5.3.5.1.4. *Output from QUASAR*

The body of the request list is a series of data names. Where a data name appears in the CIF, it will be extracted with its associated data value or values. The user need not have prior knowledge of whether a data item occurs in a looped list or not: *QUASAR* will automatically retrieve the matching values and construct a loop header if necessary. However, because the requests are served in the exact order in which they occur in the file, data items in the same list in the input CIF may be extracted into different lists upon output. Although this breaks the semantic association between items grouped in the same list (especially for CIFs described by the DDL2 relational scheme), it is a syntactically valid construction and may be a valuable feature for some processes.

5.3.5.1.4.1. *Treatment of missing data*

When a requested data item is absent from the CIF, *QUASAR* will nevertheless emit a data name with a corresponding value of '?', the conventional CIF value of null type for 'unknown quantity'. A CIF comment is also generated by *QUASAR* to indicate that the entry was missing from the input CIF. If the missing data name is found between data names that have multiple values and that occur in the same looped list, it is assumed that the missing data name should be associated with the same looped list, and it will be emitted in the loop header; the integrity of the list is then satisfied by emitting a column of unknown values. Note how this behaviour differs from that of the generic STAR File extraction utility *Star_Base* (Spadaccini & Hall, 1994), which silently ignores missing data items. However, it is a useful behaviour for applications that depend on finding a specific data item in their processing stream, even where its value is unknown.

5.3.5.1.4.2. *Matching data names*

As with the specification of data-block names, the data names in the request list may have a trailing underscore. Where this is the case, *QUASAR* will retrieve all data items where the data name starts with the specified string. For example, a request for '`_atom_site_`' will extract *all* data names starting with '`_atom_site_`'. The special case of an isolated underscore character '_' matches *all* data names present in the current data block.

5.3.5.1.4.3. *Case sensitivity*

The example demonstrates the way in which the application handles the case insensitivity of a requested data item. Data names are converted internally to a lower-case representation, both from the request list and the input CIF. Matches are therefore determined in a case-insensitive manner. However, if a data name is present in the CIF, its original case is retained on output. This permits the computationally irrelevant but cosmetically useful retention of capitalization as used in canonical CIF dictionary definitions. Where the requested data name is absent, the output is all lower-case.

5.3.5.2. *cif2cif*

cif2cif (Bernstein, 1998) is a program built with the *CIFtbx* toolkit (Chapter 5.4) to copy a CIF while checking data names against dictionaries, optionally reformatting numbers to maintain standard uncertainties within a specified range. The output CIF may contain a subset of the data in the original CIF according to a request list, in the manner of *QUASAR* (Hall & Sievers, 1993).

The program was built as a sample application using *CIFtbx* routines and grew out of requirements from several sources.

5.3.5.2.1. *Operation*

5.3.5.2.1.1. *Copying*

In its simplest application, the program copies a CIF from the standard input channel to standard output. The copy is not verbatim (standard utilities of the computer operating system should be used for that purpose), but the output CIF differs from the input only in the following respects: some comments are deleted; lines in the input longer than 80 characters are wrapped to 80 characters or less; white space between tokens may be altered, especially in an attempt to align entries in looped lists in a cosmetically pleasing manner. While none of these changes should affect robust CIF-parsing applications, they are nevertheless useful in imposing a uniform style of presentation for browsing in a text editor or other human-readable framework.

5.3.5.2.1.2. *Constraining standard uncertainties to specified ranges*

Some journals require that standard uncertainties in experimental values should be quoted within a specified range. Typically the standard uncertainty (s.u.) should be quoted as an integer in parentheses, modifying the last place or two of decimals in the experimental data, and with a value between 2 and 19. *cif2cif* permits s.u. values in the ranges 1–9, 2–19 or 3–29, selectable by a command-line switch. The effect of applying the 'rule of 19' would be to change a value of 1.458(1) in the input CIF to 1.4580(10) in the output.

5.3.5.2.1.3. *Dictionary validation*

cif2cif will open one or more CIF dictionary files as it copies the input CIF and identify certain classes of error against the dictionary definitions. The conditions that will raise an error are an unrecognized data name or a wrong data type. The program will also optionally indicate a warning if a data name has been assigned a category different from the leading portion of the data name – this may indicate an inconsistency within the dictionary itself.

5.3.5.2.1.4. *Serving a request list*

cif2cif will extract a subset of the data items contained in a CIF as specified by a request list, in the manner of *QUASAR*. The handling of data names specified in the request list is as described in Section 5.3.5.1.3 above, with the following additional feature. The special string `data_which_contains:` will extract the specified data items from the first data block in which at least one occurs; the block code need not be known in advance.

Some care must be exercised in attempting to extract data from data blocks by context without prior knowledge of the file contents. Consider the following simple example file:

```
data_A
   loop_    _A1
            _A2
             a1 a2 aa1 aa2


data_B
   loop_    _A1
            _B1
             a b aa bb
```

The loop containing **`_A1`** and **`_B1`** *cannot* be extracted with a request list of the form

```
data_which_contains:
_A1
_B1
```

because **_A1** occurs in the first data block encountered; the output from *cif2cif* in this example will be

```
data_A
    loop_     _A1
               a1   aa1
#             ---end-of-data-block---
```

The behaviour of the program differs from *QUASAR* in two other small ways. When the request list forces the output data stream to contain the same data-block header more than once, an error message is posted to the standard error channel and the data-block headers in the output stream are annotated with a comment of the form '`#<---- duplicate data block`'. In this case the output file does *not* conform to the CIF syntax rules.

When a data name is requested but no matching data item appears in the output file, *cif2cif* writes an error message to the standard error channel. However, unlike *QUASAR*, which inserts the requested data name in the output stream with an associated value of '`?`' (for *unknown*), *cif2cif* produces *no* output for the requested data item.

5.3.5.2.1.5. *Other features*

Some additional features are of use in special circumstances.

The user may preserve the layout of the contents of looped lists exactly as in the input file, or may ask the program to adjust the layout to a more visually pleasing tabular form.

The user may enable recognition of data-name aliases in the dictionaries used for validation. When the relevant command-line argument is set to *true*, user-supplied data names will be transformed to the canonical forms in the validating dictionary. This would permit, for example, a small-molecule CIF using the core dictionary definitions to be converted to mmCIF format.

The user may prefix each line of output with an identical character string. A typical reason for so doing would be to include a fragment of CIF listing within the body of an email message or some other document. Such an output would not conform to the syntax rules for CIF.

5.3.5.2.2. *Invocation of the program*

cif2cif is another application of the *CIFtbx* library by the same author, and so has a similar user interface to that of *CYCLOPS* (Section 5.3.4.1.2). Under a Unix-like operating system, the program is typically called with a command such as

`cif2cif -i` *`infile`* `-o` *`outfile`* [`-q` *`reqfile`*]

where *infile* is the name of the input file, *outfile* is the output file and *reqfile* is an optional file containing a request list for a subset of the original contents.

A more complete set of options available in a Unix-like operating environment is

`cif2cif` [`-i` *`infile`*] [`-o` *`outfile`*] [`-d` *`dictfile`*] [`-q` *`reqfile`*]
[`-f` *`cmndfile`*] [`-c` *`catck`*] [`-a` *`alias`*] [`-t` *`tab`*] [`-e` *`sulim`*]
[`-p` *`prefix`*]

where the options are as follows:

-i specifies the name of the input file, *infile*.

-o specifies the name of the output file, *outfile*.

-d specifies the name of a dictionary file, *dictfile*, against which the existence, type and category of data names are checked. The dictionary file may be *either* a CIF dictionary or a list of file names. That is, it may contain dictionary definitions in DDL format or (if the file begins with the characters **#DICT**) it may contain a list of dictionary file names to be entered. Thus, multiple dictionaries may be specified to the program.

-q specifies the name of the request file, *reqfile*, containing a list of data names (with associated data-block directives) that should be extracted as a subset of the contents of the original file.

-f specifies the name of a command file *cmndfile* that contains additional directives to the program.

-c is a flag indicating whether an error message should be raised if a data name has been assigned a category different from the leading portion of the data name itself. The Boolean variable *catck* may take the values 't', '1' or 'y' for *true*, 'f', '0' or 'n' for *false*.

-a is a flag indicating whether data-name aliases in the validating dictionary should be used to replace user-supplied names by their canonical forms. The Boolean variable *alias* may take the same values for *true* or *false* as above.

-t is a flag indicating whether the output should be reformatted with tabs to produce a regular table layout within looped lists. The Boolean variable *tab* takes the same values as above. If *true*, text is reformatted; if *false*, the original formatting is retained.

For the flags expecting Boolean values, the default is 'f' (*false*).

-e specifies the precision to retain in rounding standard uncertainty values. The permitted integer values are 9, 19 (the default) and 29.

-p takes a string value which is prefixed to every line of output. Every occurrence of the underscore character '_' in the prefix is changed to a space on output.

If no input or output file names are specified, the program will read from the standard input channel or write to standard output, respectively. The special character hyphen ('-') may also be supplied as an argument in place of a file name to indicate standard input or standard output as appropriate.

Finally, if the operating system supports the passing of environment variables to a program, the name of the input file may be passed as the value of `$cif2cif_INPUT_CIF`, and likewise the output file, `$cif2cif_OUTPUT_CIF`, dictionary file, `$cif2cif_CHECK_DICTIONARY`, and request file, `$cif2cif_REQUEST_LIST`, may be specified.

5.3.5.3. *ciftex*: translating to a typesetting language

The program *ciftex* (McMahon, 1993) was developed to create files for typesetting the journal *Acta Crystallographica* using the text-formatting language TEX (Knuth, 1986). Details of its use in the journal production process are given in Chapter 5.7. It is discussed here as an example of translating a CIF to some output format where data values are annotated with different text depending on their accompanying data names.

5.3.5.3.1. *Basic operation of ciftex*

The program is designed to act as a filter, typically in a Unix-style environment, reading a CIF on the standard input channel and outputting a modified data stream to standard output. The output is a file of TEX code that is processed by the TEX program to produce a device-independent file describing the content of a formatted typeset document. Further post-processing allows the formatted document to be viewed on the screen or printed.

Each input token (number, character or text string; data name; **loop_** or **data_** keywords) is transformed as it is identified; there is no lookahead and minimal retention of context. The data stream is treated purely syntactically; no transformations are applied on the basis of the supposed meaning of any of the file contents.