A model oriented approach to the mapping of annotation formats using standards.

Florian Zipser, Laurent Romary

A model oriented approach to the mapping of annotation formats using standards

Florian S. Zipser, HUB-IDSL

Laurent Romary, INRIA-Gemo & HUB-IDSL

Abstract: In this paper, we present, Salt, a framework for mapping heterogeneous linguistic annotation formats into each other using a model-based approach, i.e. independently of the actual formats in which the corresponding linguistic data is being expressed. As we describe the underlying concept of this framework, we identify how it echoes ongoing standardisation activities within ISO committee TC 37/SC 4, and in particular, the possible conceptual equivalences with ISO CD 24612 (LAF) combined with ISO 24610-1 (FSR), as well as the possible role of the central data category registry (ISOCat), currently under deployment. We thus show the adequacy of our methodology and its capacity to integrate a wide range of possible linguistic annotation models.

1 The issue of mapping and the current standardization landscape

1.1 The importance of mapping when managing heterogeneous language resources

Over the years, the linguistic research community has seen the development of a wide variety of tools ([schmidt02], [lezius02] and [zeldes09] specifically targeted at the extraction, representation and analysis of many different phenomena. For example, a tool such as the search tool Tiger Search [lezius02] was primary developed for syntactic analysis, whereas a tool like the annotation tool EXMARaLDA [schmidt02] covers discourse analysis. Most of these tools are built around the use of one specific format, which was developed specifically for this tool and for a certain type of analysis. The focus of such formats has in general been to supply all necessary information for the tool to proceed in an efficient manner (limited coverage, optimized representation). Because of their specialization, these formats are difficult to reuse in other contexts for which they were not intended.

Providing standardized formats is one of the possible answers to this issue. One of the benefits of a standardized format can be the interoperability between tools or the keeping of existing data for some years and being assured these will also be legible in the future. At present, however, there is very few linguistic data that is represented in standardized formats. As long as the tools do not have a direct import or export for standardized formats, it would be necessary to map the used formats from or to standardized formats. As a consequence, defining mappings between existing formats and more standardized representations represents an important component of any further development relying on the use of external data.

1.2 Difficulties related to mapping formats

Existing standards such as LAF [iso24612], MAF [iso24611] or SynAF [iso24615] mainly focus on the provision of persistent models and formats to provide a stable descriptive framework for linguistic information. In particular, they do not address the mapping between themselves and the already used formats, with the exception of ISO 16642 (TMF), which provide an explicit mapping framework across terminological data formats. It is thus necessary to define appropriate solutions to get existing data into standard formats by 1) defining a conceptual mapping between them and 2) having a concrete implementation which realizes the mapping thus defined.

Most standards, because they basically aim at providing an interchange format, include a strong

technical part to specify, for instance, how they can be implemented in a given XML representation or a relational database structure. In this context, it is quite often the case that the very existence of such format definitions, with the associated technical constraints, impact on the actual expressive power of the corresponding model. For example, an attribute value of an XML element cannot contain additional mark-up. To create a mapping, one therefore has to consider both the conceptual mapping and the technical realizations. This requires the implementer to have a good level of understanding of the underlying format description, for instance expressed by means of a schema language (DTD, RelaxNG or W3C schema) in the case of XML. Covering both aspects makes the mapping generation extremely complex, for anyone who just wants to focus on the underlying linguistic concepts or constraints.
A conceptual mapping has to cover two aspects. First, there has to be a mapping for each structural object like the representation of tokens or representations of primary data. Second, the mapping has to regard semantic mappings for data categories. In this paper we want to propose an approach to structural mappings via a model like Salt (introduced in section 2) and a semantic mapping using the ISOCat [kemps09] system (shown in section 3).

1.3 A model based approach to mapping

A solution for clarifying the actual interdependence between conceptual and technical levels is to adopt a model-based approach as for instance in MDA ([miller03]). The idea is to separate the meaning of data (the model layer) from their representation (the format layer, cf. figure 1) especially in the case of persistence constraints. When a separation between a conceptual model and a persistent format is made, one can avoid taking care of persistence issues and focus on processing data through the elicitation of a mapping between models. For example, a specialist in the linguistic domain, can create or describe a mapping between two morphosyntactic tagsets, leaving it for a further stage, and a more technical expertise, to implement a mapping for the underlying formats.

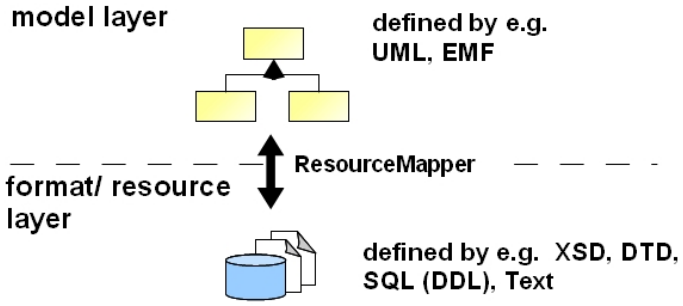

figure 1: correlation between the model and the resource or format layer

Model-based development frameworks such as MDA [miller03] or EMF [steinberg09] support 1) a graphical representation for models and 2) a generation of processable object models for further work (in terms of an API for instance). The graphical representation of a model can be used as a communication base between linguists and technical experts. The generated API can be used for implementing tools working with the model, such as an annotation tool or, in our case, a converter. The EMF framework that we use also generates a persistent format based on XML. This generated format is called a resource and can be exchanged with other formats, by re-implementing the "ResourceMapper" in figure 1.
Figure 2 shows an example of a resource mapping between the format description of Tiger XML [mengel00] and the corresponding model.

```
<element       name="graph" type="graphType" minOccurs="0"
               maxOccurs="1"/>
<complexType name="graphType">
  <sequence>
    <element   name="terminals" type="terminalsType" minOccurs="1"
       maxOccurs="1"/>
    <element   name="nonterminals" type="nonterminalsType" minOccurs="1"
       maxOccurs="1"/>
  </sequence>
  <attribute name="root" type="idrefType" use="required"/>
 </complexType>

 <complexType name="terminalsType">
  <sequence>
    <element   name="t" type="tType" minOccurs="1"
               maxOccurs="unbounded"/>
  </sequence>
 </complexType>

 <complexType name="tType">
  <attribute name="id" type="idType" use="required"/>
 </complexType>

 <complexType name="nonterminalsType">
  <sequence>
    <element   name="nt" type="ntType" minOccurs="0"
       maxOccurs="unbounded"/>
  </sequence>
 </complexType>

 <complexType name="ntType">
  <sequence>
    <element   name="edge" type="edgeType" minOccurs="0"
       maxOccurs="unbounded"/>
       <complexType>
          <attribute name="idref" type="idrefType" use="required"/>
    </complexType>
    </element>
  </sequence>
  <attribute name="id" type="idType" use="required"/>
    ...
 </complexType>
</element>
```

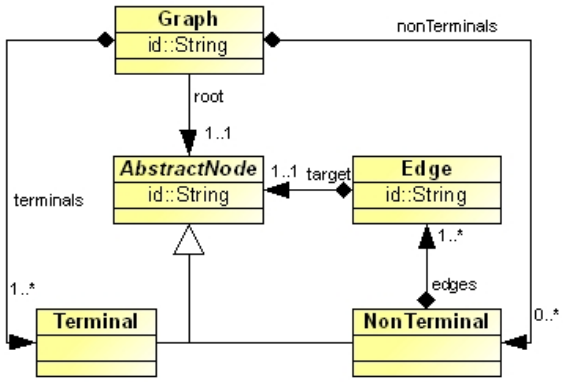

figure 2: on the left side: an excerpt of the xsd description of Tiger XML [mengel00] ; on the ride side: the correlated model for this excerpt in UML-like notation

1.4 Same but different – shared advantages with a format based approach

As pointed out in [ide07], the number of mappings can be reduced by mapping data over a common format, or in this case a common model. Instead of creating n^2-n mappings to map n models to each other in the case of 1:1 mappings, the number of mappings via a common model decreases to 2n mappings. In this paper we want to follow this approach. Figure 3 shows this approach using a common model for mappings simultaneously to the mapping of data via a pivot format defined by LAF/GrAF [ide07].

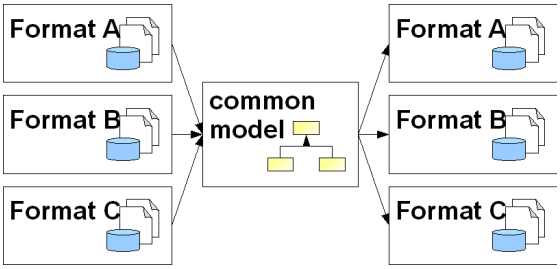

figure 3: common model as middleware between formats to import and to export

In the remaining sections of this paper, we present the main characteristics of the framework that we have developed to implement such a perspective by the comparisons of models.

2 An overview of Salt and its relation to LAF

2.1 Basic principles

Salt is a common model for linguistic annotated data. This model defines a conceptual abstraction of data, independent of persistence techniques. This means that one can use Salt as an object representation of data. This allows us to process data with respect to the object model, with no prejudice with respect to the actual storage (or linearisation) format, be it XML or a relational database, in which the data will be represented.

Salt was influenced by several existing linguistic formats such as EXMARaLDA [schmidt02] TigerXML [mengel00] and above all PAULA [dipper05]. Salt unifies the concepts of these formats e.g. common timeline, multiple layers of annotation etc. and represents them in a common model. Salt is a model for representing the underlying organization of linguistic data, and as such, does not take into consideration their underlying semantics. Furthermore, Salt is independent of specific linguistic theories or analyse.

2.2 The underlying graph structure of Salt

Salt is based upon a directed, labeled and layerable graph structure model. The model contains a graph structure component, which contains 1) a set of nodes or vertices, 2) a set of directed edges, 3) a set of layers, which embraces a set of nodes and edges and 4) a set of labels, used to label a node, an edge, a layer or a label. This means that a label can be used as a recursive structure and therefore enables the possibility to annotate an annotation.

The Salt model is a refinement of the general graph structure model, in effort to apply Salt to linguistic needs e.g. primary data, tokens, relations, annotations and so on. But every element in Salt is still an element of a general graph structure model and can be processed with general graph structure methods e.g. traversing. Figure 4 shows this refinement on the basis of some elements of Salt. Here one can see, for example that a textual representation of primary data (STextualDS) is still a node. Although nodes get a more linguistic meaning, nodes and relations are just placeholders for annotations.

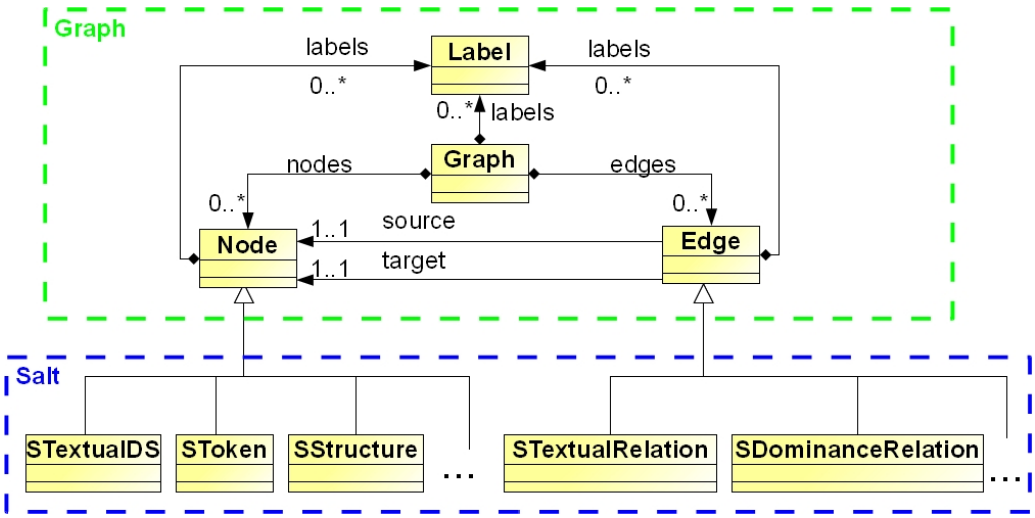

figure 4: excerpt of the refinement between the graph structure model and the common model Salt. The elements STextualDS, SToken and SStructure are still nodes and the elements STextualRelation and SDominanceRelation are still edges.

We used the element STextualDS as a model representation of the primary data. Therefore this element contains a String representation of the primary data. Continuous spans of the primary data can be addressed by using the node type SToken and the edge type STextualRelation. A node of type SToken represents the tokenization of the primary data and is the basis for further structural objects and annotation. To relate such a token node with the primary data node, an edge of type STextualRelation can be created. This edge contains the start and end position of the referred span. To create hierarchical annotation graphs for example in case of syntactic analysis one can use nodes of type SStructure and relate them via edges of type SDominanceRelation to one or more nodes of type SToken or SStructure. Figure 5 shows an example of data represented in the Salt model. Salt offers further types of nodes and edges to create annotation graphs which are not shown in figure 4 and not mentioned here. For example it contains further edge types to realize different relations between nodes.

2.3 Salt and LAF

The graph-based approach is very similar to the one taken in the linguistic annotation framework (LAF, [iso24612]). Our objective is indeed to let Salt and LAF be identified as complementary tools on their specific abstraction level. LAF can be used as a persistence and exchange format for data whereas Salt can be used 1) as a conceptual abstraction which can be easily understood by non technical experts 2) as basis for a processable API. To do so we need a mapping between the Salt object model and the XML-representation of LAF (the GrAF format [ide07]). Although both GrAF and Salt are very similar, there are some core differences between them. One is the way they deal with edges: as opposed to GrAF, Salt allows edges to be annotated. A second difference lies in the referencing to primary text: In Salt there is a relation (STextualRelation) between a token node (SToken) and the primary data node (STextualDS), whereas in GrAF there is just one span concept for both. A third difference is that in Salt a copy of primary data is part of the model in terms of a node (see SText1 in figure 5). The first two differences can be handled as shown in figure 5. The figure shows a Salt model representation and an XML representation according to GrAF. The third difference can be handled by storing primary data in a separate document or by loading primary data from a text file into the Salt model.

```
<graph>
  <!-- STextRel1 and SToken1 -->
  <span id="s0" start="0" end="4">
    <fs><f n="pos" v="verb"/></fs>
  </span>
  <!-- STextRel2 and SToken2 -->
  <span id="s1" start="5" end="12">
    <fs><f n="pos" v="noun"/></fs>
  </span>
  <!-- SDominance1 -->
  <node id="n0">
      <fs><f n="func" v="HD"/></fs>
  </node>
  <!-- SDominance2 -->
  <node id="n1">
    <fs><f n="func" v="OA"/></fs>
  </node>
  <!-- SStruct1 -->
  <node id="n2">
    <fs><f n="cat" v="SC"/></fs>
  </node>
  <!-- artificial edges -->
  <edge id="e0" from="n0" to= "s0"/>
  <edge id="e1" from="n1" to= "s1"/>
  <edge id="e2" from="n2" to= "n0"/>
  <edge id="e2" from="n2" to= "n1"/>
</graph>
```

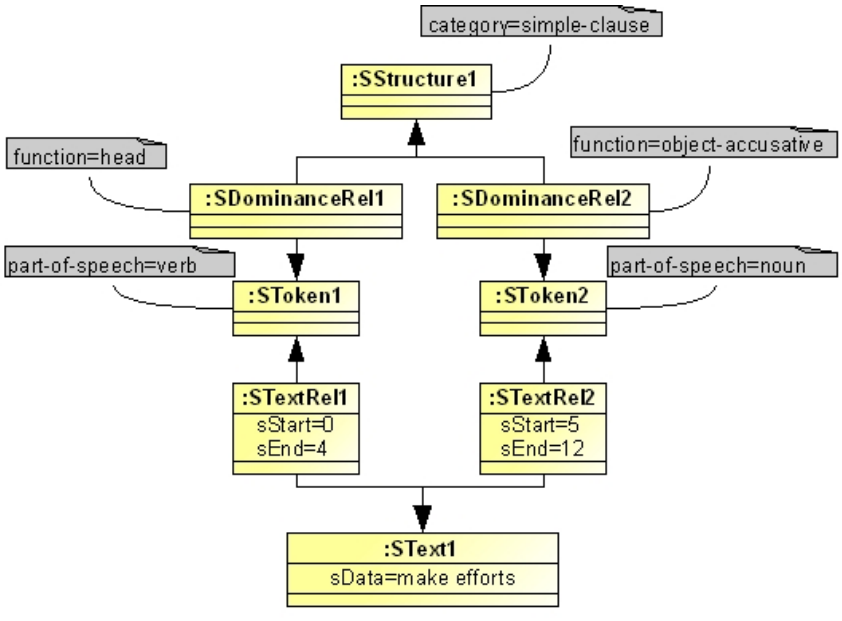

figure 5: on the left side: an example corpus represented in the format GrAF (the primary data "make efforts" can be stored in a external file); on the right side: the same example represented in a Salt model

Moreover, we developed Salt to be able to take into account some important phenomena that LAF would not handle in its current state:

- The representation of a common timeline (e.g. for audio-video and dialog data such as those produced by EXMERaLDA)
- The management of higher level structures, in particular for the implementation of the notion of corpus (in particular, embedded corpus or sub-corpus relations)
- The typing of annotations e.g. as textual, numeric or more complex values.

3 The relation of Salt to ISOCat and FSR

3.1 The need to consider the meaning of annotations

As already mentioned, Salt does not deal with the semantics of annotations. Similarly to GrAF [ide07] annotations are understood as an attribute-value pair, the entries of which do not have an interpretable meaning for the system. In the case of converting data, the meaning could be important. For example some formats like TreeTagger [schmid94] need to have part-of-speech or lemma annotations. If these data were mapped in a format or a model which handles annotations as attribute-value pair the meaning of the annotations would get lost. For example a problem occurs if one tries to map to a format which needs specific annotations, because the data for a part-of-speech annotation appear in different forms: pos=verb, POS=verb, PartOfSpeech=verb. Because of different surface representations of the attribute name for part-of-speech, annotations cannot be unified by the system. The system does not know that all these names actually have the same meaning.

It is therefore essential to have a possibility for unifying syntactical representations, or rather to make clear the meaning of such a representation. In this respect, ISOCat [kemps09] supplies the possibility of a central reference for elementary descriptors (data points) to which data model can refer. The meaning of a data point can be defined by the experts of the domain, whereas a system just has to check equality of references to the data points. In the case of part-of-speech annotations in format data, we can for instance use the reference http://www.isocat.org/datcat/DC-396, which in turn provides the actual definition of this data point as stored in ISOCat ("A category assigned to a word based on its grammatical and semantic properties").

Indeed, many formats which support attribute-value pairs for representing annotations only support String values e.g. TigerXML [mengel00], PAULA [dipper05] etc. . This means that a reference can be stored, but not necessarily interpreted as a reference. Thus we have to mark the data type of an attribute as well as of a value as references. In Salt there is a possibility for marking this, therefore we now take a closer look at an annotation. In figure 5 annotations are shown as simple attribute-value pairs beside the nodes and edges. Annotations are slightly more complex than what figure 5 shows. The annotation shown in figure 6 is the same as in figure 5 beside the node "SToken1" first as a String representation and second as a representation using ISOCat references.

:anno1	
name	=part-of-speech
nameType	=String
value	=verb
valueType	=String

:anno1	
name	=http://www.isocat.org/datcat/DC-396
nameType	=URI
value	=http://www.isocat.org/datcat/DC-1424
valueType	=URI

figure 6: on the left side: an annotation using simple string values as an attribute-value pair; on the right side: an annotation using references to ISOCat

3.2 Salt and FSR

As in GrAF, Salt nodes can be multiply annotated. For example, one can attach a part-of-speech and a lemma annotation to one node. But actually in Salt, there is no grouping function for annotations. Every annotation stands alone for itself. GrAF uses feature-structures (FSR) defined by ISO [iso24610-1] and used in the TEI P5 guidelines [burnard08]. For example some features can be grouped to a "morpho-syntactic annotation". GrAF does not yet support naming or typing of a feature structure as TEI describes (@type attribute in the <fs> element). Figure 7 shows an example taken from the TEI P5 guidelines for representing a grouping of annotations via feature structures.

```
<fs type="morpho-syntax">
  <f name="case">
    <symbol value="accusative"/>
  </f>
  <f name="gender">
    <symbol value="feminine"/>
  </f>
  <f name="number">
    <symbol value="plural"/>
  </f>
</fs>
```

figure 7: sample from the TEI P5 guidelines of grouping features by using feature structures

In Salt you can either represent the given three annotations as independent annotations, or you can represent them by using recursive annotations (means creating annotations on annotations). The second way simulates such a grouping as feature structures achieve. Both ways are shown in figure 8.

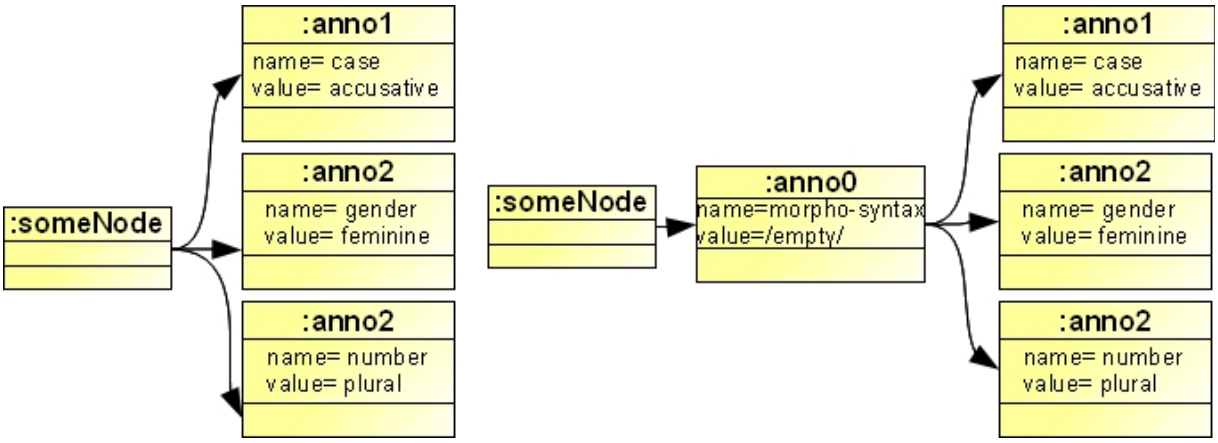

figure 8: on the left side: the sample from figure 7 without grouping; on the right side the same sample with grouping via the recursive structure of annotations in Salt

In addition to the types URI and String, we introduce additional types for annotation names and annotation values. On the one hand, there are additional simple types such as numeric (for numeric data), float, and boolean. On the other hand, there is a complex type called object. This complex type is defined in a flexible way, so that a value of this type can be any kind of object. As a consequence, it is possible to define a complex structure as a collection with conditions on their elements in terms of alternations or negations as mentioned in TEI [burnard08] chapter 18.

The main element of Salt is a SaltProject. This element contains the corpus structure. The corpus

structure is a tree, which defines super- and sub-corpus relations between corpora. A corpus contains one ore more documents in which the primary data, tokens, hierarchical structures annotations and so one can be found. Additionally to the corpus structure a SaltProject can also contain a library graph structure. This graph structure consists of nodes, which define data points as well as ISOCat do. These nodes can be referenced by URI´s using the scheme *salt*. A library structure can therefore be modeled as a graph structure. For example the STTS tagset [schiller95] for German part-of-speech can be described as shown in figure 9.

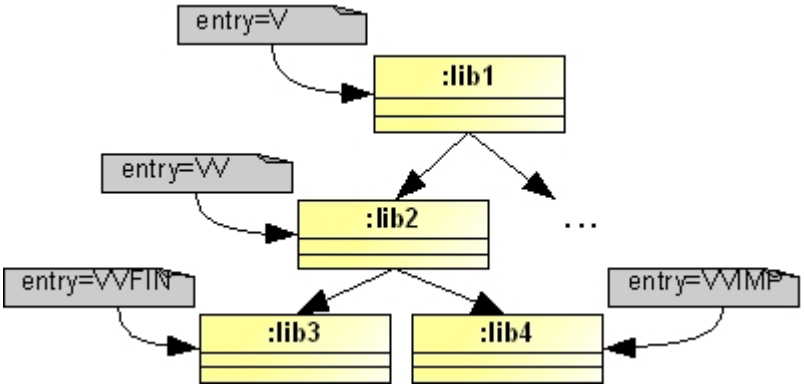

figure 9: an excerpt of the STTS tagset represented in the library graph structure of Salt. This example shows how refinements between entries can be handled.

Figure 9 contains the nodes „lib1“, „lib2“, „lib3“ and „lib4“ as data points. These nodes can be annotated with annotations like entry, for the tagset name, a description, which explains the usage of this tag and an example, which shows the usage in a specific case. The relations between the nodes “lib1”, “lib2”, “lib3” and “lib4” can be interpreted as a refinement. This means, that the node “lib3” which defines the entry “VVFIN”[1] is also of type “V”[2]. Further we propose a grouping relation to group the represented entries of several nodes under one node. This way of grouping is similar to the grouping function of the “fvLib” element of the FSR. Figure 10 shows the grouping mechanism by using a grouping relation.

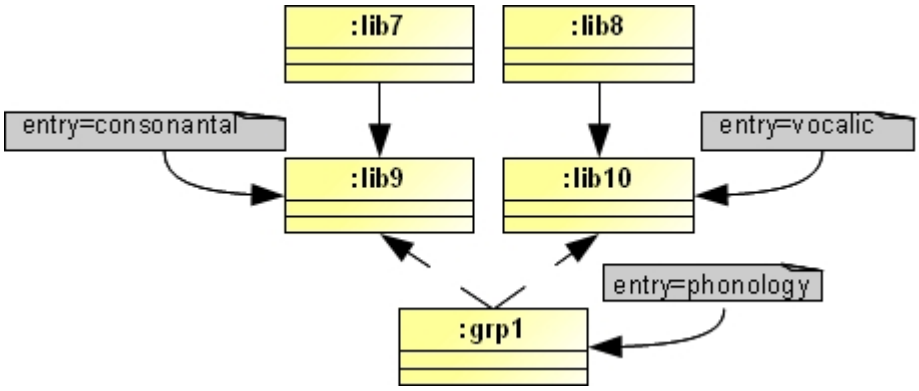

figure 10: grouping mechanism to group several data points e.g. consonantal and vocalic to one data point. This example is an excerpt from the TEI P5 guidelines (chapter 18).

The dashed arrow of figure 10 shows such a grouping relation, whereas the continuous arrow shows a refinement. The node “grp1” groups the nodes “lib3” and “lib4”, and also stands for the entry “consonantal” as well as for the entry “vocalic”.

1 tag for a finite full verb in the STTS
2 general tag prefix for a verb in the STTS

To use a data point such as a document structure, one can use the attribute value of an annotation typed as URI. The value than contains a URI entry. This URI starts with the scheme name *salt*, followed by the path which is the identifier for the library structure and the fragment which is the identifier of a node of the library structure graph. This node either can be a node standing for such an entry as "lib3" for example, or a grouping node as "grp1". Figure 11 shows the referencing mechanism for annotations using a URI value for a reference to the library graph structure.

:anno1	
name=	pos
nameType=	URI
value=	salt://featLib#lib3
valueType=	URI

:anno2	
name=	phonology
nameType=	URI
value=	salt://featLib#grp1
valueType=	URI

figure 11: on the left side: an annotation which references a library entry; on the right side: an annotation which references a grouping.

4 Validation (using Salt in Pepper)

4.1 What is Pepper?

To validate the Salt model, we define Pepper, a Salt based converter framework. This framework was developed to convert data from x formats into y different formats, with a constant number of mapping steps. As shown in figure 3 Salt and Pepper makes it possible to convert several formats via a common model into each other with a minimal number of needed mappings and just two steps.

Pepper thus forms a use case for Salt with which we can check whether Salt can represent data from several formats. Furthermore, it is possible to trace information losses during conversion operations. For example one can convert a corpus from format A into Salt and then export the data back to format A. The import and export can then be compared for losses.

4.2 How does Pepper work?

Pepper can be separated into three components: 1) the framework, 2) a common instance of the Salt model and 3) mappers to several formats. Figure 12 shows the general architecture of Pepper and the relations of the components.

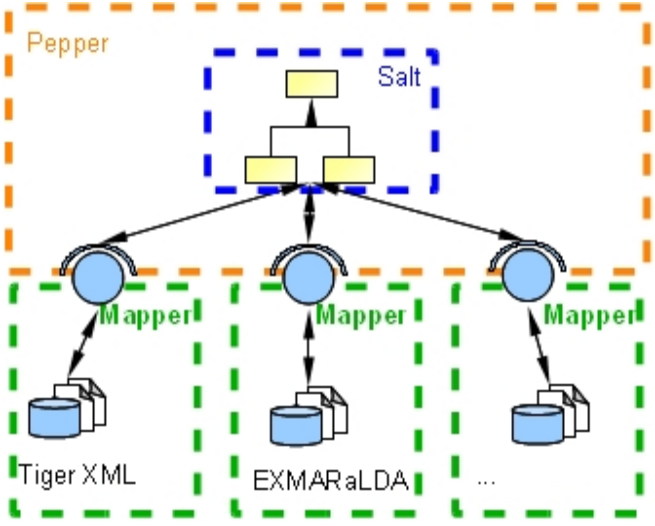

figure 12: architecture of the converter framework Pepper and the relation between the components of Pepper

The framework controls the given workflow, for example importing a corpus from TigerXML [mengel00] and exporting it to the EXMARaLDA format [schmidt02] via Salt. It creates a common instance of the Salt model, which can be used by mappers to import, or export their data. A mapper has to realize a mapping from an external format to the Salt instance, a mapping from the Salt instance to an external format, or both. A mapper is implemented in terms of a module, which can be plugged into the framework. Such a module can either be 1) an import module, 2) a manipulation module or 3) an export module.

1) An import module maps data from external formats to a Salt instance.
2) A manipulation module can manipulate a Salt instance, for example by changing the names of an annotation to upper case or to ISOCat data points.
3) An export module maps data from a Salt instance to an external format.

The example in figure 13 describes a mapping for an import module between TigerXML [mengel00] and Salt, with respect to the persistence and the model layer. The mapping can be described as

map: TigerXML → Salt

and can be done in two ways.

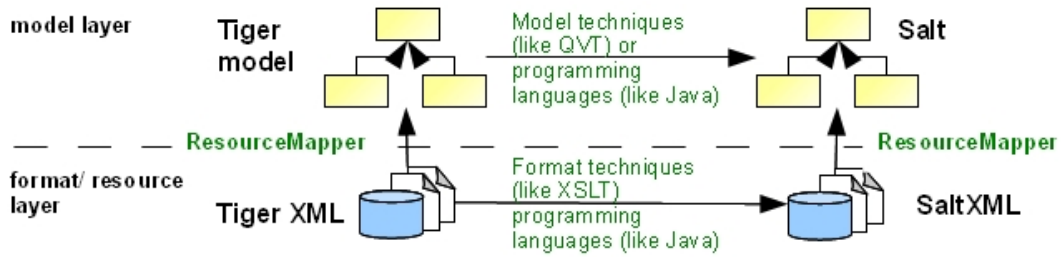

figure 13: *two different mechanisms to map data from the format Tiger XML to a Salt model (the first way via Tiger XML → Tiger model → Salt, the second way via Tiger XML → SaltXML → Salt).*

Both ways address different technical mechanisms, the first one handles the mapping via format techniques with no abstraction between persistence layer and conceptual layer and the second one handles a conceptual mapping on the conceptual layer. For the second way we need to have a mapping between model and format. For this we can modularize the tasks in creating a mapping between model and format for example to the format developer and in creating a mapping, which can be done by another person or team. Figure 14 shows the representations of the three stages of the first way: 1) the data in the origin format Tiger XML, 2) the data in a Tiger model representation and 3) the data in a Salt model representation.

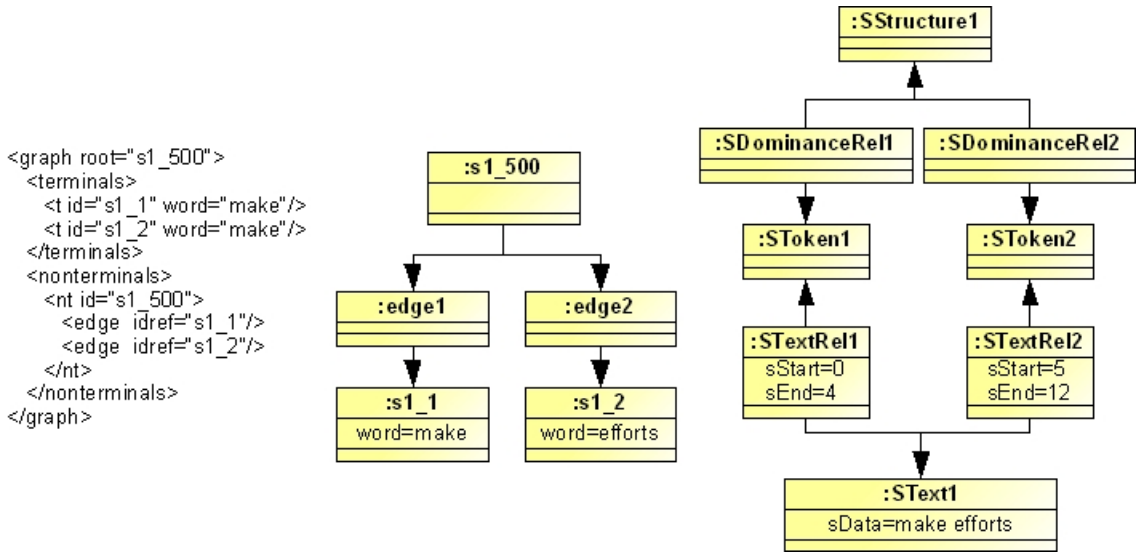

figure 14: on the left side: an example of data in the Tiger XML format; in the middle: the same example in the model of Tiger XML; on the right side: also the same data in a Salt model

Model based developing of mappings on a conceptual layer becomes much easier especially if a usable API also exists. In the case of using programming languages, one has a well-defined, context specific object model to map with, instead of working with a general model, e.g. a DOM model.

4.3 Evaluation

There are two ways To attach GrAF to Salt: 1) GrAF can be treated as an actual format, therefore a mapper can be implemented and plugged into the Pepper framework or 2) GrAF can be used as a native resource of Salt. GrAF then gains the same status as the automatically generated format Salt-XML[3]. The second approach makes Salt and GrAF become closer and will melt them as a unit consisting of a format and a model. This would be helpful for both, Salt gets a standardized format for persisting data and GrAF gets a processable API with a defined model.

Both ways need an isomorphic mapping, the general way of mapping was shown in section 2, but some losses remain in terms of the element types of Salt. As shown above, Salt elements such as edges have types: for example they can define a dominance, a coverage relation and further more between nodes. GrAF includes a type attribute for nodes, but no defined value domain, so the mapping from Salt to LAF/GraF can be made, but the way back would be difficult, if the attribute does not contain Salt-types.

Another loss also occurs for the recursive structure of annotations in Salt. As long as features in GrAF [ide07] cannot contain feature structures, an annotation of an annotation is not possible.

The current implementation of Pepper covers modules for the mapping between Salt and the formats EXMARaLDA [schmidt02], TigerXML [mengel00], TreeTagger [schmid94], PAULA [dipper05] and relANNIS (the relational format of the search and visualization system for multilevel linguistic corpora: ANNIS [zeldes09]). These data can be represented in Salt. To support other formats it must be discovered if the structure of Salt is powerful enough to cover them, or if Salt has to be expanded.

3 automatically generated by the modeling framework used, EMF [steinberg09], as mentioned in section 1

5 References

[burnard08] Burnard, L. and Bauman, S., editors (2008). TEI P5: Guidelines for Electronic Text Encoding and Interchange. Oxford. http://www.tei-c.org/Guidelines/P5/.

[dipper05] Stefanie Dipper (2005) XML-based Stand-off Representation and Exploitation of Multi-Level Linguistic Annotation. In: Eckstein R, Tolksdorf R (Hrsg.) Berliner XML Tage.

[ide07] Nancy Ide, Keith Suderman (2007) GrAF: A Graph-based Format for Linguistic Annotations. In: Proceedings of the Linguistic Annotation Workshop, Prague, Czech Republic.

[iso24610-1] ISO:24610-1 (2005). Language resource management – feature structures – part 1: Feature structure representation. ISO/DIS 24610-1, 2005-10-20.

[iso24611] ISO:24611 (2005). Language resource management – Morphosyntactic annotation framework (MAF). ISO/CD 24611, ISO TC 37/SC 4 document N225 of 2005-10-15.

[iso24612] ISO:24612 (2008). Language resource management – Linguistic annotation framework. ISO/WD 2461[2], ISO TC 37/SC 4 document N463 rev00 of 2008-05-12.

[iso24615] ISO:24615 (2009). Language resource management – Syntactic annotation framework (SynAF). ISO/CD 24615, ISO TC 37/SC 4 document N421 of 2009-01-30.

[kemps09] Kemps-Snijders, M., Windhouwer, M., Wittenburg, P., & Wright, S. E. (2009). ISOcat: Remodeling metadata for language resources. International Journal of Metadata, Semantics and Ontologies (IJMSO), 4(4), 261-276.

[lezius02] Wolfgang Lezius (2002) Ein Suchwerkzeug für syntaktisch annotierte Textkorpora. http://www.ims.uni-

[mengel00] Andreas Mengel, Wolfgang Lezius (2000) An XML-based encoding format for syntactically annotated corpora. In: Proceedings of the Second International Conference on Language Resources and Engineering (LREC 2000), Athen. stuttgart.de/projekte/corplex/paper/lezius/diss/.

[miller03] J. Miller, J. Mukerji (2003) MDA Guide Version 1.0.1. Object Management Group (OMG).

[schiller95] A. Schiller, S. Teufel, and C. Thielen (1995). Guidelines für das Tagging deutscher Textkorpora mit STTS. Technical report, Universität Stuttgart and Universität Tübingen.

[schmid94] Helmut Schmid (1994) Probabilistic Part-of-Speech Tagging Using Decision Trees. In: Proceedings of International Conference on New Methods in Language Processing.

[schmidt02] Thomas Schmidt (2002) EXMARaLDA - ein System zur Diskurstranskription auf dem Computer. Arbeiten zur Mehrsprachigkeit, Folge B 34:1 ff. http://www.exmaralda.org/files/AZM.pdf.

[steinberg09] David Steinberg, Frank Budinsky, Marcelo Paternostro and Ed Merks (2009) EMF: Eclipse Modeling Framework 2.0. Addison-Wesley Professional.

[zeldes09] Amir Zeldes, Julia Ritz, Anke Lüdeling, Christian Chiarcos (2009) ANNIS: A Search Tool for Multi-Layer Annotated Corpora. In: Proceedings of Corpus Linguistics 2009, Liverpool, July 20-23, 2009.