a space-filling molecular representation, and the drop-down menu indicates some of the options available to modify the appearance of the graphics. The molecular-graphics library used by the program is part of the larger database interface software package developed at the Cambridge Crystallographic Data Centre. In the present version, the visualizer is run only upon initial parsing of the input CIF, and therefore does not provide an ability to track visually the molecular changes associated with direct modification of the contents of the file.

5.3.3.2. *CIFEDIT*

The *CIFEDIT* program (Toby, 2003) is written in Tcl/Tk (Ousterhout, 1994) and provides an application for viewing and editing CIFs. The code is written in such a way that it can be embedded into larger programs to provide a CIF-editing interface within larger application suites.

The current version of the program is able to validate CIFs against both DDL1 and DDL2 dictionaries, although the DDL2 validation is currently less complete than for DDL1. For example, numeric values are checked against permitted enumeration ranges only for DDL1. Dictionaries are accessed through index files, each of which contains Tcl data structures that point to the location of the definitions in the dictionary file itself and store information such as units and enumeration ranges that can be used for data validation. A utility provided with the program allows a user to generate new index files when new versions of the dictionaries become available. It is intended that dictionary indexing will be incorporated within the main application in the next program release, so that interactive dictionary selection will be possible.

When a CIF is opened, the contents are parsed and validated against one or more user-selected dictionaries. Errors are displayed in a pop-up window and may be written to a file or viewed within the application. The main program window displays the contents of the CIF in two primary panes (Fig. 5.3.3.6). In the left-hand pane, a tree structure shows the data blocks in the file and the data names present in each block. The data blocks may be expanded or collapsed by the user, to present an overview or a detailed view of the data structure of the file. Underneath the icon representing the data block, non-looped data items are listed alphabetically. The figure demonstrates how a single value may be selected in the left-hand pane (`_cell_length_a`) and displayed in the main window. Physical units for the selected quantity are extracted from the corresponding dictionary definition and presented alongside the numeric value. The dictionary definition may also be displayed in a separate pop-up window using the 'Show CIF Definitions' button.

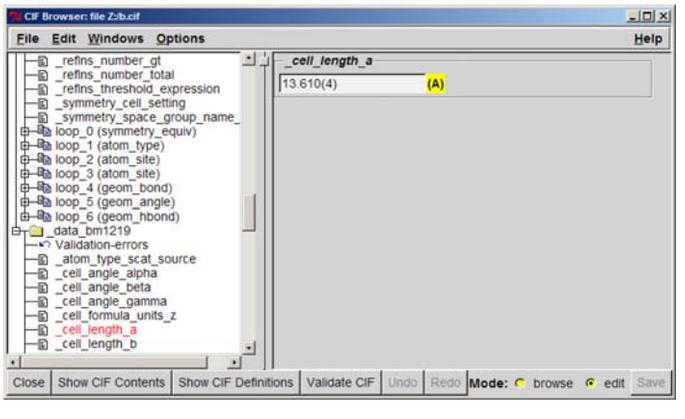

Fig. 5.3.3.6. The use of *CIFEDIT* to display and alter the contents of a CIF; here a non-looped data item is shown.

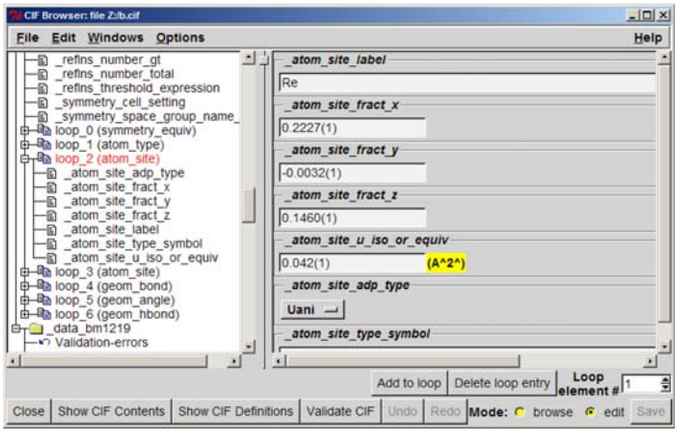

Fig. 5.3.3.7. Row-based loop editing with *CIFEDIT*; here loop_2 (comprising the ATOM_SITE category) has been selected by the user; the editing cursor begins at row 1 of the loop.

The program may be run in two modes: a 'browse' mode, where the selected value is displayed in the main pane, but may not be altered; and an 'edit' mode (as in the example) where the value appears in an editable text widget.

Data loops in the CIF are displayed after the alphabetical list of non-looped items. The loops are numbered sequentially from zero and an indication of the loop category is given in parentheses in the tree-view window. The loop 'branches' of the tree may be expanded or collapsed as the user wishes.

Loops may be viewed and edited in two ways: by row or by column. If the user selects the loop title node in the hierarchical view pane, the loop is presented by row, starting in sequence at row 1 (Fig. 5.3.3.7). Other rows may be selected by using the address box in the lower-right-hand part of the window. Alternatively, if the user selects an individual data name within the loop representation in the hierarchical view, all instances of that data item within the loop are displayed in the main pane. (In practice the number of values shown is constrained to a maximum number that the user may choose, so that the application does not run out of memory if there are very large loops.)

For items with a restricted set of permitted values in the dictionary, the editing function allows the user to select only one of the permitted options *via* a drop-down menu.

While the application is intended to be used in this structured and itemized mode, there is an option to open the entire CIF in a text-editing window if there are errors that cannot be handled in the normal mode. This is not recommended, but is occasionally convenient. While this free-text editing mode is in operation, the ability to modify the file through the structured editing pane is suspended to avoid conflicting changes.

After any change has been made, the user may revalidate the file. This is strongly recommended after making changes in the free-text editing mode.

5.3.3.3. *HICCuP*

The program *HICCuP* (Edgington, 1997) was an early graphical utility developed at the Cambridge Crystallographic Data Centre for interactive editing and validation of a CIF. It is no longer supported, having been replaced by *enCIFer* (Section 5.3.3.1). Nevertheless, it contained some interesting features and is of potential interest to developers using multiple-platform scripting languages. It was implemented in the Python language (van Rossum, 1991) and required that Tcl/Tk (Ousterhout, 1994) be also available on the host computer. The name of the program is an acronym for 'High-Integrity CIF Checking using Python'.

HICCuP was designed to allow users of the Cambridge Structural Database (Allen, 2002) to check structures intended for deposition in the database and therefore included a range of additional content checks specific to this purpose. These could, however, be disabled by the user.

5.3.3.3.1. *Interactive use of the program*

5.3.3.3.1.1. *The control window*

Because *HICCuP* was designed as an interactive tool, upon invocation it presented to the user a *control window* from which CIFs could be selected for analysis and in which summary results of the program's operations were logged. Fig. 5.3.3.8 shows an example of the control window after a single CIF has been loaded.

In the large frame below the file-entry field are listed the data blocks found by the program. The names are highlighted in various colours according to the highest level of severity of errors found within the corresponding data block.

Because the utility was designed for processing large amounts of CIF data for structural databases, it was considered useful to supply a compact visual indicator of the progress of the program through a large file. This takes the form of a grid of rectangular cells, one column for each data block present. Each column contains three cells, which monitor the performance of checks on the file syntax, conformance against a CIF dictionary, and other checks specific to the requirements of the Cambridge Crystallographic Data Centre. As each data block was checked, the corresponding cells were coloured according to the types of error found. Different colours were used to indicate: no errors; structure errors in the initial syntax tests; dictionary errors; or a deviation from certain conventions used by journals and databases in naming datablocks.

The large frame at the bottom of the control window provides a text summary of the same information, listing the number of errors found.

Check boxes and an 'Options…' button allowed some configurability of checks by the user.

5.3.3.3.1.2. *The report frame and edit window*

The user could get more details of the reported errors by clicking on the name of the data block of interest in the control window. The text of the CIF would appear in a new window positioned at the point where the program has detected the first error and a terse statement of the type of error, with a longer explanation of its nature and possible cause, would be given.

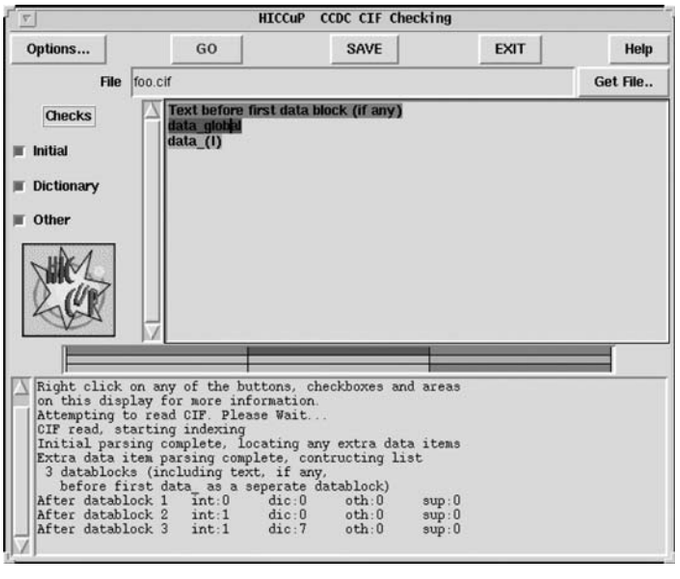

Fig. 5.3.3.8. Control window of the *HICCuP* application.

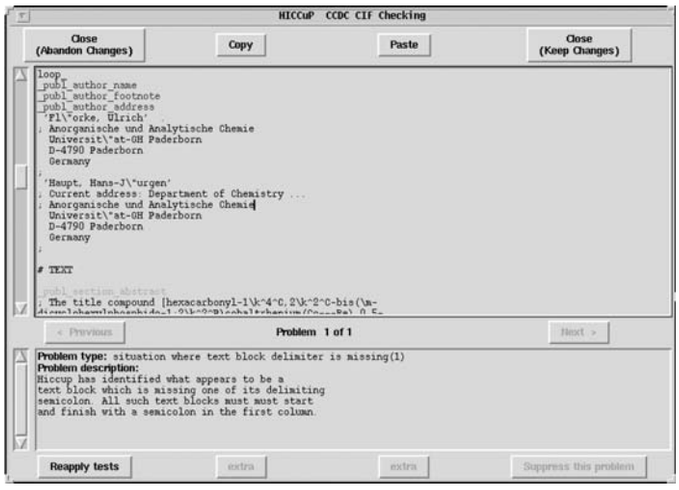

Fig. 5.3.3.9. *HICCuP* edit window and error description.

In the example of Fig. 5.3.3.9, the program has detected that there is a missing text delimiter (a semicolon character), and positions the text in the upper frame at the likely location of the error. The program has attempted to localize the region where the error may have occurred. Because a text field might contain arbitrary contents, including extracts of CIF content, it is impossible to be sure on purely syntactic grounds of the nature of the error. Nonetheless, some heuristic rules serve to identify the author's likely intent in the majority of cases. So, in this example, the user may scan the file contents in the vicinity of the line highlighted by the program and find the error within a few lines (in this example an incorrectly terminated `_publ_author_footnote` entry beginning 'Current address:').

For this example, the more literal *vcif* error analysis provides only the message

```
ERROR: Text field at end of file does not terminate
```

The upper frame in this window is an editable window, so that the user could modify the text and revalidate the current data block. Only when a satisfactorily 'clean' data block was obtained were the changes saved, and the modified data block written back into the original file.

5.3.3.3.1.3. *Dictionary browsing*

An additional useful feature of the program was its interactive link to a CIF dictionary file (Fig. 5.3.3.10). The browser window contains the definition section of the dictionary referring to

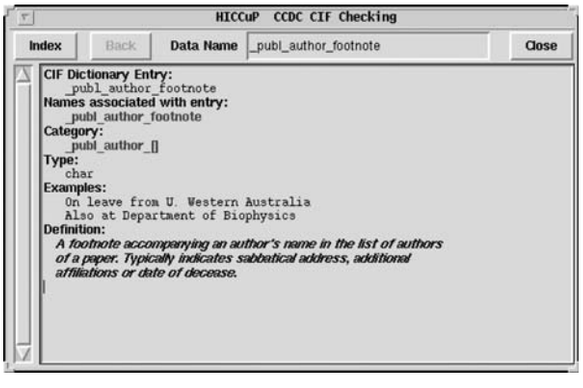

Fig. 5.3.3.10. *HICCuP* dictionary browser window.

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

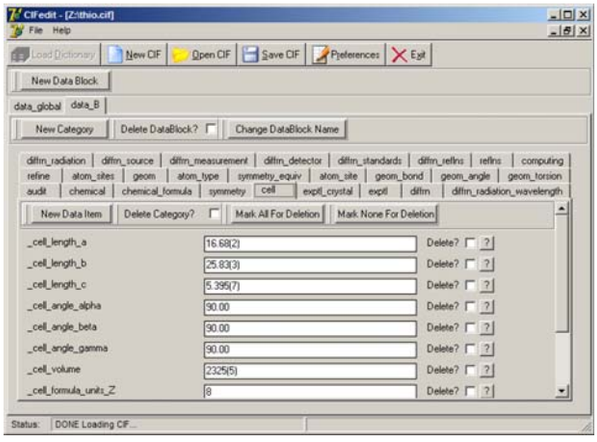

Fig. 5.3.3.11. A category view in the *beCIF* editor of a CIF with navigation by tabs.

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