31 Multiple Hierarchies

This chapter will be substantially revised and expanded at the next release of these Guidelines.

At various points in these Guidelines, the discussion has mentioned the problems which arise when using SGML or XML to encode textual features which do not take a strictly hierarchical form: features, that is, which do not necessarily nest within other features. This chapter provides an overview of the techniques defined in these Guidelines for handling such problems, and should be consulted when deciding how to deal with them.

The following examples illustrate the type of problem with which this chapter is concerned:

- in narrative, a speech by a character may begin in the middle of a paragraph and continue for several more paragraphs
- in a verse text, the encoder may need to tag both the formal structure of the verse (its stanzas and lines) and its syntactic structures (which sometimes nest within the metrical structure and sometimes cross metrical boundaries)
- in any kind of text, the encoder may wish to record the physical structure of volume, page, column, and line, as well as the formal or logical structure of chapters and paragraphs or acts and scenes, etc.
- in verse drama, the structure of acts, scenes, and speeches often conflicts with the metrical structure
- in any kind of text, an embedded text (e.g. a play within a play, or a song) may be interrupted by other matter; the encoder may wish to establish explicitly the logical unity of the embedded material (e.g. to identify the song as a single song, and to mark its internal formal structure)
- in a dictionary, different types of information (e.g. orthography, syllabification, and hyphenation) may be combined within a single notation; the encoder may wish both to preserve the presentation of the material in the source text and to disentangle the logically distinct pieces of information in the interests of more convenient processing of the lexical information

Many other examples might be given, but these should suffice to show the variety of applications where non-hierarchical or non-nesting information appears, and to illustrate the various methods for addressing the problem.

Non-nesting information poses fundamental problems for any encoding scheme, and it must be stated at the outset that no solution has yet been suggested which combines all the desirable attributes of formal simplicity, capacity to represent all occurring or imaginable kinds of structures, suitability for formal or mechanical validation, and clear identity with the notations needed for simpler cases (i.e. cases where the textual features do nest properly). The representation of non-hierarchical information is thus necessarily a matter of choices among alternatives, of tradeoffs between various sets of different advantages and disadvantages.

There are several methods used within these Guidelines to handle non-nesting information:

- *concur*: an optional feature of SGML (not available in XML) which allows multiple hierarchies to be marked up concurrently in the same document
- *milestone elements*: empty elements which mark the boundaries between elements in a non-nesting structure
- *fragmentation* of an item: the division of what logically is a single element into two or more parts, each of which nests properly within its context
- *virtual joins*: the recreation of a virtual element from fragments of text, possibly discontiguous or out of order
- redundant encoding of information in multiple forms

In the sections which follow, these techniques, their advantages, and their disadvantages, are briefly described, and instances of their use within these Guidelines are pointed out. The examples show various solutions to the problem of direct speech spanning several paragraphs in a narrative; the text in question takes the following form:

> “The first thing that put us out was that advertisement. Spaulding, he came down into the office just this day eight weeks with this very paper in his hand, and he says:—”
>
> ““I wish to the Lord, Mr. Wilson, that I was a red-headed man.””
>
> ““Why that?” I asks.”

31.1 Concurrent Markup of Multiple Hierarchies

In SGML only, CONCUR allows us to mark up the document with many different hierarchical structures, but not to reorder the tree or to have different content in different views. Note that the restriction against having different content in different views is imposed not by SGML but by the TEI Interchange Format.

For example, if quotations are marked as part of a distinct markup stream given the name “QD”, the outermost speech in our example need not be broken up into multiple elements:

```
<(QD)q who='Wilson'> ...
<(TEI.2)p>The first thing that put us out was that
advertisement.  Spaulding, he came down into the
office just this day eight weeks with this very paper
in his hand, and he says:&mdash;</(TEI.2)p>
<(TEI.2)p><(QD)q who='Spaulding'>I wish to the Lord, Mr.
Wilson, that I was a red-headed man.</(QD)q></(TEI.2)p>
<p><(QD)q who='Wilson'>Why
that?</(QD)q> I asks.</(TEI.2)p>
 ... </(QD)q></p>
```

This method has the advantages of cleanly distinguishing among separate logical hierarchies in the text, using the same structures as non-concurrent markup and thus requiring no special conventions for use (as the other methods described in this chapter do). It has the disadvantage of using a cumbersome notation, which means it could most conveniently be used within processing environments which masks the complexity of the notation from the user; unfortunately, CONCUR is an optional feature of SGML and is not supported by all SGML processors.

The major use of concurrent markup in the current version of these Guidelines is in the tag set for concurrent markup for pages, columns, and lines defined elsewhere in this chapter.

31.2 Boundary Marking with Milestone Elements

Milestones use empty elements to mark the beginnings and endings of regions of the text which have something in common; they work like COCOA tags. Examples in these Guidelines include the `<milestone>` element and the elements `<pb>`, `<lb>`, and `<cb>`.

For example, if quotations are marked using (user-defined) empty elements given the names “qb” and “qe”, the empty elements mark the beginning and end of the speeches, but because they do not contain the speech as content, there is no element that needs to be broken up into multiple elements at the paragraph breaks.[175]

```
<qb who='Wilson'/> ...
<p>The first thing that put us out was that
advertisement.  Spaulding, he came down into the
office just this day eight weeks with this very paper
in his hand, and he says:&mdash;</p>
<p><qb who='Spaulding'/>I wish to the Lord, Mr.
Wilson, that I was a red-headed man.<qe/></p>
<p><qb who='Wilson'/>Why that?<qe/> I asks.</p>
 ... <qe/>
```

[175] as elsewhere in these Guidelines, empty elements are denoted with a penultimate slash character, which is the XML syntax; in SGML, either omit the slash or modify the SGML declaration to permit it via the NET delimiter.

This has the drawback that it is difficult to tell which <qe> corresponds with which <qb> without a complex processing of the text. One way to improve on this situation would be to use the linking attribute corresp discussed in chapter 14 *Linking, Segmentation, and Alignment* to associate the milestone indicating the end of a given speech with that indicating its start, as follows:

```
<qb who='Wilson' id='W1'/> ...
<p>The first thing that put us out was that
advertisement.  Spaulding, he came down into the
office just this day eight weeks with this very paper
in his hand, and he says:&mdash;</p>
<p><qb who='Spaulding' id='S1'/>I wish to the Lord, Mr.
Wilson, that I was a red-headed man.<qe corresp='S1'/></p>
<p><qb who='Wilson' id='W2'/>Why that?<qe corresp='W2'/> I asks.</p>
 ... <qe corresp='W1'/>
```

This method has the advantage of simplicity; it provides all the information needed to reconstruct all the competing hierarchical views of the text. Many times, the only processing required for an element occurs at its start and end (or can easily be formulated to do so); this markup method handles those cases well. In other cases, however, this method incurs the disadvantage of cumbersome processing: since the elements of the analysis (e.g. the direct speech of Wilson) are not uniformly represented by nodes in the document tree, they must be reconstituted by software in an ad hoc fashion, which may be difficult and is likely to be error prone. Processing elements may often involve more than specified actions at the start and end of an element. Most important for some encoders, this method disguises the logical relationship between the beginning and the ending of each logical element, making it impossible for parsers per se to provide the same kind of validation possible elsewhere in the encoding.

31.3 Fragmentation of Elements

Fragmentation breaks up what might be considered a single element into multiple smaller elements, in order to make it fit within the hierarchy. If a passage of direct discourse begins in the middle of one paragraph and continues for several more paragraphs, for example, one could encode the passage as a series of <q> elements. This has the effect that the document contains more <q> elements than it did before; to the extent that one might be interested in counting <q> elements, this is a drawback. To the extent that the element being broken up is used primarily to signal some characteristic, (e.g. that of being spoken by someone other than the narrator) rather than some countable object, this drawback is rather minor. Direct discourse is in fact so frequently interrupted by narrative irruptions, including but not limited to reporting clauses like “he said”, that the number of <q> elements is unlikely to correspond precisely to the number of utterances, speaker turns, or any other observable unit of conversation. For that reason, some encoders prefer to solve the quotation-and-paragraph problem using fragmentation.

To tag our example with this method, the outermost speech (Wilson’s) can be broken up to fit into the series of paragraphs, using the rend attribute to record the absence of closing quotation marks at the end of each paragraph. The inner speeches, being punctuated conventionally, need not carry rend values.

```
<p><q rend="pre ldquo" who="Wilson">The first thing that put us out
    was that advertisement. Spaulding, he came down into the office
    just this day eight weeks with this very paper in his hand, and he
    says:&mdash;</q></p>
<p><q rend="pre ldquo" who="Wilson"><q who="Spaulding">I wish to the
      Lord, Mr. Wilson, that I was a red-headed man.</q></q></p>
<p><q rend="pre ldquo" who="Wilson"><q who="Wilson">Why that?</q> I
    asks.</q></p>
```

Among the places where these Guidelines recommend fragmentation as a solution to the encoding of non-nesting information are the discussion of fragmentary verse lines, fragmentary stanzas, and fragmentary embedded texts in drama.

The advantages of this method are that it is simple, that at least one of the competing hierarchies can be processed normally, and that it makes the reconstitution of virtual units much easier, using the method described in the next section. Its disadvantages are that some units are not realized at all in the markup (here, the single long outermost speech of Wilson), and that automatic processing of these units is thus impossible when this method is used without further refinement.

31.4 Reconstitution of Virtual Elements

Virtual joins may be used to indicate objects in the text which would, for whatever reason, be difficult to mark using hierarchical syntax. In the TEI encoding scheme, virtual joins are most often expressed by the `<join>` element, the `<span>` element, or the `<fs>` element. This technique covers all out-of-line or 'stand-off' annotation methods, which involve the construction, out of line, of a clean structure representing the interpretation, and the virtual join of the interpretation with the text fragment instantiating it.

The tagging of our example with this method is almost identical to that given in the preceding section, with the addition of `<join>` elements to indicate the component parts of the individual speeches which have been broken up to fit into the paragraph hierarchy:

```
<p><q id="qw1" rend="pre ldquo" who="Wilson"> The first thing that put
    us out was that advertisement. Spaulding, he came down into the
    office just this day eight weeks with this very paper in his hand,
    and he says:&mdash;</q></p>
<p><q id="qw2" rend="pre ldquo" who="Wilson"><q who="Spaulding">I
      wish to the Lord, Mr. Wilson, that I was a red-headed man.</q></q></p>
<p><q id="qw3" rend="pre ldquo" who="Wilson"> <q who="Wilson">Why
      that?</q> I asks.</q></p>
<!-- ... -->
<join targets="qw1 qw2 qw3" result="q"/>
```

Alternatively, the next and prev attributes defined in chapter 14 *Linking, Segmentation, and Alignment* may be used to join the fragmentary quotations:

```
<p><q next="qw2" id="qw1" rend="pre ldquo" who="Wilson">The first
    thing that put us out was that advertisement. Spaulding, he came
    down into the office just this day eight weeks with this very
    paper in his hand, and he says:&mdash;</q></p>
<p><q next="qw3" prev="qw1" id="qw2" rend="pre ldquo" who="Wilson"><q
      who="Spaulding">I wish to the Lord, Mr. Wilson, that I was a
      red-headed man.</q></q></p>
<p><q prev="qw2" id="qw3" rend="pre ldquo" who="Wilson"><q
      who="Wilson">Why that?</q> I asks.</q></p>
```

The major advantage of this method is that it allows all the hierarchies in the text to be handled explicitly, both the privileged one directly represented, and the alternate hierarchy which has been split up and rejoined. Its major disadvantages are that (like most of the other methods described here) it privileges one hierarchy over the others, and requires special processing to reconstitute the elements of the other hierarchies.

Instances of this markup method in these Guidelines include the part attribute on the `<l>`, `<lg>`, `<seg>`, and numbered-segment elements, the `<join>`, `<span>`, and `<fs>` elements, and the global next and prev attributes available with the additional tag set for linking and alignment.

31.5 Multiple Encodings of the Same Information

In some cases, the simplest method of disentangling two conflicting hierarchical views of the same information is to encode it twice, each time capturing a single view. Thus, for example, a dictionary headword which gives in a single place the orthography, stress pattern, syllabification, and hyphenation for a word might be encoded several times: once with all the information in a single notation (as in the print dictionary), and once again for each separate piece of information — or at least, once more for the orthography, to speed up the common operation of searching for the article for a given headword in the electronic dictionary.

The out-of-line treatment of annotation in the feature structure notation (defined in chapter 16 *Feature Structures*) may also be considered to fall under this rubric.

The advantages of this method of markup are that each way of looking at the information is explicitly represented in the data, and may be processed in straightforward ways, without requiring complex methods of disentangling information relevant to one view from information relevant only to other views. It has the disadvantage of requiring more space and of introducing redundant information into the encoding, with the resulting risk that one view may be updated without corresponding changes being made to the others, resulting in inconsistencies within the document.

31.6 Concurrent Markup for Pages and Lines

Where the main purpose for encoding alternative hierarchies in a text is to represent competing referencing schemes describing the same basic text, the CONCUR mechanism of SGML provides a very natural solution. Note, however, that the CONCUR feature is not supported by much of the available SGML software, nor is it available in XML at all.

One common form of traditional reference system specifies the page and line, or page, column, and line of a passage as it appears in some standard edition. Such references may be specified using a concurrent markup hierarchy which divides the body of a text into pages and lines or into pages, columns, and lines. Volumes may also need to be identified. The document type name should be a short identifier for the edition cited. The following tags may be used:

<vol> marks the individual volumes of a reference edition.
<page> marks pages in a reference edition.
<col> contains one column of a multi-column reference edition.
<line> contains one line of a reference edition.

Page and line numbers for an edition by Lachmann, for example, might be specified thus:

```
<(La)page n='37'> <!-- Text from Lachmann, p. 37 -->
          ...
<(La)line n='32'> <!-- Text from Lachmann, p. 37, line 32 -->
<(La)line n='33'> <!-- Text from Lachmann, p. 37, line 33 -->
<(La)line n='34'> <!-- Text from Lachmann, p. 37, line 34 -->
<(La)page n='38'> <!-- Text from Lachmann, p. 38 -->
<(La)line n='1'>  <!-- Text from Lachmann, p. 38, line 1 -->
<(La)line n='2'>  <!-- Text from Lachmann, p. 38, line 2 -->
<(La)line n='3'>  <!-- Text from Lachmann, p. 38, line 3 -->
          <!-- etc. -->
<(La)page n='39'> <!-- Text from Lachmann, p. 39 -->
 ...
<(La)line n='18'> <!-- Text from Lachmann, p. 39, line 18 -->
<(La)line n='21'> <!-- Text from Lachmann, p. 39, line 21 -->
          <!-- etc. -->
<(La)page n='40'> <!-- Text from Lachmann, p. 40 -->
 ...
          <!-- etc. -->
<(La)page n='41'> <!-- Text from Lachmann, p. 41 -->
 ...
          <!-- etc. -->
```

The markup shown above would be interleaved with the normal markup for the document. Since SGML requires tags in concurrent markup streams to be labeled with their document type, however, the 'normal' markup would need to have the notation `TEI.2` inserted before each tag's generic identifier. The combined markup might look something like this:

```
<(TEI.2)TEI.2>
 <(TEI.2)TEI.Header> ... </(TEI.2)TEI.Header>
 <(TEI.2)text> ...
 <(TEI.2)div0><(TEI.2)head> ... </(TEI.2)head>
 ...
 <(TEI.2)div1>
 <(TEI.2)div2> ...
 <(La)page n='37'> <!-- Text from Lachmann, p. 37 -->
                ...
 <(La)line n='32'> <!-- Text from Lachmann, p. 37, line 32 -->
```

```
<(La)line n='33'> <!-- Text from Lachmann, p. 37, line 33 -->
<(La)line n='34'> <!-- Text from Lachmann, p. 37, line 34 -->
<(La)page n='38'> <!-- Text from Lachmann, p. 38 -->
<(La)line n='1'>  <!-- Text from Lachmann, p. 38, line 1 -->
<(La)line n='2'>  <!-- Text from Lachmann, p. 38, line 2 -->
<(La)line n='3'>  <!-- Text from Lachmann, p. 38, line 3 -->
</(TEI.2)div2>
<(TEI.2)div2>
<(La)line n='4'>  <!-- Text from Lachmann, p. 38, line 4 -->
                <!-- etc. -->
</(TEI.2)div2>
</(TEI.2)div1>
<(TEI.2)div1><head>... </head>
                ...
<!-- Text from Lachmann, p. 39 -->
...
<(La)page n='39'> <!-- Text from Lachmann, p. 39 -->
...
<(La)line n='18'> <!-- Text from Lachmann, p. 39, line 18 -->
<(La)line n='19'> <!-- Text from Lachmann, p. 39, line 19 -->
<(La)line n='20'> <!-- Text from Lachmann, p. 39, line 20 -->
<(La)line n='21'> <!-- Text from Lachmann, p. 39, line 21 -->
                ... <!-- etc. -->
<(TEI.2)div1>
<(TEI.2)div2>
</(TEI.2)div2>
<!-- etc. -->
<!-- Text from Lachmann, p. 40 -->
...
<!-- etc. -->
</(TEI.2)text>
</(TEI.2)TEI.2>
```

The following declarations give the formal specification for the standard pre-defined document type designed for recording page and line numbers of a reference edition in an SGML concurrent markup stream.

```
<!-- 31.6: Concurrent Document Type for Page and Line References-->
<!ENTITY % version 'ref' >
<!ELEMENT %version; %om.RR;  (#PCDATA | page | vol)*>
<!ATTLIST %version;
      %a.global; >
<!ELEMENT vol %om.RR;  (#PCDATA | page)*>
<!ATTLIST vol
      %a.global;
      TEIform CDATA 'vol'  >
<!ELEMENT page %om.RO;  (#PCDATA | line | col)*>
<!ATTLIST page
      %a.global;
      TEIform CDATA 'page'  >
<!ELEMENT col %om.RO;  (#PCDATA | line)*>
<!ATTLIST col
      %a.global;
      TEIform CDATA 'col'  >
<!ELEMENT line %om.RR;  (#PCDATA)>
<!ATTLIST line
      %a.global;
      TEIform CDATA 'line'  >
<!-- end of 31.6-->
```

This concurrent hierarchy is enabled as shown below: after the document type declaration for the TEI.2 document type, the document should contain the sequence of lines:

```
<!DOCTYPE La PUBLIC
  "-//TEI P4//DTD Concurrent Document Type: Pages and Lines//EN"
       'teipl2.dtd' [
    <!ENTITY % version "La" >
    ]>
```

which call the document type for page and line references and give it the name “La”. If page and line numbers from more than one standard edition are to be marked, then the relevant lines may be repeated, each time using a different value for the document type and entity definition (where the example has “La”). For example, to show page and line numbers from the editions of Lachmann (La), Kraus (Kr), and Moser/Tervooren (MT) at the same time, one might use declarations like the following:

```
<!DOCTYPE La PUBLIC
    "-//TEI P4//DTD Concurrent Document Type: Pages and Lines//EN"
       "teipl2.dtd" [
         <!ENTITY % version "La" >
    ]>
    <!DOCTYPE Kr PUBLIC
    "-//TEI P4//DTD Concurrent Document Type: Pages and Lines//EN"
       "teipl2.dtd" [
         <!ENTITY % version "Kr" >
    ]>
    <!DOCTYPE MT PUBLIC
    "-//TEI P4//DTD Concurrent Document Type: Pages and Lines//EN"
       "teipl2.dtd" [
         <!ENTITY % version "MT" >
    ]>
```

To document a referencing system of this kind the TEI header, a formal declaration should be provided in the `<refsDecl>` element described in section 5.3.5 *The Reference System Declaration*. For the above example, a declaration such as the following would be appropriate:

```
<refsDecl doctype='La'>
    <step  from='CHILD (1 page n %1)'/>
    <step  from='CHILD (1 line n %2)'/>
</refsDecl>
<refsDecl doctype='Kr'>
    <step  from='CHILD (1 page n %1)'/>
    <step  from='CHILD (1 line n %2)'/>
</refsDecl>
<refsDecl doctype='MT'>
    <step  from='CHILD (1 page n %1)'/>
    <step  from='CHILD (1 line n %2)'/>
</refsDecl>
```

Hierarchies similar to that defined above can be provided for most common hierarchical reference systems. Hierarchies such as act / scene / line, for conventional dramatic structure, book / canto / stanza / line, for longer verse texts, or book / poem / stanza / line, for collections of verse, may be readily expressed with concurrent SGML markup. Since these hierarchical structures can readily be represented using the base tag sets described in part III of these Guidelines, however, reference systems with such structures may most readily be expressed using the `n` or `id` attributes, as described above in section 6.9.1 *Using the ID and N Attributes*.

Any text with an idiosyncratic standard reference system will require its own dtd, so that appropriately named tags can be created for the reference units. Such dtds may follow the pattern of those described in the preceding section; they should also be documented in an auxiliary tag set description file, using the tags described in chapter 27 *Tag Set Documentation*.