16 Feature Structures

16.1 Introduction

A *feature structure* is a general purpose data structure which identifies and groups together individual *features*, each of which associates a name with one or more values. Because of the generality of feature structures, they can be used to represent many different kinds of information. Interrelations among various pieces of information, and their instantiation in markup provides a *metalanguage* for representing text analysis and interpretation. Moreover, this instantiation allows feature values to be of specific *types*, and for restrictions to be placed on the values for particular features, by means of *feature system declarations*, which are discussed in chapter 26 *Feature System Declaration*. Such restrictions provide the basis for at least partial validation of the feature-structure encodings that are used.

This chapter is organized as follows. Following this introduction, section 16.2 *Elementary Feature Structures: Features with Binary Values* introduces the *binary* feature values, and shows how elementary feature structures using features with those values may be constructed. Section 16.3 *Feature, Feature-Structure and Feature-Value Libraries* introduces the tags that represent *libraries* of features, feature structures and feature values, along with methods for pointing at features, feature structures and feature values in these libraries. Section 16.4 *Symbolic, Numeric, Measurement, Rate and String Values*, presents the tags for *symbolic*, *numeric*, *measurement*, *rate*, and *string* values. Section 16.5 *Structured Values*, shows how to use feature-structures themselves as values, thus enabling feature structures to be recursively defined. Section 16.6 *Singleton, Set, Bag and List Collections of Values* demonstrates the use of multiple values for features, for encoding *set*, *bag*, and *list* collections of values. Section 16.7 *Alternative Features and Feature Values* presents various methods for representing alternations (disjunctions) of features and feature values. Section 16.8 *Boolean, Default and Uncertain Values*, presents tags for *boolean*, *default*, and *uncertain* values, along with methods for *underspecifying* feature values. Section 16.9 *Indirect Specification of Values Using the rel Attribute* shows how to specify various logical relations, such as negation and subsumption, between the expressed values for a feature and its actual values. Finally, section 16.10 *Two Illustrations*, illustrates how feature structures may be linked to to text elements.

This tag set is selected as described in 3.3 *Invocation of the TEI DTD*; in an XML document which uses the markup described in this chapter, the document type declaration should contain the following declaration of the entity TEI.fs, or an equivalent one:

```
<!ENTITY % TEI.fs 'INCLUDE'>
```

The entire document type declaration for a document using this additional tag set together with the base tag set for prose might look like this:

```
<!DOCTYPE TEI.2 PUBLIC "-//TEI P4//DTD Main Document Type//EN" "tei2.dtd" [
    <!ENTITY % TEI.XML       'INCLUDE' >
    <!ENTITY % TEI.prose     'INCLUDE' >
    <!ENTITY % TEI.fs        'INCLUDE' >
 ]>
```

The overall document type declaration for this additional tag set has the following structure:

```
<!-- 16.1: Feature Structures-->
<!--Text Encoding Initiative Consortium:
Guidelines for Electronic Text Encoding and Interchange.
Document TEI P4, 2002.
Copyright (c) 2002 TEI Consortium. Permission to copy in any form
is granted, provided this notice is included in all copies.
These materials may not be altered; modifications to these DTDs should
be performed only as specified by the Guidelines, for example in the
chapter entitled 'Modifying the TEI DTD'
These materials are subject to revision by the TEI Consortium. Current versions
are available from the Consortium website at http://www.tei-c.org-->
<!--declarations from 16.2: Feature structures, binary values inserted here -->
<!--declarations from 16.3: Feature libraries inserted here -->
<!--declarations from 16.4: Symbolic, etc. values inserted here -->
```

```
<!--declarations from 16.6: Null values inserted here -->
<!--declarations from 16.7: Alternative features and feature values inserted here -->
<!--declarations from 16.8: Boolean, default, uncertainty values inserted here -->
<!-- end of 16.1-->
```

16.2 Elementary Feature Structures: Features with Binary Values

The fundamental elements of a feature structure system are <f> (for *feature*) and <fs> (for *feature structure*). The <fs> element has a type attribute for indicating what type of feature structure it represents, and may contain any number of <f> elements. An <f> element, in turn, has a required name attribute and any number of associated *values*. These may be binary, numeric, symbolic (i.e. taken from a restricted set of legal values), or string-valued, or may consist of sets, lists, or bags of binary, numeric, symbolic, or string values. Specialized values may also be given which allow partial underspecification of the feature. These possible types are all described in more detail in this and the following sections.

This section considers the special case of feature structures that contain features whose single value is one of the *binary values* represented by the empty elements <plus> and <minus>. The elements which are used for representing feature structures, features and the binary values, along with their descriptions and attributes, are the following.

<fs> analyzes a collection of features and feature alternations as a structural unit. Attributes include:

- **type** provides a type for a feature structure.
  Values Character string, e.g. *word structure*.
- **feats** pointer to features.
  Values list of individual IDs for features.
- **rel** indicates the relation of the given content to the actual content or value of the feature structure.
  Legal values are:
  - eq indicates that the actual content is that given.
  - ne indicates that the actual content is not that given.
  - sb indicates that the actual content is subsumed by the given content.
  - ns indicates that the actual content is not subsumed by the given content.

<f> associates a name with a value of any of several different types. Attributes include:

- **name** provides a name for a feature.
  Values A name token.
- **org** indicates organization of given value or values as singleton, set, bag or list.
  Legal values are:
  - single indicates that the given value is a singleton.
  - set indicates that the given values are organized as a set.
  - bag indicates that the given values are organized as a bag (multiset).
  - list indicates that the given values are organized as a list.
- **fVal** points to the id attributes of feature values.
  Values one or more valid identifiers, separated by white space.
- **rel** indicates the relation between the values that are given as the content of the feature or pointed at by the fVal attribute and the actual values of the feature.
  Legal values are:
  - eq indicates that the given values are the actual values.
  - ne indicates that the given values are not the actual values.
  - sb indicates that the given values are a subset, subbag or sublist of the actual values.
  - ns indicates that the given values are not a subset, subbag or sublist of the actual values.

<plus> provides binary plus value for a feature.

<minus> provides binary minus value for a feature.

The attributes not discussed in this section are discussed in following sections as follows: the feats and the fVal attributes in section 16.3 *Feature, Feature-Structure and Feature-Value Libraries*, the rel attribute in section 16.9 *Indirect Specification of Values Using the rel Attribute*, and the org attribute in section 16.6 *Singleton, Set, Bag and List Collections of Values*.

An <fs> element containing <f> elements with binary values can be straightforwardly used to encode the *matrices* of feature-value specifications for phonetic segments, such as the following for the English segment [s].

```
+---          ---+
| + consonantal  |
| - vocalic      |
| - voiced       |
| + anterior     |
| + coronal      |
| + continuant   |
| + strident     |
+---          ---+
```

Using the additional tag set for feature structures, this might be encoded as follows. Note that <fs> elements may have a type attribute indicating the kind of feature structure in question.

```
<fs type="phonological segment">
   <f name="consonantal"> <plus/>  </f>
   <f name="vocalic">     <minus/> </f>
   <f name="voiced">      <minus/> </f>
   <f name="anterior">    <plus/>  </f>
   <f name="coronal">     <plus/>  </f>
   <f name="continuant">  <plus/>  </f>
   <f name="strident">    <plus/>  </f>
</fs>
```

The restriction of specific features to specific types of values (e.g. the restriction of the feature 'strident' to the values <plus/> or <minus/>) cannot be validated by a generic SGML or XML parser (though other validation mechanisms such as XML Schemas do provide such capabilities). To enable an application program to check that only legal values for particular features appear, one may write a *feature-system declaration*; see chapter 26 *Feature System Declaration*.

Here are the formal declarations of the <fs>, <f>, <plus> and <minus> elements.

```
<!-- 16.2: Feature structures, binary values-->
<!ELEMENT fs %om.RR; ((f | fAlt | alt)*)>
<!ATTLIST fs
      %a.global;
      type CDATA #IMPLIED
      feats IDREFS #IMPLIED
      rel (eq|ne|sb|ns) "sb"
      TEIform CDATA 'fs'  >
<!ELEMENT f %om.RO; ( null | ( plus | minus | any | none | dft | uncertain |
                              sym | nbr | msr | rate | str | vAlt | alt | fs )* ) >
<!ATTLIST f
      %a.global;
      name NMTOKEN #REQUIRED
      org (single|set|bag|list) #IMPLIED
      rel (eq|ne|sb|ns) "eq"
      fVal IDREFS #IMPLIED
      TEIform CDATA 'f'  >
<!ELEMENT plus %om.RO; EMPTY>
<!ATTLIST plus
      %a.global;
      TEIform CDATA 'plus'  >
<!ELEMENT minus %om.RO; EMPTY>
<!ATTLIST minus
      %a.global;
      TEIform CDATA 'minus'  >
<!-- end of 16.2-->
```

16.3 Feature, Feature-Structure and Feature-Value Libraries

As the example in the preceding section illustrates, the direct encoding of features structures can be verbose. Consequently, the effort of encoding large numbers of feature structures in this manner could be enormous, and could result in the creation of enormous files. To reduce the size and complexity of the task of encoding feature structures, one may use the feats attribute of the <fs> element to point to one or more of the features of that element. This indirect method of encoding feature structures presumes that the <f> elements are assigned unique id values, and are collected together in <fLib> elements (*feature libraries*). In turn, feature structures can be collected together in <fsLib> elements (*feature-structure libraries*). Finally, one may use the fVal attribute of the <f> element to point to its values. This indirect method of encoding feature values presumes that the value elements are assigned id specifications, and are collected together in <fvLib> elements (*feature-value libraries*). The elements which are used for representing feature, feature-structure and feature-value libraries, along with their descriptions and attributes, are the following.

<fLib> assembles library of feature elements. Attributes include:

type indicates type of feature library (i.e., what kind of features it contains).
Values Character string, e.g. *word features*.

<fsLib> assembles library of feature structure elements. Attributes include:

type indicates type of feature-structure library (i.e., what type of feature structures it contains).
Values Character string, e.g. *word structure library*.

<fvLib> assembles library of feature value elements. Attributes include:

type indicates type of feature-value library (i.e., what type of feature values it contains).
Values Character string, e.g. *symbolic values*.

For example, suppose a feature library for phonological feature specifications is set up as follows.

```
<fLib type="phonological features">
   <f id="CNS1" name="consonantal"> <plus/> </f>
   <f id="CNS0" name="consonantal"> <minus/> </f>
   <f id="VOC1" name="vocalic">     <plus/> </f>
   <f id="VOC0" name="vocalic">     <minus/> </f>
   <f id="VOI1" name="voiced">      <plus/> </f>
   <f id="VOI0" name="voiced">      <minus/> </f>
   <f id="ANT1" name="anterior">    <plus/> </f>
   <f id="ANT0" name="anterior">    <minus/> </f>
   <f id="COR1" name="coronal">     <plus/> </f>
   <f id="COR0" name="coronal">     <minus/> </f>
   <f id="CNT1" name="continuant">  <plus/> </f>
   <f id="CNT0" name="continuant">  <minus/> </f>
   <f id="STR1" name="strident">    <plus/> </f>
   <f id="STR0" name="strident">    <minus/> </f>
   <!-- ... -->
</fLib>
```

Then the feature structures that represent the analysis of the phonological segments (phonemes) /t/, /d/, /s/, and /z/ can be defined as follows.

```
<fs feats="CNS1 VOC0 VOI0 ANT1 COR1 CNT0 STR0"/>
<fs feats="CNS1 VOC0 VOI1 ANT1 COR1 CNT0 STR0"/>
<fs feats="CNS1 VOC0 VOI0 ANT1 COR1 CNT1 STR1"/>
<fs feats="CNS1 VOC0 VOI1 ANT1 COR1 CNT1 STR1"/>
```

The preceding are but four of the 128 logically possible fully specified phonological segments using the seven binary features listed in the feature library. Presumably not all combinations of features correspond to phonological segments (there are no strident vowels, for example). The legal combinations, however, can be collected together in a *feature-structure library*, with each element being given a unique id attribute, as in the following example.

```
<fsLib id="fsl1" type="phonological segment definitions">
   <!-- ... -->
   <fs id="T.DF" feats="CNS1 VOC0 VOI0 ANT1 COR1 CNT0 STR0"/>
   <fs id="D.DF" feats="CNS1 VOC0 VOI1 ANT1 COR1 CNT0 STR0"/>
```

```
   <fs id="S.DF" feats="CNS1 VOC0 VOI0 ANT1 COR1 CNT1 STR1"/>
   <fs id="Z.DF" feats="CNS1 VOC0 VOI1 ANT1 COR1 CNT1 STR1"/>
   <!-- ... -->
</fsLib>
```

Text elements can be linked to these feature structures in any of the ways described in section 15.2 *Global Attributes for Simple Analyses.* In the following example, a <linkGrp> element is used to link selected characters in the text 'Caesar seized control' to their phonological representations.

```
<text id='TXT1'>
  <!-- ... -->
  <body>
 <!-- ... -->
 <ab id='S1'>
   <w id='S1W1'><c id='S1W1C1'>C</c>ae<c id='S1W1C2'>s</c>ar</w>
   <w id='S1W2'><c id='S1W2C1'>s</c>ei<c id='S1W2C2'>z</c>e<c id='S1W2C3'>d</c></w>
   <w id='S1W3'>con<c id='S1W3C1'>t</c>rol</w>.
 </ab>
 <!-- ... -->
  </body>
  <fsLib id='FSL1' type='phonological segment definitions'>
 <!-- as in previous example -->
  </fsLib>
  <linkGrp type='phonological identification of characters'
           domains='FSL1 TXT1'
           targFunc='phonological.segment character' >
 <!-- ... -->
 <link id='LT' targets='S.DF S1W3C1'/>
 <link id='LD' targets='Z.DF S1W2C3'/>
 <link id='LS' targets='S.DF S1W2C1'/>
 <link id='LZ' targets='Z.DF S1W2C2'/>
 <!-- ... -->
  </linkGrp></text>
```

Because of the simplicity of the binary feature values, there is no particular gain in pointing at those values rather than specifying them directly. However, the mechanism of using the fVal attribute on <f> elements is useful for representing more complex feature values, and can be illustrated using binary values. Suppose the <plus> and <minus> elements are collected together in a <fvLib>, as follows.

```
<fvLib type="binary values">
   <plus id="B1"/>
   <minus id="B0"/>
</fvLib>
```

Then the feature library presented at the beginning of this section can be represented as follows.

```
<fLib type="phonological features">
   <f id="CNS1" name="consonantal" fVal="B1"/>
   <f id="CNS0" name="consonantal" fVal="B0"/>
   <f id="VOC1" name="vocalic" fVal="B1"/>
   <f id="VOC0" name="vocalic" fVal="B0"/>
   <f id="VOI1" name="voiced" fVal="B1"/>
   <f id="VOI0" name="voiced" fVal="B0"/>
   <f id="ANT1" name="anterior" fVal="B1"/>
   <f id="ANT0" name="anterior" fVal="B0"/>
   <f id="COR1" name="coronal" fVal="B1"/>
   <f id="COR0" name="coronal" fVal="B0"/>
   <f id="CNT1" name="continuant" fVal="B1"/>
   <f id="CNT0" name="continuant" fVal="B0"/>
   <f id="STR1" name="strident" fVal="B1"/>
   <f id="STR0" name="strident" fVal="B0"/>
   <!-- ... -->
</fLib>
```

Although <fs> elements are legitimate feature values (see section 16.5 *Structured Values*), they are not allowed within <fvLib> elements. They should be placed in <fsLib> elements.

Here are the formal declarations of the <fLib>, <fsLib> and <fvLib> elements.

```
<!-- 16.3: Feature libraries-->
<!ELEMENT fLib %om.RR; ((f | fAlt)*)>
<!ATTLIST fLib
      %a.global;
      type CDATA #IMPLIED
      TEIform CDATA 'fLib'  >
<!ELEMENT fsLib %om.RR; ((fs | vAlt)*)>
<!ATTLIST fsLib
      %a.global;
      type CDATA #IMPLIED
      TEIform CDATA 'fsLib'  >
<!ELEMENT fvLib %om.RR; ((plus | minus | any | none | dft | uncertain | null
                          | sym | nbr | msr | rate | str | vAlt)*)>
<!ATTLIST fvLib
      %a.global;
      type CDATA #IMPLIED
      TEIform CDATA 'fvLib'  >
<!-- end of 16.3-->
```

16.4 Symbolic, Numeric, Measurement, Rate and String Values

In section 16.2 *Elementary Feature Structures: Features with Binary Values*, we defined the two empty elements <plus> and <minus> which are used to represent binary values. In this section, we define five more feature-value elements: the empty elements <sym> for expressing *symbolic values*, <nbr> for expressing *numeric values*, <msr> for expressing *measurement values*, and <rate> for expressing *rate values*; and the element <str> for expressing *string values*. These elements, along with their descriptions and attributes, are the following.

<sym> provides symbolic values for features. Attributes include:

- **value** provides a symbolic value for a feature, one of a finite list that may be specified in a feature declaration.
  Values A string, e.g. *feminine*.
- **rel** indicates the relation of the given value to the actual value.
  Legal values are:
  - eq indicates that the actual value is that given.
  - ne indicates that the actual value is not that given.

<nbr> provides a numeric value or range of values for a feature. Attributes include:

- **value** provides a numeric value.
  Values A real number or integer.
- **valueTo** together with value attribute, provides a range of numeric values.
  Values A real number or integer.
- **type** indicates whether value or range is to be understood as real or integer.
  Legal values are:
  - int specifies that value is an integer; if noninteger is given as value of value, then only integer part is used.
  - real specifies that value is a real number.
- **rel** indicates the relation of the given value or range to the actual value or range.
  Legal values are:
  - eq indicates that the actual value or range is that given.
  - ne indicates that the actual value or range is not the value or range given.
  - lt indicates that the actual value or range is less than the given value or range.
  - le indicates that the actual value or range is less than or equal to the given value or range.
  - gt indicates that the actual value or range is greater than the given value or range.
  - ge indicates that the actual value or range is greater than or equal to the given value or range.

<msr> provides a measure value or range of values for a feature. Attributes include:

unit provides a unit for a measure feature, one of a finite list that may be specified in a feature declaration.
Values A string, e.g. *meter*.

value provides a numeric value.
Values A real number or integer.

valueTo together with value attribute, provides a range of numeric values.
Values A real number or integer.

type indicates whether value or range is to be understood as real or integer.
Legal values are:

`int` specifies that value is an integer; if noninteger is given as value of value, then only integer part is used.

`real` specifies that value is a real number.

rel indicates the relation of the given value or range to the actual value or range.
Legal values are:

`eq` indicates that the actual value or range is that given.

`ne` indicates that the actual value or range is not the value or range given.

`lt` indicates that the actual value or range is less than the given value or range.

`le` indicates that the actual value or range is less than or equal to the given value or range.

`gt` indicates that the actual value or range is greater than the given value or range.

`ge` indicates that the actual value or range is greater than or equal to the given value or range.

<rate> provides a rate value or range of values for a feature. Attributes include:

unit provides a unit for a rate feature, one of a finite list that may be specified in a feature declaration.
Values A string, e.g. *meter*.

per provides an interval for a rate feature, one of a finite list that may be specified in a feature declaration.
Values A string, e.g. *second*.

value provides a numeric value.
Values A real number or integer.

valueTo together with value attribute, provides a numeric range of values.
Values A real number or integer.

type indicates whether value is to be understood as real or integer.
Legal values are:

`int` specifies that value is an integer; if noninteger is given as value of value, then only integer part is used.

`real` specifies that value is that of a real number.

rel indicates the relation of the given value or range to the actual value or range.
Legal values are:

`eq` indicates that the actual value or range is that given.

`ne` indicates that the actual value or range is not the value or range given by the element.

`lt` indicates that the actual value or range is less than the given value or range.

`le` indicates that the actual value or range is less than or equal to the given value or range.

`gt` indicates that the actual value or range is greater than the given value or range.

ge indicates that the actual value or range is greater than or equal to the given value or range.

<str> provides a string value for a feature. Attributes include:

rel indicates the relation of the given value to the actual value.
Legal values are:

eq indicates that the actual value is that given.
ne indicates that the actual value is not that given.
sb indicates that the value given is a substring of the actual value.
ns indicates that the value given is not a substring of the actual value.
lt indicates that the actual value is less than the given value.
le indicates that the actual value is less than or equal to the given value.
gt indicates that the actual value is greater than the given value.
ge indicates that the actual value is greater than or equal to the given value.

The <sym> element is to be used for the value of a feature when that feature can have any of a small, finite set of possible values, representable as character strings. For example, consider the problem of specifying the grammatical *case*, *gender* and *number* features of classical Greek noun forms. Assuming that the case feature can take on any of the five values nominative, genitive, dative, accusative and vocative; that the gender feature can take on any of the three values feminine, masculine, and neuter; and that the number feature can take on either of the values singular and plural, then the following may be used to represent the claim that the noun form *θεάι goddesses* has accusative case, feminine gender and plural number.

```
<fs type="word structure">
   <f name="case">   <sym value="accusative"/> </f>
   <f name="gender"> <sym value="feminine"/> </f>
   <f name="number"> <sym value="plural"/> </f>
</fs>
```

Note that instead of using a symbolic value for grammatical number, one could have named the feature *singular* or *plural* and given it an appropriate binary value, as in the following example. Whether one uses a binary or symbolic value in situations like this is largely a matter of taste.

```
<fs type="word structure">
   <f name="case">     <sym value="accusative"/> </f>
   <f name="gender">   <sym value="feminine"/> </f>
   <f name="singular"> <minus/> </f>
</fs>
```

An SGML or XML parser by itself cannot determine that particular values do or do not go with particular features; in particular, it cannot distinguish between the presumably legal encodings in the preceding two examples and the presumably illegal encoding in the following example.

```
<!-- *PRESUMABLY ILLEGAL* ... -->
<fs type="word structure">
   <f name="case">   <sym value="feminine"/> </f>
   <f name="gender"> <sym value="accusative"/> </f>
   <f name="number"> <minus/> </f>
</fs>
```

There are two ways of attempting to ensure that only legal combinations of feature names and values are used. First, if the total number of legal combinations is relatively small, one can simply list all of those combinations in <fLib> elements (together possibly with <fvLib> elements), and point to them using the feats attribute in the enclosing <fs> element. This method is suitable in the situation described above, since it requires specifying a total of only ten (5 + 3 + 2) combinations of features and values. Further, to ensure that the features are themselves combined legally into feature structures, one can put the legal feature structures inside <fsLib> elements. A total of 30 feature structures (5 × 3 × 2) is required to enumerate all the legal combinations of individual case, gender and number values in the preceding illustration. Of course, the legality of the markup requires that the feat attributes actually point at legally defined features, which an SGML or XML parser, by itself, cannot guarantee.

A more general method of attempting to ensure that only legal combinations of feature names and values are used is to provide a feature system declaration which includes a <valRange> element for each feature one uses. Here is a sample <valRange> element for the 'case' feature described above; for further discussion of the <valRange> element, see chapter 26 *Feature System Declaration*; the <vAlt> element is discussed in section 16.7 *Alternative Features and Feature Values*.

```
<!-- VALRANGE specification for CASE feature -->
<valRange>
  <vAlt>
    <sym value='nominative'/>
    <sym value='genitive'/>
    <sym value='dative'/>
    <sym value='accusative'/>
    <sym value='vocative'/>
  </vAlt>
</valRange>
```

Similarly, to ensure that only legal combinations of features are used as the content of feature structures, one should provide <fsConstraint> elements for each of the types of feature structure one employs. For discussion of the <fDecl> and <fsConstraint> elements, see 26 *Feature System Declaration*. Validation of the feature structures used in a document based on the feature-system declaration, however, requires that there be an application program that can use the information contained in the feature-system declaration.

Features with <sym>, <plus>, and <minus> values may be used to encode highly structured information such as may be obtained from precoded survey instruments. We illustrate by means of a coding scheme based on the one that is used for classifying potential printed entries in the British National Corpus. The scheme uses the following features and associated values.

medium books and magazines; miscellaneous; written to be spoken

domain imaginative; applied science; arts; belief and thought; commerce and finance; leisure; natural and pure science; social science; world affairs

level high; medium; low

sampling range beginning; middle; end; whole; whole less ten percent

date of origination 1960–1975; 1975–1993

published (miscellaneous items only) yes; no

selection method (books and periodicals only) chosen on grounds of circulation or influence; chosen at random

A comprehensive feature library for this scheme is the following; the id specifications are those used by the British National Corpus (BNC) project: [127]

```
<fLib type="BNC classification features">
 <f id="ca002" name="medium"><sym value="book.or.periodical"/></f>
 <f id="ca003" name="medium"><sym value="miscellaneous"/></f>
 <f id="ca004" name="medium"><sym value="written.to.be.spoken"/></f>
 <f id="ca005" name="domain"><sym value="imaginative"/></f>
 <f id="ca006" name="domain"><sym value="applied.science"/></f>
 <f id="ca007" name="domain"><sym value="arts"/></f>
 <f id="ca008" name="domain"><sym value="belief.and.thought"/></f>
 <f id="ca009" name="domain"><sym value="commerce.and.finance"/></f>
 <f id="ca00a" name="domain"><sym value="leisure"/></f>
 <f id="ca00b" name="domain"><sym value="natural.and.pure.science"/></f>
 <f id="ca00c" name="domain"><sym value="social.science"/></f>
 <f id="ca00d" name="domain"><sym value="world.affairs"/></f>
 <f id="ca00e" name="level"><sym value="high"/></f>
 <f id="ca00f" name="level"><sym value="medium"/></f>
 <f id="ca00g" name="level"><sym value="low"/></f>
 <f id="ca00h" name="sample.type"><sym value="beginning"/></f>
 <f id="ca00j" name="sample.type"><sym value="middle"/></f>
```

[127] For more information about the British National Corpus, see the website at http://www.hcu.ox.ac.uk/BNC/

```
 <f id="ca00k" name="sample.type"><sym value="end"/></f>
 <f id="ca00l" name="sample.type"><sym value="whole"/></f>
 <f id="ca00m" name="sample.type"><sym value="whole.less.ten.percent"/></f>
 <f id="ca00n" name="published.between"><sym value="1960.1975"/></f>
 <f id="ca00p" name="published.between"><sym value="1975.1993"/></f>
 <f id="ca00r" name="published"><plus/></f>
 <f id="ca00s" name="published"><minus/></f>
 <f id="ca00t" name="selection.method"><sym value="principled"/></f>
 <f id="ca00u" name="selection.method"><sym value="random"/></f>
</fLib>
```

An entry which is a book or periodical on world affairs, medium level, sampled from the middle, published between 1975 and 1993, and selected on a principled basis could then be assigned the following feature-structure code; this code could also be placed in a feature-structure library that contains all the possible fully-specified BNC entry classifications. This library would have a total of 1620 (3 × 9 × 3 × 5 × 2 × 2) entries.

```
<fs id="ca2dfjpt" type="BNC classification for written documents"
    feats="ca002 ca00d ca00f ca00j ca00p ca00t"/>
```

The <nbr> element is to be used when the value of a feature is a number or a range of numbers. For example, suppose one wishes to encode information contained in classified advertisements for the sale or rental of real estate, such as the number of bedrooms and bathrooms in a listed property, and its advertised selling or rental price. One way of representing such information is as follows.

```
<fs type="real estate listing">
   <f name="number.of.bathrooms"><nbr value="2"/></f>
   <f name="number.of.bedrooms"><nbr value="3"/></f>
   <f name="monthly.rent"><nbr value="625.00"/></f>
</fs>
```

The information that the number of bedrooms is in the range from 3 to 5 and the monthly rent is in the range from 625.00 to 950.00 may be represented as follows, using the optional valueTo attribute.

```
<fs type="real estate listing">
   <f name="number.of.bedrooms"><nbr value="3" valueTo="5"/></f>
   <f name="monthly.rent"><nbr value="625.00" valueTo="950.00"/></f>
</fs>
```

The <nbr> (and also the <msr> and <rate> elements defined below) element also may have a type attribute to specify whether the values of the value and valueTo attributes are to be construed as integer or real numbers.

The <msr> element to be used when the value of a feature is a scalar quantity, essentially a combination of a numeric value and a symbolic value for identifying the scale on which the numeric value occurs. For example, real estate listings often provide the area (in square feet or meters) of a house or apartment and the area (in acres or hectares) of land being sold or rented. One way of representing information about such areas is as follows.

```
<fs type="real estate listing">
   <f name="interior.area"><msr value="2000" unit="sq.ft"/></f>
   <f name="property.area"><msr value="0.5" unit="acre"/></f>
</fs>
```

The value of the 'monthly.rent' feature in the two examples above might be more accurately analysed as a measurement rather than as a numeric value, since the amount of the rent in question is to be understood as payable in a specific currency (US or Canadian dollars, pounds sterling, euro, yen...) To make the currency scale explicit, the first example of this feature might be re-encoded as follows.

```
<f name="monthly.rent"><msr value="625.00" unit="USD"/></f>
```

The unit and value attributes of the <msr> element are both required. If the unit attribute is not needed (for example, if no confusion would result if the unit attribute is not specified), then the <nbr> element may be used to express the feature value.

The <rate> element is to be used when the value of a feature is a rate. This element has a required per attribute for expressing the interval over which the rate is measured (typically, but not necessarily, a temporal interval), and an optional unit attribute for expressing the scalar unit. For example, one might encode the wage rate of $8.25 per hour as follows.

```
<f name="wage.rate"><rate value="8.25" unit="USD" per="hour"/></f>
```

Note that the 'monthly.rent' feature illustrated above can be re-encoded as having a rate value, with the per attribute in this case taking the value month, as follows.

```
<f name="rent"><rate value="625.00" unit="USD" per="month"/></f>
```

To encode interest, inflation or tax rates, the unit attribute can be used to indicate that the value attribute is to be understood as a percentage. For example, an interest rate of 8.25% per year can be encoded in either of the following two ways.

```
<f name="interest"><rate value="8.25" unit="percent" per="year"/></f>
<f name="interest"><rate value="0.0825" per="year"/></f>
```

Finally, the <str> element is to be used for the value of a feature when that value is a string drawn from a very large or potentially unbounded set of possible strings of characters, so that it would be impractical or impossible to use the <sym> element. These values are expressed not as the values of the value attribute, as in the case of symbolic, numeric, measurement and rate values, but as the content of the <str> element. For example, one may encode the street address of a property in a real estate listing, as follows.

```
<fs type="real estate listing">
   <f name="address"><str>3418 East Third Street</str></f>
</fs>
```

Here are the formal declarations of the <sym>, <nbr>, <msr>, <rate>, and <str> elements.

```
<!-- 16.4: Symbolic, etc. values-->
<!ELEMENT sym %om.RO; EMPTY>
<!ATTLIST sym
      %a.global;
      value CDATA #REQUIRED
      rel (eq|ne) "eq"
      TEIform CDATA 'sym'  >
<!ELEMENT nbr %om.RO; EMPTY>
<!ATTLIST nbr
      %a.global;
      value CDATA #REQUIRED
      valueTo CDATA #IMPLIED
      rel (eq|ne|lt|le|gt|ge) "eq"
      type (int|real) #IMPLIED
      TEIform CDATA 'nbr'  >
<!ELEMENT msr %om.RO; EMPTY>
<!ATTLIST msr
      %a.global;
      value CDATA #REQUIRED
      valueTo CDATA #IMPLIED
      unit CDATA #REQUIRED
      rel (eq|ne|lt|le|gt|ge) "eq"
      type (int|real) #IMPLIED
      TEIform CDATA 'msr'  >
<!ELEMENT rate %om.RO; EMPTY>
<!ATTLIST rate
      %a.global;
      value CDATA #REQUIRED
      valueTo CDATA #IMPLIED
      unit CDATA #IMPLIED
      per CDATA #REQUIRED
      rel (eq|ne|gt|ge|lt|le) "eq"
      type (int|real) #IMPLIED
      TEIform CDATA 'rate'  >
<!ELEMENT str %om.RR; (#PCDATA)>
<!ATTLIST str
      %a.global;
      rel (eq|ne|sb|ns|lt|le|gt|ge) "eq"
      TEIform CDATA 'str'  >
<!-- end of 16.4-->
```

16.5 Structured Values

Features may have *structured values* as well; these values are represented by either the <fs> element, or the fVal attribute on the <f> element, which can point to an <fs> element. Since an <fs> or a pointer to an <fs> is permitted to occur as a value of an <f>, recursion is possible. For example, an <fs> element may contain or point to an <f> element, which may contain or point to an <fs> element, which may contain or point to an <f> element, and so on. To illustrate the use of structured values, consider the following simple model of a personal record, consisting of a person's name, date of birth, place of birth, and sex. Each personal record is a <fs type='personal record'> tag, consisting of the corresponding four features, three of which take structured values, as in the following example.

```
<fs type="personal record">
  <f name="full.name">
    <fs type="name record">
      <f name="first.name"> <str>Kathleen</str> </f>
      <f name="middle.name"> <str>Anne</str> </f>
      <f name="surname"> <str>Barnett</str> </f>
    </fs>
  </f>
  <f name="date.of.birth">
    <fs type="date record">
      <f name="year"> <nbr value="1968"/> </f>
      <f name="month"> <nbr value="4"/> </f>
      <f name="day"> <nbr value="17"/> </f>
    </fs>
  </f>
  <f name="place.of.birth">
    <fs type="place record">
      <f name="city"> <str>Austin</str> </f>
      <f name="state"> <sym value="TX"/> </f>
    </fs>
  </f>
  <f name="sex"> <sym value="female"/> </f>
</fs>
```

Now suppose that feature-structure libraries are maintained for name records and place records. Further suppose that the feature structure representing the name record in the previous example has an id attribute with the value nkab027", while the feature structure representing the place record has an id attribute whose value is txaustin.[128] Then the preceding example could also be encoded as follows. (An identifier is also provided for the personal record.)

```
<fs id="pkab027" type="personal record">
   <f name="full.name" fVal="nkab027"/>
   <f name="date.of.birth">
      <fs type="date record">
         <f name="year"> <nbr value="1968"/> </f>
         <f name="month"> <nbr value="4"/> </f>
         <f name="day"> <nbr value="17"/> </f>
      </fs>
   </f>
   <f name="place.of.birth" fVal="txaustin"/>
   <f name="sex"> <sym value="female"/> </f>
</fs>
```

This representation could be simplified further if a feature library is maintained for the year, month, day and sex features, so that the feats attribute may be used as follows.

```
<fs id="pkab027" type="personal record" feats="sxf">
   <f name="full.name" fVal="nkab027"/>
   <f name="date.of.birth"><fs type="date record" feats="y1968 m04 d17"/></f>
   <f name="place.of.birth" fVal="txaustin"/>
</fs>
```

Next, suppose that a feature-structure library is also maintained for personal records, and that the library also contains records for the parents of the individual identified in the previous example. Suppose that the

[128] Feature-structure, rather than feature-value, libraries should be used for housing collections of feature structures.

father is identified as pmfb009 and the mother as parn002. Then the personal-record feature structure could be easily augmented to include pointers to the parents, as follows.

```
<fs id="pkab027" type="personal record" feats="sxf">
   <f name="full.name" fVal="nkab027"/>
   <f name="date.of.birth"><fs type="date record" feats="y1968 m04 d17"/></f>
   <f name="place.of.birth" fVal="austintx"/>
   <f name="mother" fVal="parn002"/>
   <f name="father" fVal="pmfb009"/>
</fs>
```

If the personal records identified as parn002 and pmfb009 also contain information about the parents of those individuals, then from the present record, one would have access to that individual's grandparents as well.

Assuming that personal records of the sort described in this section are being maintained in association with text files, the records can be linked to those texts in any of the ways described in chapter 14 *Linking, Segmentation, and Alignment*, provided that identifiers are added for appropriate features, as in the following illustration.

```
<text id="bfile"><body>
    <div id="tkab027" type="birth certificate">
      <p><name id="t1kab027" type="person">Kathleen Anne Barnett</name>
        was born at <time id="t1t0659">6:59 a.m.</time> on
        <date id="t1d680417">April 17, 1968</date> in
        <name id="t1setonhsp" type="org">Seton Hospital</name> in
        <name id="t1txaustin" type="place">Austin</name> to
        <seg id="s1">Mr.</seg> and <seg id="s2">Mrs.</seg>
        <name id="t1mfb009" type="person">Michael F. Barnett</name>
        of <name id="t1sansabatx" type="place">San Saba</name>.
      </p>
      <!-- ... -->
      <join id="t1arn002" targets="s2 t1mfb009"/>
      <join id="t2mfb009" targets="s1 t1mfb009"/>
      <!-- ... -->
    </div></body>
  <fsLib id="prec" type="personal records">
    <fs id="pkab027" type="personal record" feats="sxf">
      <f name="full.name" fVal="nkab027"/>
      <f id="dkab027" name="date.of.birth">
        <fs type="date record" feats="y1968 m04 d17"/>
      </f>
      <f id="bkab027" name="place.of.birth" fVal="txaustin"/>
      <f id="mkab027" name="mother" fVal="parn002"/>
      <f id="fkab027" name="father" fVal="pmfb009"/>
    </fs></fsLib>
  <linkGrp type="record verification" domains="bfile prec" targFunc="source goal">
    <link targets="t1kab027 nkab027"/>
    <link targets="t1d680417 dkab027"/>
    <link targets="t1txaustin bkab027"/>
    <link targets="t1arn002 mkab027"/>
    <link targets="t2mfb009 fkab027"/>
  </linkGrp>
</text>
```

16.6 Singleton, Set, Bag and List Collections of Values

In the discussion to this point, we have assumed that features have exactly one simple value. However, for many purposes, it is useful to be able to consider the values of certain features to be organized in more complex ways, for example as sets, bags (or multisets), or lists. Accordingly, we provide for four different ways in which feature values may be organized, namely as *singletons*, *sets*, *bags* and *lists*. We do so by means of an org attribute on the <f> element, which takes on one of the designated values single, set, bag, and list. A feature whose value is organized as a singleton is understood as having exactly one simple value. If more than one value is specified for it, we assume that only the first one is

considered to be its true value. A feature whose value is organized as a set, bag or list may have any positive number of values as its content. In a set, items are ordered, and may not be repeated. In a bag, items are not ordered, and may repeat. In a list, items are ordered and may repeat. Sets and bags are thus distinguished from lists in that the order in which the values are specified does not matter for the former, but does matter for the latter, while sets are distinguished from bags and lists in that repetitions of values do not count for the former but do count for the latter.[129]

No default value for the org attribute is declared in the DTD; however, a default value for that attribute can be declared for particular features in the feature-system declaration; see chapter 26 *Feature System Declaration*. Note that if only one value is specified for a given <f> element, the set, bag and list values of the org are all essentially equivalent to the singleton value, so the omission of the org attribute for such a feature is not problematic.[130]

To illustrate the use of the org attribute, suppose that the illustration of personal records from the previous section is extended to include pointers to an individual's siblings. Suppose also that the individual identified as <fs id="pkab027"> has siblings identified as <fs id="panb005">, <fs id="pmfb010"> and <fs id="pzrb001"> in the personal records library. Then we may extend the personal record for <fs id="pkab027"> as follows.

```
<fs id="pkab027" type="personal record" feats="sxf">
   <f name="full.name" fVal="nkab027"/>
   <f name="date.of.birth">
      <fs type="date record" feats="y1988 m04 d17"/>
   </f>
   <f name="place.of.birth" fVal="austintx"/>
   <f name="mother" fVal="parn002"/>
   <f name="father" fVal="pmfb009"/>
   <f name="siblings" org="set" fVal="panb005 pmfb010 pzrb001"/>
</fs>
```

A more elaborate illustration of the use of the org attribute is the the following <f name="career" org="list"> element which may be added to the personal records of an individual to record the job career of that individual. The feature structures which constitute the value of this feature document the jobs which the individual has held in the order in which they were held. Note that a list has been embedded within a list by means of intervening <fs type="employment record"> and <f name="promotion.history"> elements.

```
<f name="career" org="list">
  <fs type="employment record">
    <f name="employer"><str>Safeway Stores</str></f>
    <f name="hiring.information">
      <fs type="hire structure">
        <f name="hire.date"><fs type="date structure" feats="y1988 m06"/></f>
        <f name="job.title"><sym value="stocker"/></f>
        <f name="wage"><rate value="6.00" per="hour"/></f>
        <f name="hours.worked"><rate value="40" per="week"/></f>
        <f name="status.code" fVal="sc4a"/>
      </fs>
    </f>
    <f name="promotion.history" org="list">
      <fs type="promotion record">
        <f name="date"><fs type="date structure" feats="y1988 m12"/></f>
        <f name="job.title"><sym value="cashier"/></f>
        <f name="wage"><rate value="7.00" per="hour"/></f>
        <f name="hours.worked"><rate value="40" per="week"/></f>
        <f name="status.code" fVal="sc4a"/>
      </fs>
```

[129] An SGML or XML DTD cannot however straightforwardly validate that values for features organized as sets are not repeated; such validation would have to be carried out by an application program. Our method of representing set, bag and list values also does not permit such values to be directly embedded within one another. In order to embed a set within a set, for example, one must specify the embedded set as the value of a feature of a feature-structure value of the including set. Fortunately, this is not as hard as it sounds: the embedding of a list within a list is illustrated in the second example below.

[130] Unless the value is the <null> element; see below.

```
      <fs type="promotion record">
        <f name="date"><fs type="date structure" feats="y1990 m02"/></f>
        <f name="job.title"><sym value="supervisor"/></f>
        <f name="salary"><rate value="18000" per="year"/></f>
        <f name="status.code" fVal="sc3c"/>
      </fs>
    </f>
    <f name="termination.information">
      <fs type="termination structure">
        <f name="termination.date"><fs type="date structure" feats="y1991 m04"/></f>
        <f name="status.code" fVal="sc3c"/>
        <f name="reason.for.termination"><sym value="laid.off"/></f>
      </fs>
    </f>
  </fs>
  <fs type="employment record">
    <!-- ... -->
  </fs>
  <!-- ... -->
</f>
```

The information contained in such features may be linked to textual references in the usual way. The <f name="status.code"> feature has been included to show how evaluative or interpretive information can be included along with information gleaned from textual records. The example presumes that the status code values are maintained in a designated <fvLib>.

Features with values organized as sets, bags or lists can sometimes be used instead of features organized as singletons, whose values are individual feature structures. For example, consider the following encoding of the English verb form ‘sinks’, which contains an ‘agreement’ feature whose value is a feature structure which contains ‘person’ and ‘number’ features with symbolic values.

```
<fs type="word structure">
  <!-- ... -->
  <f name="word.class"> <sym value="verb"/> </f>
  <f name="tense"> <sym value="present"/> </f>
  <f name="agreement">
    <fs type="agreement structure">
      <f name="person"> <sym value="third"/> </f>
      <f name="number"> <sym value="singular"/> </f>
    </fs>
  </f>
  <!-- ... -->
</fs>
```

If one does not care about the names of the features contained within the ‘agreement’ feature structure, the containing <f name="agreement"> element can be given an org attribute with the value set, and the contained <fs> element, together with the person and number feature elements it contained, can be eliminated, as follows.

```
<fs type="word structure">
  <!-- ... -->
  <f name="word.class"> <sym value="verb"/> </f>
  <f name="tense"> <sym value="present"/> </f>
  <f name="agreement" org="set"><sym value="third"/><sym value="singular"/></f>
  <!-- ... -->
</fs>
```

The encoding in the preceding example presumes that the <fDecl> element for the ‘agreement’ feature would look something like the following; for further details, see chapter 26 *Feature System Declaration*.

```
<fDecl name="agreement" org='set'>
  <!-- ... -->
  <vRange>
    <vAlt>
      <sym value='first'/>
      <sym value='second'/>
```

```
      <sym value='third'/>
      </vAlt>
     <vAlt>
      <sym value='singular'/>
      <sym value='plural'/>
      </vAlt>
     </vRange>
    <!-- ... -->
    </fDecl>
```

The set, bag or list which has no members is known as the null (or empty) set, bag or list. To refer to it, the <null> element is provided; its description and attributes are as follows.

<null> represents the null set, bag, or list, depending on whether the org attribute of its parent <f> has the value set, bag, or list; has no interpretation if the org attribute of its parent <f> element has the value single.

So, for example, to indicate that the individual identified above by the <fs id="pkab027"> element has no siblings, we may specify the 'siblings' feature as follows.

```
<f name="siblings" org="set"> <null/> </f>
```

The <null> element when used with a feature organized as a singleton is a semantic error; however, its appearance as a value for such a feature cannot be flagged by SGML or XML parsers. The <null> element, when it appears as a feature value, must be the only value.

Here is the formal declarations of the <null> element.

```
<!-- 16.6: Null values-->
<!ELEMENT null %om.RO; EMPTY>
<!ATTLIST null
      %a.global;
      TEIform CDATA 'null'  >
<!-- end of 16.6-->
```

16.7 Alternative Features and Feature Values

In this section, two methods of representing the alternation (ambiguity or uncertainty) of features and feature values are presented. The first of these methods is to be used for nonsystematic or sporadic markup of alternation of individual features or values; it makes use of the special-purpose <fAlt> and <vAlt> elements. The other is to be used for systematic markup of alternation and for the alternation of groups of features or values; it makes use of the general-purpose <alt> element introduced in section 14.8 *Alternation*. The <fAlt> and <vAlt> elements have the following description and attributes.

<fAlt> provides alternative features for a feature structure or other feature alternation. Attributes include:

mutExcl indicates whether values are mutually exclusive.

Legal values are:

- Y indicates that the values are mutually exclusive.
- N indicates that the values are not mutually exclusive.

<vAlt> provides alternative (disjunctive) values for a feature. Attributes include:

mutExcl indicates whether values are mutually exclusive.

Legal values are:

- Y indicates that the values are mutually exclusive.
- N indicates that the values are not mutually exclusive.

To illustrate the use of the <fAlt> element to represent the alternation of features, suppose one is uncertain whether a particular real estate advertisement describes a house with two bedrooms or with two bathrooms. This uncertainty can be represented as follows.

```
<fs type="real estate listing">
   <fAlt>
      <f name="number.of.bathrooms" > <nbr value="2"/> </f>
      <f name="number.of.bedrooms"> <nbr value="2"/> </f>
```

```
   </fAlt>
</fs>
```

This representation leaves unspecified whether or not the alternation is *mutually exclusive* (i.e. whether having two bathrooms excludes the possibility of having two bedrooms and vice versa). To make this aspect of the alternation explicit, one can specify a value for the mutExcl attribute, as follows.

```
<fs type="real estate listing">
   <fAlt mutExcl="N">
      <f name="number.of.bathrooms" > <nbr value="2" /> </f>
      <f name="number.of.bedrooms" > <nbr value="2" /> </f>
   </fAlt>
</fs>
```

The <fAlt> element can also be used to represent uncertainty about whether the number of bathrooms is two or three, as follows; note that the attribute value mutExcl="Y" can be inferred for the <fAlt> element in this example.

```
<fs type="real estate listing">
   <fAlt>
      <f name="number.of.bathrooms" > <nbr value="2" /> </f>
      <f name="number.of.bathrooms" > <nbr value="3" /> </f>
   </fAlt>
</fs>
```

Since the 'number.of.bathrooms' feature in this example can be factored out of the alternation, a <vAlt> element could be used in place of it to represent the alternation of the feature values more simply, as follows:

```
<fs type="real estate listing">
   <f name="number.of.bathrooms" >
      <vAlt>
         <nbr value="2" />
         <nbr value="3" />
      </vAlt>
   </f>
</fs>
```

The <fAlt> and <vAlt> elements can also be used to indicate certain alternations among values of features organized as sets, bags or lists. For example, suppose one uses a <f name="extras" org="set"> element in feature structures for real estate listings to represent items that are mentioned to enhance a property's sales value, such as whether it has a pool or a good view. Now suppose for a particular listing, the extras include an alarm system and a fenced-in yard, and either a pool or a jacuzzi (but not both). This situation could be represented, using the <vAlt> element, as follows.

```
<fs type="real estate listing">
   <!-- ... -->
   <f name="extras" org="set" >
      <str>alarm system</str>
      <str>fenced-in yard</str>
      <vAlt mutExcl="Y">
         <str>pool</str>
         <str>jacuzzi</str>
      </vAlt>
   </f>
   <!-- ... -->
</fs>
```

Now suppose the situation is like the preceding except that one is also uncertain whether the property has an alarm system or a fenced-in yard, or possibly both. This can be represented as follows.

```
<fs type="real estate listing">
   <!-- ... -->
   <f name="extras" org="set" >
      <vAlt mutExcl="N">
         <str>alarm system</str>
         <str>fenced-in yard</str>
      </vAlt>
```

```
      <vAlt mutExcl="Y">
         <str>pool</str>
         <str>jacuzzi</str>
      </vAlt>
   </f>
   <!-- ... -->
</fs>
```

Finally, suppose that the listing specifies that the property has a finished basement, and that it also has either an alarm system and a pool or a fenced-in yard and a jacuzzi. This situation cannot be represented using the <vAlt> element, because the alternation holds between subsets of two values each. It can, however, be represented using the <fAlt> element, as follows; note that the <str> element with the value finished basement element must be repeated.

```
<fs type="real estate listing">
   <!-- ... -->
   <fAlt mutExcl="Y">
      <f name="extras" org="set" >
         <str>finished basement</str>
         <str>alarm system</str>
         <str>pool</str>
      </f>
      <f name="extras" org="set" >
         <str>finished basement</str>
         <str>fenced-in yard</str>
         <str>jacuzzi</str>
      </f>
   </fAlt>
   <!-- ... -->
</fs>
```

If a large number of ambiguities or uncertainties involving a relatively small number of features and values need to be represented, it is recommended that the general-purpose <alt> element discussed in section 14.8 *Alternation* be used, rather than the special-purpose <fAlt> and <vAlt> elements. The use of the <alt> element avoids the need to explictly represent the alternating elements more than once.

For example, suppose one has set up a <fsLib> element containing feature structures representing the morphological structures of classical Greek inflected words, along with collections of individual features and feature values, encoded by <fLib> and <fvLib> elements as appropriate. The following example shows how one might then represent the morphological structure of a feminine gender, accusative case, plural number noun form in classical Greek, such as '*θεάι*' *goddesses* discussed in section 16.4 *Symbolic, Numeric, Measurement, Rate and String Values*:

```
<fsLib type="noun structures">
   <!-- ... -->
   <!-- plural accusative feminine noun -->
   <fs id="wngfkanp" type="noun structure" feats="wn gf ka np"/>
   <!-- ... --> </fsLib>

<fLib type="morphological features">
   <f id="wn" name="word.class"  fVal="nn"/>
   <!-- ... -->
   <f id="gf" name="gender"  fVal="fe"/>
   <!-- ... -->
   <f id="ka" name="case"  fVal="ac"/>
   <!-- ... -->
   <f id="np" name="number"  fVal="pl"/>
   <!-- ... --> </fLib>

<fvLib type="morphological feature values">
   <!-- ... -->
   <sym id="nn" value="noun" />
   <!-- ... -->
   <sym id="fe" value="feminine" />
   <!-- ... -->
```

```
   <sym id="ac" value="accusative" />
   <!-- ... -->
   <sym id="pl" value="plural" />
   <!-- ... --> </fvLib>
```

Now consider the noun form '*θεάι*' *goddesses*, which is analyzable as a feminine plural noun form in either the nominative or the vocative case. We may represent this ambiguity by adding the following entries to the <fsLib>, <fLib>, and <fvLib> elements in the preceding example; assume that appropriate entries for unambiguous nominative and vocative case forms have already been entered.

```
<!-- Add the following to the feature-structure library  -->
   <!-- plural nominative-or-vocative feminine noun -->
   <fs id="wngfknvnp" type="noun structure" feats="wn gf knv np"/>
<!-- Add the following to the feature library  -->
   <!-- CASE='nominative' or vocative -->
   <f id="knv" name="case"  fVal="novo"/>
<!-- Add the following to the feature value library -->
   <!-- nominative or vocative -->
   <alt id="novo" targets="no vo"/>
```

If the <fvLib> element is not used, and specifications for particular feature values are entered as content of the <f> elements in the <fLib> element, then the ambiguity can be represented as follows.

```
<fsLib type="noun structures">
   <!-- ... -->
   <!-- plural nominative-or-vocative feminine noun -->
   <fs id="wngfknvnp" type="noun structure" feats="wn gf knv np"/>
   <!-- ... -->
</fsLib>
<fLib type="morphological features">
   <!-- ... -->
   <f id="kn" name="case" >
      <sym value="nominative" />
      <!-- ... -->
   </f>
   <f id="kv" name="case" >
      <sym value="vocative" />
      <!-- ... -->
      <alt id="knv" targets="kn kv"/>
      <!-- ... -->
   </f>
</fLib>
```

The <alt> element together with the <join> element can, unlike the <fAlt> and <vAlt> elements, be used to express alternations between sets of features. An example of such an alternation is found in certain feminine gender Greek noun forms ending in -*ας*, such as *πείρας attempt(s)*, which may be analyzed as having either genitive case and singular number features or accusative case and plural number features, as follows (again, assuming the existence of other elements and identifier attributes for simple features and values).

```
<!-- Add the following to the feature structure library -->
   <!-- feminine noun, either genitive singular or accusative plural -->
   <fs id="wngfkg.nska.np" type="noun structure" feats="wn gf kg.nska.np"/>
<!-- Add the following to the feature library  -->
   <join id="kg.ns" targets="kg ns"/><!-- genitive singular -->
   <join id="ka.np" targets="ka np"/><!-- accusative plural -->
   <!-- alternation:  gen. sg. or acc. plural -->
   <alt id="kg.nska.np" targets="kg.ns ka.np"/>
```

Here are the formal declarations of the <fAlt> and <vAlt> elements.

```
<!-- 16.7: Alternative features and feature values-->
<!ELEMENT fAlt %om.RR; ((f | fs | fAlt), (f | fs | fAlt)+)>
<!ATTLIST fAlt
      %a.global;
      mutExcl (Y|N) #IMPLIED
```

```
      TEIform CDATA 'fAlt'  >
<!ELEMENT vAlt %om.RR; ((plus | minus | any | none | dft | uncertain | null |
                         sym | nbr | msr | rate | str | vAlt | fs),
                        (plus | minus | any | none | dft | uncertain | null |
                       sym | nbr | msr | rate | str | vAlt | fs)+) >
<!ATTLIST vAlt
      %a.global;
      mutExcl (Y|N) #IMPLIED
      TEIform CDATA 'vAlt'  >
<!-- end of 16.7-->
```

16.8 Boolean, Default and Uncertain Values

In this section we define four special empty elements used as feature values: the *boolean value* elements <any> and <none>, the <dft> element, and the <uncertain> element.

The boolean value elements are used to indicate whether the features they are associated with have values. The element <any> corresponds to the boolean value `true` (i.e., that the feature it is associated with has a value — not the same as the binary value `plus`), and the element <none> corresponds to the boolean value `false` (i.e., that the feature it is associated with has no value). The <dft> element is used to indicate that the feature it is associated with has its default value in the feature structure in which it appears. Finally, the <uncertain> element may be used to indicate uncertainty about what value, if any, its associated feature has; it is equivalent to the alternation of the <any> and <none> elements. To indicate uncertainty about which of the possible legal values a particular feature has, one should use the <any> element.

The descriptions and attributes of these elements are as follows.
<any> represents boolean *true* value variable.
<none> represents boolean *false* value variable.
<dft> provides default value for a feature.
<uncertain> provides uncertainty value for a feature.

The values <null> and <none> are distinct. The former is to be used with a feature organized as a set, bag, or list to indicate that its value is the null set, bag, or list in a particular feature structure. The latter is to be used with such a feature to indicate that it has no value in a particular feature structure.

The *boolean* values <any> and <none> are also distinct from the *binary* values <plus> and <minus>. The latter pair are specific possible values for features, whereas the former pair represent *ranges* of possible values, not specific possible values, for features. For example, suppose that the <valRange> element for the 'auxiliary' feature is declared as follows in the feature structure declaration, so that either boolean value is legal.

```
<vRange><vAlt><plus/><minus/></vAlt></vRange>
```

Given this <vRange>, then the following pair of specifications is distinct:

```
<f name="auxiliary"><plus/></f>
<f name="auxiliary"><any/></f>
```

In this situation, the <any> element is equivalent to the alternation of the <plus> and <minus> values.

Given the same <vRange>, then the following pair of specifications is also distinct.

```
<f name="auxiliary"><minus/></f>
<f name="auxiliary"><none/></f>
```

The <none> element is equivalent to the negation of the alternation of the <plus> and <minus> elements.

However, if the auxiliary feature is declared to take only the <plus> value, then the specifications below are equivalent:

```
<f name="auxiliary"><plus/></f>
<f name="auxiliary"><any/></f>
```

If the auxiliary feature is declared to take only the <plus> value, then the specifications below are not equivalent; in fact, the specification is invalid.

```
<!-- invalid! -->
<f name="auxiliary"><minus/></f>
<f name="auxiliary"><none/></f>
```

It is even possible to declare that a particular feature can never have values, as follows for the 'impossible' feature:

```
<vRange><null/></vRange>
```

In this case, the following specifications are equivalent.

```
<f name="impossible"><any/></f>
<f name="impossible"><none/></f>
```

The elements <any> and <dft> are also designed to be used in conjunction with the <fDecl> and <valDefault> elements in the feature system declaration discussed in chapter 26 *Feature System Declaration*. First, consider the <any> element, and suppose that the <vRange> element in the <fDecl> element for the 'gender' feature is specified as follows.

```
<vRange>
  <vAlt>
    <sym value='feminine'/>
    <sym value='masculine'/>
    <sym value='neuter'/>
    </vAlt>
  </vRange>
```

Then the following two representations are equivalent.

```
<f name="gender"> <any/> </f>
<f name="gender">
   <vAlt>
      <sym value="feminine"/>
      <sym value="masculine"/>
      <sym value="neuter"/>
   </vAlt>
</f>
```

Second, consider the <dft> element, and suppose that the default value for the 'gender' feature (as specified by the <valDefault> element of its <fDecl> element) is feminine. Then the following three representations are equivalent; note that if an <f> element appears without content and without a valid fVal attribute, then it is equivalent to the same element with the <dft> element as its content.

```
<f name="gender"/>
<f name="gender"> <dft/> </f>
<f name="gender"> <sym value="feminine"/> </f>
```

Using the <any> and <dft> elements, together with an <fDecl> element for the corresponding feature in the feature system declaration, provides a method for *underspecifying* the value of that feature. The <any> element means that the associated feature has a legal value but what value it has is not specified. The <dft> element means that the associated feature has the value which the encoder has declared is the normal value of the feature.

The boolean elements <any> and <none> also have specific uses within <fsConstraints> and <fDecl> elements in feature system declarations, as described in chapter 26 *Feature System Declaration*. For example, the element <any> can appear as the value of a feature contained within an <fs> of a particular type which appears in the <cond> element of an <fsConstraints> element, to indicate that the feature must appear in feature structures of the designated type (i.e., that it is obligatory) and that when it does appear, it may appear with any of its legal values. Similarly, <none> can appear in this way to specify that the feature cannot be present in feature structures of the indicated type (i.e., that it is obligatorily absent from such feature structures). All other features that are declared to have values are understood to be optional in such feature structures.

For example, the following may appear as part of the <fsConstraints> of a feature system declaration to indicate that an 'agreement structure' feature structure must contain a legal 'number' feature, but must not contain a 'category' feature.

```
<cond> <fs type='agreement structure'></fs>
   <then/><fs>
           <f name='number'><any/></f>
           <f name='category'><none/></f>
         </fs>
</cond>
```

Further constraints can be imposed on a feature structure of a particular type in the <vRange> elements of features which take feature structures of that type as values. For example, suppose that verb and adjective agreement in German are represented by feature structures of the following sorts, in which verb forms agree in person and number with their subjects and adjective forms agree in gender, case, and number with their subjects.

```
<fs type="verb structure">
   <!-- ... -->
   <f name="verbAgreement">
      <fs type="agreement structure">
         <f name="person"> <sym value="first"/> </f>
         <f name="number"> <sym value="plural"/> </f>
      </fs>
   </f>
   <!-- ... -->
</fs>
<fs type="adjective structure">
   <!-- ... -->
   <f name="adjagreement">
      <fs type="agreement structure">
         <f name="gender"> <sym value="feminine"/> </f>
         <f name="case"> <sym value="accusative"/> </f>
         <f name="number"> <sym value="plural"/> </f>
      </fs>
   </f>
   <!-- ... -->
</fs>
```

In order to ensure that an ‘agreement structure’ feature structure which appears as the value of a ‘verbAgreement’ feature may be specified for any person and number feature, but for no gender and case feature, we may provide a <vRange> element for the ‘verbAgreement’ feature as follows.

```
<vRange>
  <fs type='agreement structure'>
    <f name='person'><any/></f>
    <f name='case'><none/></f>
    <f name='gender'><none/></f>
    <f name='number'><any/></f>
    </fs>
  </vRange>
```

Similarly, to ensure that an ‘agreement structure’ feature structure which appears as the value of a ‘adjAgreement’ feature may be specified for any case, gender, and number feature, but for no person feature, we may provide a <vRange> element for the ‘adjAgreement’ feature as follows.

```
<vRange>
  <fs type='agreement structure'>
    <f name='person'><none/></f>
    <f name='case'><any/></f>
    <f name='gender'><any/></f>
    <f name='number'><any/></f>
    </fs>
  </vRange>
```

The combination of declarations like these and the principle of *subsumption* discussed in section 16.9 *Indirect Specification of Values Using the rel Attribute*, allows feature structures to be underspecified in text markup. For example, to indicate that a given adjective inflection feature (tagged <f name="adjInflection">) is a feature structure (tagged <fs type="inflection structure">) specifying plural number and any gender and case, we can omit the elements for gender and case on the <fs> element, as follows.

```
<f name="adjinflection">
   <fs type="inflection structure">
      <f name="number"> <sym value="plural"/> </f>
   </fs>
</f>
```

When supplied as the value of a 'verbInflection' feature, the same feature structure would be interpreted as an inflection structure specifying plural number and any person.

If an optional feature is not specified in a feature-structure value, then it is assumed to occur with the <uncertain> value. For further discussion, see section 16.9 *Indirect Specification of Values Using the rel Attribute*.

Here are the formal declarations of the <any>, <none>, <dft>, and <uncertain> elements.

```
<!-- 16.8: Boolean, default, uncertainty values-->
<!ELEMENT any %om.RO; EMPTY>
<!ATTLIST any
      %a.global;
      TEIform CDATA 'any'  >
<!ELEMENT none %om.RO; EMPTY>
<!ATTLIST none
      %a.global;
      TEIform CDATA 'none'  >
<!ELEMENT dft %om.RO; EMPTY>
<!ATTLIST dft
      %a.global;
      TEIform CDATA 'dft'  >
<!ELEMENT uncertain %om.RO; EMPTY>
<!ATTLIST uncertain
      %a.global;
      TEIform CDATA 'uncertain'  >
<!-- end of 16.8-->
```

16.9 Indirect Specification of Values Using the rel Attribute

The rel attribute is provided for the feature value elements <sym>, <nbr>, <msr>, <rate>, <str>, <fs>, and <default> (but not <plus>, <minus>, <null>, <vAlt>, <any>, <none>, and <uncertain>). This attribute may be used for specifying which of various logical relations the given value has to the actual value of the feature. For all value elements for which the rel attribute is defined, except for <fs>, the default value for that attribute is eq, which means that the actual value is equal (or identical) to the given value. Accordingly, the following representations are both interpreted to mean that the value of the 'case' feature is the <sym value="genitive"> element.

```
<f name="case"> <sym value="genitive"/> </f>
```

```
<f name="case"> <sym rel="eq" value="genitive"/> </f>
```

16.9.1 The Not-Equals Relation

The rel attribute can also be specified as having the value ne, which means that the associated feature has a value which is not equal to the given value. For example, the value <nbr rel="ne" value="1"> in the following example denotes any numeric value other than 1 for the feature 'number.of.bathrooms'.

```
<f name="number.of.bathrooms"> <nbr value="1" rel="ne"/> </f>
```

If an <fDecl> element has been provided which defines the legal values for the associated feature, then the value ne can be given a positive interpretation. For example, suppose that the <vRange> element is declared in the <fDecl> element for the 'case' feature as follows.

```
<vRange>
  <vAlt>
    <sym value='nominative'/>
    <sym value='genitive'/>
    <sym value='dative'/>
    <sym value='accusative'/>
    <sym value='vocative'/>
```

```
    </vAlt>
  </vRange>
```

Suppose also that the ‘case’ feature is declared as obligatory in a particular feature structure. Then the following specifications are equivalent in that structure.

```
<f name="case"> <sym value="genitive" rel="ne"/> </f>
```

```
<f name="case">
   <vAlt>
      <sym value="nominative"/>
      <sym value="dative"/>
      <sym value="accusative"/>
      <sym value="vocative"/>
   </vAlt>
</f>
```

That is, when the rel attribute occurs with the value ne in the value of an obligatory feature in a feature structure, the actual value of that feature may be any of its legal values *other than* the specified value.

On the other hand, if the ‘case’ feature is declared as optional in a particular feature structure, then the following specifications are equivalent in that structure.

```
<f name="case"> <sym value="genitive" rel="ne"/> </f>
```

```
<f name="case">
   <vAlt>
      <sym value="nominative"/>
      <sym value="dative"/>
      <sym value="accusative"/>
      <sym value="vocative"/>
      <none/>
   </vAlt>
</f>
```

That is, when the rel attribute has the value ne in the value of an optional feature in a feature structure, the actual value of that feature may be any of its legal values other than the specified value, or <none>.

If the rel attribute is specified with the value ne for a <nbr>, <msr>, or <rate> element for which the valueTo attribute is also specified, then the actual range may be any range distinct from that given. For example, the following means that the number of bathrooms is a range distinct from 3 to 5 (e.g., 3 to 4, 3 to 6, 4 to 5, 4 to 6, 0 to 2, etc.).

```
<f name="number.of.bathrooms"> <nbr value="3" valueTo="5" rel="ne"/> </f>
```

16.9.2 Other Inequality Relations

For the elements <nbr>, <msr>, <rate>, and <str>, the rel attribute may also take on the following values; the use of these values for the <str> element presumes that a particular character and string ordering (or *sorting*) convention is understood.

- **lt** The actual value or range is any legal value or range less than the specified value or range.
- **le** The actual value or range is any legal value or range less than or equal to the specified value or range.
- **gt** The actual value or range is any legal value or range greater than the specified value or range.
- **ge** The actual value or range is any legal value or range greater than or equal to the specified value or range.

These attribute values may be used as shown in the following examples. The first states that the number of bedrooms is less than 5; the second that an illegal speed is any speed greater than 65 miles per hour; the third that a lot size is in a range which is less than or equal to the range of from 5 to 10 acres;[131] the fourth that the last name is any string greater than the empty string (i.e., any nonempty string, given normal string-ordering conventions); and the fifth that for a feature whose value is a list of two strings, the first precedes the string ‘M’ and the second is the string ‘M’, or any string following it.

[131] We say that one range is less than or equal to another if both the value and valueTo attributes of the first are less than or equal to the corresponding attributes of the second.

```
<f name="number.of.bedrooms"> <nbr value="5" rel="lt"/> </f>
<f name="illegal.speed"> <rate value="65" unit="miles" per="hour" rel="gt"/> </f>
<f name="lot.size"> <msr value="5" valueTo="10" unit="acre" rel="le"/> </f>
<f name="last.name"> <str rel="gt"/> </f>
<f name="pairs" org="list"> <str rel="lt">M</str> <str rel="ge">M</str> </f>
```

16.9.3 Subsumption and Non-subsumption Relations

When the rel attribute is given the values sb or ns, the markup expresses the claim that the value given subsumes, or does not subsume, the actual value for the feature in question.

On the <str> element, these values are used to specify that the string value given in the <str> element is or is not a *substring* of the actual value of the feature. The first example below specifies that the actual feature value may be any string at all (since the empty string is a substring of every string), the second that it might be any string in which the string 'the' occurs as a substring, and the third that it might be any string in which the string 'the' does not occur as a substring.

```
<str rel="sb"/>
<str rel="sb">the</str>
<str rel="ns">the</str>
```

On the <fs> element, the attribute values sb and ns indicate that the given feature structure does or does not legally *subsume* the actual feature structure. By definition, one feature structure subsumes another if the second feature structure is identical to the first or contains more information than the first. The default value for the rel attribute of the <fs> element is sb. The subsumption of feature structures is illustrated by the following four examples; suppose that the 'person' and 'number' features are either optional or obligatory in these <fs type="agreement structure"> example elements.

```
<fs id="p3ns" type="agreement structure">
   <f name="person"> <sym value="third"/> </f>
   <f name="number"> <sym value="singular"/> </f>
</fs> <!-- third person singular -->
<fs id="p3nx" type="agreement structure">
   <f name="person"> <sym value="third"/> </f>
</fs> <!-- third person -->
<fs id="pxns" type="agreement structure">
   <f name="number"> <sym value="singular"/> </f>
</fs> <!-- singular -->
<fs id="pxnx" type="agreement structure"/> <!-- -->
```

The fourth example, pxnx, subsumes all four of the examples, since each contains at least as much information as does feature structure pxnx. Conversely, the first example, p3ns, subsumes only itself. Finally, the second and third examples, identified as p3nx and pxns attributes, subsume themselves and the first feature structure, but not each other.

If both person and number are obligatory features of agreement structure elements, then the last three elements in the preceding list have the same interpretation as their counterparts in the following list.

```
<fs id="p3na" type="agreement structure">
   <f name="person"> <sym value="third"/> </f>
   <f name="number"> <any/> </f>
</fs> <!-- third person -->
<fs id="pans" type="agreement structure" >
   <f name="person"> <any/> </f>
   <f name="number"> <sym value="singular"/> </f>
</fs> <!-- singular -->
<fs id="pana" type="agreement structure" >
   <f name="person"> <any/> </f>
   <f name="number"> <any/> </f>
</fs> <!-- -->
```

On the other hand, if both person and number are optional features of agreement structures, then those three elements have the same interpretation as their counterparts in the following list.

```
<fs id="p3nu" type="agreement structure">
   <f name="person"> <sym value="third"/> </f>
   <f name="number"> <uncertain/> </f>
</fs> <!-- 3d person -->
<fs id="puns" type="agreement structure">
   <f name="person"> <uncertain/> </f>
   <f name="number"> <sym value="singular"/> </f>
</fs> <!-- singular -->
<fs id="punu" type="agreement structure">
   <f name="person"> <uncertain/> </f>
   <f name="number"> <uncertain/> </f>
</fs> <!-- -->
```

That is, if an optional feature is omitted from a feature-structure representation, then that feature may have any of its legal values or the value `<uncertain>`.

The value `sb` is chosen as the default value for the `rel` attribute of the `<fs>` element, because it provides for the most economical means for underspecifying them. One situation in which it may be preferable to specify `<fs rel="eq">` is when the feature structure has many optional features and it is known that none of them occurs.

The specification `<fs rel="ns">` is used to denote the feature structures that the specified feature structure does not subsume. This provides a handy way of saying that a certain combination of features is not present, for example the combination of third person and singular number, as in the agreement structure of the English verb form ‘sink’, understood as a present tense verb form. The following example expresses the claim that third-person and singular-number features are not both present in the agreement feature, but makes no further claim about what is present.

```
<f name="agreement">
   <fs id="np3ns" type="agreement structure" rel="ns">
      <f name="person"> <sym value="third"/> </f>
      <f name="number"> <sym value="singular"/> </f>
   </fs>
</f>
```

In most real situations, of course, one can infer, from the range of possible values for person and number, what the remaining possibilities are. Suppose, for example, that in the relevant feature system declaration, the features ‘person’ and ‘number’ are given the following `<vRange>` elements:

```
<vRange><!-- for the PERSON feature  -->
  <vAlt>
    <sym value='first'/>
    <sym value='second'/>
    <sym value='third'/>
    </vAlt>
  </vRange>
<vRange><!-- for the NUMBER feature  -->
  <vAlt>
    <sym value='singular'/>
    <sym value='plural'/>
    </vAlt>
  </vRange>
```

Suppose, further, that the person and number features are obligatory in feature structures of the type `agreement structure`. Then the element `<fs id="NP3NS">` above is equivalent to the following alternation; the features whose value is `<any>` may be omitted, since they are implied by the default value of `sb` for the `rel` attribute in the enclosing `<fs>` elements.

```
<vAlt id="p12na-panp">
   <fs id="p12na" type="agreement structure">
      <f name="person">
         <vAlt> <sym value="first"/> <sym value="second"/> </vAlt>
      </f>
      <f name="number"> <any/> </f>
   </fs>
   <fs id="panp" type="agreement structure">
```

```
      <f name="person"> <any/> </f>
      <f name="number"> <sym value="plural"/> </f>
   </fs>
</vAlt>
```

If, on the other hand, the person and number features were optional in feature structures of type `agreement structure`, then the interpretation of an underspecified feature structure will change. The element `<fs id="NP3NS">` given above is then equivalent to the following alternation; the features whose value is `<uncertain>` may be omitted as they are implied by the default subsumption relation holding between the structure given and the actual structure.

```
<vAlt id="p12Onu-punpO">
   <fs id="p12Onu" type="agreement structure">
      <f name="person">
         <vAlt> <sym value="first"/> <sym value="second"/> <none/> </vAlt>
      </f>
      <f name="number"> <uncertain/> </f>
   </fs>
   <fs id="punpO" type="agreement structure">
      <f name="person"> <uncertain/> </f>
      <f name="number">
         <vAlt> <sym value="plural"/> <none/> </vAlt>
      </f>
   </fs>
</vAlt>
```

16.9.4 Relations Holding with Sets, Bags, and Lists

The rel attribute is also provided for the `<f>` element, but is designed to be used with that element only when its org attribute (see section 16.6 *Singleton, Set, Bag and List Collections of Values*) is `set`, `bag`, or `list`. When associated with the `<f>` element, the rel attribute may take on any of the following four values: `eq`, `ne`, `sb`, and `ns`. The default value is `eq`. Consider first the use of the rel attribute with the `<f>` element when the given value of the feature is `<null>`.

```
<f name="extras" org="set"> <null/> </f>
<f name="extras" org="set" rel="ne"> <null/> </f>
<f name="extras" org="set" rel="sb"> <null/> </f>
<f name="extras" org="set" rel="ns"> <null/> </f>
```

The first example states that the ‘extras’ feature has the null set as its value. The second example states that the ‘extras’ feature is a set which is not equal to the null set. That is, its actual value might be any non-null set. The third example states that the ‘extras’ feature has as its value a set of which the null set is a subset; that is to say, any set at all, including the null set. Note that this is not equivalent to the following, which states that the extras feature has as its value a single element which is any legal value for the ‘extras’ feature, including for example a `<str>` element containing the value `pool`.

```
<f name="extras" org="set"> <any/> </f>
```

Finally, the fourth example states that the ‘extras’ feature has as its value a set of which the null set is not a subset. Since the null set is a subset of every set, the fourth example in effect claims that the ‘extras’ feature has no legal value; it is thus equivalent to the following, which states directly that the ‘extras’ feature has no value.

```
<f name="extras" org="set"> <none/> </f>
```

Consider next the use of the rel attribute with the `<f>` element when the given value of the feature is a single `<str>` element with the content `pool`:

```
<f name="extras" org="set"> <str>pool</str> </f>
<f name="extras" org="set" rel="ne"> <str>pool</str> </f>
<f name="extras" org="set" rel="sb"> <str>pool</str> </f>
<f name="extras" org="set" rel="ns"> <str>pool</str> </f>
```

The first example states that the value of the ‘extras’ feature is a set consisting of a single member, namely a `<str>` element containing the value `pool`. The second example states that the ‘extras’ feature has as

its value a set which is not equal to the set consisting of this particular member. It could, however, be a two-membered set, one of whose members is some other value. This example is thus not equivalent to the following, which states that the 'extras' feature has as its value a set comprising a single member other than a <str> element with the content pool:

```
<f name="extras" org="set"> <str rel="ne">pool</str> </f>
```

The third example states that the 'extras' feature has as its value any set of which the set consisting of the single member specified is a subset (i.e., any set which contains the element <str> with the value pool, and possibly others). Finally, the fourth example states that the 'extras' feature has as its value any set which does not contain this element as a member.

16.9.5 Varieties of Subsumption and Non-subsumption

The rel values sb and ns have different meanings depending on whether they occur within a <str>, <fs> or <f> element. However, the use of a common name for the value reflects a fundamental similarity in those meanings. For example, the value sb can be used in all three elements to indicate that the actual value is any string, any feature structure, or any set, bag or list, as follows. In the second example below, the rel attribute has not been specified, since it has the value sb by default on <fs> elements.

```
<str rel="sb"></str>
<fs></fs>
<f name="..." org="set" rel="sb"> <null/> </f>
<f name="..." org="bag" rel="sb"> <null/> </f>
<f name="..." org="list" rel="sb"> <null/> </f>
```

Because the value sb is not defined for the attribute rel on the <nbr>, <msr> and <rate> elements, the indication that a value may be any number, measure or rate is sometimes not quite as simple. Here is one way of specifying any positive or negative integer numeric value.[132]

```
<vAlt>
   <nbr value="0" rel="gt" type="int"/>
   <nbr value="0" rel="le" type="int"/>
</vAlt>
```

The value ns also is understood in similar ways in the different elements in which it may occur. Above in this section, the equivalence of the following representations under certain conditions was shown (the id attributes and the redundant features with <any/> values have been omitted).

```
<f name="agreement">
   <fs type="agreement structure" rel="ns">
      <f name="person"> <sym value="third"/> </f>
      <f name="number"> <sym value="singular"/> </f>
   </fs>
</f>
<f name="agreement">
   <vAlt>
      <fs type="agreement structure">
         <f name="person">
            <vAlt> <sym value="first"/> <sym value="second"/> </vAlt>
         </f>
      </fs>
      <fs type="agreement structure">
         <f name="number"> <sym value="plural"/> </f>
      </fs>
   </vAlt>
</f>
```

The value ns has an analogous meaning when the value in question is a set rather than a feature structure. Recast in such terms, the equivalence above still holds good:

```
<f name="agreement" org="set" rel="ns">
   <sym value="third"/>
```

[132] Typically, there will be no need to use an encoding like this one as the value of a feature, since the <any> element is available for that purpose. However, in setting up the feature declaration for that feature, it may be necessary to use such an encoding, precisely so as to provide an interpretation for the use of the <any> element as the value of that feature.

```
   <sym value="singular"/>
</f>
<f name="agreement" org="set" rel="sb">
   <vAlt>
      <vAlt> <sym value="first"/> <sym value="second"/> </vAlt>
      <sym value="plural"/>
   </vAlt>
</f>
```

16.10 Two Illustrations

In this section, we present two illustrations of how to associate feature structures and their components with textual elements. Both are taken from the same example text, an article called *Memoirs of a Dog Shrink* that appeared in the popular magazine *Dogs Today* in August 1991, and was also included in the British National Corpus (BNC). The first illustration associates the text with a structure that represents a significant portion of the information contained in the text. The second marks up the grammatical structure of the orthographic words and certain other comparable units in the text. Here is the text, with markup provided down to the level of <s> elements. The n attribute values are taken from the BNC markup; the id attribute values have been added for purposes of these illustrations.

```
<div1 id="dt91mds" type="article">
  <head rend="italic"><s id="mds01" n="00732">Memoirs of a Dog Shrink</s></head>
  <head rend="italic" type="sub">
    <s id="mds04" n="00735">Cartoonist Russell Jones takes a ramble
      through Peter Neville's files</s> </head>
  <list>
    <item><s id="mds05" n="00736">Case number: 72</s></item>
    <item><s id="mds06" n="00737">Name: Jessie</s></item>
    <item><s id="mds07" n="00738">Breed: Collie</s></item>
    <item><s id="mds08" n="00739">Problem: Light bulb phobia</s></item>
  </list>
  <p><s id="mds09" n="00740">Jess the collie was a laid-back sort of
      hound who spent most of his life stretched out on a fireside rug
      in his large Surrey home.</s></p>
  <p><s id="mds10" n="00741">The closest he came to exercise was to
      open one eye every so often, if someone entered the room, or to
      open both eyes, smile, and wag his tail as he'd done on one
      occasion when confronted by a housebreaker!</s></p>
  <p><s id="mds11" n="00742">This extremely lazy lifestyle was one long
      yawn from dawn to dusk.</s>
    <s id="mds12" n="00743">Only the odd bouts of involuntary
      twitching in his sleep reassured his owner that Jess was still
      safe and sound in the land of the living!</s></p>
  <p><s id="mds13" n="00744">One winter night, as the mutt twitched
      away in front of the fire, his mind somewhere between
      Basingstoke and the twilight zone, a 100-watt light bulb in the
      standard lamp above his head suddenly exploded without
      warning!</s></p>
  <p><s id="mds14" n="00745">According to his owner, who witnessed the
      spectacle, Jessie rose gracefully toward the ceiling like a
      space shuttle and, after lingering in mid-air for what seemed an
      eternity, crashed to the floor and fled the house with a speed
      and agility the owner found quite amazing.</s></p>
  <p><s id="mds15" n="00746">Jessie did not return home for several hours.</s>
    <s id="mds16" n="00747">When he eventually did show up, it
      was obvious to all that he was a changed dog!</s>
    <s id="mds17" n="00748">What plodded through the front door was
      not the lovable, lazy hound who had once lived there but a
      grim-faced light bulb serial killer!</s></p>
  <p><s id="mds18" n="00749">Within seconds of his return, Jessie
      launched a vicious attack on a table lamp, popping the bulb and
      wrecking the shade before charging into the lounge.</s>
    <s id="mds19" n="00750">There, in a frenzy of violence, he reduced
      the standard lamp to a table lamp in 10 seconds flat!</s></p>
```

```
  <p><s id="mds20" n="00751">After a room-to-room chase lasting
      several minutes, during which every lamp in the house was turned
      to sawdust, the dog was finally caught and wrestled to the
      ground.</s></p>
  <p><s id="mds21" n="00752">With his house plunged into darkness,
      Jessie's owner sought my help.</s></p>
  <div2 type="part">
    <head> <s id="mds22" n="00753">SIMPLE SOLUTION</s> </head>
    <p><s id="mds23" n="00754">When I first saw the dog, it was quite
        obvious he'd been deeply affected by the explosion and had
        developed a 100-watt phobia for light bulbs!</s></p>
    <p><s id="mds24" n="00755">By placing his feeding bowl closer each
        day to a table lamp the dog gradually learned to live with his
        enemy.</s>
      <s id="mds25" n="00756">Within a couple of weeks, his killer
        instincts had disappeared and he was back where he belonged
        &mdash; twitching away peacefully on the fireside rug.</s></p>
  </div2>
</div1>
```

The first illustration is based on the observation that from the example text, it is possible to infer a fairly extensive medical history for the dog described in it. Suppose that we have a definition of a feature structure that represents a canine medical history. Then we can fill in feature values in that history from the text, and prepare a `<linkGrp>` element that specifies the links between the text segments and the various features specified in the feature structure. Here is a hypothetical example of such a filled-in feature structure and associated link group.

```
<fs id="j37" type="canine medical history">
  <f id="j37pn" name="name"><str>Jessie</str></f>
  <f id="j37pc" name="called.by" org="set">
    <str>Jessie</str>
    <str>Jess</str>
  </f>
  <f id="j37b" name="breed"><sym value="collie"/></f>
  <f id="j37o" name="owner">
    <fs type="owner description">
      <f name="name"><uncertain/></f>
      <f id="j37or" name="address"><str>Surrey</str></f>
    </fs>
  </f>
  <f id="j37i" name="illness" org="list">
    <fs id="j37i1" type="case history">
      <f id="j37i1sn" name="name.of.specialist">
        <fs type="name structure">
          <f name="last.name"><str>Neville</str></f>
          <f name="first.name"><str>Peter</str></f>
        </fs>
      </f>
      <f name="title.of.specialist"><uncertain/></f>
      <f id="j37i1n" name="case.number"><nbr value="72"/></f>
      <f name="age.at.incidence"><uncertain/></f>
      <f name="date.of.incidence"><uncertain/></f>
      <f id="j37i1b" name="baseline.condition" org="set">
        <sym value="lazy"/>
        <sym value="friendly"/>
        <sym value="indoor"/>
      </f>
      <f id="j37i1s" name="symptoms">
        <fs type="symptom structure">
          <f id="j37i1sb" name="behaviors" org="set">
            <sym value="agitated"/>
            <sym value="destructive"/>
            <sym value="unfriendly"/>
          </f>
          <f id="j37i1sp" name="particulars">
            <str>ran off, then returned and
```

```
            destroyed every lamp in the house</str>
          </f>
        </fs>
      </f>
      <f id="j37i1d" name="diagnosis">
        <fs type="diagnosis structure">
          <f name="date.of.diagnosis"><uncertain/></f>
          <f id="j37i1dd" name="disease"><str>light bulb phobia</str></f>
          <f id="j37i1dc" name="presumed.cause">
            <str>explosion of light bulb over
              patient's head</str>
          </f>
        </fs>
      </f>
      <f id="j37i1t" name="treatment">
        <fs type="treatment history">
          <f name="medicine"><none/></f>
          <f id="j37i1tr" name="regime"><str>positive reinforcement</str></f>
          <f id="j37i1tp" name="particulars">
            <str>systematically decreased
              distance between feeding bowl and table lamp</str>
          </f>
          <f id="j37i1td" name="duration.of.treatment"><msr value="2" unit="week"/></f>
        </fs>
      </f>
      <f id="j37i1r" name="result"><str>return to baseline condition</str></f>
    </fs>
  </f>
</fs>
```

```
<linkGrp domains="j37 dt91mds" targFunc="feature segment">
   <link targets="j37pn mds06"/><link targets="j37pc mds06"/>
   <link targets="j37pc mds09"/><link targets="j37pc mds12"/>
   <link targets="j37pc mds14"/><link targets="j37pc mds18"/>
   <link targets="j37pc mds21"/><link targets="j37b mds07"/>
   <link targets="j37b mds09"/><link targets="j37or mds09"/>
   <link targets="j37i1sn mds04"/><link targets="j37i1n mds05"/>
   <link targets="j37i1dc mds23"/><link targets="j37i1tr mds24"/>
   <link targets="j37i1td mds25"/><link targets="j37i1r mds25"/>
</linkGrp>
```

From this illustration, we see that links can be made not only between text and feature structure elements, but also between text and feature elements. For that matter, links between text and feature value elements can also be made.

The second illustration takes advantage of the fact that this text, like others that appear in the BNC, has been provided with detailed grammatical markup of most of its orthographic words and certain other comparable structural units. For example, in an early form of the BNC markup, the second paragraph of the above text was marked up as follows.

```
<s n="00741">The&AT0; closest&AJS; he&PNP; came&VVD; to&PRP;
exercise&NN1; was&VBD; to&TO0; open&VVI; one&CRD; eye&NN1; every
so often&AV0;,&PUN; if&CJS; someone&PNI; entered&VVD; the&AT0;
room&NN1;,&PUN; or&CJC; to&TO0; open&VVI; both&DT0;
eyes&NN2;,&PUN; smile&VVI;,&PUN; and&CJC; wag&VVI; his&DPS;
tail&NN1; as&CJS; he&PNP;'d&VHD; done&VDN; on&PRP; one&CRD;
occasion&NN1; when&AVQ; confronted&VVN; by&PRP; a&AT0;
housebreaker&NN1;!&PUN;</s>
```

The entities that appear in this fragment may be expanded into pointers to feature structures that represent grammatical structure by means of entity definitions as follows.

```
<!-- ... -->
<!ENTITY   AJS    "<ptr target='AJS'/>"                          >
<!-- ... -->
<!ENTITY   AT0    "<ptr target='AT0'/>"                          >
<!-- ... -->
```

This method of associating feature structures with textual elements has a number of drawbacks, most important of which is the fact that the association is implicit, relying on the relative position of pointer and associated text, rather than being explicit. A better method therefore (which was subsequently adopted by the BNC project) is to segment the text into the units under analysis, and point to the feature structures from within the unit tags, by means of the ana attribute (see sections 15.2 *Global Attributes for Simple Analyses* and 15.4 *Linguistic Annotation*).

```
<s n="00741">
   <w ana="at0">The</w>
   <w ana="ajs">closest</w>
   <w ana="pnp">he</w>
   <w ana="vvd">came</w>
   <w ana="prp">to</w>
   <w ana="nn1">exercise</w>
   <w ana="vbd">was</w>
   <w ana="to0">to</w>
   <w ana="vvi">open</w>
   <w ana="crd">one</w>
   <w ana="nn1">eye</w>
   <phr ana="av0">
      <w>every</w>
      <w>so</w>
      <w>often</w>
   </phr>
   <c ana="pun">,</c>
   <w ana="cjs">if</w>
   <w ana="pni">someone</w>
   <w ana="vvd">entered</w>
   <w ana="at0">the</w>
   <w ana="nn1">room</w>
   <!-- ... -->
</s>
```

To provide pointers in both direction between text and structural analysis, one may supply both the text segments and the feature-structure tags with identifiers, and associate the segments with their analysis by means of a <linkGrp> (see section 14.1 *Pointers*), as follows.

First, we define a feature-structure library to represent all of the grammatical structures that are used in the BNC encoding scheme. (For illustrative purposes, we cite here only the structures needed for the first six words of the sample sentence):

```
<fsLib id="bncgs" type="BNC grammatical structures">
   <!-- ... -->
   <fs id="ajs" type="grammatical structure" feats="wj ds"/>
   <fs id="at0" type="grammatical structure" feats="wl"/>
   <fs id="pnp" type="grammatical structure" feats="wr rp"/>
   <fs id="vvd" type="grammatical structure" feats="wv bv fd"/>
   <fs id="prp" type="grammatical structure" feats="wp bp"/>
   <fs id="nn1" type="grammatical structure" feats="wn tc ns"/>
 <!-- ... -->
</fsLib>
```

It will be noted that each feature structure in this library bears an identifier corresponding with the code supplied as the value for the ana attribute in the sample sentence. The component features of each feature structure are further specified by the feats attribute. These identify one or more <f> elements in the following feature library (again, only a few of the available features are quoted here):

```
<fLib type="BNC grammatical features">
   <!-- ... -->
   <f id="bv" name="verbbase"> <sym value="main"/> </f>
   <f id="bp" name="prepbase"> <sym value="lexical"/> </f>
   <f id="ds" name="degree">   <sym value="superlative"/> </f>
   <f id="fd" name="verbform"> <sym value="ed"/> </f>
   <f id="ns" name="number">   <sym value="singular"/> </f>
   <f id="rp" name="prontype"> <sym value="personal"/> </f>
   <f id="tc" name="nountype"> <sym value="common"/> </f>
```

```
   <f id="wj" name="class">    <sym value="adjective"/> </f>
   <f id="wl" name="class">    <sym value="article"/> </f>
   <f id="wn" name="class">    <sym value="noun"/> </f>
   <f id="wp" name="class">    <sym value="preposition"/> </f>
   <f id="wr" name="class">    <sym value="pronoun"/> </f>
   <f id="wv" name="class">    <sym value="verb"/> </f>
   <!-- ... -->
</fLib>
```

Next, here is a markup of the start of our sample sentence being analyzed, with identifiers for each segment; see section 15.1 *Linguistic Segment Categories* for discussion of the <phr>, <w>, <m> and <c> elements used here.

```
<s id="mds09" n="00741">
   <w id="mds0901">The</w>
   <w id="mds0902">closest</w>
   <w id="mds0903">he</w>
   <w id="mds0904">came</w>
   <w id="mds0905">to</w>
   <w id="mds0906">exercise</w>
   <w id="mds0907">was</w>
   <w id="mds0908">to</w>
   <w id="mds0909">open</w>
   <w id="mds0910">one</w>
   <w id="mds0911">eye</w>
   <phr id="mds0912">
      <w>every</w>
      <w>so</w>
      <w>often</w>
   </phr>
   <c id="mds0913">,</c>
   <w id="mds0914">if</w>
   <w id="mds0915">someone</w>
   <w id="mds0916">entered</w>
   <w id="mds0917">the</w>
   <w id="mds0918">room</w>
   <c id="mds0919">,</c>
   <w id="mds0920">or</w>
   <w id="mds0921">to</w>
   <w id="mds0922">open</w>
   <w id="mds0923">both</w>
   <w id="mds0924">eyes</w>
   <c id="mds0925">,</c>
   <w id="mds0926">smile</w>
   <c id="mds0927">,</c>
   <w id="mds0928">and</w>
   <w id="mds0929">wag</w>
   <w id="mds0930">his</w>
   <w id="mds0931">tail</w>
   <w id="mds0932">as</w>
   <w>
      <w id="mds0933">he</w>
      <m id="mds0934">'d</m>
   </w>
   <w id="mds0935">done</w>
   <w id="mds0936">on</w>
   <w id="mds0937">one</w>
   <w id="mds0938">occasion</w>
   <w id="mds0939">when</w>
   <w id="mds0940">confronted</w>
   <w id="mds0941">by</w>
   <w id="mds0942">a</w>
   <w id="mds0943">housebreaker</w>
   <c id="mds0944">!</c>
</s>
```

Finally, here is a <linkGrp>, which contains all of the <link> elements that associate the text segments in the example sentence with their respective grammatical structures.

```
<linkGrp domains="mds09 bncgs" targFunc="segment analysis">
   <link targets="mds0901 at0"/><link targets="mds0902 ajs"/>
   <link targets="mds0903 pnp"/><link targets="mds0904 vvd"/>
   <link targets="mds0905 prp"/><link targets="mds0906 nn1"/>
   <link targets="mds0907 vbd"/><link targets="mds0908 to0"/>
   <link targets="mds0909 vvi"/><link targets="mds0910 crd"/>
   <link targets="mds0911 nn1"/><link targets="mds0912 av0"/>
   <link targets="mds0913 pun"/><link targets="mds0914 cjs"/>
   <link targets="mds0915 pni"/><link targets="mds0916 vvd"/>
   <link targets="mds0917 at0"/><link targets="mds0918 nn1"/>
   <link targets="mds0919 pun"/><link targets="mds0920 cjc"/>
   <link targets="mds0921 to0"/><link targets="mds0922 vvi"/>
   <link targets="mds0923 dt0"/><link targets="mds0924 nn2"/>
   <link targets="mds0925 pun"/><link targets="mds0926 vvi"/>
   <link targets="mds0927 pun"/><link targets="mds0928 cjc"/>
   <link targets="mds0929 vvi"/><link targets="mds0930 dps"/>
   <link targets="mds0931 nn1"/><link targets="mds0932 cjs"/>
   <link targets="mds0933 pnp"/><link targets="mds0934 vhd"/>
   <link targets="mds0935 vdn"/><link targets="mds0936 prp"/>
   <link targets="mds0937 crd"/><link targets="mds0938 nn1"/>
   <link targets="mds0939 avq"/><link targets="mds0940 vvn"/>
   <link targets="mds0941 prp"/><link targets="mds0942 at0"/>
   <link targets="mds0943 nn1"/><link targets="mds0944 pun"/>
</linkGrp>
```

This grammatical markup represents the text as completely unambiguous, despite the fact that instances of the same textual unit are associated with different structure elements (e.g. the word 'to'); moreover at least one sequence (the words 'to exercise', with identifiers mds0905 and mds0906), is in fact structurally ambiguous in English: it may be analyzed as a preposition followed by a singular noun (as this markup asserts) or as the infinitive marker followed by an uninflected form of a main verb.

To represent the ambiguity of words like 'to' and 'exercise', and of phrases like 'to exercise', we may use the <alt> and <join> elements defined in sections 14.8 *Alternation* and 14.7 *Aggregation*, as follows. First, we define <alt> elements for the ambiguous word classes, and add these to the <fsLib>.

```
<alt id="prp-to0" targets="prp to0"/>
<alt id="nn1-vvi" targets="nn1 vvi"/>
```

Next, we change the <link> elements for the text elements with identifiers mds0905 and mds0906:

```
<link targets="mds0905 prp-to0"/>
<link targets="mds0906 nn1-vvi"/>
```

As the encoding now stands, the phrase 'to exercise' has four structural analyses associated with it: preposition followed by noun, preposition followed by verb, infinitive marker followed by noun and infinitive marker followed by verb. To narrow the choices down to the desired two, namely preposition followed by noun and infinitive marker followed by verb, we next form <join> elements to represent the desired sequences.

```
<join id="prp.nn1" targets="prp nn1"/>
<join id="to0.vvi" targets="to0 vvi"/>
```

We then define an <alt> element to express the alternation between the two <join> elements.

```
<alt id="prp.nn1-to0.vvi" targets="prp.nn1 to0.vvi"/>
```

Next, we add a <phr> element in the encoding of the text for the phrase 'to exercise'.

```
<phr id="mds090506">
   <w id="mds0905">to</w>
   <w id="mds0906">exercise</w>
</phr>
```

Finally, we add to the <linkGrp> element a <link> element connecting that phrase to the <alt> that represent its two analyses.

```
<link targets="mds090506 prp.nn1-to0.vvi"/>
```

Note that the technique of forming <join> elements for sequences of structure elements and associating them with textual units can also be used to provide a complete structural analysis for the complex word ‘he’d’. First, we add an id attribute for the word.

```
<w id="mds093334">
   <w id="mds0933">he</w>
   <m id="mds0934">'d</m>
</w>
```

Next, we form a join of the structures associated separately with the subelements ‘he’ and ‘’d’.

```
<join id="prp.vhd" targets="prp vhd"/>
```

Finally, we define a link between the complex word and the new <join> element.

```
<link targets="mds093334 prp.vhd"/>
```