5.3.7.2. *CifSieve*: automatic construction of CIF input functions

Among the utilities described in the ZINC package above was a tool to generate Fortran namelist files. It is a common requirement of applications developers that they should be able swiftly to convert existing programs to read CIF data. While libraries such as *CIFtbx* (Chapter 5.4) and *CIFLIB* (Westbrook *et al.*, 1997) offer very powerful functions for building CIF applications, it can be time-consuming to integrate them with existing software. It is a goal of *CifSieve* (Hester & Okamura, 1998) to enable the rapid creation of new CIF-conversant software by using a CIF dictionary as a template for input data structures.

The *CifSieve* program runs on Unix systems with installed versions of the software utilities and programming languages *bison* or *yacc*, *flex*, Perl (Wall *et al.*, 2000) and C.

5.3.7.2.1. *Overview of the process*

The data names in a CIF are defined in a dictionary written in DDL1 or DDL2 formalism. Therefore, information about the data type and array structure of data variables is already to hand for a software author wishing to determine how to read CIF data into a program's data structures. The *CifSieve* process requires that the programmer augment the relevant CIF dictionary by adding to a copy of the definition of desired items a new attribute, named `_variable_name`, that passes to the application program the name of the associated program variable.

A program *BuildSiv* then reads the augmented dictionary and produces a subroutine capable of reading a CIF and transferring the data items tagged in the augmented dictionary to internal variable storage. The associated data structure is presented in an ancillary file which must be linked to the application program.

CifSieve can produce input subroutines and header or include files for C and Fortran language programs. For C applications, the input subroutine is called *cifsiv_* and is invoked with arguments *cifsiv_(CIF, block)* where *CIF* is the name of the input CIF and *block* is the name of the data block from which data should be read. The data structure is declared in a header file cifvars.h which must be included in subroutines that manipulate the data input from the CIF. For Fortran applications, the input subroutine is also called *cifsiv_*, but takes an additional argument, *blockbeg*, which is the address of the common block containing the input variable names, declared in the include file forcif.inc.

5.3.7.2.2. *The augmented DDL dictionary*

Fig. 5.3.7.4 is an example of the annotations necessary to flag the data names that refer to data items desired to be input from a CIF. The current implementation requires that a copy of the DDL dictionary relevant for the CIF be physically edited to include the new `_variable_name` attribute. The inclusion of such a new attribute will not affect the use of the CIF dictionary for other purposes and by other software.

The definition blocks of data items that are not to be read by the application should be left unchanged.

The value assigned to the `_variable_name` attribute is the name of the variable declared in the application program for storing the input data item. If the items to be input are part of an array (*i.e.* they exist in the CIF as a looped list), the variable name should be supplied as a dimensioned array variable, *e.g.* `atsiteu[1000]` in the example of Fig. 5.3.7.4.

The same attribute (`_variable_name`) may be inserted in DDL1 or DDL2 dictionaries. Separate parsers are supplied for use with either format. When *BuildSiv* is invoked, the parser reads the augmented dictionary and identifies the data items required by the target input subroutine by the presence of a `_variable_name` attribute in the definition block. The definition is read and the relevant values of the type (DDL attribute `_type`), item name (`_name`) and variable name are output in a simple tag–value format and in a standard order. For DDL2 dictionaries, values of `_item_aliases.alias_name` and `_item_linked.parent_name`, if present, are also output. The DDL parser thus transforms and simplifies the dictionary contents.

Where the item-name attribute occurs inside a loop (*i.e.* several data names occur in a single definition block in the dictionary), the variable name for that particular definition block will be given an extra array dimension by *CifSieve*, equal to the number of names in the loop. When a name from this loop is found in a CIF, the value will be read into the respective array location. If an `_item_aliases.alias_name` attribute is present (DDL2), the alias will also be recognized in CIF input files. If this attribute occurs together with looped item names in the domain dictionary, an attempt is made to determine the parent `_item.name` in the loop to which this `_item_aliases.alias_name` refers. This is done within the *BuildSiv* program by examining `_item_linked.parent_name` entries within the same definition block.

Data typing is simplified; the `_item_type.code` values of DDL2 dictionaries are collapsed onto primitive 'numb' or 'char' types. Values of type numb are declared and stored as type double (C) or REAL*8 (Fortran), while values of type char are stored as character arrays char[84] (C) or CHARACTER*84 (Fortran). In consequence, multiple lines of text *cannot* be retrieved with this version of *CifSieve*. Note in particular that values declared as of type 'int' in DDL2 dictionaries will be stored as double-precision real.

```
data_atom_site_aniso_label
    _name                 '_atom_site_aniso_label'
    _category               atom_site
    _type                   char
    _variable_name          mylabel[50]
    _list                   yes

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
    _variable_name          atsiteu[1000]
    _type                   numb

data_reflns_number_
    loop_ _name           '_reflns_number_total'
                          '_reflns_number_observed'
    _category               reflns
    _type                   numb
    _enumeration_range      0:
    _variable_name          reftot

data_refine_ls_extinction_method
    _variable_name          extmet
    _name                 '_refine_ls_extinction_method'
    _category               refine
    _type                   char
    _enumeration_default  'Zachariasen'
```

Fig. 5.3.7.4. Extracts from an augmented DDL1 dictionary (version 1.0 of the core CIF dictionary). The additional `_variable_name` entry is shown in italics.

references