Published in: Literary and Linguistic Computing vol. 20 (2005) no. 1, pp. 103-116.

Unification of XML Documents with Concurrent Markup

Andreas Witt, Daniela Goecke, and Felix Sasaki
Bielefeld University, Bielefeld

Harald Lüngen
Justus-Liebig-Universität Gießen

Abstract

An approach to the unification of XML (Extensible Markup Language) documents with identical textual content and concurrent markup in the framework of XML-based multi-layer annotation is introduced. A Prolog program allows the possible relationships between element instances on two annotation layers that share PCDATA to be explored and also the computing of a target node hierarchy for a well-formed, merged XML document. Special attention is paid to identity conflicts between element instances, for which a default solution that takes into account metarelations that hold between element types on the different annotation layers is provided. In addition, rules can be specified by a user to prescribe how identity conflicts should be solved for certain element types.

1 Introduction

There is a growing need to annotate a text or a whole text corpus according to multiple information levels, especially in the field of linguistics. Language data are provided with SGML-based markup encoding phonological, morphological, syntactic, semantic, and pragmatic structure analyses. It might also be desirable to annotate alternative structures for one description level, such as syntactic structures according to the LFG (Lexical Functional Grammar) versus HPSG (Head-Driven Phrase Structure Grammar) grammatical frameworks. Parallel to this development, more and more documents published on the world wide web are provided with different kinds of metadata in the form of XML (Extensible Markup Language) markup, following the Semantic Web Initiative. When more annotation levels are added, the need to represent *multiple hierarchies* becomes obvious. The annotation of multiple hierarchies with SGML-based markup systems is still one of the fundamental problems of text-technological research. Formally, using SGML-based markup, exactly one element hierarchy can be represented. This is a consequence of the fact that

Correspondence:
Harald Lüngen,
Justus-Liebig-Universität Gießen,
FB05 – Angewandte
Sprachwissenschaft
und Computerlinguistik,
Otto-Behaghel-Str. 10 D,
D-35394 Gießen.
E-mail:
luengen@uni-giessen.de

SGML was developed with a bias towards applications in printed publishing, where text was considered as an OHCO (ordered hierarchy of content objects, cf. DeRose *et al.*, 1990). Sometimes, the single hierarchy restriction is not perceived as a drawback because annotations with concepts from different information levels can often be integrated in a single hierarchy. But often it will be impossible to combine annotations on many layers in a single hierarchy, because ranges of text marked up by SGML or XML must not overlap.

In our approach to representing multiple hierarchies, each annotation layer is represented in a separate and independent well-formed XML document, i.e. the same textual content is annotated several times. The identical textual content then serves as the link between the different annotation layers and is used to make explicit the relations holding between them. If an integrated view is desired, the different annotation layers can be *unified*, i.e. merged into one well-formed XML document containing the annotation from multiple layers plus the textual content. The remainder of this article deals with introducing our framework of XML-based multilayer annotation and an implemented prototype system that allows for the unification of separately annotated text documents.

2 XML-based Multilayer Annotation

Several solutions to the problem of representing multiple, possibly overlapping hierarchies have been proposed previously, e.g. in chapter 31 of the TEI-Guidelines (Sperberg-McQueen and Burnard, 1994), and in Barnard *et al.* (1995). Moreover, there are approaches in the context of non-SGML-based markup languages (Huitfeldt and Sperberg-McQueen, 2001; Tennison, 2002).

The solution we have adopted is XML-based and avoids the drawbacks of the approaches above (discussed in Witt, 2004). It is summarized well in the following quotation:

> 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 (Sperberg-McQueen and Burnard, 1994, p. 94).

One advantage of annotating the same data multiply but separately is that the modelling of information on one level is not dependent on (the existence of) a primary modelling level, as in the standoff annotation technique suggested by Thompson and McKelvie (1997). Our strategy results in an information modelling where each layer can be viewed separately and new layers can be added at any time. Each document instance containing one annotation layer uses its own Document Type Definition (DTD) (or schema), i.e. annotation formats can be maintained independently. Existing layer-specific annotation and transformation tools can be employed, or new ones can be implemented.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE article SYSTEM "docsimple.dtd">
<article>
  <!--snip-->
  <sect1>
    <title>3 Erfassungmodus der Stellenanzeigen</title>
    <para> Bei der Erfassung der Stellenanzeigen wurden sowohl überregionale
       und regionale Zeitungen als auch Fachzeitschriften berücksichtigt. Zu
       den bearbeiteten überregionalen Zeitungen zählen:
    </para>
    <itemizedlist>
            <listitem>
                    <para> * Frankfurter Rundschau</para>
            </listitem>
            <listitem>
                    <para> * Süddeutsche Zeitung</para>
            </listitem>
            <listitem>
                    <para> * Die Zeit</para>
            </listitem>
            <!--snip-->
    </itemizedlist>
    <!--snip-->
  </sect1>
</article>
```

Fig. 1 Annotation according to simplified DocBook

Consider the XML document in Fig. 1, which shows an extract from a linguistic research report (Greve *et al.*, 2002) marked up according to a simplified DocBook DTD (cf. Walsh and Muellner, 1999) (henceforth referred to as 'Layer `doc`'). Exactly the same text is provided with an annotation that marks topic types of text segments in Fig. 2 (henceforth 'Layer `seg`').[1]

While such multiple annotations are kept and maintained independently of each other, they can still be connected using the identical textual content as a link to create a common view as proposed in Witt (2002). This common view can then be exploited to infer relations between the multiple layers (cf. Bayerl *et al.*, 2003), to provide a framework for editing the common textual source (cf. Witt, 2004), and also to unify the multiple annotation layers for creating an integrated XML view.

1 The PCDATA of the texts are identical modulo whitespace adjustments that we applied here for pretty printing. The conversion component XML2PROLOG can be parametrized to add or remove whitespace anywhere if whitespace is the reason for differences of the PCDATA.

3 Unification

Sometimes an integrated XML representation of multiple annotations is required. For this purpose, we have developed a Prolog-based unification program for multiple annotations. Presently, two document layers can be merged. Figure 3 displays the architecture of the unification tool, i.e. all formats and transformations involved.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE ScientificArticle SYSTEM "pretty.dtd">
<ScientificArticle>
  <segment topic="method_evd" id="s49" parent="g">3 Erfassungmodus der
     Stellenanzeigen</segment>
  <segment topic="data" id="s50" parent="g"> Bei der Erfassung der
     Stellenanzeigen wurden sowohl überregionale und regionale Zeitungen als
     auch Fachzeitschriften berücksichtigt.</segment>
  <segment topic="data" id="s51" parent="g"> Zu den bearbeiteten
     überregionalen Zeitungen zählen: * Frankfurter Rundschau * Süddeutsche
     Zeitung * Die Zeit<!--snip--></segment>
  <!--snip-->
</ScientificArticle>
```

Fig. 2 Annotation of topic type segments

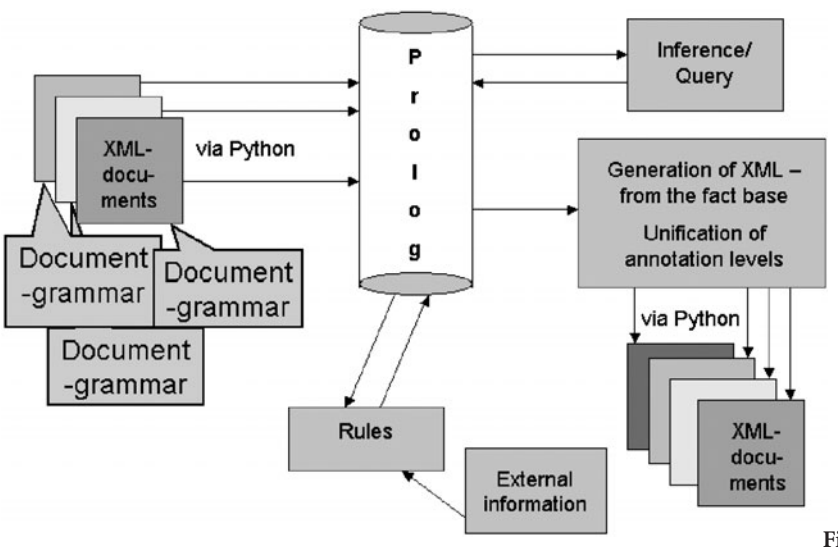

Fig. 3 The unification process

The interface to the central merging component SEMT is the Prolog predicate `semt/5`, which receives five arguments.

(1) The name of Layer A (the file name of the original XML document); this layer is considered the base layer during merging (see Section 3.3).
(2) The name of Layer B (the file name of the original XML document).
(3) A name for the Prolog result fact base (see below).
(4) A (possibly empty) list of elements that should be deleted during merging.
(5) The name of a rule file for the external specification of rules that should be applied in case of identity conflicts (see Section 3.3.2).

The different annotation layers (XML documents) and their identical textual content are transformed into an internal uniform Prolog representation via the Python script XML2PROLOG. The characteristics of this Prolog fact base are discussed in Section 3.1. It contains all the information of the different annotation layers and the textual data, and it is the basis for the relation inference component and the merging component, which are also implemented in Prolog. The result of the merging process is again a Prolog fact base, representing the document tree for the XML result document. This new fact base is re-converted to well-formed XML using another Python script called PROLOG2XML.

An automatically unified XML document for the two XML documents shown in Figs 1 and 2 is shown in Fig. 4. Basically, it contains all tags from the two input layers. The merging component SEMT was parametrized such that in case of *identity conflicts* (element instances from Layer A and Layer B that span the same range of textual data, such as `<title>` in Layer `doc` and the first `<segment>` in Layer `seg`), the element from Layer seg appears nested inside the element from Layer `doc`. (Further strategies to resolve identity conflicts are discussed in Section 3.3.2). Also note that the text in the first `<para>` element in Layer `doc` properly overlaps with the text in the third `<segment>` element in Layer `seg`. In the result document in Fig. 4, the element `<segment>` from Layer `seg` is absent so that the XML result document is well-formed. However, empty *milestone elements* (Sperberg-McQueen and Burnard, 1994) mark the beginning and end of the original overlapped `<segment>` element to preserve the information. In the following sections, the components of the unification program are discussed further.

3.1 A single representation: Prolog fact base

The Prolog fact base that represents a common view on the textual content and its multiple annotations is based on the format introduced by Sperberg-McQueen *et al.* (2001) to represent *meaning and interpretation of markup* of single XML documents. In this format, all elements, attributes and text nodes are saved as Prolog predicates. We have extended their approach such that all pieces of information from multiple annotations as described in Section 2 are represented. Thus, Prolog facts for elements and attributes contain the following.

(1) The node type (i.e. `attr` for attributes or `node` for elements) as the name of the predicate.
(2) The name of the annotation layer as the first argument.
(3) The absolute start position of the annotated text passage as the second argument.
(4) The absolute end position of the annotated text passage as the third argument.
(5) The position of the unit in the document tree as the fourth argument.
(6) The name of the element or attribute as the fifth argument.

```
<doc_article>
    <doc_sect1>
        <seg_ScientificArticle>
            <doc_title>
                <seg_segment topic="method_evd" id="s49" parent="g">
                    3 Erfassungmodus der Stellenanzeigen
                </seg_segment>
            </doc_title>
            <doc_para>
                <seg_segment topic="data" id="s50" parent="g"> Bei der
                   Erfassung der Stellenanzeigen wurden sowohl überregionale
                   und regionale Zeitungen als auch Fachzeitschriften
                   berücksichtigt.
                </seg_segment>
                <milestone id="m-0" element="segment"/> Zu den bearbeiteten
                   überregionalen Zeitungen zählen:
            </doc_para>
            <doc_itemizedlist>
                <doc_listitem>
                    <doc_para> * Frankfurter Rundschau</doc_para>
                </doc_listitem>
                <doc_listitem>
                    <doc_para> * Süddeutsche Zeitung</doc_para>
                </doc_listitem>
                <doc_listitem>
                    <doc_para> * Die Zeit
                    <milestone id="m-1" reference="m-0" element="segment"/>
                    </doc_para>
                </doc_listitem>
            </doc_itemizedlist>
        </seg_ScientificArticle>
    </doc_sect1>
</doc_article>
```

Fig. 4 Unification of `doc` and `seg` annotation layers

Figure 5 shows the common fact base of all nodes representing element instances and attributes from the annotation layers `doc` and `seg` introduced in Section 2. The facts representing attributes additionally contain a sixth argument for the value of the attribute. The Prolog representation of attributes is somewhat redundant in that the indication of the character positions could also automatically be inferred from the respective element node information, but an explicit indication of the range also for attributes can speed up processing.

Finally, the identical textual content (the PCDATA), is represented character-wise by a predicate called `pcdata_node`, whose three arguments are one character and its start and end positions as shown below.

```
node('seg', 0, 273, [1], element('ScientificArticle')).
node('seg', 0, 36, [1, 1], element('segment')).
node('seg', 36, 169, [1, 2], element('segment')).
node('seg', 169, 273, [1, 3], element('segment')).
node('doc', 0, 273, [1], element('article')).
node('doc', 0, 273, [1, 1], element('sect1')).
node('doc', 0, 36, [1, 1, 1], element('title')).
node('doc', 36, 222, [1, 1, 2], element('para')).
node('doc', 222, 273, [1, 1, 3], element('itemizedlist')).
node('doc', 222, 244, [1, 1, 3, 1], element('listitem')).
node('doc', 222, 244, [1, 1, 3, 1, 1], element('para')).
node('doc', 244, 264, [1, 1, 3, 2], element('listitem')).
node('doc', 244, 264, [1, 1, 3, 2, 1], element('para')).
node('doc', 264, 273, [1, 1, 3, 3], element('listitem')).
node('doc', 264, 273, [1, 1, 3, 3, 1], element('para')).
attr('seg', 0, 36, [1, 1], 'topic', 'method_evd').
attr('seg', 0, 36, [1, 1], 'id', 's49').
attr('seg', 0, 36, [1, 1], 'parent', 'g').
attr('seg', 36, 169, [1, 2], 'topic', 'data').
attr('seg', 36, 169, [1, 2], 'id', 's50').
attr('seg', 36, 169, [1, 2], 'parent', 'g').
attr('seg', 169, 273, [1, 3], 'topic', 'data').
attr('seg', 169, 273, [1, 3], 'id', 's51').
attr('seg', 169, 273, [1, 3], 'parent', 'g').
```

Fig. 5 Prolog fact base

```
pcdata_node(17, 18, 'd').
pcdata_node(18, 19, 'e').
pcdata_node(19, 20, 'r').
...
```

The prolog result fact base, which is the result of applying SEMT to the original fact base in listing 4, is shown in Fig. 6.

3.2 Relations between annotation layers

To determine the hierarchical structure of the intended merger between two annotation layers A and B, it is reasonable to consider the possible relationships between element instances on layer A and element instances on layer B on account of their shared or unshared PCDATA. These have been analyzed and classified in Durusau and O'Donnell (2002). In the overview given in Fig. 7, we have collapsed and renamed some of the relationships for the illustration of our approach.

The question is, what should a unified hierarchy look like when two element instances <a> and <b> from Layers A and B, respectively, share PCDATA?

```
node('seg_fragment', 169, 273, [1, 3], element('segment')).
attr('seg_fragment', 169, 273, [1, 3], 'topic', 'data').
attr('seg_fragment', 169, 273, [1, 3], 'id', 's51').
attr('seg_fragment', 169, 273, [1, 3], 'parent', 'g').

node('output', 0, 273, [1], element('doc_article')).
node('output', 0, 273, [1, 1], element('doc_sect1')).
node('output', 0, 273, [1, 1, 1], element('seg_ScientificArticle')).
node('output', 0, 36, [1, 1, 1, 1], element('doc_title')).
node('output', 0, 36, [1, 1, 1, 1, 1], element('seg_segment')).
attr('output', 0, 36, [1, 1, 1, 1, 1], 'topic', 'method_evd').
attr('output', 0, 36, [1, 1, 1, 1, 1], 'id', 's49').
attr('output', 0, 36, [1, 1, 1, 1, 1], 'parent', 'g').
node('output', 36, 222, [1, 1, 1, 2], element('doc_para')).
node('output', 36, 169, [1, 1, 1, 2, 1], element('seg_segment')).
attr('output', 36, 169, [1, 1, 1, 2, 1], 'topic', 'data').
attr('output', 36, 169, [1, 1, 1, 2, 1], 'id', 's50').
attr('output', 36, 169, [1, 1, 1, 2, 1], 'parent', 'g').
node('output', 222, 273, [1, 1, 1, 3], element('doc_itemizedlist')).
node('output', 222, 244, [1, 1, 1, 3, 1], element('doc_listitem')).
node('output', 222, 244, [1, 1, 1, 3, 1, 1], element('doc_para')).
node('output', 244, 264, [1, 1, 1, 3, 2], element('doc_listitem')).
node('output', 244, 264, [1, 1, 1, 3, 2, 1], element('doc_para')).
node('output', 264, 273, [1, 1, 1, 3, 3], element('doc_listitem')).
node('output', 264, 273, [1, 1, 1, 3, 3, 1], element('doc_para')).

pcdata_node(0, 1, '3').
pcdata_node(1, 2, ' ').
pcdata_node(2, 3, 'E').
...
```

Fig. 6 Prolog fact base representing the merger (unification) of `doc` and `seg`

When the relation of proper nesting holds between the two element instances (*included*(*b,a*)), the result layer can retain all tags as in the originals as sketched below.

```
Layer A:       <a>.....................</a>
Layer B:                 <b>........</b>
Unification:   <a>.......<b>........</b>.......</a>
```

Occurrences of *non-proper inclusion*, i.e. where an element instance on Layer A either starts or ends at a position where an element instance on Layer B also starts or ends (*endPoint_identical*(*a,b*) or *startPoint_identical*(*a*, *b*)), can also be merged in a straightforward way. The desired nesting of elements in the target layer must be inferred by comparing the length of the respective text spans on both layers—the

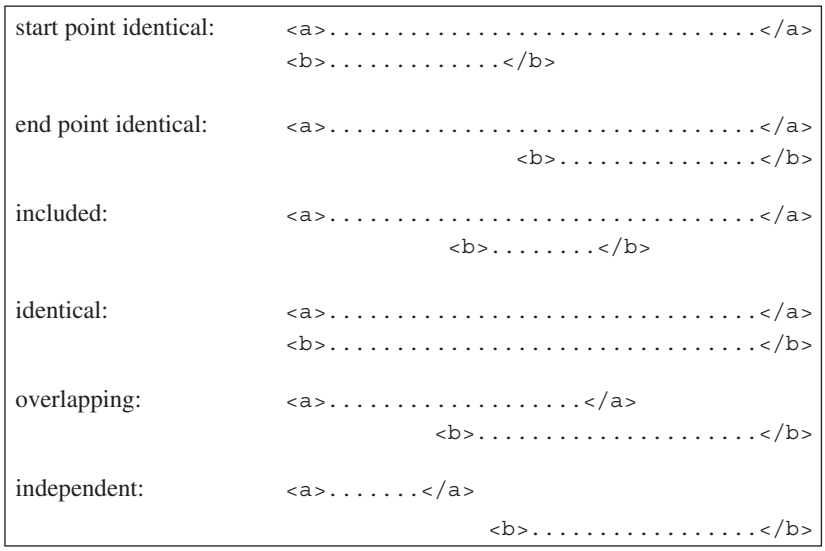

Fig. 7 Possible relations between pairs of elements instances from different layers after Durusau and O'Donnell (2002)

element spanning more characters then becomes the outer element, i.e. the parent node in the result tree:

```
Layer A:      <a>.....</a>
Layer B:      <b> ..... </b>
Unification:  <a> <b> ..... </b> ..... </a>
```

3.3 Resolution of conflicts during merging

3.3.1 *Overlap conflicts*

In case the overlap relation holds between two element instances of Layer A and Layer B, the result of a simple merger cannot be represented in a tree structure and consequently not be mapped to a single well-formed XML document in a straightforward way. Thus, SEMT writes the element instance that is not from the base layer to a difference list. In order to obtain a well-formed XML result file, PROLOG2XML generates empty milestone elements (Sperberg-McQueen and Burnard, 1994) from this difference list that mark the beginning and end of the original, overlapped element (see the example below and also cf. Fig. 4).

```
Layer A:      <a> .....</a>
Layer B:            <b> .....</b>
Unification:  <a>.....<milestone id="m-0"
              element="b"/>..... </a> ... <milestone
              id="m-1" reference="m-0" element="b"/>
```

Alternatively, in our implementation it is possible to generate element fragments from a node in the difference list, i.e. to split an originally overlapped element into parts, each of which is properly nested in its context (Sperberg-McQueen and Burnard, 1994).

3.3.2 *Identity conflicts*

An identity conflict exists when two element instances from the two annotation layers span an identical portion of text, as in the example of the annotation of the Japanese pronoun *anata* ('you') according to a reference type level (Layer A), and a phrasal level (Layer B).[2]

Layer A: `<ref type="anaphoric">anata</ref>`
Layer B: `<np>anata</np>`

In such a case, a non-overlapping merged tree can be constructed, but the decision which of the two should become the parent node, i.e. the outer element in the result document, is non-trivial.

There are three alternative possible result tree configurations:

(1) A hierarchy is created, i.e.

- Unification: `<np><ref type="anaphoric">anata</ref></np>` or
- Unification: `<ref type="anaphoric"><np>anata</np></ref>`

(2) One of the element instances is deleted, i.e.

- Unification: `<ref type="anaphoric">anata</ref>` or
- Unification: `<np>anata</np>`

(3) One of the element instances is deleted and its attributes are integrated into the other elements, i.e.

- Unification: `<np type="anaphoric">anata</np>`

To decide which strategy should be chosen, a heuristics has been implemented that infers possible *metarelations* that hold between an element type a on Layer A and an element type b on Layer B. The metarelation *identity*(a, b) holds, if for all pairs of instances (a_i, b_j) of a and b that share PCDATA, the relation *identical*(a_i, b_j) holds. The metarelation *inclusion*(a,b) holds if, for all pairs of instances (a_i,b_j) of a and b that share PCDATA, either the relation *identical*(a_i, b_j), or *included*(b_j, a_i), or *endPoint_identical*(a_i, b_j), or *startPoint_identical*(a_i, b_j) holds, and at the same time the metarelation *identity*(a, b) does not hold. At the beginning of the merging process, for each pair of element types (a, b) where a is an element type from Layer A and b is an element type from Layer B, it is checked whether the metarelations *inclusion*(a, b) or *inclusion*(b, a) hold, and the result is stored in a file.

2 This example illustrates the benefit of our methodology for linguistic research on the relation between annotations of semantic relations ('anaphoric') and syntactic means of their realization ('np'). Moreover, an annotation and analysis of such relations is needed for computational applications such as automatic anaphora resolution.

As a default, this file is consulted whenever an identity conflict between two element instances is encountered. If an identity conflict is found to exist between the two element instances a_i and b_j (from Layers A and B, respectively), then if the metarelation *inclusion*(*a*, *b*) holds, a_i becomes the parent node in the merged document as in the first alternative under (1) above. If the metarelation *inclusion*(*b*,*a*) holds, b_j becomes the parent node as in the second alternative under (1) above. If neither of these two metarelations holds, the element from the base layer becomes the parent node. Additionally, and alternatively, external rules can be specified that prescribe which strategy for the solution of identity conflicts should be chosen in SEMT. Our implementation is designed to deal with three types of rules:

PRIORITIZE(a,b):	always render conflicting a and b as `<a><b>...</b></a>` in the target document
DELETE(a):	Retain only b in the target document
ATTRIBUTE(a):	Retain only b in the target document, and add the attributes of a to b

In a future implementation of the merging process, it will be possible to specify further external rules types, where the elements to which a rule applies can be subcategorized according to their context.

(1) Rules without restrictions. These rules apply to all instances of the elements, for example to all `<np>` and `<ref>` elements.
(2) Rules with attribute restrictions. The rules apply only to elements with certain attributes and values, for example all `<ref>` elements with the attribute specification `type="anaphoric"`.
(3) Rules which describe configurations between elements. These rules apply only to elements that are in a specified configuration with other elements, for example all `<ref>` elements that are contained in a `<syntacticDescription>` element.
(4) Rules which combine (2) and (3), for example affecting all `<ref>` elements which are contained in a `<corpus>` element which has the attribute specification `corpus-type="syntax has priority"`.

As an alternative to encoding such rules in Prolog, an RDF Schema representation (Dan and Guha, 2004) has been developed (cf. Sasaki, 2004). The representation in Fig. 8 makes use of the triple notation for logical statements as defined for RDF and RDF Schema. Statements are formulated with predicates such as 'subConceptOf' and arguments such as 'sekStruk:NpGeneral'. The concepts of a given domain are declared and integrated in a taxonomy. The taxonomic relation is expressed via the predicate 'subConceptOf'. The statement 'sekStruk:NpAnaphoric subConceptOf sekStruk:NpGeneral', for example expresses that the concept 'sekStruk:NpAnaphoric' is subordinate to the concept 'sekStruk: NpGeneral'. In addition to the taxonomic relation to other concepts,

```
TAXONOMY OF CONCEPTS:

sekStruk:syntacticDescription subConceptOf rdf:resource.
sekStruk:LinguisticUnit subConceptOf rdf:resource.
sekStruk:NpGeneral subConceptOf sekStruk:LinguisticUnit.
sekStruk:NpAnaphoric subConceptOf sekStruk:NpGeneral.
sekStruk:RefUnitGeneral subConceptOf sekStruk:LinguisticUnit.
sekStruk:RefUnitInSyntacticDescription sekStruk:subConceptOf RefUnitGeneral.

MAPPING BETWEEN CONCEPTUAL HIERARCHY AND MARKUP CONSTRUCTS.
sekStruk:NpGeneral sekStruk2PrimStruk "element np".
sekStruk:NpAnaphoric sekStruk2PrimStruk "attribute type {'anaphoric'}".
sekStruk:RefUnitGeneral sekStruk2PrimStruk "element ref".
sekStruk:RefUnitInSyntacticDescription sekStruk2PrimStruk
 "layerRel:included_A_in_B sekStruk:syntacticDescription".
sekStruk:syntacticDescription sekStruk2PrimStruk
 "element corpus { attribute corpus-type {'syntax has priority'}}".
```

Fig. 8 RDF-oriented representation of rules

concepts are related to markup constructs (here represented in the compact syntax of RELAX NG) via the predicate 'sekStruk2primStruk'. The statements 'sekStruk:NpAnaphoric sekStruk2PrimStruk2 ''attribute type anaphoric'' ', for example, expresses that the concept 'sekStruk:NpAnaphoric' is related to the attribute 'type' by its value 'anaphoric'. Since the concept 'sekStruk:NpAnaphoric' is subordinate to 'sekStruk:NpGeneral', the relation of 'sekStruk:NpGeneral' to the 'np' element can be inferred. Hence, the inferred rule for this concept is 'element np attribute type ''anaphoric'' '.

4 Conclusion

The framework of XML-based multilayer annotation is designed for the representation of multiple hierarchies, which are often encountered when annotating language data. In this approach, textual content is annotated several times, and each annotation layer is stored in a separate and independent XML document, such that each layer may be independently maintained and processed. The approach presents amongst other things a solution to the problem of representing overlapping hierarchies. To continue to provide an integrated view of separate annotations, we introduced a Prolog implementation for the unification of two XML documents with identical textual content and concurrent markup. By means of exploring the possible relationships between element instances from different annotation layers that share PCDATA, it was demonstrated what their unified hierarchies look like. In case two element instances stand in the identity relation, the metarelations that hold between their respective element types are

computed to decide which instance is to become the parent node in the target hierarchy. In addition, rules can be specified by a user, which prescribe how identity conflicts should be solved for certain element types. The Prolog implementation is currently employed in two subprojects of the Research Group *Text-technological modelling of information* funded by the German Research Foundation and dealing with the representation of textual structures on multiple information levels, and with the analysis of relations between document grammars for linguistic annotations.

In the future, it will be possible to process more specific rules for solving identity conflicts during unification, by defining subsets of element instances according to their attribute specifications or their hierarchical context. The implementation will also be extended to be deal with multiple XML documents with concurrent markup rather than only two.

References

Barnard, D. T., Burnard, L., Gaspart, J.-P., Price, L. A., Sperberg-McQueen, C., and Varile G. B. (1995). Hierarchical encoding of text: Technical problems and SGML solutions. In Ide, N. and Véronis, J. (eds), *TEI Background and Context*, Kluwer, pp. 211–31.

Bayerl, P.S., Goecke, D., Lüngen, H., and Witt, A. (2003). Methods for the semantic analysis of document markup. In *Proceedings of the ACM-Symposium on Document Engineering* (DocEng). Grenoble, France, pp. 161–170.

Dan, B. and Guha, R. (2004). *RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation.* http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

DeRose, S., Durand, D., Mylonas, E., and Renear, A. (1990). What is text, really? *Journal of Computing in Higher Education*, **1(2)**: 3–26.

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

Greve, M., Iding, M., and Schmusch, B. (2002). Geschlechtsspezifische Formulierungen in Stellenangeboten. *Linguistik Online*, **11**. http://www.linguistik-online.de.

Huitfeldt, C. and Sperberg-McQueen, C. (2001). *TexMECS: An Experimental Markup Meta-language for Complex Documents*. http://www.hit.uib.no/claus/mlcd/papers/texmecs.html.

Sasaki, F. (2004). Secondary information structuring—a methodology for the vertical interrelation of informational resources. In *Proceedings of the Extreme Markup Languages*, Montreal. Available at http://www.idealliance.org/proceedings/extreme (accessed 12 January 2005).

Sperberg-McQueen, C. and Burnard, L. (eds) (1994). *Guidelines for Electronic Text Encoding and Interchange (TEI P3)*. Chicago and Oxford: Text Encoding Initiative.

Sperberg-McQueen, C., Huitfeldt, C., and Renear, A. (2001). Meaning and interpretation of markup. *Markup Languages*, **2(3):** 215–34.

Tennison, J. W. P. (2002). The layered markup and annotation language. In *Proceedings of the Extreme Markup Languages*, Montreal.

Thompson, H. S. and McKelvie, D. (1997). Hyperlink semantics for standoff markup of read-only documents. In *Proceedings of SGML Europe '97: The next decade—Pushing the Envelope*, Barcelona, pp. 227–9.

Walsh, N. and Muellner, L. (1999). *DocBook: The Definitive Guide*. Beijing: O ' Reilly.

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

Witt, A. (2004). Multiple hierarchies: New aspects of an old solution. In *Proceedings of the Extreme Markup Languages*, Montreal. Available at http://www.idealliance.org/proceedings/extreme (accessed 12 January 2005).