Published in: Witt, Andreas/Metzing, Dieter (eds.): Linguistic Modeling of Information and Markup Languages. Contributions to Language Technology. - Dordrecht: Springer, 2010. pp. 1-22. (Text, Speech and Language Technology 41)

Different Views on Markup

Distinguishing Levels and Layers

Daniela Goecke, Harald Lüngen, Dieter Metzing, Maik Stührenberg, and Andreas Witt

Abstract In this chapter, two different ways of grouping information represented in document markup are examined: *annotation levels*, referring to conceptual levels of description, and *annotation layers*, referring to the technical realisation of markup using e.g. document grammars. In many current XML annotation projects, multiple levels are integrated into one layer, often leading to the problem of having to deal with overlapping hierarchies. As a solution, we propose a framework for XML-based multiple, independent XML annotation layers for one text, based on an abstract representation of XML documents with logical predicates. Two realisations of the abstract representation are presented, a Prolog fact base format together with an application architecture, and a specification for XML native databases. We conclude with a discussion of projects that have currently adopted this framework.

Keywords Concurrent markup · XML · Annotations

1.1 Introduction

An annotated text document firstly contains the primary information, i.e. the text, and secondly, meta information, i.e. its annotation. Typically, the meta information structures the text according to a certain *view* on the text. Since language is a highly complex object of investigation, linguists often want to express more than a single view on a text. This chapter deals with problems that can arise when annotating multiple, different views on a text. We propose a practical and terminological distinction between aspects related to modelling issues and aspects related to the actual annotations. Such a distinction helps overcome problems that are often encountered when dealing with heterogeneously structured text.

This chapter is organised as follows: In Section 1.2, the terms *level* and *layer* are introduced, the relationship between levels, layers, and markup languages is

D. Goecke (✉)
Bielefeld University, Bielefeld, Germany
e-mail: daniela.goecke@uni-bielefeld.de

described in Section 1.3. In Section 1.4 approaches to XML-conformant annotation of multiple levels are presented. In Section 1.5 the need for an abstract representation of XML markup is motivated and two approaches are presented: In Section 1.5.1, inference tools for concurrent markup that have been developed during the first phase of the project *Sekimo – Secondary structuring of information*, are introduced. Here, an architecture for the integration of heterogeneous resources, which utilises the set of Sekimo tools, is described. In Section 1.5.2, it is demonstrated how text data with concurrent markup can be represented and stored in XML databases. The chapter concludes with a discussion of the relevance of the distinction between level and layer with references to selected publications.

1.2 Levels and Layers

Markup expresses additional information about text, e.g. authorship, or the part of speech of each word in it. Often it makes implicit information explicit, e.g. information encoded in the physical layout structure (such as paragraph and word boundaries).

The amount of information that is associated with text by means of markup has been constantly growing over the past few years. This development involved an organisation and structuring of the multitude of information. Structuring information by means of markup implies a conceptual process as well as a technical one. The conceptual and the technical process do not necessarily result in identical combinations of the pieces of information. We introduce the following terminology to clarify the two principally different ways of grouping units of information occurring in markup structures.

- *Annotation level* – referring to the conceptual level of information represented in markup
- *Annotation layer* – referring to the technical realisation of markup

For short, the term "level" refers to a model involving theoretical concepts e.g. of a research discipline. In linguistics, there are several subdisciplines which investigate different aspects and modalities of natural language and natural language description such as phonology, morphology, syntax, and semantics, which are often called the linguistic *levels of description*. Thus, an annotation unit (an XML element or attribute) will refer to one level while another annotation unit may refer to another level of linguistic description. In that sense, different *levels of markup* can be found in one annotated text. But even on one linguistic description level, different types of analyses can be represented which we still consider as different conceptual levels of markup. On the level of syntax, for example, alternative analyses according to different syntactic theories (e.g. Lexical Functional Grammar, Tree Adjoining Grammar, Categorial Grammar) may exist, and the annotation used to express any one of them refers to its own level of markup.

The term "layer", on the other hand, refers to the technical realisation of a modelling task. What it means exactly thus depends on the annotation system

employed. In transcription systems based on the annotation graph framework (Bird and Liberman 2001), for example, a layer corresponds to a single labeled path which spans the transcribed text. Typically, an annotation graph consists of several such paths, thus multiple layers can be realised in one annotation graph. Another example of a technical realisation is the use of different XML documents to store annotations of one text (Witt 2005), each XML file then corresponds to one annotation layer.

Distinguishing between levels and layers calls for an explication of the possible relations between the two. Does one annotation layer always correspond to one annotation level and vice versa? A closer look at the theoretical conceptualisations of annotations and their practical realisations reveals that levels and layers can stand in an $1:1$, $1:m$, $n:1$, or an $n:m$ relation (where $n, m > 1$). Examples of these relations are shown in Fig. 1.1.

In an $1:1$ relation, one conceptual level is mirrored by exactly one annotation layer, and each technical layer realises exactly one annotation level. Given an $1:1$ relation, one layer can be easily removed or exchanged without changing other layers. In an $1:m$ relation, units of one level are distributed over several layers, e.g. for the POS level, a different layer might be created for each word class. In an $n:1$ relation, two or more descriptive levels are integrated into one annotation layer, e.g. syntax and morphology annotations are often encoded in one XML document. An $n:m$ relation involves both the splitting and the mixture of conceptual levels and is seldom found in annotated corpora.

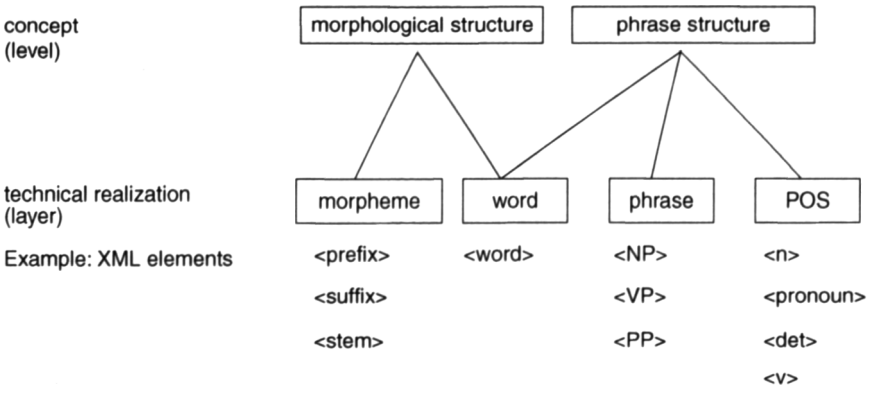

Fig. 1.1 Possible relations between level and layer

1.3 Levels, Layers, and Markup Languages

Markup Systems are formally defined. Consequently all markup languages are constrained. For SGML and XML, two restrictions are relevant when one aims at marking up information stemming from different linguistic levels of description:

- Firstly, they require that the elements used in a document instance must nest properly, i.e. the beginning and the end of a range of text annotated by one element must be contained in the same parent element.
- Secondly, one document instance can be associated with at most one document grammar.

The first restriction concerns the modelling aspect in a document instance and often results in the question of how to arrange different elements of a single layer in XML. Because a string annotated by one element must either be fully included in or totally separate from a string annotated by another element,[1] i.e. elements must not overlap, this problem is also called the *overlap problem.*

The second restriction often leads to difficulties when different levels are to be represented in one single XML document and thus addresses properties of markup systems. This problem concerns the use of document grammars, therefore we refer to it as the *document grammar problem.*

Ideally, an annotation level should be formally defined in one document grammar and a document grammar should only define one level of annotation in order to allow for a clear identification of the ontological status of the elements under consideration. Nevertheless, quite often a document grammar defines an annotation inventory for several levels. E. g., some versions of the DTDs for HTML allow for annotating both the text structure (paragraphs, headings, hypertext relations, etc.) and features that are intended to be used by the rendering engine when displaying the text (font-size, color, etc.). In the same way, one could try to merge different annotation vocabularies into one integrated document grammar. In order to define a common document grammar which permits the representation of units from several levels, the interrelation of the concepts of the different levels that one wants to express using markup has to be analysed. An integrated document grammar, however, is not recommended for a clear identification of the ontological status of elements.

Soon after the standardisation of XML the need for disentangling annotation vocabularies arose, since XML is not only used for annotating documents but also for as different purposes as programming (e.g. XSLT), the definition of document grammars (e.g. XSchema, Relax NG), and – through SVG – even for vector graphics.

The namespace standard was thus introduced to prefix element and attribute names for indicating the annotation level to which they belong. Hence, in principle, namespaces can be used to disentangle subsets of markup which belong to different linguistic levels of description. Other markup languages, e.g. LMNL (Cowan et al. 2005), provide a similar mechanism to refer to different levels. This approach is better suited than the construction of an integrated document grammar, since different annotation levels are associated with different namespace prefixes.

However, especially in XML, integrating different linguistic levels into one annotation often leads to the first problem, the overlap problem.

Despite the different nature of the two kinds of problems, the overlap problem and the document grammar problem, both often occur together. The reason for

this is obvious: When different levels are annotated and their document grammars have been designed independently, overlaps are bound to occur frequently. Hence, it might be helpful to try and solve both problems at once.

A solution to both problems was provided by XML's predecessor SGML. When the additional SGML feature "concur" is enabled, it is possible to independently annotate a text according to different document grammars. In recent years, new proposals to introduce this feature in XML, too, have been put forward (Hilbert et al. 2005, Schonefeld and Witt 2006). In this context the development of techniques for the definition of document grammars for concurrent markup has been started (Sperberg-McQueen 2006, Schonefeld and Witt 2006, Tennison 2007).

Moreover, some non-SGML-based markup languages have been designed which allow for the annotation of overlapping elements as well as for the annotation on different levels, e.g. TexMECS (Huitfeld and Sperberg-McQueen 2001) and LMNL (Cowan et al. 2005).

1.4 XML-Conformant Annotation of Multiple Levels

Since there is often a need for annotating structures which would result in overlapping XML elements, several solutions have been proposed. The two most frequently employed techniques to avoid overlap are firstly the so-called *milestone elements* and secondly the *fragmentation* of elements.

- *Milestone elements:* This term describes the use of empty elements to mark only the boundaries of a text range which would otherwise be contained in a non-empty element.[2] Since elements with text content can in principle also be used as empty elements, there is a variant of the milestone technique in which it is recommended to use all elements which originally were not intended to be empty (e.g. the element `<line>`) as milestone elements. Special attributes then indicate the status of these elements. This approach is known under at least three different names: Trojan Milestones, Horse and CLIX (cf. DeRose 2004).
- *Fragmentation:* A text sequence included in an element which would otherwise be affected by overlap is artificially split into several text sequences. Each of the resulting strings is included in a fragment element. An element which marks up a fragment content (e.g. a part of a `<sentence>` has attributes that indicate its special status and point to the preceding and the following element fragment(s).

The second problem, caused by the single document grammar constraint, can also be tackled by different strategies:

- Find a new integrated document grammar. In order to define a common document grammar which permits the representation of units from several levels, the interrelation of the concepts on the different levels that one wants to express using markup has to be analysed.

- Use different markup vocabularies for an annotation according to different document grammars. The namespace technique can be used to point to the document grammar from which an element or attribute is taken.

Standoff annotation, introduced by Thompson and McKelvie (1997) as a technique to split annotations from the textual data, can be used both to separate multiple linguistic levels from each other and to avoid overlapping structures. Standoff annotation has been primarily intended to present a solution to read-only textual data, copyright issues and overlapping structures. Instead of embedding markup in the text, markup and text are separated. Technically, standoff annotations are realised by referencing textual data through character offsets, e.g. an element <s> begins at the 17th and ends at the 42nd character of the text in a given file "transcript.txt". More often than raw text data a primary annotation is used as the target of a reference, e.g. a primary annotation annotates all the tokens of a text and introduces identifiers which can be used as link targets by the standoff annotation.

Standoff markup is often regarded as appropriate markup technique when linguists have to deal with complex annotations (see Pianta and Bentivogli 2004). For real annotation tasks, however, standoff annotations easily become quite complex and therefore they are only editable and even human readable with specialised software (Dipper et al. 2007).

An alternative way to solve the problems is the simplest solution to represent multiple, possibly overlapping hierarchies: The same text is annotated several times, and for each level of description, a separate markup layer is introduced, resulting in an 1:1 relation between markup levels and layers. Obviously, the creation of redundant copies of the textual base might be conceived as a drawback, because if changes to the textual base have to made, they have to be repeated for each annotation layer. On the other hand, this approach offers a range of possibilities for working with multiple description levels, especially in the field of humanities computing where it is the rule that one text is associated with manifold analyses and interpretations.

In this solution, care has to be taken that all layers contain the same *primary data* (i.e. the text which is to be annotated): When editing multiply annotated text, the identity of the primary data has to be maintained. If the text base is changed in one of the XML layers but not in another one, identity conflicts may arise, and a connection between the different XML layers could be no longer established. However, several editors have been developed that facilitate the editing of multiply annotated text, see Witt (2005) for a detailed description of the tools that have been implemented in the Sekimo project. Further editors that support multi-layered annotations are described in Schmidt (2004), Dipper and Götze (2005) and Schonefeld and Witt (2006).

Despite the need to ensure the identity of the primary data, the proposed solution offers the advantage that representations of different description levels may be developed independently of each other. A distributed development of XML layers allows experts to create markup independently of categories and structures from different levels. Furthermore, markup for additional levels may be added at any time without changing already existing XML layers.

1.5 Modelling, Representation, Annotation

In the previous section we have reviewed multiple annotations as a way to represent multiple, possibly overlapping hierarchies. To obtain a common view on the textual data and their multiple annotations, all markup is separated from its textual base. In order to split markup and text, an *abstract representation* of XML markup in logical predicates has been developed. Making a distinction between the textual base (primary data) and (possibly) multiple markup has been proposed in Witt (2002a,b), and has been further developed in Bayerl et al. (2003).

In the NITE project, the *Nite Object Model* (*NOM*, Carletta et al. 2003) has been defined, which is similar to DOM, the standardised W3C Document Object Model, used for the representation of HTML and XML documents. The most important difference between DOM and NOM is that a DOM corresponds to a tree with a single root node for the outermost element in an XML document and the leaf nodes for the textual content of the elements. The underlying data structure of a NOM, however, is not one tree with a single root, but several interconnected trees. Since each of their roots (indirectly) spans the same leaves, i.e. the textual data, we use the term *multi-rooted tree* to refer to this data structure.

In the Sekimo project, two approaches to the realization of an abstract representation have been developed: In Section 1.5.1 Prolog-based inference tools for concurrent markup are introduced. Section 1.5.2 describes the *Sekimo Generic Format* (*SGF*), an abstract XML-based representation format. In contrast to NOM, SGF uses a single-rooted tree but allows for several annotation levels corresponding to the same primary data using the formal model of a multi-rooted tree.

1.5.1 Multi-Layered XML Documents and Prolog

In the following, both a realization of an abstract representation as well as an application of multiple annotations is presented. The realization is done in terms of a Prolog fact base, the application of the Prolog fact base focuses on the analysis and combination of different layers.

The Prolog fact base format is an extension of previous work by Sperberg-McQueen et al. (2000). In this approach, all XML markup is translated into Prolog predicates that describe both the textual data as well as XML markup in terms of elements or attributes. The original format of Sperberg-McQueen et al. (2000) (see also Sperberg-McQueen et al. 2002a), which we consider as an intra-layer approach, has been extended in order to allow both for intra-layer analyses as well as for inter-layer analyses (see Witt et al. 2005 for details). In the original format, each XML element is translated into a Prolog fact with two arguments, e. g.

```
node([1,5,2],element(p)).
```

Attributes are translated into facts with three arguments, e. g.

```
attr( [1,5,2], id, implied).
```
(examples taken from Sperberg-McQueen et al. (2000), p. 219).

When representing multiple annotations, it must be recorded for each element or attribute to which layer it belongs. The multiple annotations can be connected using the identical textual content as a link between the separate layers. Therefore the original `node` and `attr` facts have been extended with arguments including layer information as well as information on the start and end position of the elements under consideration. In addition, the PCDATA (i.e. the underlying textual data) is translated into separate facts with three arguments (start position, end position, and the character at that position). These facts have been included for the purpose of reconverting the Prolog fact base into an XML representation. Thus, there are three types of Prolog predicates with their argument positions:

- Predicates for XML elements:
  `node` (*Layer, StartPosition, EndPosition, PositionDocumentTree, ElementName*) .
- Predicates for XML attributes:
  `attr` (*Layer, StartPosition, EndPosition, PositionDocumentTree, AttributeName, Attribute-Value*) .
- Predicates for PCDATA:
  `pcdata_node` (*StartPosition, EndPosition, Character*) .

A simple sentence like the one in Fig. 1.2 shall be used as an example to demonstrate the architecture of the application of multiple annotations. The textual content is represented character-wise by the multiple occurrence of the predicate `pcdata_node`, which has the arguments start position, end position and the character at that position as shown in Fig. 1.3. The offset of the character position is used on the one hand as a reference for different layers of markup and on the other hand in order to generate new XML output from the Prolog fact base.

```
00|01|02|03|04|05|06|07|08|09|10|11|12|13|14|15|16|17|18|19
  T  h  i  s     i  s     a     s  e  n  t  e  n  c  e  .
```

Fig. 1.2 A simple sentence

```
1 pcdata_node(0, 1, 'T').
2 pcdata_node(1, 2, 'h').
3 pcdata_node(2, 3, 'i').
4 pcdata_node(3, 4, 's').
5 pcdata_node(4, 5, '␣').
6 ...
7 pcdata_node(18, 19, '.').
```

Fig. 1.3 PCDATA nodes in the Prolog fact base

```
1  <s xml:lang="en">
2   <np>
3     <pron>This</pron>
4   </np>
5   <vp>
6     <v>is</v>
7     <np>
8       <det>a</det>
9       <n>sentence</n>
10    </np>
11  </vp>.
12 </s>
```

```
1  <syll>
2   <s>This</s>
3   <s>is</s>
4   <s>a</s>
5   <s>sen</s>
6   <s>tence</s>.
7  </syll>
```

Fig. 1.4 Formatted markup of POS/syntactic (*above*) and syllable (*below*) level of the same text segment

For the example sentence we have created XML markup for the levels *Syllable Structure* (Layer `syll`) and *POS/syntactic Information* (Layer `pos`) (see Fig. 1.4). In Fig. 1.5, the Prolog fact base of all nodes representing element instances and attributes from the layers `pos` and `syll` is shown. The word *This* (character positions 0–4, see lines 1–4 in Fig. 1.3), for example, is annotated as a pronoun on the layer `pos` and as a syllable on the layer `syll` (Fig. 1.4, above on line 3, and below on line 2). In the Prolog representation in Fig. 1.5, these elements can be found on lines 3 and 10.

```
1  node('pos.xml', 0, 19, [1], element('s')).
2  node('pos.xml', 0, 4, [1, 1], element('np')).
3  node('pos.xml', 0, 4, [1, 1, 1], element('pron')).
4  node('pos.xml', 5, 18, [1, 2], element('vp')).
5  node('pos.xml', 5, 7, [1, 2, 1], element('v')).
6  node('pos.xml', 8, 18, [1, 2, 2], element('np')).
7  node('pos.xml', 8, 9, [1, 2, 2, 1], element('det')).
8  node('pos.xml', 10, 18, [1, 2, 2, 2], element('n')).
9  node('syll.xml', 0, 19, [1], element('syll')).
10 node('syll.xml', 0, 4, [1, 1], element('s')).
11 node('syll.xml', 5, 7, [1, 2], element('s')).
12 node('syll.xml', 8, 9, [1, 3], element('s')).
13 node('syll.xml', 10, 13, [1, 4], element('s')).
14 node('syll.xml', 13, 18, [1, 5], element('s')).
15
16 attr('pos.xml', 0, 19, [1], 'xml:lang', 'en').
```

Fig. 1.5 Element and Attribute information in the Prolog fact base

The Prolog representation of the XML markup can be used in order to query the corpus, e.g. for an analysis of the relation between elements from different layers. Taking the start and end positions of two (or more) elements from separate annotation layers into account, different relations between these elements can be identified, e.g. the element `np` from layer `pos` *includes* the element `s` from layer `syll`. The information on relations between elements from different layers is important in order to create an annotation *merger*, i.e. for a *markup unification* of different annotation layers. The process of markup unification is described in detail in Witt et al. (2005). Basically, the merger contains all markup from the input layers and the interrelationship of elements determines the hierarchical structure of the intended merger. Besides unifying different annotation layers by markup unification to *merge* different annotation layers it is also possible to *split* a single annotation layer into different partitions. However – unlike merging – splitting is already possible with standard XML tools (e.g. XSLT or XQuery). Thus, the Prolog fact base format has mainly been extended for the purpose of markup unification.

The Prolog fact base representing the merger of the facts in Fig. 1.5 is shown in Fig. 1.6; an XML output file generated from the merged fact base and the textual content is shown in Fig. 1.7. In order to avoid merging conflicts caused by identically named elements on different layers, each element name is provided with a prefix indicating its original layer.

An XML layer is the realisation of a data model, i.e. of a conceptual level. When having different layers that describe different aspects of the data, an analysis of the data should give answers to the question of how the different layers interact, i.e. which relations hold between the markup elements. The possible relationships between elements from two layers have been arranged and classified in Durusau and O'Donnell (2002). They are of central interest for the merging process as they give information as to what the hierarchical structure of the merged XML annotation should look like. In the overview given in Fig. 1.8, some of the relationships given in Durusau and O'Donnell (2002) have been collapsed and renamed for the illustration of our approach.

```
1  node('output', 0, 19, [1], element('pos.xml_s')).
2  attr('output', 0, 19, [1], 'xml:lang', 'en').
3  node('output', 0, 19, [1, 1], element('syll.xml_syll')).
4  node('output', 0, 4, [1, 1, 1], element('pos.xml_np')).
5  node('output', 0, 4, [1, 1, 1, 1], element('pos.xml_pron')).
6  node('output', 0, 4, [1, 1, 1, 1, 1], element('syll.xml_s')).
7  node('output', 5, 18, [1, 1, 2], element('pos.xml_vp')).
8  node('output', 5, 7, [1, 1, 2, 1], element('pos.xml_v')).
9  node('output', 5, 7, [1, 1, 2, 1, 1], element('syll.xml_s')).
10 node('output', 8, 18, [1, 1, 2, 2], element('pos.xml_np')).
11 node('output', 8, 9, [1, 1, 2, 2, 1], element('pos.xml_det')).
12 node('output', 8, 9, [1, 1, 2, 2, 1, 1], element('syll.xml_s')).
13 node('output', 10, 18, [1, 1, 2, 2, 2], element('pos.xml_n')).
14 node('output', 10, 13, [1, 1, 2, 2, 2, 1], element('syll.xml_s')).
15 node('output', 13, 18, [1, 1, 2, 2, 2, 2], element('syll.xml_s')).
```

Fig. 1.6 The merged Prolog fact base

```
<pos_s xml:lang="en">
 <syll_syll>
  <pos_np>
   <pos_pron>
    <syll_s>This</syll_s>
   </pos_pron>
  </pos_np>
  <pos_vp>
   <pos_v>
    <syll_s>is</syll_s>
   </pos_v>
   <pos_np>
    <pos_det>
     <syll_s>a</syll_s>
    </pos_det>
    <pos_n>
     <syll_s>sen</syll_s>
     <syll_s>tence</syll_s>
    </pos_n>
   </pos_np>
  </pos_vp>.
 </syll_syll>
</pos_s>
```

Fig. 1.7 The merged output XML file

Using the tools described in Witt et al. (2005) interrelationships between different annotation layers can be analysed, and also two layers can be merged into a single XML document via the process of markup unification. Bayerl et al. (2003) describe the inter-layer analysis for three XML layers: the text's document structure on the one hand and the XML markup of two kinds of semantic levels on the other hand (the thematic level, i.e. topics in the text world that the article is about, and

```
start point identity: <a>.....................................</a>
                      <b>...........</b>

end point identity:   <a>.....................................</a>
                                              <b>...........</b>

inclusion:            <a>.....................................</a>
                                    <b>...........</b>

identity:             <a>.....................................</a>
                      <b>.....................................</b>

overlap:              <a>...................</a>
                                    <b>.....................</b>

independent elements: <a>..........</a>
                                              <b>.............</b>
```

Fig. 1.8 Possible relations between pairs of element instances (cf. Durusau and O'Donnell 2002)

the functional or rhetorical level). Goecke and Witt (2006) describe the inter-layer analysis of a text's document structure and the anaphoric relations that hold within the text. Apart from analysing elements from different layers, elements within one layer (intra-layer analysis) may be compared, too. In case of an n:1 relation between n levels and one layer, for example, an analysis of the relations between elements might help to split the layer into several layers, i.e. one for each level.

1.5.2 Multi-Layered Documents and XML-Databases

A different way of viewing and working with multi-layered documents is available when using an XML-based abstract representation format in connection with a native XML database. An XML-based format permits the application of several XML-related tools such as XPath, XSLT or XQuery. Dealing with multi-layered documents, however, bears the problem of overlapping structures which cannot be handled in plain XML (cf. Section 1.3). For this reason, we propose the abstract XML representation format *SGF* (*Sekimo Generic Format*) for multi-layered XML documents to be stored in a native XML database.[3] An overview of the architecture is shown in Fig. 1.9.

SGF is similar to the Prolog representation (cf. Section 1.5.1) in that the same mechanism for referencing characters and whitespaces is used: the offset position of each character. These are used to span sequences of character data over the text, which can be referred to in a second step as tokens in the annotation process. The root element `corpus` contains the element `corpusData` with its required attribute `xml:id`, the value of which is a unique identifier of the given input text, and a `type` attribute, determining the type of corpus data (textual or multimodal).

The different annotation layers appear in a structured fashion as child elements of the `annotation` element. Each layer belongs to a namespace indicating it. A mandatory `primaryData` element is used to store and structure the primary textual data. The abstract representation of the example sentence shown in Fig. 1.2 can be seen in Fig. 1.10.

The primary layer is flat in hierarchical terms. The `primaryData` element contains the complete whitespace-normalised textual input including whitespace and punctuation characters (for shorter texts) or a reference to a file in which the whitespace-normalised textual input is stored in (via the `location` element – not shown in the example). In the latter case an optional element `checksum` can be used to protect the integrity of the input data, providing both the computed checksum and the algorithm used. The element `segments` is used to store several `segment` elements, each of which contains an identifier, the segment type (in this case character) and the segment span (ranging from the `start` to the `end` attribute, referring to the offset of the first and last character of the string. Additional occurrences of the empty element `segment` can be used to define the position of whitespace character data, including a character reference. Relying on character offsets allows for dealing with different possible tokenisations (e.g. output from different text analysis tools).

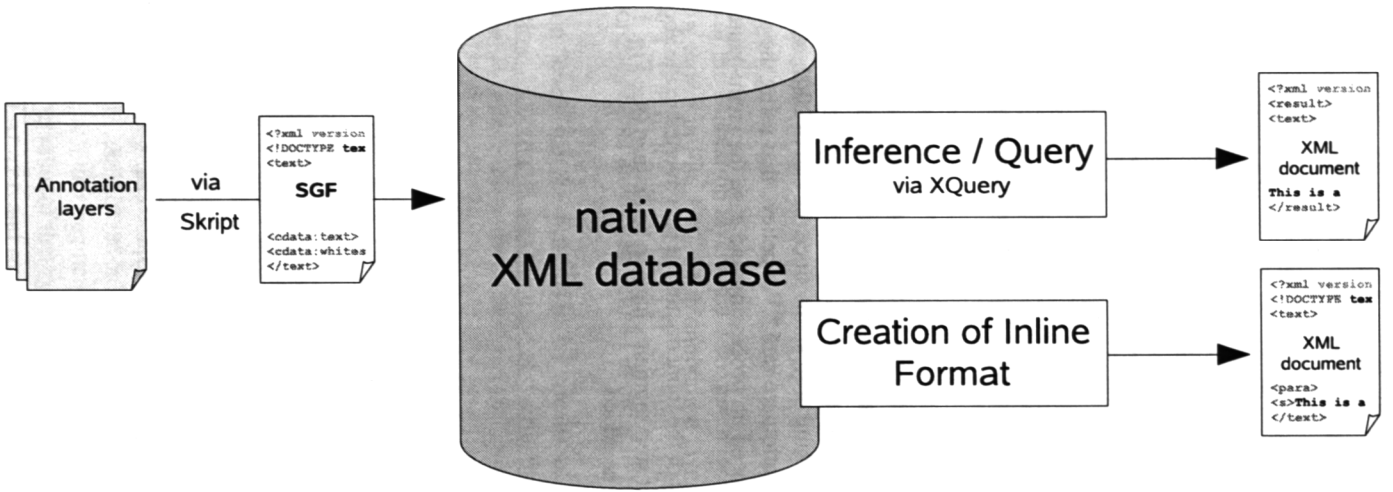

Fig. 1.9 An overview of the architecture

```
<base:corpus xmlns="http://www.text-technology.de/sekimo"
 xmlns:base="http://www.text-technology.de/sekimo">
 <base:corpusData xml:id="c1" type="text">
  <base:meta>
  <!-- [...] -->
  </base:meta>
  <base:primaryData start="0" end="19" xml:lang="en">
   <base:textualContent>This is a sentence.</textualContent>
  </base:primaryData>
 </base:corpusData>
 <base:corpusData xml:id="c2" type="text">
 <!-- [...] -->
 </base:corpusData>
</base:corpus>
```

Fig. 1.10 The root element of the abstract XML representation format

Instead of using the `start` and `end` attributes, a use of the XPointer *xpointer() Scheme* (and especially the *string-range* function) would have been possible (e.g. as in the PAULA format, cf. Dipper et al. 2007). However, the *xpointer() Scheme* has been pending in the working draft status since the end of 2002, and implementations of the string-range function are rare.[4] We believe that using the simpler concept of two attributes could speed up processing and ease the (semi-)automatic annotation process.

The textual content of the input document is converted to the above described primary layer. In the next step segments have to be defined (see Fig. 1.11). Afterwards, annotation layers can be added.

```
<base:corpus xmlns="http://www.text-technology.de/sekimo"
 xmlns:base="http://www.text-technology.de/sekimo">
 <base:corpusData xml:id="c1" type="text">
  <base:meta>
  <!-- [...] -->
  </base:meta>
  <base:primaryData start="0" end="19" xml:lang="en">
   <base:textualContent>This is a sentence.</textualContent>
  </base:primaryData>
  <base:segments>
   <base:segment xml:id="seg1" type="char" start="0" end="19"/>
   <base:segment xml:id="seg2" type="char" start="0" end="4"/>
   <base:segment xml:id="seg4" type="char" start="5" end="18"/>
   <base:segment xml:id="seg5" type="char" start="5" end="7"/>
   <base:segment xml:id="seg7" type="char" start="8" end="18"/>
   <base:segment xml:id="seg8" type="char" start="8" end="9"/>
   <base:segment xml:id="seg10" type="char" start="10" end="18"/>
  </base:segments>
 </base:corpusData>
</base:corpus>
```

Fig. 1.11 Adding segments in the instance document

In case that already existing inline annotation layers shall be used, the following steps have to be done for conversion:

1. A namespace referring to a converted representation of the schema of the annotation level is also added, following the notation http://www.text-technology.de/sekimo/[layer], and all elements of the imported layer are prefixed with the corresponding namespace prefix. If there are multiple annotations referring to the same schema (e.g. in case of an analysis of intra-layer relations), different namespace prefixes for the same namespace shall be used.[5] For this reason we refer to the prefix as *annotation layer prefix* rather than *namespace prefix*.
2. An optional `meta` element can be used to describe the annotation layer (e. g. its origin, the annotator, etc.). Apart from the `description` element, other elements derived from different namespaces are allowed as children of the *meta* element.
3. The attribute `segment` of the primary layer is added to each element.
4. Elements with a `PCDATA` content model are converted to empty elements, mixed content elements are converted to container elements. This is possible because all character content is already stored in the `primaryData` element of the primary layer or in another file.

A conversion of the annotation layers that were given in Fig. 1.4 would result in the representation shown in Fig. 1.12. Each annotation layer is stored in a `layer` element which is a child of the `annotation` element of the primary layer. Note, that only two more segments have been defined in order to represent the additional syllables layer.

The annotation levels can be prioritised by means of the optional attribute `priority` to allow for correct nesting in case of overlapping structures.

Keeping the explicit structural information of the non-terminal elements is a benefit provided by the XML-based representation format in contrast with other possible representation formats, allowing validation of annotation layers with only slightly changed version of the original document grammars (including cross-layer validation). As a second advantage, the XML-based representation format allows for storing meta-data such as the language of a sentence. Nontextual elements like images and figures can be embedded in special elements (e.g. `<nontext type="image" src="image.jpg"/>`). Since the scope of this work is the annotation of *textual* documents, the treatment of non-textual elements is not pursued further here. However, this framework can be used for the annotation of multimodal corpora, too, by using timecode or frame positions as values for the `start` and `end` attributes and using `multimodal` as value for the `type` attribute of the `corpusData` element. In addition, it should be mentioned that constructing larger segments by referencing to other segments is possible as well (including disjoint segments). In this case the value of the `type` attribute of the `segment` element is set to `seg` and instead of `start` and `end` attributes a `segments` attribute is used (containing the identity references of the corresponding segments).

```
<base:corpus xmlns="http://www.text-technology.de/sekimo"
 xmlns:base="http://www.text-technology.de/sekimo">
 <base:corpusData xml:id="c1" type="text">
  <base:primaryData start="0" end="19" xml:lang="en">
   <base:textualContent>This is a sentence.</textualContent>
  </base:primaryData>
  <base:segments>
   <base:segment xml:id="seg1" type="char" start="0" end="19"/>
   <base:segment xml:id="seg2" type="char" start="0" end="4"/>
   <base:segment xml:id="seg4" type="char" start="5" end="18"/>
   <base:segment xml:id="seg5" type="char" start="5" end="7"/>
   <base:segment xml:id="seg7" type="char" start="8" end="18"/>
   <base:segment xml:id="seg8" type="char" start="8" end="9"/>
   <base:segment xml:id="seg10" type="char" start="10" end="18"/>
   <base:segment xml:id="seg11" type="char" start="10" end="13"/>
   <base:segment xml:id="seg12" type="char" start="13" end="18"/>
  </base:segments>
  <base:annotation>
   <base:level xml:id="pos" priority="0">
    <base:layer xmlns:pos="http://www.text-technology.de/pos"
     xsi:schemaLocation="http://www.text-technology.de/pos␣p.xsd">
     <pos:s base:segment="seg1">
      <pos:np base:segment="seg2">
       <pos:pron base:segment="seg2"/>
      </pos:np>
      <pos:vp base:segment="seg4">
       <pos:v base:segment="seg5"/>
       <pos:np base:segment="seg7">
        <pos:det base:segment="seg8"/>
        <pos:n base:segment="seg10"/>
       </pos:np>
      </pos:vp>
     </pos:s>
    </base:layer>
   </base:level>
  </base:annotation>
  <base:annotation>
   <base:level xml:id="syll" priority="0">
    <base:layer xmlns:syll="http://www.text-technology.de/syll"
     xsi:schemaLocation="http://www.text-technology.de/syll␣s.xsd">
     <syll:syll base:segment="seg1">
      <syll:s base:segment="seg2"/>
      <syll:s base:segment="seg5"/>
      <syll:s base:segment="seg8"/>
      <syll:s base:segment="seg11"/>
      <syll:s base:segment="seg12"/>
     </syll:syll>
    </base:layer>
   </base:level>
  </base:annotation>
 </base:corpusData>
</base:corpus>
```

Fig. 1.12 The converted SGF representation of the two annotation layers

A successor of SGF, called XStandoff, is already available as development release (Stührenberg and Jettka 2009). For sustainability reasons, the current version of the Sekimo Generic Format has undergone a feature-freeze in that way that the format and its corresponding tools are considered as stable.

Storing the multi-layered documents in a native XML database allows for using query and analysis mechanisms which are similar to those provided for the Prolog fact base. Most native XML database systems support XPath and at least a subset of XQuery and some sort of update mechanism (e.g. XUpdate as defined by the XML:DB Initiative[6]) or the upcoming XQuery Update Facility which is capable of processing and updating instances of the XQuery/XPath Data Model (XDM) (Chamberlin et al. 2008). The W3C is working on the extension of XQuery 1.0 with full-text search capabilities (Amer-Yahia et al. 2006).

Tests with the Open Source native XML databases eXist,[7] the Berkeley DB XML[8] and the commercial but freely available IBM DB2 Express-C[9] showed a good performance. Mechanisms like the above mentioned XUpdate or the upcoming XQuery Update Facility allow for updating the instance files. By now the use of a native XML database allows for easy intra- and inter-layer analysis. Having a powerful query language like XQuery allows for quite complex analyses. For a more detailed description of the Sekimo Generic Format and performance measures on a per-file basis in a real-world application (cf. Stührenberg and Goecke 2008).

1.6 Conclusions

Information contained in textual markup can be grouped according to two distinct principles: On the one hand, (annotation) *level* refers to a conceptual level of information such as the phonological, syntactic and semantic levels of description familiar from linguistics. (Annotation) *layer*, on the other hand, refers to the technical realisation of markup, e.g. one document grammar or one labelled path in an annotation graph defines one annotation layer. The ideal case in text-technological information modelling is that of a 1:1 correspondence between levels and layers. However, due to the single document grammar restriction for SGML-based markup languages, linguistic levels are often integrated into one annotation layer, resulting in a need to solve the so-called overlap problem. We showed that previous solutions to the overlap problem exhibit some drawbacks, so that we vote for a framework of XML-based multi-layer annotation where the same text is annotated several times, and a separate markup layer is introduced for each description level. That way, experts can create and maintain markup for their description levels independently of the structures defined for the same text by the experts for a different description level. Markup for additional levels can be added without having to make changes to existing markup layers. The use of special editors guarantees the identity of the primary data of each layer. An abstract representation in terms of logical predicates defines a common view on multiply annotated layers of one text. We presented two realisations of such abstract representations,

firstly a Prolog fact base format, and secondly, a realisation that makes use of existing XML database facilities. For the Prolog fact base, we presented an application architecture in which multiply XML-annotated documents can be unified, and relations between element types in annotation layers can be inferred. We also showed that utilising XML databases for a realisation of the abstract representation format SGF, XML standards and tools such as XPath, XQuery and XUpdate can be used conveniently for retrieving and updating annotations within this framework.

Our framework of XML-based multiple annotations is currently applied in several text-technological projects for the automatic linguistic analysis of XML-annotated texts.

The annotations of the different levels of discourse structure described in Lüngen et al. (in this volume), for example, have been annotated separately in a corpus of scientific journal articles. The discourse parser described is realised in Prolog and takes the Prolog fact base derived from the multiple annotations of one document as its input and adds the independent annotation layer of rhetorical structure as its output.

Stührenberg et al. (2006) apply the framework within the context of anaphora resolution. Necessary resources for the resolution process (e.g. morphology, syntax, logical document structure, ontological knowledge) have been annotated separately and the resulting annotation layers are combined in the representation format. On the basis of the combined XML data, feature vectors have been extracted that serve as input for corpus analyses and the resolution process.

Both, the Sekimo Generic Format and its currently developed successor, XStandoff, are freely available under the LGPL 3 license including the accompanied tools, other interested parties and projects are invited to use and enhance this framework.[10]

For a broader discussion of the issue of sustainability of multiply structured linguistic data see Stührenberg et al. (2008), Rehm et al. (2009), and Witt et al. (2009).

Notes

1. This is the reason for stating that SGML or XML documents form an "ordered hierarchy of content objects" (OHCO).
2. E.g. instead of using the element `<line>` such that it contains the text of a single print line, two empty elements `<lb/>` could be employed to annotate the line breaks before and after a line.
3. Storing on a per-file basis or in a relational database is possible as well.
4. Cf. http://www.w3.org/XML/2000/09/LinkingImplementations.html
5. It would also be possible to declare multiple namespaces as an ad hoc solution, but this would be against the intention of the XML namespace standard.
6. Cf. http://xmldb-org.sourceforge.net/xupdate/
7. Cf. http://www.exist-db.org
8. Cf. http://www.sleepycat.com/products/bdbxml.html
9. Cf. http://www.ibm.com/software/data/db2/express/
10. Cf. http://www.xstandoff.net for further details.

References

Amer-Yahia, S., Botev, C., Buxton, S., Case, P., Doerre, J., Holstege, M., McBeath, D., Rys, M., and Shanmugasundaram, J. (eds.) (2006). *XQuery 1.0 and XPath 2.0 Full-Text.* W3C Candidate Recommendation 16 May 2008 http://www.w3.org/TR/2008/CR-xpath-full-text-10-20080516/

Barnard, D., Burnard, L., Gaspart, J., Price, L. A., Sperberg-McQueen, C. M., and Varile, G. B. (1995). *Hierarchical encoding of text: technical problems and SGML solutions.* In: Computers and the Humanities. 29:211–231.

Bayerl, P. S., Goecke, D., Lüngen, H., and Witt, A. (2003). *Methods for the semantic analysis of document markup.* In: Roisin, C., E. Munson and C. Vanoirbeek (eds.). Proceedings of the 3rd ACM Symposium on Document Engineering (DocEng), Grenoble: 161–170.

Bird, S. and Liberman, M. (2001). *A formal framework for linguistic annotation.* In: Speech Communication 33(1,2):23–60.

Carletta, J., Kilgour, J., O'Donnnell, T., Evert, S., and Voormann, H. (2003). *The NITE Object Model Library for Handling Structured Linguistic Annotation on Multimodal Data Sets.* In: Proceedings of the EACL Workshop on Language Technology and the Semantic Web (3rd Workshop on NLP and XML, NLPXML-2003). Budapest.

Chamberlin, D., Florescu, D., and Robie, J. (eds.) (2008). *XQuery Update Facility.* W3C Candidate Recommendation 1 August 2008 http://www.w3.org/TR/2008/CR-xquery-update-10-20080801/

Clark, H. (1977) *Bridging.* In: Johnson-Laird P.C. and P.N. Wason (eds.), Thinking: Readings in Cognitive Science, Cambridge University Press, Cambridge: 411–420.

Cowan, J., Tennison, J., and Piez, W. *LMNL update.* In: Proceedings of Extreme Markup Languages 2006, Montreal.

Czmiel, Al. (2004) *XML for Overlapping Structures (XfOS) Using a Non XML Data Model.* In: Proceedings of the Joint Conference of the ALLC and ACH, Göteborg, Sweden.

DeRose, S. *Markup overlap: a review and a horse.* In: Proceedings of Extreme Markup Languages 2004, Montreal.

DeRose, S. J., Durand, D. G., Mylonas, E., and Renear, A. (1990). *What is text, really?* Journal of Computing in Higher Education, ACM Press, 1:3–26.

Dipper, S. and Götze, M. (2005). *Accessing heterogeneous linguistic data – generic XML-based representation and flexible visualization.* In: Proceedings of the 2nd Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, Poznan:206–210.

Dipper, S., Götze, M., Küssner, U., and Stede, M. (2007). *Representing and querying standoff XML.* In: G. Rehm, A. Witt, and L. Lemnitzer (eds.), Data Structures for Linguistic Resources and Applications. Proceedings of the Biennial GLDV Conference 2007, Gunter Narr Verlag, Tübingen:337–346.

Durusau, P. and O'Donnell, M. B. (2002). *Concurrent markup for XML documents.* In: Proceedings of XML Europe 2002.

Goecke, D. and Witt, A. (2006). *Exploiting logical document structure for anaphora resolution.* In: Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006). Genoa, Italy.

Hilbert, M., Schonefeld, O., and Witt, A. *Making CONCUR work.* In: Proceedings of Extreme Markup Languages 2005, Montreal.

Huitfeld, C. and Sperberg-McQueen, C. M. (2001). *TexMECS: An experimental markup metalanguage for complex documents.* http://xml.coverpages.org/MLCD-texmecs20010510.html.

Karttunen, L. (1976). *Discourse referents.* In: Syntax and Semantics: Notes from the Linguistic Underground, 7:363–385.

Mitkov, R. (2002). *Anaphora resolution.* Longman, London.

Pianta, E. and Bentivogli, L. (2004). *Annotating discontinuous structures in XML: the multiword case.* In: Proceedings of the LREC-Satellite Workshop on XML-based Richly Annotated Corpora. Lisbon 2004.

Piez, W. (2004) *Half-steps toward LMNL.* In: Proceedings of Extreme Markup Languages 2004, Montreal.

Rehm, G., Schonefeld, O., Witt, A., Hinrichs, E., and Reis, M. (2009). *Sustainability of annotated resources in linguistics: a web-platform for exploring, querying and distributing linguistic corpora and other resources.* In: Literary and Linguistic Computing 2009 24(2):193–210.

Renear, A., Mylonas, E., and Durand, D. (1996). *Refining our notion of what text really is: The problem of overlapping hierarchies.* In: N. Ide and S. Hockey (eds.) Research in Humanities Computing. Selected Papers from the ALLC/ACH Conference, Christ Church, Oxford, April 1992, 4:263–280.

Schonefeld, O. and Witt, A. *Towards validation of concurrent markup.* In: Proceedings of Extreme Markup Languages 2006, Montreal.

Schmidt, T. (2004). *EXMARaLDA – ein System zur computergestützten Diskurstranskription.* In: Mehler, A. and Lobin, H. (eds.) Automatische Textanalyse: Systeme und Methoden zur Annotation und Analyse natürlichsprachlicher Texte. Wiesbaden: VS Verlag:203–218.

Simons, G., Lewis, W., Farrar, S., Langendoen, T., Fitzsimons, B., and Gonzalez, H. (2004). *The semantics of markup: mapping legacy markup schemas to a common semantics.* In: Proceedings of the ACL 2004 Workshop on RDF/RDFS and OWL in Language Technology (NLP XML-2004), Barcelona.

Sperberg-McQueen, C. M., Huitfeldt, C., and Renear, A. (2002). *Meaning and interpretation of markup.* In: Markup Languages: Theory & Practice 2.3 (2000):215–234.

Sperberg-McQueen, C. M., Dubin, D., Huitfeldt, C., and Renear, A. (2002). *Drawing inferences on the basis of markup.* In: Proceedings of Extreme Markup Languages 2002, Montreal.

Sperberg-McQueen, C. M. and Burnard, L. (eds.) (2002). *TEI P4: guidelines for electronic text encoding and interchange.* Text Encoding Initiative Consortium. XML Version: Oxford, Providence, Charlottesville, Bergen.

Sperberg-McQueen, C. M. (2006). *Rabbit/duck grammars: a validation method for overlapping structures.* In: Proceedings of Extreme Markup Languages 2006, Montreal.

Stührenberg, M., Witt, A., Goecke, D., Metzing, D., and Schonefeld, O. (2006). *Multidimensional markup and heterogeneous linguistic resources.* In: Proceedings of the 5th Workshop on NLP and XML (NLPXML-2006): Multi-Dimensional Markup in Natural Language Processing. April 4, 2006. Trento, Italy.

Stührenberg, M. and Goecke, D. (2008). *SGF – an integrated model for multiple annotations and its application in a linguistic domain.* In: Proceedings of Balisage: The Markup Conference 2008, Montreal.

Stührenberg, M., Kühnberger, K.-U., Lüngen, H., Mehler, A., Metzing, D., and Mönnich, U. (2008) *Sustainability of text-technological resources.* In: Proceedings of the LREC 2008 Workshop Sustainability of Language Resources and Tools for Natural Language Processing, Marrakech, Morocco:33–40.

Stührenberg, M. and Jettka, D. (2009). *A toolkit for multi-dimensional markup – the development of SGF to XStandoff.* In: Proceedings of Balisage: The Markup Conference 2009, Montreal.

Strube, M. and Müller, C. (2003). *A machine learning approach to pronoun resolution in spoken dialogue.* ACL 03.

Tennison, J. (2007). *Creole: validating overlapping markup.* In: Proceedings of XTech 2007, Paris.

Thompson, H. S. and McKelvie, D. (1997). *Hyperlink semantics for standoff markup of read-only documents.* In: Proceedings of SGML Europe '97, Barcelona.

Trippel, T. Sasaki, F., Hell, B., and Gibbon, D. (2003). *Acquiring lexical information from multilevel temporal annotations.* 8th European Conference on Speech Communication and Technology.

Vieira, R. and Teufel, S. (1997). *Towards resolution of bridging descriptions.* In: Proceedings of ACL/EACL, Madrid.

Webber, B. L. (1988) *Discourse deixis: reference to discourse segments.* In: Proceedings of the ACL:113–122.

Witt, A. (2002). *Meaning and interpretation of concurrent markup.* In: Proceedings of the Joint Conference of the ALLC and ACH, Tübingen, Germany.

Witt, A. (2002). *Multiple Informationsstrukturierung mit Auszeichnungssprachen. XML-basierte Methoden und deren Nutzen für die Sprachtechnologie.* Phd Thesis, Universität Bielefeld.

Witt, A. (2005). *Multiple hierarchies: new aspects of an old solution.* Re-published in: Dipper, S., M. Götze, and M. Stede (eds.) Heterogeneity in Focus: Creating and Using Linguistic Databases. Volume 2 of Interdisciplinary Studies on Information Structure (ISIS), Working Papers of the SFB 632. Universitätsverlag Potsdam, Germany.

Witt, A., Goecke, D., Sasaki, F., and Lüngen, H. (2005). *Unification of XML documents with concurrent markup.* Literary and Linguistic Computing 2005 20(1):103–116.

Witt, A., Rehm, G., Hinrichs, E., Lehmberg, T., and Stegmann, J. (2009). *SusTEInability of Linguistic Resources through Feature Structures.* In: Literary and Linguistic Computing 2009; doi: 10.1093/llc/fqp024.