictionaries have been published by the IUCr with COMCIFS approval. They are described in the remaining chapters in this part of the volume. This section provides an overview of the large-scale structure of these dictionaries and forms a general introduction to Chapters 3.2 to 3.8.

The public CIF dictionaries have been constructed by experts in a number of different crystallographic fields. They are intended to serve the individual fields in which they have been commissioned and therefore vary in character depending on the requirements