5. APPLICATIONS

the selected data name and hyperlinks to definitions of other data names referred to. Additionally, there is a small text-entry box allowing a specific definition to be retrieved and an 'Index' button to list all available definitions.

5.3.3.3.2. *Options*

As already mentioned, the user could modify the detailed mode of operation of the program. Any or all of the 'initial', 'dictionary' or 'other' checks could disabled.

The 'dictionary' checks could be modified by the user through the 'Options' button of the main control window. The CIF dictionary for validation could be specified; the dictionary itself had to be translated from a source file in DDL format to a Python data structure.

The types of dictionary-based validation supported by the program were:

(i) *List Status* (checking whether a data value should be included in a looped list),

(ii) *Limited Enumeration Options* (checking that a data value is one of the permitted codes where such a constraint exists),

(iii) *Incorrect Enumeration Case* [a special case of (ii), where a data value matches a permitted code except for incorrect alphanumeric case],

(iv) *Enumeration Range* (the data value falls outside the range permitted),

(v) *Value Type (numb or char)* (the data value has the wrong type),

(vi) *List Link Parent* (a data item is present within the data block, but its mandated parent item is not – for example, the data item `_atom_site_aniso_label` should not be present without its parent data item `_atom_site_label`),

(vii) *List Reference* (the required data name used to reference the loop in which the current data name appears is missing),

(viii) *Esd Allowable* (a data value appears to have a standard uncertainty value where one is not expected).

The user could also supply the program with a list of data names that do not appear in the validation dictionary but for which no warning message should be raised. The program normally flagged such nonstandard data names as possible errors and suggested the possible form of a standard data name that might have been intended. This was useful in catching misspellings of additional data items entered by hand.

The program could also be run in a batch mode when the objective was to work through a large volume of CIF data and identify the data blocks that require attention. This mode of operation is particularly useful in databases or publishing houses. In this mode, input is from a named file or from the standard input channel; output is written to standard output or redirected to a results file. The operation of the program may be controlled by the application of various command-line flags.

5.3.3.4. Platform-specific editors

As well as the tools described earlier in this section, which are designed to run under a variety of common operating systems, there are some applications restricted to users of particular types of computer. Here we mention two that run in the popular Microsoft Windows environment on personal computers.

5.3.3.4.1. *beCIF*

The Windows program *beCIF* (Brown *et al.*, 2004) is still in prototype. It is a DDL1-dictionary-driven CIF manipulation tool that does not require detailed knowledge of CIF or dictionary
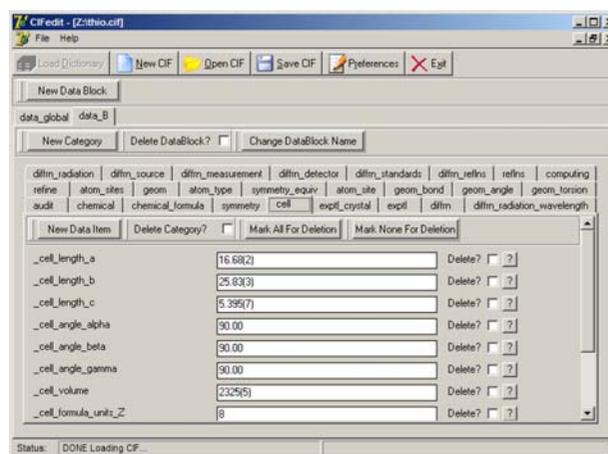

Fig. 5.3.3.11. A category view in the *beCIF* editor of a CIF with navigation by tabs.

structures. It provides a rather different view of the contents of a CIF from the applications discussed above through an interface that will be familiar to users of Microsoft Windows applications. When the application is opened, the user is prompted to provide the location of a CIF dictionary (at any one time, only a single dictionary file may be loaded). This dictionary is loaded into memory and used to validate CIFs upon input. As a data file is read, discrepancies from the types and value ranges permitted by the dictionary are listed in an information window.

The file contents are presented in a number of panels, one per dictionary category, between which the user may navigate by selecting the tab with the desired category name (Fig. 5.3.3.11).

At the highest level, tabs allow the user to choose the data block of interest. Buttons are provided to delete a data block entirely, to rename it or to create a new data block.

Within each data block, the user may add new categories. Again, to help the novice user, when the button 'New Category' is selected, a list of only those categories described in the current dictionary but absent from the current data block is presented to the user. Each category present in the data file is accessed through its own tabbed display panel.

Where the category contains non-looped data items, values may be edited within individual text widgets; data items may be removed by selecting the adjacent check box; or new data items may be added by selecting the 'New Data Item' button to create a dialogue box offering a choice of the remaining data items in the dictionary category. Against each data item a button provides access to a pop-up window containing the relevant dictionary definition.

For a category with looped data, the contents are displayed in a spreadsheet-style representation, with columns headed by the matching data name and rows numbered for convenience (Fig. 5.3.3.12).

The changes requested to the CIF are only effected when the user selects the 'Save CIF' button. Unlike many other of the CIF editors previously discussed, this program does not make any effort to retain the initial ordering of the input data, nor does it preserve comments. The edited CIF may therefore be superficially very different from the input file; however, the only significant differences in content will be those introduced through use of the editing functions within the application.

5.3.3.4.2. *printCIF for Word*

The tools described so far emphasize the data content of a CIF. *printCIF for Word* (Westrip, 2004), on the other hand, was commissioned to help prospective authors of structure reports in the

references