Extreme Markup Languages 2005®

Montréal, Québec
August 1-5, 2005

TEI HORSEing Around

Syd Bauman
Brown University Women Writers Project

Abstract

The Text Encoding Initiative's typed segment-boundary delimiter method is only one of several proposed mechanisms for handling overlap in TEI documents. HORSE (aka CLIX) defines a method by which an XML element is used normally when possible and as an improved version of the typed segment-boundary delimiter method when an overlap problem is encountered. A significant portion of the rules necessary for validation of HORSE markup can be expressed using Schematron. This, combined with an utter hack that can "HORSEify" the declaration of elements in a TEI Relax NG grammar, can provide a potential significant step forward in handling overlap in TEI documents.

A Conference of IDEAlliance
Extreme Markup 2005 Languages®

TEI HORSEing Around

Table of Contents

A Conference of IDEAlliance
Extreme Markup Languages® 2005

TEI HORSEing Around

Syd Bauman

NOTE: A more recent copy of this paper may be available at **http://dev.stg.brown.edu/staff/Syd_Bauman/papers/EML2005Baum1020.xml** and **http://dev.stg.brown.edu/staff/Syd_Bauman/papers/EML2005Baum1020.html**.

§ Introduction

As a theoretical problem, overlap is an intriguing issue for the markup community generally.[1] For may users of the TEI [Text Encoding Initiative] Guidelines ([TEIP4]), however, it is particularly urgent and more practical. Because TEI users are typically transcribing texts rather than authoring them, they must accurately capture the features of their sources, even if these involve overlapping hierarchies. And since the TEI community so often creates digital texts that must serve a multidisciplinary audience, they must often mark many different types of textual features, which often tend to overlap, in a single text.

Steve DeRose's recent paper ([HORSE]) introduced YAMFORX [yet another method for overlap representation in XML]. DeRose et al. had discarded their original name HORSE [hierarchy-obfuscating really spiffy encoding] in favor of CLIX [canonical LMNL in XML], and went on to assert that any LMNL document could be expressed in CLIX.[2] [3] The approach that DeRose proposed was discussed at the October meeting of the TEI SIG on overlapping hierarchies in Baltimore. The SIG discussed CLIX's applicability to TEI, and thought that the methodology was promising. The group was particularly interested in exploring a system for handling overlap in which either hierarchy could be represented by CLIX, thus permitting the other to be validated by standard XML tools. It was thought that perhaps this could be accomplished by a transform based on an extended version of XSLT ([EXSLT]). However, the details of this system have not been worked out, and it was recognized that CLIX markup would not be useful in the TEI community unless it could be properly constrained and machine validated.

This paper provides an analysis of CLIX's constrainability, particularly for use with TEI, building on the results of that meeting. However, since the goal here is not generic representation of LMNL, but rather using CLIX-style markup in TEI XML documents only where overlap occurs, it would seem best to come up with a more appropriate name. Several suggestions have been put forth, including:

- COLT [co-indexed overlap terminus]
- PONIE [partially overlapped nested index elements]
- SADDLE [structured addition of dual-delimited linked encoding]

For this paper, it seems more reasonable to return to the original name, HORSE.

In this paper I will

1. quickly review HORSE (aka CLIX), comparing it to TEI typed segment-boundary delimiters
2. demonstrate how a significant portion of the rules necessary for validation of HORSE markup can be expressed via a RelaxNG schema
3. demonstrate how most of the rules necessary for validation of HORSE markup can be expressed via a Schematron script
4. demonstrate an shameless hack that will “HORSEify” the declaration of elements in a TEI P5 RelaxNG (compact syntax) grammar
5.
   - discuss the desire to “swap” hierarchies, applying a “just in time” approach
   - discuss the algorithm for swapping a simple case
   - discuss a major reason why such swapping is problematic
6. in an appendix, provide a nomenclature for the discussion of overlapping hierarchies

§ HORSE

In 1989 a TEI working paper was begun [MLW18] which eventually grew into a published article [CHUM]. In this paper, a number of solutions to the overlap problem were proposed, including one called “typed segment-boundary delimiters”. In this method of encoding, an element type that is expected to

Figure 1

```
<l>And I said to him, <q endpnt="start" id="q1"/>Superman, have you not seen,</l>
<l>The embarrassment havoc I'm wreaking?<q endpnt="end" startid="q1"/></l>
```

overlap other structural elements would be encoded with two empty elements of the same element type, each of which points to the other (although only one of those pointers is an ID/IDREF link that would be validated by a parser). The first, "starting" occurrence was assigned an `id=` attribute and carried an `endpnt="start"`; the second, "ending" occurrence was assigned a `startid=` IDREF attribute which pointed to the first occurrence, as well as an `endpnt="end"`. In order to indicate that an occurrence of the element type really had no content, only a single (as opposed to paired) empty element would appear, and its `endpnt` attribute would be set to the value `"point"`.

The following fanciful poem (with apologies to Gilbert & Sullivan — if it's any consolation, the real words from the *Mikado* are used for the example in the Guidelines [TEIP4]) has a quotation overlapping the boundary between two metrical lines:

```
I found at a conference C M Sperberg-McQueen
Sang "closing, keynoting, I'm speaking"
And I said to him, "Superman, have you not seen,
The embarrassment havoc I'm wreaking?"
```

[4] If we presume an encoder wishes to preserve the prosodic structure by encoding each metrical line in a TEI `<l>` element, and the quotation structure by encoding each quotation in a TEI `<q>` element, we have a standard overlap problem in the 3rd and 4th lines.

A typed segment-boundary delimiter encoding of these two lines might be as seen in figure 1.

Last year Tommie Usdin presented Steve DeRose's paper [HORSE], in which it was claimed that a mildly different syntax of this method of encoding permits complete representation of any LMNL (see [LMNL]) document in XML.[2] The encoding method presented was initially called HORSE, but the name was changed to CLIX, which then grew to mean "canonical LMNL in XML". Both because there already exists a CLIX language (Constraint Language in XML [CLIX]), and because the use of this technique that I am discussing has no direct relationship to LMNL, I have chosen to revert to DeRose et al.'s original name, HORSE.

In HORSE, a content object which is likely to overlap another XML element is used normally whenever it **doesn't** cause an overlap problem, and is encoded using the same element type, but with an improved version of the typed segment-boundary delimiter method whenever it **does**. Thus:

```
<l>I found at a conference C M Sperberg-McQueen</l>
<l>Sang <q>closing, keynoting, I'm speaking</q></l>
<l>And I said to him, <q sID="q2"/>Superman, have you not seen,</l>
<l>The embarrassment havoc I'm wreaking?<q eID="q2"/></l>
```

Here the existence of the `sID=` attribute indicates that the 2nd occurrence of the `<q>` element is actually a segment-boundary delimiter, the start of what would be a normal `<q>` element if it could.

The rules of HORSE markup are reasonably simple, although no closed schema language of which I'm aware can enforce them all. They are (copied from DeRose's paper [HORSE]):

1. The element must be empty whenever either the `sID=` or `eID=` attribute is specified[5].
2. When `eID=` is present, no other attributes are permitted.[6]
3. each `sID=`/`eID=` value should occur only twice (once on `sID=` and once on `eID=`)
4. Empty elements with matching `sID=` and `eID=` values should match up in proper pairs and in order.

The implication of rule #3 is that the value of an `sID=` or `eID=` can only appear twice in an entire instance. Rule #4 seems to further constrain the two values such that they must occur on the same element type. However, to find the other end of an overlapping content object marked by a HORSE element, a processor need only look for an element of the same type (i.e., with the same GI) that has a matching `sID=` or `eID=`. Thus it is not at all clear why a pair of `sID=`/`eID=` values could not be permitted once for any given element type. So for example, perhaps rule #3 should read "Each value of `sID=` or `eID=` should occur

only twice for any given element type (i.e., element with the same name). The value should occur once on an sID=, and once, presumably later in the text, on an eID=."

Under this revised constraint #3, the following would be permissible HORSE markup:

```
  <lg sID="a" type="verse"/>
    <l>Asked a girl what she wanted to be</l>
    <l sID="a"/>She said <q>baby, can't you see<l eID="a"/>
    <l>I wanna be famous, a star on the screen</l>
    <l>But you can do something in between</l>
  <lg eID="a"/>
  <lg sID="b" type="chorus"/>
    <l>Baby you can drive my car</l>
    <l>Yes I'm gonna be a star</l>
    <l>Baby you can drive my car</l>
    <l sID="b"/>And baby I love you</q><l eID="b"/>
  <lg eID="b"/>
```

At first glance, rule #4 seems ambiguous. Does "match up in proper pairs and in order" mean that the "elements" defined by the HORSE markup should nest properly within each other, such that were they to become "normal" elements in their own hierarchy they would be well-formed? This seems like an odd restriction for a system of markup designed to handle overlap that is by nature not well-formed. Some have suggested that it merely means that such elements must appear in pairs, i.e. be of the same element type, with the occurrence bearing sID= preceding the occurrence bearing eID=.

§ Validating the Constraints

None of these rules, or constraints, can be expressed in DTDs. All of them (I believe) could be validated by special-purpose software. It would not be difficult, for example, to write a routine that used SAX events and counted sID= and eID= attributes to ensure they were paired properly. With a simple stack of open elements (including HORSEified elements), it could also issue error messages if any did not nest properly. My goal, however, is to investigate validation of HORSE markup using commonly available schema languages. In particular I will concentrate on the "closed" schema language RelaxNG, and the "open" schema language Schematron. This is not because these two languages are the best in their respective categories, but rather because they are the ones in each category with which I am familiar.[7]

RelaxNG a HORSE[8]

Constraints #1 and #2 above can be easily expressed in RelaxNG. Here is an example RelaxNG (compact syntax) declaration of a normal element, and one for the HORSEable equivalent.

```
# normal declaration
element q {
   attribute dir { xsd:boolean }?,  # direct vs indirect speech
   text
}
```

This declares a <q> element that has one attribute, the optional dir=, and can have only text (aka PCDATA) as content.

```
# HORSEable declaration
element q {
   ( attribute sID { xsd:NCName }, attribute dir { xsd:boolean }? )  # HORSE start
 | ( attribute eID { xsd:NCName } )                                  # HORSE end
 | ( attribute dir { xsd:boolean }?, text )                          # normal
}
```

Note that it is not necessary that sID= and eID= be declared as xsd:NCName, as the only requirement is that a processor be able to match their values to each other. Nonetheless, this seems like a prudent restriction to apply that is not likely to cause any burden, and will likely make it simpler to write software to process them.

Schematroners Sing This Song

Constraints #3 and #4 above can not be expressed in RelaxNG.[9] However, they can be expressed, with some difficulty, in Schematron (see [dsdl3]).

The following fragment of Schematron performs very basic validation of constraint #3 for the <q> element. That is, it tests that any value of sID= occurs twice, once on the sID= of the <q> element that we are currently examining, and once on an eID= of a <q> element. Supposedly RelaxNG has already validated for us that the eID= does not occur on the same occurrence of <q> as the sID=, so that possibility is not tested here. This fragment of Schematron does not issue any useful diagnostic messages. Here the "s:"

prefix is for Schematron elements, and unprefixed names are in the TEI namespace. (The same holds for all subsequent fragments of Schematron.)

```
<s:pattern name="2 occurrences of HORSEID, one sID, one eID">
  <s:rule context="q[@sID]">
    <s:let name="myID" value="./@sID"/>
    <s:report test="count(//q[@sID=$myID]) > 1">The value of sID= must be unique
    to sID=s.</s:report>
    <s:report test="count(//q[@eID=$myID]) = 0">An sID= value must match an
    eID= value.</s:report>
    <s:report test="count(//q[@eID=$myID]) > 1">An sID= value must match only
    1 eID= value.</s:report>
  </s:rule>
</s:pattern>
```

Note that this is ISO Schematron code (Schematron 1.5 does not have a `<let>` element).

The Needed XPath's Five Miles Long

Neither interpretation of constraint #4 can be expressed in RelaxNG. If the constraint merely implies that for a given HORSE pair the occurrence with `sID=` must precede the occurrence with `eID=`, then this is easily expressible in Schematron. If it means that HORSE elements must "nest" such that they would be well-formed if they were real elements, it may also be expressible in Schematron, but I have so far been unable to test this sufficiently.[10]

The following fragment of Schematron ensures that an `sID=` occurs before its matching `eID=`. Here "matching" means both that the special attribute values match and that the element types match. (This both prevents a mis-match between a HORSE start tag that matches a HORSE end tag of a different color, and simultaneously permits the same `sID=` value to be used multiple times in one document, so long as it is on different element types.) Unlike the previous example, it is not specific to the `<q>` element.

```
<s:rule context="*[@sID]">
  <s:assert test="following::*[name(.)=name(current()) and ./@eID=current()/@sID]">
    An sID= without a matching eID= following it
  </s:assert>
</s:rule>
```

Oh, Do It All At Once

The following Schematron schema validates all four constraints, presuming the special attributes may occur once per element type, and that constraint #4 does not imply proper nesting.

```
<?xml version='1.0' ?>
<!-- Copyleft 2005 Syd Bauman -->
<!-- This is intended for P5, so default namespace is TEI -->
<s:schema xmlns:s="http://www.ascc.net/xml/schematron"
          xmlns="http://www.tei-c.org/ns/1.0"
          xml:lang="en">

  <s:title>Validate HORSE Markup</s:title>

  <s:pattern name="HORSE rule #1">
    <s:rule context="*[@sID or @eID]">
      <s:report test="child::* or child::text()">
        Elements with sID= or eID= must be empty.
      </s:report>
    </s:rule>
  </s:pattern>

  <s:pattern name="HORSE rule #2">
    <s:rule context="*[@eID]">
      <s:report test="@*[name()!='eID']">
        Elements with eID= must not have other attributes.
      </s:report>
    </s:rule>
  </s:pattern>

  <s:pattern name="HORSE modified rule #3">
    <s:rule context="*[@sID]">
      <s:assert test="count( //*[name(.)=name(current())
                                 and @sID=current()/@sID] ) = 1">
        An sID= value should not occur on more than one &lt;<s:name path="."/>> element.
      </s:assert>
    </s:rule>
    <s:rule context="*[@eID]">
      <s:assert test="count( //*[name(.)=name(current())
                                 and @eID=current()/@eID] ) = 1">
        An eID= value should not occur on more than one &lt;<s:name path="."/>> element.
      </s:assert>
    </s:rule>
  </s:pattern>

  <s:pattern name="HORSE rule #4">
```

```
    <s:rule context="*[@sID]">
      <s:report test="count( following::*[name(.)=name(current())
                                          and ./@eID=current()/@sID] ) = 0">
        An sID= occurs that does not have a matching eID= following it.
      </s:report>
      <s:report test="count( following::*[name(.)=name(current())
                                          and ./@eID=current()/@sID] ) &gt; 1">
        An sID= occurs with more than one matching eID= following it.
      </s:report>
    </s:rule>
  </s:pattern>

</s:schema>
```

§ The TEI Stables

So the question arises, how could we go about using HORSE markup in TEI files?

The TEI Guidelines are encoded in a single XML document which is itself a TEI document. Various forms of output may be generated from this source file, including formal reference documentation, the descriptive chapters of the TEI Guidelines, and the schema files (in various languages). In the conventional terminology of literate programming it is a "tangle" process that produces the P5 RelaxNG schema files.

Because the TEI P5 RelaxNG schema files are machine-generated, the syntax used is only a subset of that permitted by RelaxNG, and the format is 100% consistent and predictable. So while RelaxNG permits any amount of white-space between, say, a name of a pattern being defined and the following equals sign ("="), in TEI P5 RelaxNG (compact syntax) files, there is **always** one space character preceding the equals sign ("=").

Furthermore, because of the consistent indirection of the TEI P5 RelaxNG compact syntax schema files, the format of every element declaration is exactly the same, differing only in the name of the element being declared. An example here will be worth a thousand words.

Figure 2: declaration of the <q> element

```
element q { q.content, q.attributes }
```

This predictability permits us to transform any given element declaration into a similar declaration that could be used for HORSE markup as well.

Figure 3 is a fragment of Perl code that could perform this task.

Figure 3: Perl fragment transforms standard TEI into HORSE

```
# $foal contains GI of the element that is becoming a HORSE element
while(<>) {
  s[^(  element ($foal) \{) \2\.content, \2\.attributes \}]
   [\1
        ( horse.start.attributes, \2.attributes )
      | ( \2.content, \2.attributes )
      | ( horse.end.attributes )\n    }]gx;
  print;
  }
```

[11] When applied to the declaration for <q>, above (see figure 2), the output is as found in figure 4.

Figure 4: Output of figure 3 applied to figure 2

```
  element q {
        ( horse.start.attributes, q.attributes )
      | ( q.content, q.attributes )
      | ( horse.end.attributes )
    }
```

All that remains is to declare the patterns used for indirection.

Figure 5

```
<lg type="verse">
  <l>I found at a conference C M Sperberg-McQueen</l>
  <l>Sang <q>closing, keynoting, I'm speaking</q></l>
  <l sID="L3"/>And I said to him, <q>Superman, have you not seen,<l eID="L3"/>
  <l sID="L4"/>The embarrassment havoc I'm wreaking?</q><l eID="L4"/>
</lg>
```

```
horse.start.attributes = attribute sID { tei.datatype.horseID }
horse.end.attributes   =  attribute eID { tei.datatype.horseID }
tei.datatype.horseID   =  xsd:NCName
```

Here the standard mechanism for indirection of attribute types is used.

§ Jockeying for Position

When two content objects overlap, and are both encoded in XML, one using "normal" well-formed XML elements, and the other using some other mechanism (e.g., the TEI `part=` attribute, the TEI `next=` and `prev=` attributes, or the HORSE mechanism described here), it seems to be generally agreed that the one encoded with "normal" well-formed XML elements has been favored over the other. While it is not clear to the author that this is necessarily the case, it does seem to be practically the case: off-the-shelf software will read, parse, and process the normal element, but there exists very little, if any, software that will know what to do with the "specially" encoded content object.

However, someone who has documents with overlapping content objects of interest may not wish to permanently favor one hierarchy over the other. It is likely that such a user would like to have equal access to both hierarchies at once, all the time.[12] Failing that, it is reasonable to believe that users would want to "favor" different hierarchies at different times. There are many reasons for wanting to do this; the most obvious of which is for validation purposes. While the *position* of a HORSE-element in the structure of a document can be constrained by its schema (an advantage of using the same element type), the *content* is not. In fact, not only do standard validation techniques not validate the content of a HORSE "element", its content is validated as if it were the content of the parent element or elements. More on this below.

But validation is not the only reason one might wish to "swap" the dominant hierarchy with the subjugated (i.e. HORSEd) hierarchy. E.g., one might wish to have metrical lines marked up with normal well-formed XML elements when performing metrical analysis (for which quotations are not generally of interest), and to have quotations marked up with normal well-formed XML elements when performing textual analysis intended for authorship attribution — during which a quotation may deserve to be treated differently than words implicitly in the author's "normal" voice.

So the previous example (see figure 1which had `<q>` HORSEified to avoid overlap with `<l>`) altered so that `<l>` is HORSEified to avoid overlapping normal `<q>` elements, and wrapped in an `<lg>` (the TEI "line group" element contains things like verses or stanzas) is found in figure 5.

While this is advantageous for various potential processes, most notably validating the content of `<q>`, it has the enormous disadvantage of not being valid TEI markup as it stands. This is because text content (aka PCDATA) is not permitted as a direct child of a TEI `<lg>` element.

I have decided to refer to this problem as the "bare back" problem: the PCDATA is now sitting as the bare child of an element which, back in the schema, is declared as element content. It will occur whenever a child of an element being HORSEified is not permitted as content of the parent of the element being HORSEified.

It is my guess that in most common cases of overlap within the TEI, the bare back problem either will not occur or will occur only when one hierarchy is subjugated, but not the other. I have not tested this hypothesis yet, however.

One algorithm for converting a HORSEd element to a standard element, while correspondingly converting its parent elements to HORSE markup using a stream parser is as follows.

a. read in data, keeping track of open elements, until you get to an element with `sID=` or `eID=` (we'll call it `<B>`; note that the `sID=` should always occur first)

b. scan left to the first unclosed start-tag (we'll call it <A>)

c. convert <A> to a HORSE start-tag; i.e., make it an empty element with the same attributes as it originally had plus an sID=

d. convert the <B> HORSE start-tag to a normal start-tag, i.e., make it a normal start-tag (instead of an empty element) and remove the sID=; OR convert the <B> HORSE end-tag to a normal end-tag, i.e., make it a normal end-tag (instead of an empty element) and remove the eID=

e. continue to read until you get to the end-tag of the <A> whose start-tag was just converted (which is not necessarily the first <A> end-tag to be encountered)

f. convert this end-tag to a HORSE end-tag by making it an empty element with an eID= that matches that of the <A> HORSE start-tag generated above

§ Conclusion

Every good scientific research paper in medicine ends with roughly the same conclusion: "more study is needed in this area". It's true here, too. While I have demonstrated that HORSE markup can be validated with schemas written in popular schema languages, have demonstrated that TEI could make use of some of this validation without a major change to its ODD system, and discussed some of the problems inherent in HORSE markup in general that becomes particularly evident when one changes the dominant hierarchy, I have not put forth a formal algorithm for changing such a hierarchy, proposed solutions for the invalidities that result, nor even mentioned the problems style-sheet authors and other document processors will run into when handling documents with HORSE markup.

§ Appendix I

It may be worth considering doubly-linked elements instead, e.g.:

```
<q dir="yes" xml:id="q07s" myEnd="#q07e"/>
  <!-- "content" of an overlapping <q> element here  -->
<q xml:id="q07e" myStart="#q07s"/>
```

This would permit myStart= and myEnd= to be declared as XPointers. It's not at all clear what the complete ramifications of this are. (E.g., what does it mean if my end-tag is not only in another document on another part of the web, but is dynamically generated?)

§ Appendix II: Suggested Nomenclature

In order to discuss the various solutions to encoding features that do not fit a single strict hierarchy, it is useful to have a common nomenclature with which to discuss the pieces of the problem. Here are the definitions used by the Brown University Women Writers Project as we discussed this problem.

overlapping elements	two (or more—eek!) encoded textual objects that do not properly nest; i.e., a textual object that starts in one element but ends in another.
fragmentation	the division of what logically is a single element that doesn't fit into the XML hierarchy into multiple pieces which do
text fragments	the pieces of text resulting from fragmentation; each one fits into the hierarchy, but you would treat the combination of them as a unit, if you could
partial element	A complete element (has a start-tag, content, and an end-tag, although one or both may be minimized) whose content is one of the text fragments
aggregate element	a pseudo-element that is composed of the partial elements that were used to encode each text fragment
aggregation	the process of pointing (aka "chaining"), hunting/gathering, or joining partial elements into an aggregate element
pointing	the process of using a (hyperlink) reference mechanism so that each of the partial elements that make up an aggregate element points to one or more other partial elements of the same aggregate element
hunting/gathering	the process of aggregating an element that has been marked as aggregate by use of the part= attribute on each of the partial elements (in TEI, the part= attribute is a legal attribute of <l>, <lg>, (including all

	the <lgN> and <divN> elements), <c>, <cl>, <m>, <phr>, <s>, <seg>, and <w>).
joining	to combine partial elements into an aggregate element by using the <join> element
horse markup	The Trojan (i.e., same name) typed segment-boundary delimiter elements (indicated by the existence of sID= and eID=) at the start and end of all but one of a set of overlapping elements.
changing horses	The process of swapping which element type of a set of overlapping elements is the one which is not indicated using horse markup (i.e., is encoded normally with a start tag and an end tag).

Notes

1. Although most of the present paper is about the "HORSE" part of DeRose's paper *Markup Overlap: A Review and a Horse* ([HORSE]), the "review" part provides a good introduction to what the XML overlap problem is. There are dozens, if not hundreds of other discussions of this problem, many of which are summarized and referred to on the Cover Pages ([Cover]).
2. An anonymous reviewer of the current paper points out, correctly I believe, that DeRose overstated the case — arbitrary LMNL can not be expressed in CLIX as currently defined, because LMNL annotations (roughly analogous to XML attributes) can have structure, potentially overlapping structure, of their own. It may be the case that some minor modifications to CLIX markup, in particular permitting CLIX elements to have CLIX children, would overcome this problem, but this issue is not considered here.
3. If you like puns, this paper is a must-read even if you already know about CLIX. The one missed opportunity is that because Steve mis-represented segment-boundary delimiters as milestones he could not make puns about Trojan boundaries — something about being awash in aging C, or perhaps taking advantage of the island of Lesbos being nearby and that start- and end-segment-boundary delimiter elements are sometimes called "sisters".
4. See the introduction of the closing keynote address of Extreme Markup Languages 2004 ([SI]).
5. SGML could accomplish this with its much-maligned CONREF attributes
6. Just as some restrictions on XML come with the attached information "for compatibility", this one should come with "for ease of processing" attached, as it is not strictly necessary. One can imagine a processor that reads attributes from either the HORSE-start or HORSE-end tag and attaches them to the proper node of its in-memory representation. One can imagine a processor that finds attributes on HORSE-end tags and transfers them to the corresponding HORSE-start tag. And, as an anonymous reviewer of the current paper points out, when translating to LMNL there is no problem, as LMNL permits annotations on close range markers.
7. It is at this point that I wish I had a co-author who was an expert in the W3C XML Schema language and Constraint Language in XML.
8. No, I don't mean Thorazine®, the use of which is not recommended in horses due to side effects (e.g., ataxia, panic reaction).
9. And I do not think they can be expressed in W3C XML Schema language, either, but I am far from an expert.
10. An anonymous reviewer suggested the following:

```
test='not( count( following::q[@sID][following::q[@eID=$myID]] )
        != count( following::q[@eID][following::q[@eID=$myID]] ) )'
```

    from whence this section gets its name.
11. This code matches the output of the TEI's current tangle process exactly. In truth, the program I used to test this was deliberately more permissive with respect to white-space. The "search" regular expression

```
^(\s*element\s+($foal)\s*\{}\s*\2\.content\s*,\s*\2\.attributes\s*\}
```

will not be disturbed by differences in white-space.

12. In which case, perhaps a non-XML encoding should be considered.

Acknowledgements

The members of the TEI special interest group on overlapping hierarchies who were present at the meeting from which the ideas for this paper sprang included

- Syd Bauman
- Patricia Bart
- Jelks Cabaniss
- Patrick Durusau
- Dorothy Porter (convener)

The author would like to gratefully acknowledge the following people.

The various anonymous reviewers of this paper provided incredibly helpful input including many points well taken and useful suggestions that have drastically improved the content of this paper.

In addition, Wendell Piez contributed some helpful input towards the end stages of production. Lastly, and far from least, Julia Flanders helped invaluably with the actual manuscript preparation.

Bibliography

[CHUM] Barnard, D.T., L. Burnard, J.-P. Gaspart, L. Price, C.M. Sperberg-McQueen, and G.B. Varile; "Hierarchical Encoding of Text: Technical Problems and SGML Solutions" *Computers and the Humanities* 29:3 (1995) 211–231. Papers in this special issue are also published in: N. Ide and J. Veronis, (eds.); *The Text Encoding Initiative: Background and Context* Kluwer Academic Publishers, Dordrecht (1995).

[CLIX] Marconi, M. and C. Nentwich, eds. "Constraint Language in XML", January 2004. **http://www.clixml.org/**.

[Cover] Cover, R. ed., "Markup Languages and (Non-) Hierarchies" in *The Cover Pages* hosted by OASIS. **http://xml.coverpages.org/hierarchies.html**

[dsdl3] ISO/IEC 19757-3: Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron

[EXSLT] Iacob, Emil, "The Extended XPath language (EXPath) for querying Concurrent Markup Hierarchies" **http://dblab.csr.uky.edu/~eiaco0/docs/expath/index.html**

[HORSE] DeRose, S.J., "Markup Overlap: A Review and a Horse" in *Extreme Markup Languages 2004®: Proceedings* **http://www.mulberrytech.com/Extreme/Proceedings/xml/2004/DeRose01/EML2004DeRose01.xml**

[LMNL] Tennison, J. and W. Piez. "The Layered Markup and Annotation Language (LMNL)" Late breaking paper presented at Extreme Markup, Montreal, 2002. **http://www.mulberrytech.com/Extreme/Proceedings/xml/2002/Tennison02/EML2002Tennison02.xml**

[MLW18] TEI Metalanguage & Syntax Working Committee, *Notes on SGML Solutions To Markup Problems*, 1992-04-16. **http://www.tei-c.org/Vault/ML/mlw18.txt**.

[SI] Bauman, S., introduction to Sperberg-McQueen, C. M. "Runways, product differentiation, snap-together joints, airplane glue, and switches that really switch", the closing keynote address at Extreme Markup Languages 2004 **http://www.mulberrytech.com/Extreme/Proceedings/xml/2004/Sperberg-McQueen02/EML2004Sperberg-McQueen02.xml#id2610845**

[TEIP4] Sperberg-McQueen, C. M. and Lou Burnard, eds. *Guidelines for Electronic Text Encoding and Interchange*, March 2002. **http://www.tei-c.org/P4X/**.

The Author

Syd Bauman
Brown University Women Writers Project

Syd Bauman is the Programmer/Analyst for the Women Writers Project, where he has worked since 1990, designing and maintaining a significantly extended TEI-conformant DTD for encoding early printed books. He also serves as the North American Editor of the Text Encoding Initiative Guidelines. He has an AB from Brown University in political science and has worked as an Emergency Medical Technician since 1983.

Extreme Markup Languages 2005®
Montréal, Québec, August 1-5, 2005
This paper was formatted from XML source via XSL
by Mulberry Technologies, Inc.