

## Mapping Phrase Structures to Dependency Structures in the Case of (Partially) Free Word Order Languages

Bernd Bohnet  
University of Stuttgart  
Intelligent Systems Institute  
Universitätsstr. 38  
70569 Stuttgart, Germany  
Bernd.Bohnet@informatik.uni-stuttgart.de

### Résumé – Abstract

Les corpus sont très utiles pour de nombreuses tâches dans le domaine du traitement automatique des langues naturelles. Les corpus annotés syntaxiquement sont devenus une ressource importante en TAL. Ils sont couramment utilisés, par exemple comme banc d'essai pour la génération, l'analyse et la désambiguïsation sémantique, et comme source pour l'acquisition de ressources (collocations, information sur la sous-catégorisation, extraction de grammaire). Lorsqu'on utilise les structures de dépendance pour le TAL, le manque de corpus annotés en structures de dépendance constitue un handicap. Nous présentons une approche fondée sur une grammaire de graphes pour convertir des corpus annotés en structures syntagmatiques en corpus annotés en dépendances. Cette approche fonctionne pour des langues à ordre de mots (partiellement) libre et fixe.

Corpora are very useful for many tasks in the area of natural language processing. Syntactically annotated corpora became an important resource in NLP. They are widely used, for example as test bed for generation and parsing, word sense disambiguation, as source for the acquisition of resources (e.g. collocations, subcategorization information, grammar extraction). When one uses dependency structures for NLP, the lack of corpora annotated by dependency structures is a handicap. We will present a graph grammar approach to map corpora annotated by phrase structures into corpora annotated by dependency structures. The approach works for (partially) free and fixed word order languages.

### Mots-clefs – Keywords

Corpus des structures de dépendance, grammaire de graphes, Compilation de structure syntagmatiques en structures de dépendance  
dependency annotated corpora, graph grammar, compiling PS into DS

# 1 Introduction

A considerable number of corpora annotated with phrase structures for different languages are available. For instance, for English the International Brown Corpus, the Christine Corpus, Penn Treebank and so on; for Chinese the Chinese Penn-Treebank, for Korean the Korean Penn-Treebank, for German the NEGRA and Verbmobil Corpus, etc. But there are only a few corpora annotated with dependency structures available, no dependency-annotated corpus for German exists. The question is how to get a corpus annotated with dependency structures with low effort. What suggests itself more than to convert a phrase structure annotated corpus into a dependency structure annotated corpus?

In this paper we adopt this strategy to map the phrase structure annotations of the NEGRA corpus, a middle size corpus of German onto dependency annotations. The mapping is implemented in terms of a graph grammar (Rozenberg, 1997). A graph representation is the most universal for linguistic structures. It accounts for (semantic) predicate-argument structures, (syntactic) tree structures, and linearized structures. Graph grammars combine the potentials and advantages of graphs and rules.

In order to perform that task we use the transductive graph grammar interpreter of the *MATE*-environment, cf. (Bohnet & Wanner, 2001), (Bohnet *et al.*, 2000). Only few extensions were necessary. The advantage of doing this over the implementation of a special purpose algorithm is that only rules have to be implemented which contain the knowledge in a declarative manner.

The structure of the paper is as follows: In section 2, we introduce the phrase structure formalism used in the Negra corpus, the dependency structures (Surface Syntactic Structures from the Meaning-Text Theory), and the tasks which have to be carried out to map PS into DS. In section 3, we describe then the mechanism for mapping the PS onto the DS and give an example. In section 3.5, we present the results of the application of our approach to the NEGRA Corpus. Finally in section 4, we compare our approach with related work.

## 2 Mapping Phrase Structures into Dependency Structures

In this section we describe the scenario of mapping phrase structures onto dependency structures. We introduce first the PS of the NEGRA Corpus and the DS, Surface Syntactic Structures (SSynS) as defined in (Mel'čuk & Pertsov, 1987). After that we describe the necessary tasks to map the structures.

### 2.1 NEGRA Phrase Structures

The phrase structures as used in the NEGRA Corpus are supposed to be theory independent (Wojciech *et al.*, 1997). Therefore they are rather flat. In Figure 1 an example is shown. The syntactic categories (NN (“normal noun”), KON (“coordinated conjunction”), etc.) are encoded as node labels and the grammatical functions (such as HD (“head”), SB (“subject”), etc.) as edge labels. The used part of speech tags and edge labels are summarized in Table 1.





2. **Determining dependency relations.** The edge labels are mapped onto surface syntactic dependency relations. Three main cases have to be distinguished:
  - a) direct mapping of NEGRA edge labels onto surface syntactic relations, e.g. SB is mapped onto 'subjective'.
  - b) mapping of ambiguous edge labels onto SSynt relations using the government pattern of the lexeme in question, e.g. *liegen* 'to lie' contains a PP with *in* and is mapped onto the SSynt relation *pobjective*.
  - c) mapping of edge labels onto SSynt relations using part of speech tags, e.g. words labeled with ADV are mapped onto the SSynt relation *adverbial*
3. **Determining node labels.** The node labels of DS i.e., basic forms of lexemes, are determined by node labeling rules. The node labeling rules are looking up the basic word form in the lexicon and label the target nodes of DS with it. In some cases it is necessary to split the node, e.g. *im* is divided into the node *in* and the node *dem*, *auszureizen* (Engl. to exhaust), which contains an incorporated *zu* 'to' particle, is divided into *zu* 'to' and *auszureizen* 'exhaust' (cf. also Figure 3 and 2). The verb prefix 'aus' is not separated because it cannot occur separate in the sentence. On the other hand but for the same reason we keep the verb prefix 'ein' of the verb 'eingehen' separate because the separation is prerequisite for the linearization task.
4. **Determining elisions.** For phrases that have an elided head a node with the label ELISION is introduced. Further auxiliary rules are applied, if no appropriate head is found.

### 3 Mapping of the NEGRA Corpus

In this section we describe the implementation aspects of the mapping. As mentioned above, the mapping process is realized by a graph grammar. In order to map PS to DS it is first necessary to convert the PS into graphs<sup>2</sup> and second to translate the phrase graphs into DS.

In what follows we introduce first the basic notion of graph mapping. Then we briefly discuss the preprocessing, i.e., the conversion PS into a graph and finally our application.

#### 3.1 Graph Mapping with Rules

In the mapping procedure a compiler applies grammar rules to the input graphs. A graph rule is a tuple  $R\langle G_l, G_r, C_D, C_R \rangle$ . A rule consists of a left-hand side graph  $G_l$ , a right-hand side  $G_r$ , the set of conditions  $C_D$  which must hold in order for the rule to be applicable and the set  $C_R$  which specifies correspondences between parts of  $G_l$  and  $G_r$ .

If a fragment of the source graph matches the left-hand side of a rule, and the condition specified met, the fragment is mapped onto the subgraph specified in the right-hand side. An important point is that the source graph is not altered, but a completely new target graph is build up by the applied rules. Additionally, rules can contain context information both in the left-hand side and in the right-hand side. Left-hand side context means that a rule can share an input fragment

---

<sup>2</sup>PS and DS are of course trees. But, because we use a graph grammar formalism, we use graphs to represent them.

with other rules; right-hand side context contains a target fragment that must have already been created by other rules. This sounds procedural but rules are completely declarative. They only state that a part must be available. This means for the rule interpreter that the mapping of a source graph to a target graph is performed iteratively.<sup>3</sup> It consists of five steps that are applied to all rules in parallel:

1. **Matching.** All occurrences of the left-hand side graph  $G_l$  of all rules are identified in the source structure. This procedure is efficient because the left-hand side graphs are small; the attributes and labels of nodes have a further restrict the matching process.
2. **Evaluation.** The conditions are evaluated and the matches which do not fulfill the condition are filtered out.
3. **Clustering.** During the clustering step, rules which are applicable together to the source structure are grouped into clusters.
4. **Application.** During the application step, the rules of each cluster are applied. For each applicable rule, an isolated elementary graph is created.
5. **Unification.** In the last step, the elementary structures are glued together. This is done by the unification of nodes that correspond to the same source node.

### 3.2 Preprocessing

As mentioned above, to process phrase structures using graph grammars, it is necessary to convert them into graphs. Each constituent is converted into a hyper node<sup>4</sup>. Additionally, the word order is made explicit by introducing edges labeled with b (meaning before) between a word and its direct successor. After the b-relations are established between words, they are recursively defined between constituents, between coordinated phrases, appositions, relative clause and its head, etc. The word order is made so explicit because the rules of the next step need the information about the position to determine the heads of a phrase and the edge labels.

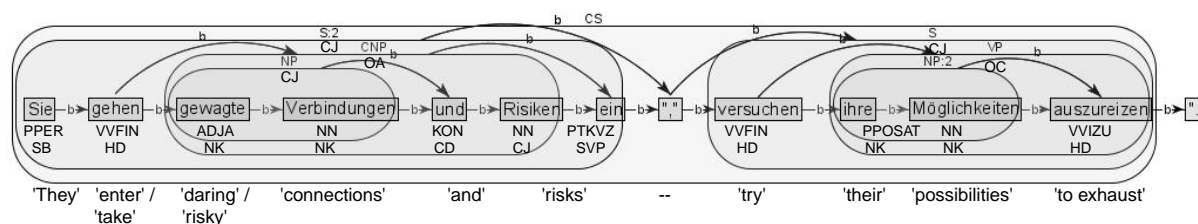


Figure 3: Phrase structure as graph.

The graph representation of the phrase structure in Figure 1 is shown in Figure 3. It shows a chain of nodes labeled with words, part of speech tags, and edge labels of the phrase structure.

### 3.3 Mapping the Phrase Graph

After the preprocessing step, the mapping takes place, i.e. the application of the graph rules to the phrase graph. The rules are executed in two steps. In the first step, those rules that determine heads, those that determine node labels, and those that detect elisions are executed. In the second, step the edge-determining rules are executed.

<sup>3</sup>In the case of the mapping of Phrase Structures to Dependency Structures two iterations are necessary.

<sup>4</sup>A hyper node contains other nodes.

The reason for processing the structure in two steps is that otherwise each edge-determining rule would additionally have to determine the head or the elision of the head. This would lead to an excessive number of rules.

### 3.4 Sample Grammar

In this subsection we describe the application of rules to a part of the structure, which is shown in Figure 3. The rules are executed in two steps as illustrated in Figure 4.

In the first step, the rules are matched against the source graph only. They create fragments of the target graph and correspondences between nodes of the source and target graphs. The created fragments are shown in Figure 4 immediately to the right of the vertical lines. Correspondences are indicated by dashed lines. In the example, the rules `Head_NP_ART`, `Head_VP_vvi zu`, and `standard_lex` are applied to the fragment shown.

In the second step, the edge-determining rules are matched against the source graph, the target graph and the correspondences. The applicable rules are `dobj` and `det`.

In what follows we describe the rule `Head_NP_ART` and `dobj` in general to introduce the rule formalism and then application of all rules to the example.

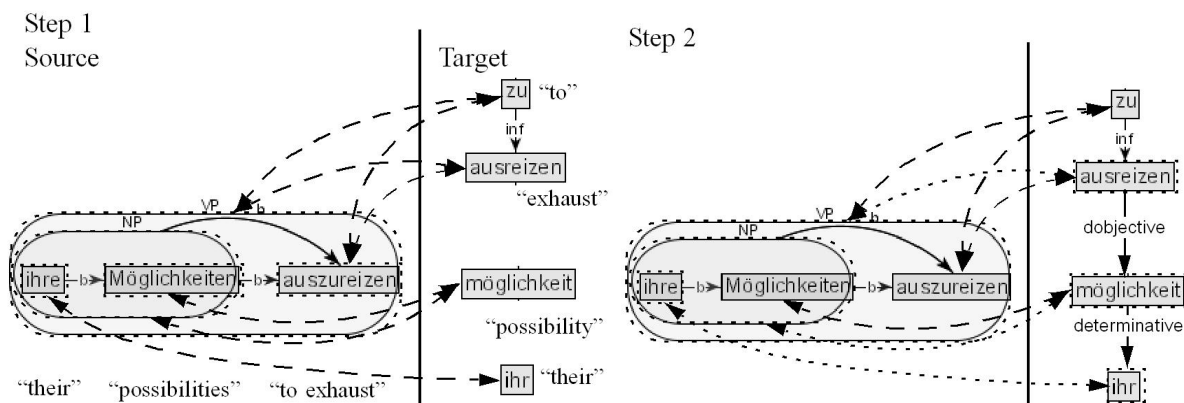


Figure 4: Application of some rules and their result

**Determining heads.** In general, a head-determining rule does the following: it identifies a head of a constituent in the source structure. Then it creates a node in the target structure that corresponds to this head. To mark the head for further processing, a correspondence link between the hyper node in the source structure that represents the constituent and the target head node is established.

```
Rule: Head_NP_ART Type: PS<=>DS
LS: ?XP{ cat=NP|PP ?A {pos=ART|PPOSAT} ?X {} }
RS: ?Xs{<=>?X
    <=>?XP}
CD: ?X first (?X.pos=NN|TRUNC and not (?X.edge=GL));
```

The example rule `Head_NP_ART` calculates the head of a NP in the case that it contains an article and at least one noun. The variable `?XP` of the left-hand side (LS) matches a constituent, which must have an attribute `cat` with the value `NP` or `PP`. The vertical stroke `|` in the expression `cat=NP|PP` stands for the disjunction `OR`. In the case of a `PP`, the head is of course the preposition, but the direct dependent of the preposition is computed in the same way as the head of a `NP` (see above). The phrase which is matched by `?XP` has to contain two other nodes: an

article which is matched by the variable ?A and a noun which is matched by the variable ?X. ?X is further constrained by a condition that is defined in the condition slot (CD). The condition expresses that ?X has to be the first node in the phrase which is a noun (NN) or a truncated noun and it must not have an edge labeled by GL (genitive left). At the right-hand side, the variable ?Xs is defined. It corresponds to the variables ?X and ?XP of the source side. Therefore, the rule creates a node and two correspondences.

In the head determining step, which is illustrated left in Figure 4, the applicable rules are the head determining rules `Head_NP_ART` (described above) and `Head_VP_vvizu` as well as the node labeling rule `standard_lex`. The fragments of the source structure which are matched by the rules are marked with dotted lines. The rule `Head_NP_ART` matches the hyper node labeled with NP and the node `Möglichkeit`. It introduces two correspondences, one starting at the NP and the other starting at the node `Möglichkeiten`. Both go to the node which has been labeled by the rule `standard_lex` as `möglichkeit`.

```
Rule: Head_VP_vvizu Type: PS<=>DS
LS: ?XP{ cat=VP|S ?X {pos=VVIZU} }
RS: ?Xs { lex=zu
         <=>?X
         <=>?XP
         inf-> ?Ys{<=>?X
                  <=>?XP}}
```

The rule `Head_VP_vvizu` matches the node VP and `auszureizen`. It creates two target nodes because at the SSynt level the incorporated zu-particle is separated. Therefore, it is necessary to introduce two correspondences that start at the hyper node VP of the source structure, one ending at the target node labeled by particle `zu` and one ending at the target node labeled by `ausreizen`. The head of the VP is the zu-particle, all children in the VP are attached to `ausreizen`. This means that one of the correspondences is introduced to facilitate the finding of the “parent of the VP” by edge determining rules, and the other one to facilitate the finding of the head for the children.

**Determining edges.** Edge determining rules introduce labeled edges in the target structure. The rules do the following: First, they match, a hyper node in the source structure against the corresponding head in the target graph, and a specific child of the constituent in the hyper node against the corresponding node in the target structure. In the case of a match, a labeled edge is introduced. Let us consider a sample rule.

```
Rule: dobj Type: PS<=>DS
LS: ?XP {tag=VP|S <=> ?Vs
        ?X { edge=OA <=> ?Ns }}
RS: rc:?Vs { rc:cat=verb
             dobjective-> rc:?Ns {} }
```

The rule `dobj` first matches a VP or S which is referenced by ?XP and the correspondence between the head of VP/S and the target node referenced by ?Vs. Then, it matches a child of the phrase VP/S which has the attribute `edge=OA` and the correspondence between an NP and the target node ?Ns. If the rule is applicable, an edge labeled `dobjective` is introduced between the nodes which are referenced by ?Vs and ?Ns.

In the edge determining step of our example, the edge determining rules `dobj` and `det` are applied which is illustrated right part of Figure 4. The rule `dobj` (described above) matches the VP hyper node in the source structure, the node `ausreizen` in the target structure, and the correspondence between these two nodes. Furthermore, it matches the NP hyper node in the source structure, the node `möglichkeit` in the target structure, and the correspondence between the two nodes. If it is applicable, it creates the relation labeled `dobjective`.

## Mapping Phrase Structures to Dependency Structures

```
Rule: det Type: PS<=>DS
LS: ?XP{ cat=NP|PP ?D {pos=ART|PPOSAT} }
RS: rc:?Xs { <=>?XP
      determinative-> ?Ys{ <=> ?D}}
```

The rule `det` matches the NP hyper in the source structure, the node `möglichkeit` in the target structure, and the correspondence between them. It matches also the node `ihre` in the source structure, the node `ihr` in the target structure, and the correspondence between them. In case it is applicable, it creates the edge labeled with `determinative`. The result of the application of all rules to the structure in Figure 3 is shown in Figure 2.

### 3.5 Results of the Mapping of the Negra Corpus

With the approach described in the previous sections, we translated the PS annotations of the NEGRA Corpus into DS annotations. The corpus contains 20602 sentences. The average number of words per sentence in the NEGRA Corpus is about 17.0 and the number of words contained in the dependency tree was 12.8. The reason therefore is that the probability for larger structures is higher than necessary rules are missing. The current grammar contains 143 rules. This grammar achieves an accuracy of 74%, i.e., we receive about 15600 of correct SSynt structures. 3.5 % of the SSyntS are wrong because they are not trees or have superfluous relations. The rest of the incorrect SSyntS is incomplete. This means that additional rules are necessary to achieve a higher accuracy.

## 4 Related work

There are some approaches to the conversion of phrase structure annotations to DS and to LTAG-trees; cf. (Lin, 1995), (Collins, 1997), (Xia & Palmer, 2001), (Xia, 2001), (Frank, 2001). The approach that is most similar to ours are those by Dekang Lin and Fei Xia. Lin converts phrase structures into dependency structures. His algorithm is similar to that one used by (Collins, 1997) which both draw upon (Magerman, 1994)'s procedure to determine heads. Xia extends the approach of Lin using argument and tagset tables to distinguish arguments and adjunct, and to label dependency links, but so far as we know only for english.

Magerman's algorithm determines the head of a phrase using a table (see below). Each entry is a triple (`parent direction head-list`), where `parent` is a grammatical category, `direction` is either `right-to-left` or `left-to-right` and `head-list` is a list of grammatical categories.

```
(S right-to-left (Aux VP NP AP PP))
(VP right-to-left (V VP))
(NP right-to-left (Pron N NP))
```

The first entry means: the head of S-Phrase is the first child from left to right with the category `Aux`; if it isn't the case then the first VP-child from left to right is the head. Based on this assumption, the main idea of Lin's algorithm is to create a dependency tree introducing edges from the head of a phrase to all of its children heads. The algorithm is as follows (taken from (Lin, 1995)):

```
Tree makeDeps(Tree root, DepTree deps) {
  if (root is a leaf node) return root;
  Tree headChild = findHeadOfChild(root);
  Tree lexHead = makeDeps(headChild, deps);
  for each non-head child of root {
    lexHeadOfChild = makeDeps(child, deps);
    addDepRel(lexHead, lexHeadOfChild)}}}
```

The function `findHeadChild` returns the head of a child, the function `addDepRel` adds a relation to a dependency tree. Because of the fairly free word order in German, Lin's and Xia's approaches are only partly applicable to German.

## 5 Conclusion and Future Work

In this paper, we described a rule-based approach for mapping phrase structures of (partially) free word order languages to surface syntactic structures. This approach has been applied to the PS annotations of the German NEGRA corpus. The advantage is obvious; the approach allows us to reuse phrase structure resources. This approach can also be used to map PS annotations to DS annotations of corpora of fixed word order languages, e.g. English. In this case, the head determining rules would use Magerman's tree head table.

## References

- BOHNET B., LANGJAHR A. & WANNER L. (2000). A Development Environment for an MTT-Based Sentence Generator. In *Proceedings of the First International Natural Language Generation Conference*.
- BOHNET B. & WANNER L. (2001). On Using a Parallel Graph Rewriting Formalism in Generation. In *Eight European Workshop on Natural Language Generation*, Toulouse: ACL.
- COLLINS M. (1997). Three Generative, Lexicalized Models for Statistical Parsing. In P. R. COHEN & W. WAHLSTER, Eds., *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics*, p. 16–23, Somerset, New Jersey: ACL.
- FRANK A. (2001). Treebank Conversion – Converting the NEGRA treebank to an LTAG grammar. In *Proceedings of the Workshop on Multi-layer Corpus-based Analysis*, Iasi, Romania.
- LIN D. (1995). A Dependency-based Method for Evaluating Broad-Coverage Parsers. In *IJCAI*, p. 1420–1427.
- MAGERMAN D. M. (1994). *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Stanford University.
- MEL'ČUK I. A. & PERTSOV N. (1987). *Surface-syntax of English, a formal model in the Meaning-Text Theory*. Benjamins, Amsterdam/Philadelphia.
- G. ROZENBERG, Ed. (1997). *Handbook of Graph Grammars and Computing by Graph Transformation*. Singapore, New Jersey, London, Hong Kong: World Scientific.
- WOJCIECH S., KRENN B., BRANTS T. & USZKOREIT H. (1997). An Annotation Scheme for Free Word Order Languages. In *Proceedings of the ANLP Conference*.
- XIA F. (2001). *Automatic Grammar Generation From Two Different Perspectives*. PhD thesis, University of Pennsylvania.
- XIA F. & PALMER M. (2001). Converting Dependency Structures to Phrase Structures. In *The Proc. of the Human Language Technology Conference*, San Diego.