A LAF/GrAF based Encoding Scheme for underspecified Representations of syntactic Annotations

Manuel Kountz*, Ulrich Heid†, Kerstin Eckart†

*Graduate Programme GRK 609, †Institute for Natural Language Processing (IMS)
Azenbergstraße 12, D-70174 Stuttgart
{kountzml|heid|eckartkn}@ims.uni-stuttgart.de

Abstract

Data models and encoding formats for syntactically annotated text corpora need to deal with syntactic ambiguity; underspecified representations are particularly well suited for the representation of ambiguous data because they allow for high informational efficiency. We discuss the issue of being informationally efficient, and the trade-off between efficient encoding of linguistic annotations and complete documentation of linguistic analyses. The main topic of this article is a data model and an encoding scheme based on LAF/GrAF (Ide and Romary, 2006; Ide and Suderman, 2007) which provides a flexible framework for encoding underspecified representations. We show how a set of dependency structures and a set of TiGer graphs (Brants et al., 2002) representing the readings of an ambiguous sentence can be encoded, and we discuss basic issues in querying corpora which are encoded using the framework presented here.

1. Introduction

Ambiguity poses a problem for syntactic annotation of text corpora. Although a parser may produce a fully disambiguated analysis, as e.g. a probabilistic (tree-bank) parser would do, one may opt for a non-disambiguated analysis for several reasons. First, one might be interested in the precise nature, distribution and frequency of ambiguities which can be observed in corpus data[1], in which case a disambiguated corpus is obviously of no use. Second, automatic disambiguation is not perfect nor uncontroversial (and neither is manual disambiguation), thus we might loose the syntactic reading intended by the author by forcedly disambiguating the syntactic structure of a sentence. Third, we might not be able or not interested in calculating fully disambiguated structures, because the nature of our linguistic task does not require that, or because a certain state of tools and resources is to be documented and shared, without any stipulations.

In all these cases, we need an annotation data model which allows us to represent ambiguous structures without producing invalid data structures (e.g. tree fragments instead of a tree) or forcing us to use means of representation in a non-canonical way (fragmentary parses, for example, are often represented with the root nodes of tree fragments connected to a non-linguistic artificial root node).

In this paper we will outline a data model for underspecified representations of syntactic structures in text corpora, based on the upcoming ISO standard *Linguistic Annotation Framework* (LAF; Ide and Romary, 2006). Our representation is designed to be a generic encoding scheme for underspecified representations. We use dependency structures to exemplify our encoding scheme.

The main requirement to our representation is to be *informationally efficient*:

- The representation should encode as much information as is necessary to correctly describe those parts of an analysis which are not subject to ambiguity;
- The representation should avoid unnecessary overhead in encoding variation, e.g. between the structures of different readings.

This sounds trivial, but is not easy to achieve. The second requirement above actually constitutes a trade-off, as the threshold for being unnecessary may in practice vary with the task. In section 2. we examine generic requirements for underspecified representations of dependency structures.

Informational efficiency intrinsically allows for compact storage, as only a minimum amount of information needs to be stored. It furthermore helps to reduce inconsistencies in the annotations, as it eliminates the need to check for consistency several encoded instances of the same piece of information.

While treebanks are often conceived of as containing only intended, thus (manually) disambiguated syntactic structures, some parsers produce *packed parse forests* (Dörre, 1996) to represent those structural ambiguities which cannot be eliminated during parsing (e.g. Bitpar: Schmid, 2004). Packed parse forests essentially encode local differences between structures by means of spelling out local alternatives. Our representation allows to specify such spelled-out disjunctions, but goes further by allowing to specify interval-like constraints (e.g. for a highest and a lowest node to which a PP can attach). Furthermore, we allow for interdependencies between constraints to be specified explicitly, i.e. not only by mere parallelism between disjunctions.

Using a pre-existing framework as the basis for a new encoding scheme increases reusability of data encoded in the new scheme. Reusability here has two aspects:

1. *Re-using tools:* Existing tools can be used to analyse or to modify the data, perhaps with little modifications to the tools. This facilitates investigations based on the data.
2. *Re-using the data:* The data encoding can be interpreted more easily because it shares concepts, data model, and representational means with other encod-

[1]The data model described in this paper was originally designed for a corpus of syntactically ambiguous sentences collected for the study of syntactic ambiguity.

ing formats. Thus, sharing the data among projects and re-using it in later projects becomes easier.

We base our encoding on an upcoming ISO standard for corpus encoding, thus we should be able to profit from upcoming tools and methods based on this standard, as well as being able to contribute to the development of this standard and the collection of pertaining tools.

2. Requirements for a dependency-based Representation

Dependency structures encode governor-dependent relations between individual words; a dependent is seen to bear a certain role with regard to its governor. Thus, a dependency structure can be represented by a directed acyclic graph, with tokens as nodes and pair-wise dependency relations as edges. Each node is labelled with features pertaining to the particular word it represents, like part-of-speech tag, lemma, and morphosyntactic analysis, while an edge is annotated with the grammatical relation it encodes. The distinction between governor and dependent is encoded in the direction of the edge, which is either governor to dependent or dependent to governor (as used by e.g. Schiehlen, 2003; Schröder, 2001). The following example shows dependency edges directed from dependent to governor taken from an analysis for *John buys a book*:

John $\stackrel{SB}{\longrightarrow}$ buy

book $\stackrel{OBJ}{\longrightarrow}$ buy

a $\stackrel{SPEC}{\longrightarrow}$ book

There are three kinds of possible ambiguities in dependency structures:

1. *Alternative labellings* of edges or nodes: Edge label alternatives occur e.g. with subject-object ambiguities, node label alternatives e.g. with homographic forms which may receive multiple part of speech tags. Examples:

   (1) Hans $\stackrel{SUBJ|OBJ}{\longleftarrow}$ liebt $\stackrel{OBJ|SUBJ}{\longrightarrow}$ Maria

   (2) Er bevorzuge/VVFIN ein gutes Arbeitsklima, sie dagegen sichere/VVFIN|ADJA Arbeitsplätze

2. *Alternative topologies*, where a particular node can be a dependent of two (or more) possible governors: This is the case e.g. for PP attachment ambiguities. Example:

   (3) Klaus beobachtet den Mann mit dem Fernrohr.
   Klaus watches the man with the telescope.

3. *Interdependent ambiguities*, where one option out of one set of possible alternatives depends on the choice of an option in another set of alternatives: for example, an ambiguitiy in the attachment of a PP to either a verb or a noun may only occur if the PP can be analysed as adjunct to the verb. If the PP can be alternatively labelled as subcategorised PP (i.e. a prepositional object), the ambiguity disappears. Thus, the structural ambiguity depends on the labelling ambiguity.

   To outline our approach, we will refer to example (4) throughout the paper:[2]

   (4) Karl sieht nur Schrott in seinem Wagen
   - a *Karl sees nothing but scrap, which is inside his car.*
     (PP as noun adjunct)
   - b *Karl is in his car, and sees nothing but scrap*
     (PP as verb adjunct)
   - c *Karl regards his car as nothing better than scrap.*
     (PP as POBJ, attachment not ambiguous)

Encoding by constraints. Our approach to encoding these types of ambiguities is to specify *constraints* for those parts of the annotation which are different between individual readings. Such constraints need to be as generic as possible to cover all possible readings, while being as specific as is necessary to prevent illegal instantations which lead to illegal structures or unwanted readings (e.g. readings ruled out by partial disambiguation). Each type of ambiguity introduces an own class of constraints, namely *labelling constraints*, *structural constraints*, and *constraint interdependencies*. These are discussed in detail in section 4.1.
The number of nodes in a dependency structure is directly correlated with the number of words in a sentence. Thus an underspecified representation of dependency structures normally can not contain constraints which are instantiated by nodes. Representations of phrase structures, on the other hand, may contain constraints for the existence of nodes, namely if a constraint controls optional adjunction. In example (4) we can optionally build a complex NP from *Schrott* and the PP *in seinem Wagen*, which means that we can have an additional node above the node pertaining to *Schrott* (cf. section 5.).
This means that there may be nodes which instantiate constraints describing an alternative local phrase structure. However, this does not introduce a new type of constraint; if we can put constraints on the arrangement of partial structures, new nodes can be (part of the) instantiations of such arrangement constraints.
Representations for all three kinds of ambiguities (as outlined above) should be encoded with as few modifications and extensions to GrAF as possible.

3. The GrAF Data Model and packed Structures

As far as we can see, there have not been many proposals for representing syntactic ambiguity in corpora. In treebanks, the problem does not occur, because annotators are trained to identify the intended reading of the sentences they deal with. On the other end of the scale, formats for detailed corpus annotation, such as Annotation Graphs (Bird and Liberman, 2001) and the ATLAS tools

[2]The scope ambiguity introduced by the adverb is silently ignored for simplicity.

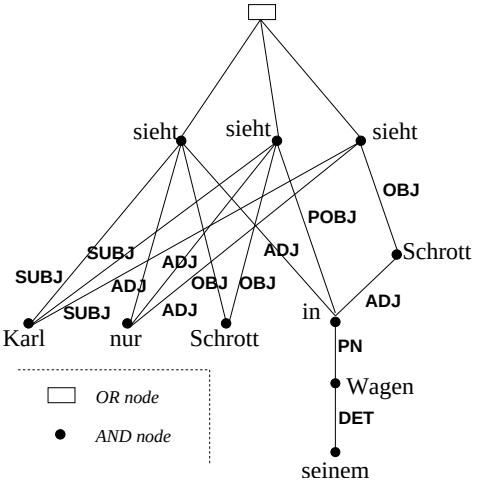

Figure 1: Example (4) represented by an AND-OR tree of dependency structures.

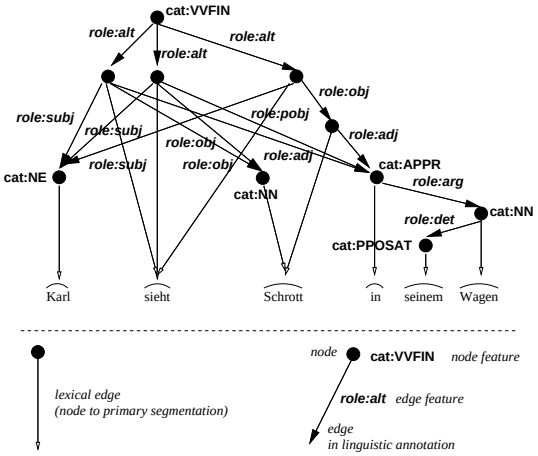

Figure 2: Graphical depiction of the GrAF representation of example (4)

(Laprun et al., 2002), or NITE and NXT (Carletta, et al., 2004) aim at representing alternative annotations of arbitrary complexity. For syntactic ambiguities, this implies that such formats don't aim at informational efficiency, but rather list different completely disambiguated structures, even if they partly overlap. More recent proposals, such as Eckart and Teich (2007), focus so far on other types of linguistic annotations. The SALSA corpus (cf. Burchardt et al., 2006) is based on the TiGer treebank (Brants et al., 2002) and thus does not deal with structural ambiguity; for its semantic annotation, it allows for local underspecification at the level of semantic frames (of Frame Semantics, cf. Baker et al., 2003) and roles: if a given syntactic structure may receive two alternative frame semantic interpretations, different roles may point to the same constituent. As also this is only a partial account, our proposals will need to be compared with the LAF/GrAF approach proposed by Ide and Romary (2006).

In LAF/GrAF, the most basic layer is a so-called *primary segmentation*, e.g. a conventional tokenisation, of the input or *primary data*[3] which is represented as a set of edges defined on elementary parts of the primary data. Edges in the primary segmentation are viewed as nodes when being referred to by annotations from higher levels, so-called *linguistic annotations*. Linguistic annotations consist of nodes and edges; annotations are encoded in feature structures attached to nodes and edges.

If a span of primary data receives multiple alternative annotations, e.g. several syntactic structures, this can be handled in two ways in GrAF: either both annotations are stored independently, or the alternative structures are *merged* into a combined structure. When two or more nodes cover the same sub-span, they can be joined by a dummy parent node, which in turn has the two alternative nodes as descendants (cf. figure 2).

This representation scheme resembles packed parse forests represented as AND/OR trees as decribed e.g. by Dörre (1996), cf. figure 1.

GrAF appears to restrict annotations of edges to technical items like indices within multipart tokens and suggests that dependency relations be annotated to special nodes inserted into the word-word edges. We assume in the following that edges *are* labelled with relations, but obviously the transformations necessary to obtain a representation fully compliant with GrAF are fairly easy to implement.

4. A flexible encoding scheme based on LAF/GrAF

The aims of the representation scheme we propose are as follows:

1. The representation scheme should be informationally efficient,
2. the representation scheme must allow to cleanly encode underspecified representations of syntactic structures,
3. and the representation scheme must be as flexible as possible.

Requirement (1) has been discussed in the introductory section 1. The requirement for a clean encoding of underspecified representations (2) both forbids to use representational means in non-canonical ways and implies to produce encodings easy to read and to understand. For example, using pseudo-nodes to represent constraints or disjunctions (cf. AND-OR trees, Dörre 1997) would mean to introduce a special category of nodes (and edges), which are not part of the linguistic description as such, but serve technical purposes; cf. for example the OR-node and the vast number of reduplicated edges in figure 1. On the other hand, several alternative elements (e.g. the edge linking the PP in example (4) to either verb or noun) can just be included in the

[3] Primary data is, in the sense of a true stand-off annotation, only referenced, never copied or modified.

representation without further marking them as belonging to the same set of options; reconstructing the structure of one specific reading means that first we must detect (by examining a large part of the representation) that there is a set of options, before one specific option can be selected.

The requirement of flexibility (3) dictates that the representation format must not be limited to means of representation which are specific to a particular type of underspecified representations. If, for example, the only means of representation is disjunction, difficulties arise with interdependent ambiguities, as interdependencies cannot be expressed[4]. Furthermore, the means of representation must not be tied to a specific kind of annotation or annotation scheme, but should operate on the abstract properties of the annotations (e.g. on the fact that both dependency structures and TiGer structures are graphs). Thus, the format need not be redesigned at large for adapting it to a new type of syntactic annotation format.

To ensure this, we propose an extension to the LAF/GrAF data model which allows to specify constraints over structures in a generic way. The data model defined by LAF/GrAF, which has been described in section 3., represents syntactic ambiguity by a construction equivalent to packed parse forests. However, the resulting representation introduces dummy nodes and edges marking alternatives, which means that strictly syntactic parts of the description are interspersed with strictly technical elements. To avoid this, we extend the data model by generic *constraints*.

4.1. Constraint extension of the LAF data model

Our extension of the LAF data model consists of a *constraint list*, which contains all constraints for arranging and labelling the structures described by a particular document. The constraint list may contain three kinds of constraints, namely *structural constraints*, *labelling constraints*, and *constraint interdependencies*.

The constraint list contains all constraints imposed on a certain collection of partial graphs. These partial graphs are encoded as defined by LAF/GrAF, i.e. as a list of edges and a list of nodes.[5] The partial graphs (or graph fragments) need not be connected; for increased compatibility with tools which are not aware of underspecified representations as defined in this paper, one reading may be picked arbitrarily.[6]

Structural constraints define relations between those parts of the structure which can be deterministically recognised, that is, which do not contain ambiguities. Parts of annotations in LAF/GrAF can be referenced by their *identifiers* (the XML attribute `id` on nodes and edges). Thus, a structural constraint is a triple $\langle N_G, N_F, t\rangle$, where N_F is the root node of a graph fragment which needs to be placed in the structure. It may be placed inside a subgraph whose root is the node N_G. t tells the way of combining N_G and N_F. We define three types[7] as follows:

1. BELOWAPPROPRIATELY means that N_F will be placed in any *appropriate* place inside the structure whose root N_G is. The grammar defines which places are appropriate.
   If N_F is a preposition in a dependency structure, and N_G is a noun, with other prepositional phrases between N_G and N_F, then N_F may be adjoined to any noun between N_G and N_F which sits on the right edge of a noun phrase. This is the case in *Peter calls the* ***man***$_{N_G}$ *behind the* ***telescope*** *on*$_{N_F}$ *the hill*, with the nodes marked in bold being possible governor nodes for the preposition *on*.

2. BELOWWITHOPTIONS is essentially the same as BELOWAPPROPRIATELY, but the possible points of attachment are given in the constraint instead of by grammatically defined appropriateness.

3. ATTACHMENTOPTION means that N_F may be attached to N_G in particular, but to no other node inside the fragment whose root N_G is. This constraint is not very useful in isolation, but needs to be combined with other constraints.

Instantiations of structural constraints are generally edges. In dependency structures, all nodes are known in advance, thus it is sufficient to allow constraints to just be instantiated by adding an edge between two existing nodes. But in other formalisms not all nodes may be present in advance. In TiGer structures (see section 5.), for example, some nodes may only be present if needed as source or target of an edge; if present, they incur creation of additional edges. For example, a bare noun (without article, adjective, or prepositional adjuncts) may be directly attached to a verb as an object. If a PP is attached to the noun, an NP node is constructed which becomes the object of the verb, and the noun and the PP are attached to this new NP node. That is, attaching the PP results in creating an NP node, the edge from the PP to the NP, and a new edge linking the NP to its head noun.

The semantics for the instantiation of a structural constraint we assume is a three-step procedure: The first step is to construct the *target edge*, as immediately specified by the constraint. This edge may require as a *precondition* a node to be present, which is first sought within the structure; if it exists, it is used. Otherwise, a new node is created, and along with this new node all edges are created which the grammar requires, as a *side effect*. See section 5. for an example of the above case of attaching a PP to an NP.

Labelling constraints define options for labelling parts of the structure. A labelling constraint is basically a triple $\langle S, L, t\rangle$, where S is a structural element which receives a

[4] Parallel disjunctions may incur duplication of elements.

[5] Mathematically: sets of uniquely identifyable elements.

[6] In a scenario where a parser always delivers one unique structure which is guaranteed to contain all fragments of possible structures, and which is 'augmented' later on with information which permits to construct the structures of all possible readings (cf. Spranger and Kountz, 2007), this structure may be picked as the reading spelled out. However, this reading has no particular status with regard to the other readings whose structures are not stored explicitly.

[7] More types may need to be defined in order to encode more sophisticated underspecified representations than those we conceive of here.

label as specified by L. t indicates the interpretation of L as follows:

- LABELSET indicates that L is a set of possible labels for S. S may receive exactly one element out of the set given in L.
- LABELSUBTYPE indicates that L is a supertype of the possible labels which S may receive. S is labelled with a label which has a terminal subtype of the type given in L.
  That is, if there are subtypes of the type given as L which themselves are supertypes $L_1, \ldots$ of subtypes $l_{1,1} \ldots$, then these non-terminal subtypes $L_1, \ldots$ cannot be labels of S. Instead, only any of their terminal subtypes can be an instance of the label of S.

  Many structural elements have more than one label, thus it is necessary to specify the type F of label which is constrained; Labelling constraints thus become *quadrupels* $\langle S, F, L, t \rangle$, with S,L,t as specified above.

The object S to which a labelling constraint refers may be an existing node or edge, or a node or edge which results from instantiating a constraint. In the latter case, there is no identifier to which a labelling constraint could refer; thus, a labelling constraint has to refer to the structural constraint which, by instantiation, allows to introduce the labelled node or edge.

Constraint interdependencies allow to specify interdependencies between particular instantiations of other constraints. Basically, a constraint interdependency encodes a relation between two (classes of) instantiations v_A and v_B of two constraints A and B. We specify the following relationships:

- ENFORCES: Instantiating constraint A by v_A enforces the choice of v_B as instantiation of constraint B.
- PRECLUDES: If v_A instantiates constraint A, then v_B may not instantiate constraint B.

A constraint interdependency is described by a quintuple $\langle A, B, v_A, v_B, t \rangle$, where A, v_A, B, v_B are constraints and possible instantiations as described above, and t is one of said types ENFORCES or PRECLUDES.
References to structural constraints are made in a shorthand manner: The structural constraints we defined above can only be instantiated by edges from *some* governor to *one* particular fragment root, thus only one end of the edge varies. This is exploited by referring to the varying node in v_A or v_B attributes of a constraint interdependency.

4.2. The GrAF encoding of the constraint extension exemplified

We exemplifiy the linearisation of the extended LAF data model described above by giving one possible encoding for example (4), analysed in dependency structures.[8] Figure 3 shows the fragments of the analyses of readings (4a,b,c). The nodes v_1, v_4, and v_5 are those nodes that need to be taken into consideration when constraining the arrangement of these two fragments.

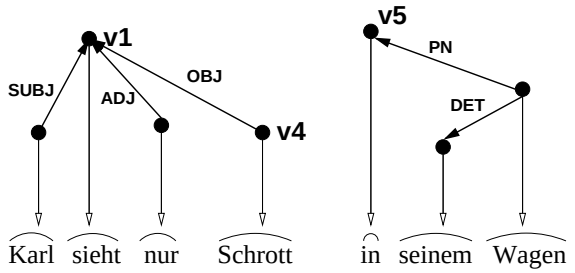

Figure 3: Fragments and nodes referenced by constraint in example (4)

Example (4) can be encoded using any of the three types of structural constraints given in section 4.1. Here, we opt for a constraint of the type BELOWAPPROPRIATE and explain which grammatical knowledge must be available when spelling out the particular readings. We also shortly discuss the differences between this encoding and an encoding using BELOWWITHOPTIONS.
The fragments given in figure 3 can be encoded as given in figure 4. The arrangement of the two fragments is ruled by following constraint:

$$\langle v_1, v_5, \text{BELOWAPPROPRIATE} \rangle$$

In order to instantiate a constraint of type BELOWAPPROPRIATE, a notion of "being appropriate" is needed. Here, a prepositional phrase is to be attached; thus, only verbs and nouns are appropriate candidates to which the PP might be attached. Furthermore, a PP can only attach to a noun immediately left of it, which rules out *Karl*.
The constraint list we need contains exactly one constraint, which is the structural constraint described above:

```
<constraint-list>
  <structural-constraint
       id="c1" type="BelowAppropriate"
       gov="v1" frag="v5" />
</constraint-list>
```

As indicated above, we chose this type of constraint over the alternative encoding using a structural constraint of the type BELOWWITHOPTIONS. In this alternative solution with BELOWWITHOPTIONS, a list of possible points of attachment would have been necessary, as in the following XML fragment:

```
<constraint-list>
  <structural-constraint
        id="c1" type="BelowWithOptions"
        gov="v1" frag="v5"
        options="v1 v4" />
</constraint-list>
```

[8] We represent the predicative NP *Schrott* in the reading *consider sth. as sth* as an accusative object. Obviously, the predicative could equally be represented by means of a specific grammatical relation; however, the analysis as an accusative object is in line with the TiGer treebank and Engel (1996).

```
 1 <!-- primary segmentation omitted  -->
  ...
10 <!-- nodes -->
11 <node id="v1">
12   <f name="cat" value="V"/>
13 </node>
14 <node id="v2">
15   <f name="cat" value="NE"/>
16 </node>
17 <node id="v3">
18   <f name="cat" value="ADV"/>
19 </node>
20 <node id="v4">
21   <f name="cat" value="N"/>
22 </node>
23 <node id="v5">
24   <f name="cat" value="PREP"/>
25 </node>
26 <node id="v6">
27   <f name="cat" value="PPOSS"/>
28 </node>
29 <node id="v7">
30   <f name="cat" value="N"/>
31 </node>
32 <!-- dependency edges -->
33 <edge type="dep-rel"
       id="e1" from="v2" to="v1">
34    <f name="role" value="SB"/>
35 </edge>
36 <edge type="dep-rel"
       id="e2" from="v3" to="v1">
37    <f name="role" value="ADJ"/>
38 </edge>
39 <edge type="dep-rel"
       id="e3" from="v7" to="v5">
40    <f name="role" value="PN"/>
41 </edge>
42 <edge type="dep-rel"
       id="e4" from="v4" to="v1">
43    <f name="role" value="OA"/>
44 </edge>
45 <edge type="dep-rel"
       id="e5" from="v6" to="v7">
46    <f name="role" value="SPEC"/>
47 </edge>
48 <!-- lexical edges
   (from nodes v1..v7 to the primary segments)
    omitted -->
```

Figure 4: Fragments in figure 3 encoded in XML

The advantage of using a BELOWWITHOPTIONS constraint is that the whole annotation is self-contained insofar as no additional knowledge about appropriateness needs to be available. A query tool, for example, could just produce the two instantiations by examining the constraint. On the other hand, for this solution we must encode more information, namely that v_1 and v_4 are possible points of attachment.

In any case the instantiation of the above structural constraint depends on the label assigned to the edge which is to instantiate the structural constraint. If the edge is to be labelled as a prepositional object (POBJ), the instance $\langle v_4, v_5 \rangle$ (attachment to noun) is ruled out – the noun *Schrott* cannot have a prepositional object.

We first define which labels are possible for this edge by a labelling constraint (which has the ID `c2` in the XML fragment below), then constrain the choice of the edge which instantiates the above structural constraint (`c1`) by a constraint interdependency (`c3`).

```
<constraint-list>
  <structural-constraint
       id="c1" type="BelowAppropriate"
       gov="v1" fragment="v5" />
  <labelling-constraint
       id="c2" type="LabelSet"
       reference="c1"
       labels="ADJ POBJ" />
  <constraint-interdependency
       id="c3" type="Enforces"
       a="c2" aValue="POBJ"
       b="c1" bValue="v1" />
</constraint-list>
```

Efficiency vs. Informativity. Chosing the best encoding of a certain phenomenon of ambiguity turns out to be a trade-off. The encoding scheme we propose is designed to be flexible enough to accomodate both the needs of a query tool, namely self-containedness, and the need of encoding information about ambiguities as efficiently as possible.

Corpus annotation in general has to trade off informativity for space. In the first case, the corpus fully documents the analyses of the sentences with regard to a particular grammar. A tool reading this corpus does not need much more than built-in knowledge about the representation format as such to decode the corpus. On the other hand, storing only the minimun amount of information necessary to describe the linguistic analysis is difficult or impossible. In the second case, the encoded corpus data as such becomes small, at the expense of efficiency in software which reads the corpus, such as query processors. The software reading a corpus needs very detailed grammatical knowledge (such as the information that only verbs and nouns may be modified by PPs). Furthermore, the corpus does not fully serve as a documentation of linguistic phenomena, as part of the knowledge about the constructions is only available as a separate grammar.

Our aim is informational efficiency, thus we basically adopt the second view with regard to the *representations* we propose. However, the *encoding scheme* laid out here is designed to be versatile, and thus it should serve both for encoding maximally efficient representations and for encoding more self-contained annotations.

5. Representation of TiGer structures

In the TiGer treebank (Brants et al., 2002) sentences from newspaper text are annotated with syntactic analyses using a special, hybrid syntax which describes a sentence in terms of dependency and phrase structure. The phrase structure

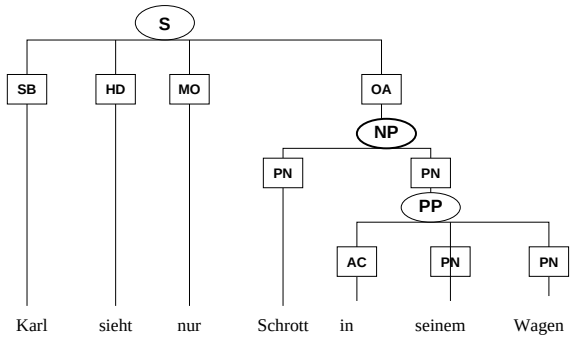

Figure 5: Noun adjunct reading (4a) of example (4) in TiGer structures

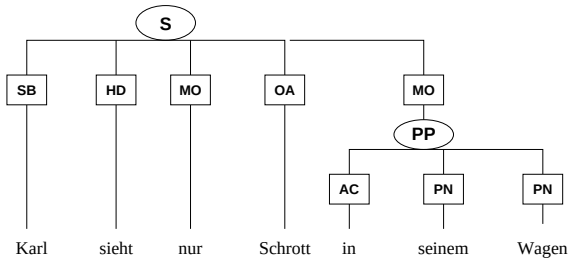

Figure 6: Verb adjunct reading (4b) of example (4) in TiGer structures

annotation is flat, and nodes are only introduced if they cover more than one terminal elements (no unary branching nodes are used). Dependency relations are as well intentionally underspecified in many cases, e.g. the many possible sisters of a noun (like article, adjective, and PP) are annotated as PN (part of noun).[9]

For the TiGer treebank as such, there is no need to represent ambiguity, as it is seen as part of the task of the annotators to fully disambiguate the syntactic analyses they assign to sentences. However, one might be interested in representing ambiguities while using TiGer-like structures as the syntactic annotation format in a NLP tool, e.g. to be able to compare output from the tool against the TiGer treebank.

In this section, we exemplify how a representation of syntactic ambiguities based on TiGer structures could be defined within the framework we lay out in this paper. Again, we use example (4); assuming the current TiGer grammar (release 2.1, 2006-Aug-24), the *in*-PP is uniformly annotated as `MO` (modifier), thus there is no equivalent to reading (4c). For reading (4a), an analysis by a TiGer structure is given in figure 5, while reading (4b) receives the structure given as figure 6.

Reading (4a) differs from reading (4b) in so far that it has an additional node: the NP headed by *Schrott* is annotated with a separate node, as it covers more than one token. This node and the necessary `PN` edge from the NP node to the head noun is created automatically when the structural constraint which rules the attachment of the PP becomes instantiated. We use a BELOWAPPROPRIATE constraint; otherwise, we would need to store a precomputed NP node in the corpus.

```
<constraint-list>
  ...
  <structural-constraint
      id="c2" type="BelowAppropriate"
      gov="n501" fragment="n502" />
</constraint-list>
```

6. Querying

In this section we outline how a corpus could be queried which is encoded according to the framework presented in this paper. This will be done in two steps: First, we sketch the querying of a graph representation as the one GrAF uses, i.e. a representation using separate node and edge lists. Second, we show how the challenge of taking constraints as the ones we specified above can be solved.

6.1. Querying the basic Graph Representation

As our encoding scheme basically allows to constrain the arrangement of graph fragments, it builds on a representation for graphs; in our case, the basis is the representation specified by GrAF. GrAF represents graphs as two sets (encoded as lists), one containing all nodes and the other all edges. Edges refer to the nodes by XML identifiers. Both nodes and edges may be labelled with feature structures (cf. section 3. for the LAF/GrAF data model and encoding).

Queries for graphs essentially specify constraints for structure and labelling of parts of the graphs which are desired as a result. For the labels, this amounts to giving feature structures and checking whether a given feature structure in the query subsumes a feature structure in the node or edge list of a sentence in the corpus. This leads to sets of candidate nodes and edges for each part of the query, represented by their identifiers.

Structural parts of queries basically consist of sets of edge templates, where the endpoint nodes of the edges are specified as said above. The matching procedure now has to select those entries in the edge lists of sentences which match the conjoined edge templates as given by the structural part of the query.

Besides edge sets proper, search predicates allow to search lists of candidate edges by applying more abstract checks.

[9] TiGer structures are usually depicted as trees, where all phrase structure nodes are represented by their respective label, enclosed in an elliptical frame. Edge labels are drawn as small rectangels on the edges. The TiGer edge categories used in the figures are:
`SB` — subject, `HD` — head (e.g. of a sentence), `MO` — modifier (adverb or PP), `AC` — 'adpositional case marker', i.e. the preposition in a PP, `PN` — 'part of noun', i.e. determiners, nouns, and adjectives in NPs and PPs. `OA` — accusative object. Especially PPs and NPs are annotated in a way that avoids decisions about which daughter node is the head of the structure. Cf. Brants et al. (2002).

These checks may, for example, examine the intersection of the candidate edges and the transitive closure of edge templates given in the query, e.g. for transitive dominance checks.
Implementation of abstract search predicates is much more complex than just to allow users to specify parts of the graphs they are interested in, because the search engine has to construct intermediate partial representations dynamically.

6.2. Querying a Representation containing Constraints

The constraints we specify are fundamentally similar to search predicates insofar as they also allow to construct additional structure during the instantiation of constraints. They are basically a declarative specification of parts of the structure which can be constructed by rule, in the same way as expressions over search predicates are declarative specifications of structure templates which are built when the search is executed.
In principle, the constraints given in the representation of a particular sentence have to be instantiated when a search pattern is checked whether it matches that sentences. Unlike the procedure for 'spelling out' high level search predicates, this process of instantiating constraints is not limited by a pre-defined set of edges and nodes, but has a much larger domain of candidate nodes and candidate edges. This is especially true for representations of sturctures for which additional nodes and edges have to be introduced during constraint instantiation, cf. section 5.

7. Conclusion

We presented an extension to the LAF data model and its GrAF serialisation which allows to encode underspecified representations of ambiguous sentences in a variety of underlying annotation structures. We exemplified this for dependency structures and TiGer structures; the latter impose the additional problem of nodes which need to be created during instantiation of constraints but do not exist in the structures of all possible readings. We showed that the framework presented here can handle such cases.

Future Work. The constraint inventory as presented in this paper is far from complete. Although the basic constraints we specify cover a garden variety of phenomena of ambiguity, it remains to be extended to more complicated cases.
The logic behind the interdependency of constraints is quite complex. We are only tackling very basic cases here, but it must be investigated further how exactly constraints can interact, and which restrictions can be placed on the interaction of constraints.

8. References

C. F. Baker,C. J. Fillmore, B. Cronin. 2003. The Structure of the FrameNet Database. *International Journal of Lexicography,* 16(3).

S. Bird, M. Liberman. 2001. A formal Framework for linguistic Annotation. *Speech Communication,* 33(1,2):23–60.

S. Brants, S. Dipper, S. Hansen, W. Lezius, G. Smith. 2002. The TIGER treebank. *Proceedings of the Workshop on Treebanks and Linguistic Theories.* Sozopol/Bulgaria.

A. Burchardt, K. Erk, A. Frank, A. Kowalski, S. Padó, M. Pinkal. 2006. The SALSA Corpus: a German corpus resource for lexical semantics. *Proceedings of the 5th International Conference on Language Resources and Evaluation.* Genoa/Italy.

J. Carletta,D. McKelvie, A. Isard, A. Mengel, M. Klein, M. B. Møller. 2004. A generic approach to software support for linguistic annotation using XML. In G. Sampson and D. McCarthy, editors, *Corpus Linguistics: Readings in a Widening Discipline,* chapter 39. Continuum International, London/New York.

J. Clark, S. DeRose. 1999. XML Path Language (XPath) Version 1.0 W3C Recommendation. The W3C Consortium.

J. Dörre. 1996. Efficient Construction of Underspecified Semantics under Massive Ambiguity. *Proceedings of the Meeting of the Association for Computational Linguistics,* (1997). 386–393.

R. Eckart, E. Teich. 2007. An XML-based data model for flexible Representation and Query of linguistically interpreted Corpora. In G. Rehm, A. Witt, L. Lemnitzer, editors, *Data Structures for Linguistic Resources and Applications,* (= Proc. of GLDV Conf. 2007), Tübingen/Germany.

U. Engel. 1996. *Deutsche Grammatik.* Julius Groos, Heidelberg/Germany, 3. korrigierte Auflage.

C. Laprun, J. G. Fiscus, J. Garofolo, S. Pajot. 2002. A practical Introduction to ATLAS. *Proceedings of the 3rd International Conference on Language Resources and Evaluation.* Las Palmas/Spain.

N. Ide, L. Romary. 2006. Representing Linguistic Corpora and Their Annotations. *Proceedings of the 5th Language Resources and Evaluation Conference.* Genoa/Italy.

N. Ide, K. Suderman. 2007. GrAF: A Graph-based Format for Linguistic Annotations. *Proceedings of the Linguistic Annotation Workshop,* 1–8, Prague/Czech Republic.

M. Schiehlen. 2003. A Cascaded Finite-State Parser for German. *Proceedings of the Research Note Sessions of the 10th Conference of the European Chapter of the Association for Computational Linguistics.* Budapest/Hungary.

H. Schmid. 2004. Efficient Parsing of Highly Ambiguous Context-Free Grammars with Bit Vectors. *Proceedings of the 20th International Conference on Computational Linguistics.* Geneva, Switzerland (2004).

I. Schröder. 2002. *Natural Language Parsing with Graded Constraints.* PhD thesis, Universität Hamburg, Fachbereich Informatik.

K. Spranger, M. Kountz. 2007. Efficient Ambiguity-Handling using underspecified Representations. In G. Rehm, A. Witt, L. Lemnitzer, editors, *Data Structures for Linguistic Resources and Applications* (= Proc. of GLDV Conf. 2007), Tübingen/Germany.