```
data_1
     _example   foo
data_2
     _example   bar
```

despite the latter's equivalence purely in terms of the non-contextual values returned.

### 5.2.3. *Star_Base*: a general-purpose data extractor for STAR Files

The stand-alone application *Star_Base* (Spadaccini & Hall, 1994) provides a facility for performing database-style queries on arbitrary STAR Files. It is generic in nature and makes no assumptions about the nature or organization of the data in a STAR File. It may indeed be used as an application-specific database tool if the user has prior knowledge of the relationships between included data items. However, by faithfully returning context as well as value in the way outlined in Section 5.2.2, it can be applied to any STAR File even without such prior knowledge.

#### 5.2.3.1. Program features

*Star_Base* is a fully functional STAR File parser and may be used to test the syntactic validity of an input STAR File. It may be used to write an input STAR File directly to the output stream, while validating the structural integrity as the contents are parsed. The input format and comments are discarded on output.

Given a valid input file, *Star_Base* guarantees to write output in fully compliant STAR format.

If a data name is supplied as a request item, *Star_Base* will return the single or multiple values associated with that data name and their associated context according to the principles of Section 5.2.2, *i.e.* all loop structures, data-block headers and global headers will be returned, and save frames will be expanded as required to accommodate de-referencing of frame codes as returned values.

Where multiple data items are requested, *Star_Base* will write their occurrences to its output stream in the order they were requested, *not* in the order of appearance in the input file. This may disturb data relationships that are implicit in the ordering or association of values in the input file, but it is the responsibility of the user to track and retain such associations where they are an essential part of an application-level data model. As emphasized before, a generic STAR tool will make no assumptions about data models and will simply return values and contexts as requested.

To illustrate the effect of this, consider a request for the following data items from the example file of Fig. 5.2.2.1:

```
_basis_set_atomic_name
_basis_set_atomic_symbol
_basis_set_contraction_scheme
```

*Star_Base* will return the following result:

```
data_Gaussian
loop_
     _basis_set_atomic_name
     _basis_set_atomic_symbol
    loop_
     _basis_set_contraction_scheme
    stop_

    hydrogen H
         (2)->[2]  (2)->[2]  (2)->[1]  (3)->[2]  stop_

    lithium Li
         (4)->[4]  (9,4)->[3,2]  (4,3)->[3,2]  stop_
```

However, if the same items are requested in a different order,

```
_basis_set_atomic_name
_basis_set_contraction_scheme
_basis_set_atomic_symbol
```

the result is structured differently:

```
data_Gaussian
loop_
     _basis_set_atomic_name
    loop_
     _basis_set_contraction_scheme
    stop_
     _basis_set_atomic_symbol

    hydrogen
         (2)->[2]  (2)->[2]  (2)->[1]  (3)->[2]  stop_
    H
    lithium
         (4)->[4]  (9,4)->[3,2]  (4,3)->[3,2]  stop_
    Li
```

In the examples so far, one or more data items have been requested by name. *Star_Base* extends the type of requests that can be made through its own query language. This gives it much of the power of a database query language such as SQL. Three types of query are supported, known as *data*, *conditional* and *branching* requests.

A *data request* is a straightforward generalization of the request by data name. Individual data items may be requested by name, as may individual data blocks or save frames. Wild carding is permitted to generalize the requests. More details are given in Section 5.2.3.2.

A *conditional request* involves one or more conditions; only data items satisfying the conditions are returned. More details are given in Section 5.2.3.3.

A *branching request* applies similar conditions to establish the context in which matching data items occur within the file, but may also apply scoping rules to select among the available contexts. Only data items matching both the conditions imposed on their values *and* the requested scope are returned. It is the existence of such branching conditions that gives *Star_Base* the ability to select data matching the specific requirements of overlying data models. Again, however, it is emphasized that the program itself operates without any semantic awareness of the significance of the data that is implied within the overlaid data model. More details on branching requests are given in Section 5.2.3.4.

#### 5.2.3.2. The *Star_Base* data request

A *data request* is the simplest type of query used to extract single items from a file. It may be formed from any of the following string types:

(i) a **name** string, *e.g.* `_atom_identity_symbol`;

(ii) a **block** string, *e.g.* `data_Gaussian`;

(iii) a **frame** string, *e.g.* `save_methyl`.

In accordance with the principles set out earlier in this chapter, data requests satisfy the following rules:

(i) Requested data items are returned with their associated context (*i.e.* including the headers of any containing data blocks, save frames and loop structures).

(ii) A request for a data block returns all preceding global blocks (since the data block will contain by inheritance all values in the global blocks).

(iii) A request for a save frame also returns the header of the data block encompassing the save frame. All frame-pointer codes are resolved so that if a requested save frame contains pointer codes to other save frames, these are also returned.

(iv) A request for `global_` returns all global blocks, together with all data-block headers in their scope.

(v) The request need not be specified explicitly. Two wild-card characters are permitted. An asterisk (*) represents *any sequence*