```
data_reaction

save_methyl
  loop_
     _atom_identity_node
     _atom_identity_symbol      1  C  2  C
  loop_
     _attached_hydrogen_node
     _attached_hydrogen_count   1  3
save_

save_ethyl
  loop_
     _atom_identity_node
     _atom_identity_symbol      1  C  2  C  3  C
  loop_
     _attached_hydrogen_node
     _attached_hydrogen_count   1  3  2  3
save_

save_R1
  loop_
     _variable_alternative_number
     _variable_identifier_symbol
     _variable_node     1  $methyl  1   2  $ethyl  1
save_

save_carboxylic_acid
  loop_
     _atom_identity_node
     _atom_identity_symbol      1 $R1  2  C  3  O  4
O
  loop_
     _attached_hydrogen_node
     _attached_hydrogen_count   2  0  3  0  4  1
save_

loop_
     _reaction_component_number
     _reaction_component_symbol
     _reaction_component_type
            1  $carboxylic_acid  reactant
```

Fig. 5.2.2.3. Example STAR data structure where save frames encapsulate related data sets. See text for details.

The scope of data values is well defined (see Section 2.1.3.9). Only data expressed in a global block have values that are inherited in later portions of the STAR File. Data values in data blocks or save frames are restricted in scope to the current data block or save frame, respectively.

A consequence of these rules of scope and encapsulation is that a full description of the context of a STAR data value must also reflect any values carried through as global data or by de-referencing associated save frames. The results are not always intuitive.

Consider Fig. 5.2.2.3, which represents a partial description of a chemical reaction where one of the reactants is expressed as a generic structure described by the save frame `save_R1`. However, the generic structure in this case is restricted to a small number of alkyl groups, each described in its own save frame. Setting aside this prior knowledge, we see that a request for `_atom_identity_symbol` must return not only the data values in their embedded save frames, but also the save frames in their entirety and the higher-order data values that reference the matching save frames. It is only in this way that we can guarantee that the value can be used by any application. Fig. 5.2.2.4 demonstrates the full context of the returned requested data values.

Notice that the requested item occurs (among other places) in the save frame `save_carboxylic_acid` and this instance of the

```
data_reaction

save_methyl
loop_
     _atom_identity_node
     _atom_identity_symbol     1 C 2 C
loop_
     _attached_hydrogen_node
     _attached_hydrogen_count  1 3
save_

save_ethyl
loop_
     _atom_identity_node
     _atom_identity_symbol     1 C 2 C 3 C
loop_
     _attached_hydrogen_node
     _attached_hydrogen_count  1 3 2 3
save_

save_R1
loop_
     _variable_alternative_number
     _variable_identifier_symbol
     _variable_node     1  $methyl  1  2  $ethyl  1
save_

save_carboxylic_acid
loop_
     _atom_identity_symbol $R1 C O O
save_

loop_
     _reaction_component_symbol $carboxylic_acid
```

Fig. 5.2.2.4. Context for the requested values of `_atom_identity_symbol` in the preceding example. See text for details.

item is presented solely in the context of the save header and closure strings (it is shown in italics in Fig. 5.2.2.4). However, one of the values extracted from this location is the save-frame reference pointer $R1 that identifies `save_R1`, and the complete contents of this save frame are presented (because the data structure represented by the save frame *is* itself one of the values of the requested data item). Further de-referencing of the save-frame pointers within `save_R1` results in the extraction also of the complete save frames `save_methyl` and `save_ethyl`. In this example it is coincidental that there are instances of the requested data item (`_atom_identity_symbol`) within these returned save frames as well.

Notice, however, that establishing the full context of the returned data demands also that data values referencing the `save_carboxylic_acid` frame be presented. In this example, the value of `_reaction_component_symbol` at the outermost level of the data block is returned, a result that may at first seem surprising. It is only in this way that one can be sure that an arbitrary application will have access to the full semantic information carried by the data item.

Data values declared in global blocks should be presented in the same spirit of supplying the complete context in which the value was instantiated, and not simply the value in isolation. For example, given the trivial STAR File

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

a request for `_example` should return the identical file, *not* the interpolated result