Treebank Conversion
– Converting the NEGRA treebank to an LTAG grammar –

Anette Frank
DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany
frank@dfki.de

Abstract

We present a method for rule-based structure conversion of existing treebanks, which aims at the extraction of linguistically sound, corpus-based grammars in a specific grammatical framework. We apply this method to the NEGRA treebank to derive an LTAG grammar of German. We describe the methodology and tools for structure conversion and LTAG extraction. The conversion and grammar extraction process imports linguistic generalisations that are missing the in original treebank. This supports the extraction of a linguistically sound grammar with maximal generalisation, as well as grammar induction techniques to capture unseen data in stochastic parsing. We further illustrate the flexibility of our conversion method by deriving an alternative representation in terms of topological field marking from the NEGRA treebank, which can be used as input for stochastic topological parsing approaches. On a broader perspective our approach contributes to a better understanding on where corpus-linguistics and theoretical linguistics can meet and enrich each other.[1]

1 Introduction

Parsed corpora are widely used as training material for stochastic parsing and other learning approaches. Annotation schemata of existing treebanks vary, often motivated by language-specific characteristics. Dependency-based annotations are generally preferred for languages with (relatively) free word order, and are considered particularly well suited as "neutral" encoding schemes for parser evaluation (cf. Carroll et al. (2000), Skut et al. (1998)). By contrast, phrase structure (PS) oriented annotations allow for a range of variation, which can affect the evaluation of (stochastic) parsers (Johnson, 1999). In general, treebank annotations should be designed to be rather "theory neutral", that is, not tailored to the assumptions of a particular grammatical framework, to allow these highly expensive resources to be widely applicable and reusable.

The availability of annotated corpora allowed corpus-linguistic methods to rapidly extend to many areas studied in theoretical and computational linguistics. On the other hand, the use of "theory neutral" corpora can lead towards a gap between corpus-based research and well-studied linguistic theories if corpus annotations can not be mapped, for example, to the basic assumptions of a particular syntactic framework. While dependency-oriented stochastic parsers have been trained on dependency treebanks (Collins et al., 1999), it is more difficult to make direct use of dependency or PS annotated corpora for syntactic frameworks with special phrase structure assumptions. Not surprisingly then, where stochastic or learning methods are applied to frameworks like LFG or HPSG (e.g. Bod and Kaplan (1998), Riezler et al (2000), Cancedda and Samuelsson (2000), Neumann and Flickinger (1999)), no use is made of available large-scale corpora. A notable exception is recent work on LTAG grammar extraction from the Penn Treebank (Xia,

[1]We would like to thank Günter Neumann, Josef van Genabith, Stefanie Dipper and Detlef Prescher for discussions and for comments on earlier versions of this paper. Special thanks go to Hubert Schlarb and Holger Neis, who evaluated the results for coordination restructuring.

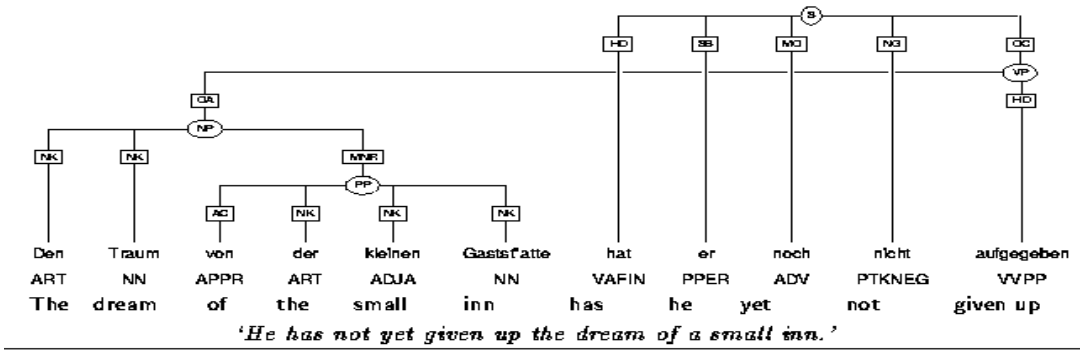


Figure 1: NEGRA annotation example (Brants et al., 1997)

1999), (Xia et al., 2000), which was used for stochastic LTAG parsing (Chen et al., 1999).

The need of large-scale corpora for higher-level syntactic frameworks is addressed in Sadler et al (2000), Frank (2000), Frank et al (2001), who develop methods to enrich treebanks with higher-level syntactic information tailored to specific syntactic frameworks. Partial restructuring of treebank trees was undertaken to bridge crude mismatches between treebank vs. general linguistic structural assumptions. In general, however, the phrase structure of treebank trees was left unaltered. It is therefore difficult to mix such corpus-derived grammars with independently developed grammar resources, e.g. to boost their coverage.

In this paper we present a method for "treebank conversion", which is intended to bridge this gap between corpus-based and theoretical syntax. We apply treebank conversion to the NEGRA treebank (Skut et al., 1998), (Brants et al., 1997),[2] an annotated corpus of German newspaper text, to extract an LTAG grammar of German. Sentences are annotated for POS; syntactic structure encodes both constituency and grammatical function information in a dependency-based annotation format with crossing edges (Fig. 1).[3]

In Neumann (1998, 2001) the NEGRA corpus was used for extraction of a stochastic lexicalised tree grammar (SLTG), following the method of SLTG extraction from an HPSG corpus in (Neumann and Flickinger, 1999). The extraction of tree fragments from corpora is, however, strongly dependent on the underlying annotation scheme and the resulting criteria for tree fragmentation. Compared to the HPSG treebank, the tree structures assigned in the NEGRA corpus are rather flat. As a consequence, extracted tree fragments are strongly contextually restricted. The resulting grammar provides a high degree of ambiguity reduction, yet at the same time does not generalize well enough to unseen data.

The present paper extends the work set out in Neumann (1998, 2001). Our method differs from (Xia, 1999)[4] in that it makes use of a general tree description language (cf. (Frank, 2000)) for structure conversion and fragment extraction. It allows for flexible and fine-grained definition of treebank conversion procedures, which is particularly useful given that LTAG grammar extraction from the NEGRA corpus is challenging both for special aspects of German syntax, and due to the flat NEGRA annotation scheme.

The paper is structured as follows. Section 2 motivates treebank conversion for extraction of linguistically sound grammars. Section 3 describes our method for treebank conversion and grammar extraction. Section 4 provides more in-depth discussion of linguistic aspects in tree transformation, such as rule-based induction of linguistic knowledge, extraction of subcategorisation information, and observations on where to find the border-line between theoretical and corpus-based syntax. Section 5 illustrates the flexibility of our conversion method, by deriving topological field structures from the NEGRA treebank. We finish with final discussion and conclusions in Section 6.

[2]The corpus consists of 20000 sentences, but is being extended to 50000 in the TIGER project (Dipper et al., 2001).

[3]Circled labels encode constituency; boxed labels encode grammatical functions like SB, H(EA)D. Crossing edges are alternatively recompiled to a classical constituent structure, the so-called Penn-format, by insertion of traces.

[4]Or related work on CCG in Hockenmaier et al (2000).

2 From Treebanks to Corpus-Based LTAG Grammars

For extraction of "treebank" grammars it is essential to factor optional constituents and constructional variants from occurrences found in the corpus, to obtain maximal generalisation, and thus maximal coverage on unseen data.

2.1 (P)CFG extraction from corpora

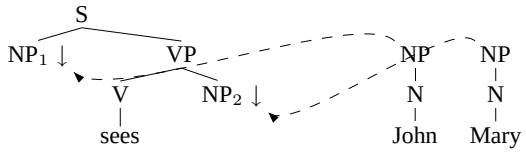
Simple methods in grammar extraction from corpora (Charniak, 1996) have been shown to yield large, yet incomplete grammars with nondecreasing rule novelty rates ("accession rates") due to flat rule encodings. Krotov et al. (1998, 2000) propose rule compaction techniques which filter infrequent rules, or rules that can be pruned without loss in coverage or relative parse probability. Yet, while coverage is maintained, the latter compaction techniques are non-structure preserving. The compacted grammars are not guaranteed to preserve linguistically correct structure assignments. Hepple and van Genabith (2000) propose a structure-preserving compaction method which generalises fine-grained category labels into "supercategories". This method is structure-preserving, but can lead to overgeneration by collapsing discriminating categories. It does also not address the problem of optional constituents.

In recent approaches to statistical parsing[5] the problem of optional constituents in flat treebank grammars is addressed by use of a generative model based on Markov Models ("Markov Grammars"). The symbolic grammars extracted from the corpus, however, do not reflect this statistical knowledge about optionality, nor do we see how this statistical approach can be easily extended to capture regular syntactic variation.

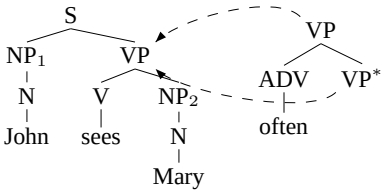
2.2 LTAG extraction from corpora

LTAG grammar extraction differs from extraction of (P)CFG grammars in that the extracted grammar components are strictly lexicalised elementary *trees*, which locally encode all arguments of the lexical head as substitution nodes, defining an "extended domain of locality" (Joshi and Schabes, 1997). LTAG syntax models modification and recursively embedding structures in terms of tree *adjunction*, as opposed to *substitution* (see (1) and (2)). Optional modifiers and other recursively embedding structures must therefore be factored from (usually flat) treebank trees, and rearranged as tree-adjunction structures. The fragmentation rules for LTAG grammar extraction are therefore considerably more complex than simple CFG extraction, and the corpus trees must meet strong structural criteria in terms of construction-specific tree adjunction configurations.

(1) Substitution



(2) Adjunction



Neumann (1998) proposed extraction of stochastic lexicalised tree grammars (SLTG) from the NEGRA corpus. SLTGs are close to LTAGs, but do not necessarily factor modifiers in terms of tree adjunction. For SLTG extraction, corpus trees are recursively decomposed by identifying heads (labelled `HD`, `NK`), and cutting off all non-lexical non-head constituents, marking them as substitution nodes. Since NEGRA structures are rather flat, modification is not represented in terms of tree adjunction. Instead, optional modifiers (labelled `MO`) are "unattached" in a copy of the extracted fragments. Recursion is therefore limited by the maximal number of modifiers in the observed tree, and extracted modifiers cannot be freely inserted in trees which did *not* occur with modifiers in the corpus. LTAG tree adjunction structures, by contrast, guarantee full generalisation to unseen structures.

Examples of the NEGRA annotation scheme are given in Figs. 2, 3. NP structures are flat, unless a daughter has its own dependents; PPs do not embed NP constituents. Restructuring is needed to obtain appropriate LTAG adjunction structures for modifying adverbs, adjectives or other noun modifiers. To obtain linguistically sound repre-

[5] See e.g. Collins (1997), Charniak (2000).

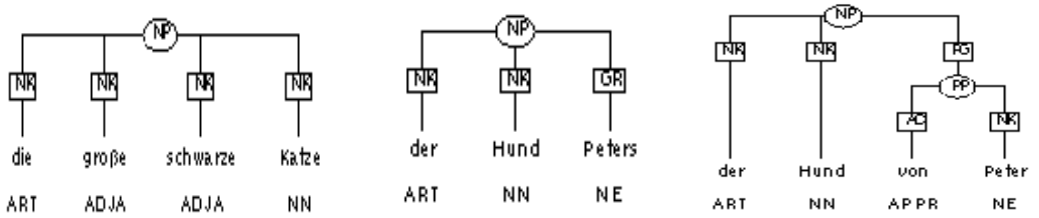


Figure 2: NEGRA NP annotation scheme

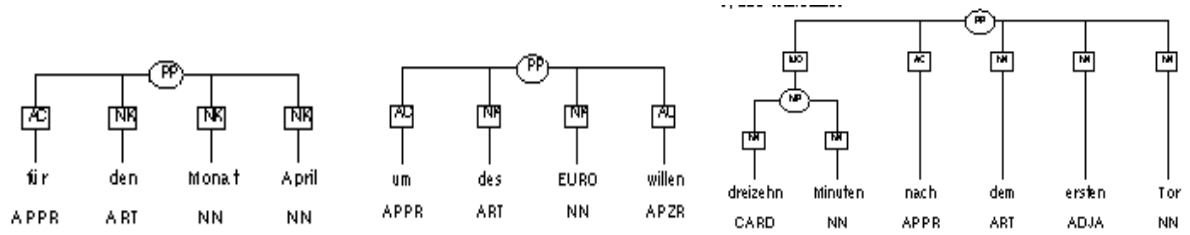


Figure 3: NEGRA PP annotation scheme

sentations for PPs, these require additional restructuring, so as to take NP complements. Irrespective of linguistic motivations, a flat PP encoding as in Fig. 3 results in a large number of corpus-derived PP rules (or SLTG trees), which are nevertheless likely to contain gaps. Grammars extracted from this flat format will miss linguistic generalisations, which is detrimental for coverage on unseen sentences. Restructuring towards linguistic generalisations, by contast, will contribute to a higher degree of modularity, and therefore, coverage on unseen data.

Similar observations hold for clausal constructions: Some verb might have been observed in verb-first position only, or in verb-second, verb-last, or infinitival construction, respectively. If grammar extraction does not abstract over such constructional variants, the resulting grammar will not generalise well enough to unseen data. The same holds for syntactic variations like passivization or reflexivization.

In LTAG, such constructional variants are encoded in the grammar's elementary trees. LTAG grammar extraction therefore requires conversion *and* abstraction over the extracted corpus-trees, to identify construction occurrences (elementary trees), along with the induction of (possibly unseen) constructional variants predicted by linguistic knowledge. These are pre-defined in "tree families". By importing this kind of abstraction and generalisation, linguistic knowledge that is left implicit in corpus annotation, is made explicit in the resulting extracted grammar.

Xia (1999), extending the work of (Neumann, 1998), describes an approach for LTAG grammar extraction from the Penn Treebank. In this paper we describe a method for fine-grained treebank conversion and tree fragmentation on the basis of a general tree description language, which we apply both for extraction of a German LTAG grammar from the NEGRA corpus, and for treebank conversion towards topological field structures for German. Sec. 3 describes treebank conversion and fragment extraction, Sec. 4 provides more in-depth discussion of linguistic issues.

3 Treebank Conversion

We developed software components that compile bracketed tree structures (so-called Penn-format) into a term representation language. These tree representations are input to a cascade of conversion rules which continuously rewrite the structure of the trees. The converted corpus is input to fragment extraction. A set of rules defines fragmentation criteria to extract LTAG elementary trees from the restructured corpus. The transformed corpus and extracted elementary trees are finally reconverted to standard bracketing format.

3.1 A constraint language for trees

We compile canonical bracketed tree structures into a constraint language for trees which allows

us to specifiy nodes, mother-daughter and precedence relations in a modular way, and which can be extended to encode higher-order syntactic structures (grammatical relations, or feature structures in LFG, HPSG, etc. (Frank, 2000)).

The basic tree description predicates are stated in (3.a). Further predicates, like first/last daughter, transitive closure of dominance and precedence, and other shorthand predicates in (3.b) are derived using the definitions in (4).[6]

(3a) basic tree predicates (`A,B` node identifiers)

`arc(A,LA,B,LB)`	B daughter of A, w/category labels LA,LB
`arc(A,LA,FA,B,LB,FB)`	w/functional labels FA,FB
`prec(A,B)`	immediate precedence between siblings A,B
`lex(A,LA,Lex)`	lex. node A w/cat label LA
`lex(A,LA,FA,Lex)`	id. w/functional label FA

(3b) derived tree predicates (`A,B` node identifiers)

`dom(A,B)`	immediate dominance
`dom_x(A,C)`	dominance
`prec_x(A,C)`	precedence (of siblings)
`first_d(A,F)`	F first daughter of A
`last_d(A,L)`	L last daughter of A
`c_label(A,CA)`	CA functional label of A
`f_label(A,FA)`	FA functional label of A ...

(4)

```
dom(A,B) :- arc(A,_,_,B,_,_).
dom_x(A,C) :- dom(A,C) ∨
              (dom(A,B), dom_x(B,C)).
prec_x(A,C) :- prec(A,C) ∨
               (prec(A,B), prec_x(B,C)).
first_d(A,F):- arc(A,_,_,F,_,_), ~prec(_,F).
last_d(A,L):- arc(A,_,_,L,_,_), ~prec(L,_).
c_label(A,CA) :- arc(_,_,_,A,CA,_).
f_label(A,FA) :- arc(_,_,_,A,_,FA).
```

(5) displays (some predicates of) the term representation derived for the rightmost tree in Fig. 2.

(5)

```
first_d(0,1), last_d(0,3),
arc(0,NP,_,1,ART,NK), lex(1,ART,NK,der),
prec(1,2),
arc(0,NP,_,2,NN,NK), lex(2,NN,NK,Hund),
prec(2,3),
arc(0,NP,_,3,PP,GR),
 first_d(3,4), last_d(3,5),
 arc(3,PP,GR,4,APPR,AC),lex(4,APPR,AC,von),
 prec(4,5),
 arc(3,PP,GR,5,NE,NK), lex(5,NE,NK,Peter)
```

3.2 Conversion rules

After conversion to this tree representation language, the corpus trees are restructured by means of a set of declarative conversion rules, which are applied to each corpus tree, in a cascade. *Conversion Rules* consist of a *Rule Identifier*, a set of *Constraints*, and a set of *Actions*.

(6) `RuleId :: Constraints ≫ Actions.`

Constraints specify partial configurations by means of tree description predicates (3), or more complex constraints using predefined templates. *Actions* specify tree modifications by removing (`−p`), changing, or adding (`+p`) tree description predicates `p`. A small set of general and recurrent transformation *Actions* are pre-defined in generic, parameterised templates (see (7) below).

A *Rule* is recursively applied to *each* partial tree configuration that satisfies all *Constraints*. This test is carried out by simple term unification. If the test succeeds, the *Actions* are applied to the term representation. The output is a modified tree representation. If the test doesn't succeed, the rule does not apply, the tree representation remains unaltered. Conversion rules are stated in a sequence, and apply in a cascade: the output resulting from application of some rule r_i provides the input to the following rule r_{i+1}.

For illustration, the conversion rule `Compl` (7) identifies a complementizer node `B` (labelled 'CP') in a flat sentence structure. It identifies the span to the right of B up to the last daughter Y in A (using the predefined template `span_next_to_last/4`), and triggers lowering of the span from X to Y to a new subtree with root node N labelled 'VP-OF',[7] (`lower_subtree_to/6`), which replaces the lowered span. (8) displays an example, with input (a) and output (b) of rule `Compl`, as well as the fully converted structure (c).

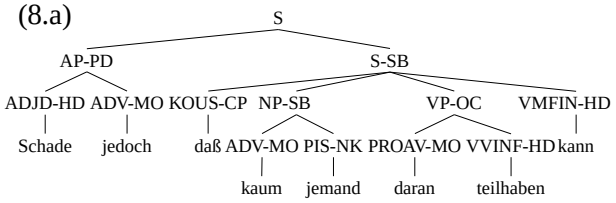
(7)

```
Compl ::
     dom(A,B), f_label(B,'CP'),
     span_next_to_last(A,B,X,Y)
  ≫ lower_subtree_to(A,X,Y,N,'VP','OF').
```

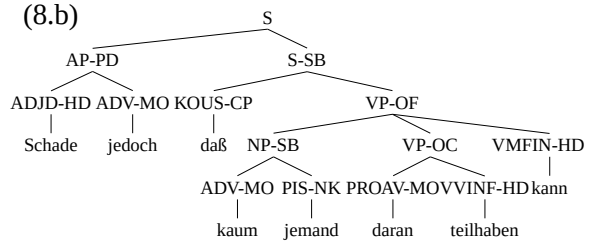
[6] The division into basic vs. derived predicates is useful for concise definition of conversion rules. While all structural changes are internally compiled in terms of basic predicates, conversion rules can refer to derived properties (3.b) as well as predefined templates (s. below), which are computed from the current set of basic predicates "on the fly".

[7] 'OF' stands for 'functional object'.

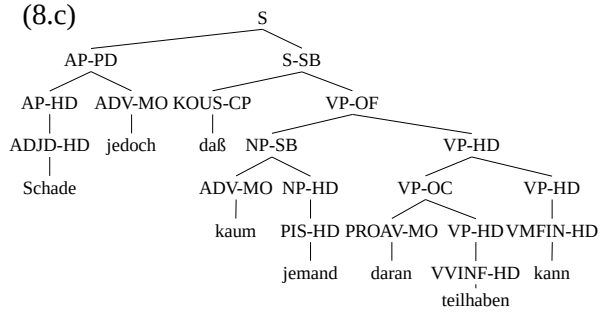
(8.a)



(8.b)



(8.c)



"Schade jedoch, daß kaum jemand daran teilhaben kann."

A pity, though, that almost no one can take part in it.

3.3 Cascaded processing

Cascaded processing can make the definition of conversion rules order dependent in that some rule may depend, in its application constraints, on a tree configuration which was brought about by some preceding rule application. In principle, order dependence in cascaded rewriting systems can be avoided if *all* rules apply to the *same* input structure, which they *non-destructively* enrich.[8] In our *tree conversion* scenario, where the objective is to modify the input structure of treebank trees, this cannot hold by necessity.

3.4 Fragment extraction

After conversion, the restructured corpus trees are input to fragment extraction.

Fragmentation points are identified by rules which again specify *Conditions* and *Actions*. *Conditions* state constraints to identify complementation or adjunction configurations in the (restructured) corpus trees. Fragmentation criteria are treebank-specific categorial and/or functional annotations, as well as specially induced properties based on linguistic knowledge, which are not explicitly encoded in the corpus (see Section 4.1).

In (9.a), the first rule identifies modifier constituents (`MO, JU`) that are in left-adjunction position to a node `C` marked as head (`HD`), yet excluding "VP[V2]" nodes.[9] The second rule identifies extraposed VP modifiers in VP structures.

(9.b) illustrates complementation rules. Direct or indirect objects (functional labels `OA`, `DA`) are identified as substitution nodes - to the exclusion of reflexive and relative pronouns, which are preserved as lexical anchors. More sophisticated rules deal with VP complements vs. VPs in complex tenses, or modal verbs in transitive use.

The *Actions* are predefined to (a) cut out auxiliary trees at specified root (`A`) and foot (`C`) nodes (`cut_mods/6`), or (b) to cut off subtrees at identified complement nodes `B` (`cut_comps/2`), which are marked as substitution nodes: `B`↓. Their effects are illustrated in (10.a) and (10.b).

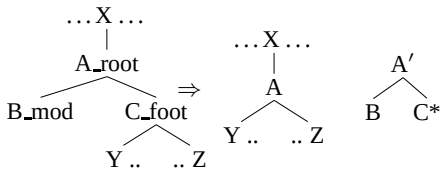
(9.a) Identifying Auxiliary Trees

```
arc(A,_,_,B,_,FB), (FB="MO" ; FB="JU"),
prec(B,C),
arc(A,_,_,C,CC,"HD"), CC ≠ "VP[V2]"
≫ cut_mods(A,C,CC,B,CB,FB).

arc(A,_,_,C,"VP","HD"), prec(C,B),
arc(A,"VP",_,B,CB,"MO")
≫ cut_mods(A,C,"VP",B,CB,"MO").
```

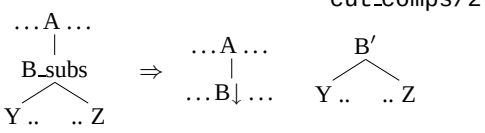
(9.b) Identifying Substitution Nodes

```
comp_node(B,CB,FB), dom(A,B)
≫ cut_comps(A,B).

comp_node(_,CB,"OA"):- CB≠"PRF",CB≠"PRELS".
comp_node(_,CB,"DA"):- CB≠"PRF",CB≠"PRELS".
```

(10.a) Cutting off adjunction trees: `cut_mods/6`



(10.b) Cutting trees at substitution nodes: `cut_comps/2`



[8] This property is e.g. guaranteed in the translation architecture of Dorna and Emele (1996), and can be observed in tree annotation with feature structures (Frank, 2000).

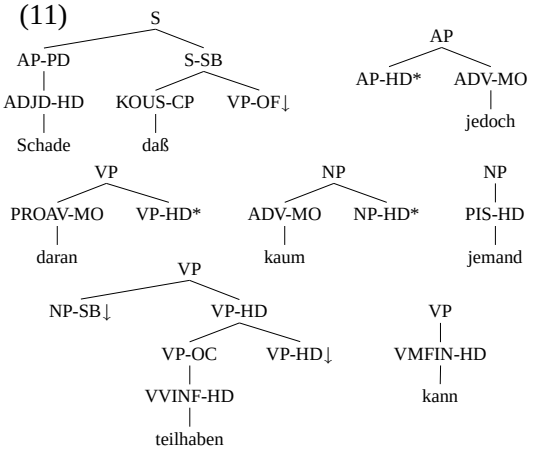
[9] I.e. VP nodes that follow the Vorfeld position in verb second (V2) clauses (cf. Sec. 4.1.1).

# sentences	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
# fragments	12095	23872	35062	46446	57868	69464	81315	91399	102716	113350
# frag/1000 sents	12095	11777	11190	11384	11422	11596	11851	10084	11317	10634
# typed frags	9118	17994	26370	35009	43685	52539	61456	69166	77735	85735
% typed frags	75,38	75,37	75,2	75,37	75,49	75,63	75,57	75,67	75,67	75,65
# types	174	209	225	236	251	262	276	284	294	297
# new types	174	35	16	11	15	11	14	8	10	3
# tokens/type	52,4	86,1	117,2	148,3	174	200,5	222,6	243,4	264,4	288,7
# non-typed frags	2977	5878	8692	11437	14183	16925	19859	22233	24981	27615
# new non-typed	2977	2901	2816	2745	2746	2742	2934	2374	2748	2634

Figure 4: Results for fragment extraction with tree type accession rates

(11) displays tree fragments extracted from (8.c)[10]

(11)



The set of extracted tree fragments is filtered by mapping them against pre-defined tree templates. Fragments that match these templates are typed accordingly, and set apart from the remaining non-typed trees (cf. Sections 4.2.1, 4.2.2).

3.5 Current State and Preliminary Results

Treebank conversion and fragment extraction are not yet completed. We currently concentrated on clause and verb structures, as well as PPs, with only partial treatment of NP-internal structure. The following figures are therefore preliminary. Treebank conversion is defined by 44 conversion rules;[11] fragmentation rules sum up to 13 adjunction and 8 complementation rules. For fragment filtering we are currently using 2155 tree templates, 2075 of these are automatically generated from 65 subcategorisation frames.

Tree conversion on the currently available 10.027 NEGRA sentences triggers 142.649 rule applications (14.2 rule applications/sentence). From the restructured corpus we extract 113.525 tree fragments. Out of these, 85.876 (75.6%) are well-typed according to the tree templates. The remaining 27.649 trees (24.3%) are not (yet) covered by our tree templates.

To measure generalisation of the extracted grammar we determined an LTAG equivalent of rule accession rates, by counting the number of newly encountered (well-formed) tree types in successively extended portions of the corpus. Since in LTAG elementary trees are the only grammar components, abstraction from lexical anchors in tree templates (types) gives an adequate measure of generalisation in the extracted grammar. The results displayed in Fig. 4 are encouraging. With a linearly increasing number of tree fragments, we determine extremely small, and overall decreasing novelty rates at the level of newly encountered fragment types (#new types), with very small global growth rates (#types), and increasing density of tree types (#tokens/type).[12]

4 Linguistic Issues in TB Conversion

We now focus on linguistic issues, in particular how to make use of linguistic knowledge and treebank-specific annotations for fine-grained grammar extraction from corpora.

[10] Note that the NEGRA annotations do not distinguish PP or adverbial (PROAV) arguments from modifiers (see also Section 4.2.1).

[11] These cover V1/V2 vs. Vlast sentences; relative clauses, subordinated complement and adjunct clauses; extraposition; VP and sentential coordination; modal verbs; embedded VP complements/modifiers; complex tenses; adjective modification (partial); PPs (with pre-, post- and circumposition, as well as determiner incorporation); genitive NPs.

[12] These figures are to be considered with care, since $\approx$ 25% of the fragments are still untyped. With completion of the grammar extraction process we expect the number of (new) untyped fragments to show overall decreasing tendency, and the percentage of (new) typed fragments to converge.

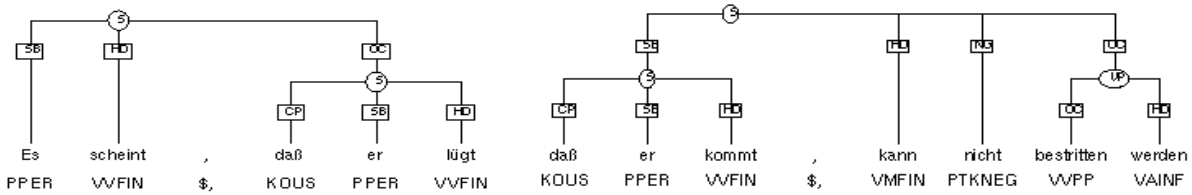


Figure 5:

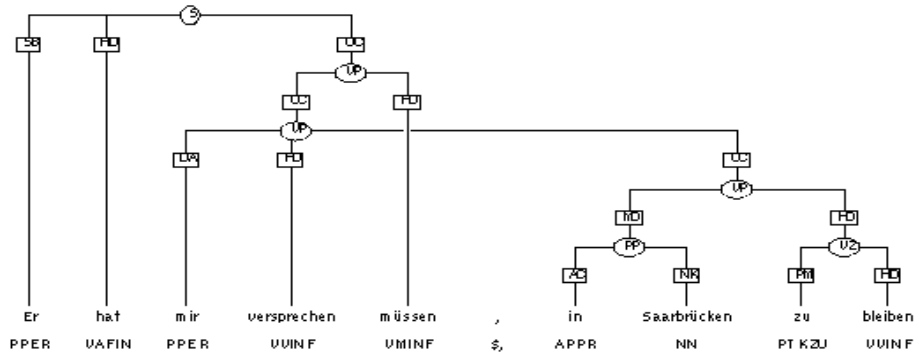


Figure 6:

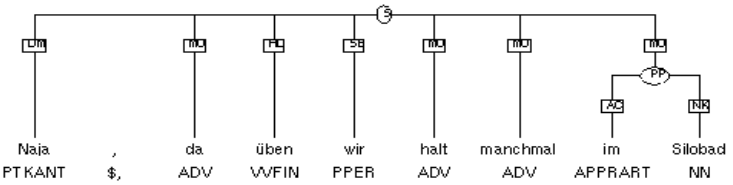


Figure 7:

It should be evident that corpus-based extraction of linguistically sound grammars requires a highly informative annotation scheme. The NEGRA corpus is in this respect well suited, assigning both hierarchical and functional information. While phrase structures are relatively flat, tree conversion rules can induce more hierarchical structure, by reference to phrasal and/or functional labels. Especially functional labels provide very useful general constraints for conversion and fragmentation; refinements are then steered by finer distinctions at the categorial level.

Still, in many cases annotations are not ideal, or miss out crucial information. We show how fine-grained conversion rules, by importing external linguistic knowledge, can induce missing information from secondary properties encoded in the corpus. This is illustrated by looking at German sentence structure and sentence coordination.

4.1 Induction of Linguistic Knowledge

4.1.1 German sentence structure

German sentence structure is traditionally analysed in terms of its "Field Structure", or topological structure, based on the position of the finite verb in left (LB) or right (RB) bracket position (12). In main clauses the finite verb typically occupies the second constituent position, following the so-called "Vorfeld" (VF) (V2 clauses). The Vorfeld can be missing in yes/no questions or embedded conditional clauses (V1 clauses), as well as in subordinate clauses with complementizer. In subordinate clauses the complementizer (or wh-/rel-phrase) marks the LB, the finite verb is in RB position (Vlast clauses). Arguments and modifiers between LB and RB occupy the "middle field" (MF), extraposed material is found to the right of the right bracket, in the "Nachfeld" (NF).

(12)	Vorfeld	Left Bracket	Middle Field	Right Bracket	Nachfeld
V2	topic/ wh-phrase	finite verb	args/ adjs	(verbal complex)	extraposed constituents
V1	- -	finite verb	args/ adjs	(verbal complex)	extraposed constituents
Vlast	- wh-phrase - rel-phrase -	compl	args/ adjs	(verbal complex) +finite verb	extraposed constituents

4.1.2 Identifying clause types

In the NEGRA corpus, topological structure is not encoded, but can - to a certain extent - be derived from the corpus by importing linguistic knowledge. Vlast sentences are clearly identified by an introducing complementizer (CP) or relative clause label S-RC.[13] Here, the position of the finite verb marks the RB (Figs. 5.a,b). Constituents that follow the RB constitute the Nachfeld. The LB of V1 sentences is easily identified as the clause initial finite verb position, but less so the LB of V2 sentences: occasionally more than one constituent precedes the finite verb (Fig. 7). However, with cascaded processing we first mark the LB in (the clearly identified) Vlast sentences. Subsequent rules mark the LB of (the unmarked) V1 and V2 clauses,[13] where all sisters to the left of the finite verb - if any - are marked as VF constituents. This captures the Vorfeld and left brackets in Figs. 5.a,b, 6, 7. More difficult is identification of the Nachfeld in V1/V2 sentences. With compound tenses, the verbal complex (without finite verb) marks the right bracket, and thereby the Nachfeld (Fig. 6). With simple tense clauses (Fig. 5.a) there is no verbal complex to mark the RB, except for verbs with separable prefixes, where the prefix marks the RB. Our conversion rules identify such V1/V2 clauses, and test for S and VP constituents that can occur in the Nachfeld, possibly separated by punctuation, to postulate the "invisible" RB that separates Middlefield and Nachfeld. Vlast vs. V2 wh-clauses are more difficult to distinguish. Both types occur in embedded (13) and matrix clauses (14). Currently we apply criteria corresponding to the distinction (13.a/b) vs. (14.a), i.e. embedded vs. matrix clause.

(13) a. Er fragte, wer kommt. Vlast
b. Er fragte, wen er kennt. Vlast
c. Er fragte mich: "Wer kommt?" V2
d. Er fragte mich: "Wen kennt er?" V2

(14) a. Wer kommt? V2
b. A: "Er fragte, wer kommt."
B: "Wer kommt?" Vlast
c. Wer da wohl kommt? Vlast

Cases like (14.c) are accounted for if constituents occupy the Middlefield. Taking punctuation into account will help us to capture (13.c/d) correctly. With evaluation of our results we hope to sharpen the discriminating criteria.

[13] For wh-sentences see below.

4.1.3 Clause types in LTAG

German sentence structure in LTAG differs considerably from standard assumptions for English. Figs. 8 - 10 illustrate our analysis of the basic clause types.

A binary branching VP provides a uniform Middlefield for V1/V2 and Vlast sentences, with left adjunction of modifiers, or right adjunction for extraposed (NF) constituents (Fig. 10). The alternation between Vlast and V1/V2 sentences is modeled by coindexation of the finite verb and an empty ("-") base position in V1/V2 clauses.[14] In V2 sentences the Vorfeld is either an argument substitution node (8.a) or a verbal projection (8.b) of the lexical tree, or else it is filled by a modifier which takes a VP[V2] *substitution node*, rather than a foot node (Fig. 8.c) to exclude multiple constituents in Vorfeld position. For infinite VP complements and modifiers we induce an empty subject node (not displayed), in accordance with LTAG's concept of extended domains of locality.

In sum, the notion of topological structure which is crucial for a general syntactic treatment of German in LTAG (and other theories) is not encoded in the NEGRA corpus. We show that – by fine-grained definition of conversion rules, making use of the criteria outlined in Sec. 4.1.2 – we can import external linguistic knowledge to automatically enrich the corpus with this additional linguistic information. We obtain maximal generalisation in the extracted grammar, which further supports grammar induction (Sec. 4.2.2).

4.1.4 Coordinated subjects in topic position

For some types of sentence coordination, NEGRA annotations do not follow general linguistic assumptions. The subject in Fig. 11 is not represented as the joint subject of the two conjuncts, but as the subject of the first conjunct, while the second conjunct is missing a subject. The NEGRA representation is not only problematic on theoretical grounds, it leads to wrong results in any corpus-based approach which makes use of the concept of subcategorisation. The information that in Fig. 11 *versuchen* takes a subject, or

[14] The assumption of a trace is not essential.

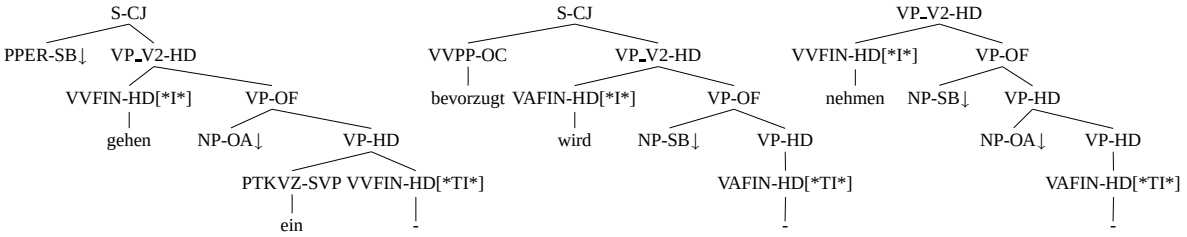


Figure 8: V2 clauses with (a) argument, (b) verb, (c) modifier in Vorfeld

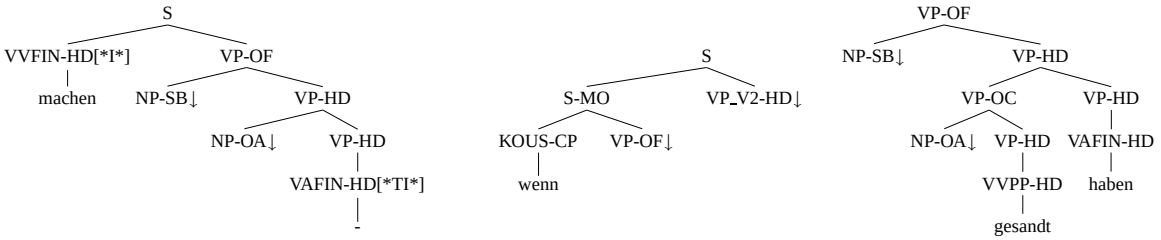


Figure 9: (a) V1 clause; (b) modifying Vlast clause in Vorfeld of V2 clause, (c) vlast clause

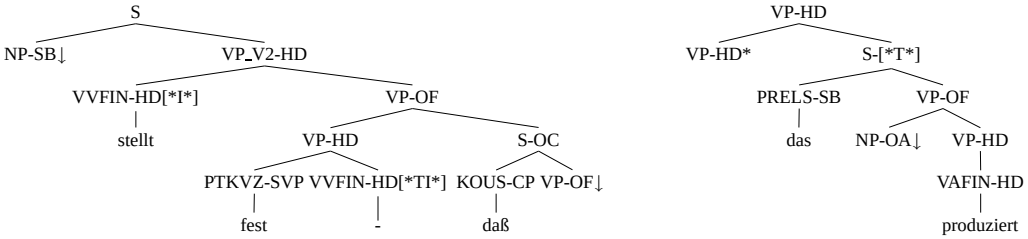


Figure 10: Extraposed (a) vlast S argument; (b) vlast relative clause

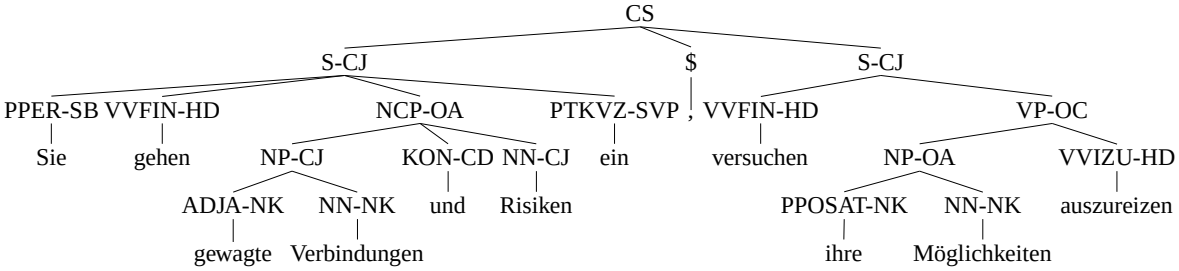


Figure 11: NEGRA coordination structure

Sie gehen gewagte Verbindungen und Risiken ein, versuchen ihre Möglichkeiten auszureizen

They engage in daring relations and risks, try to challenge their possibilities

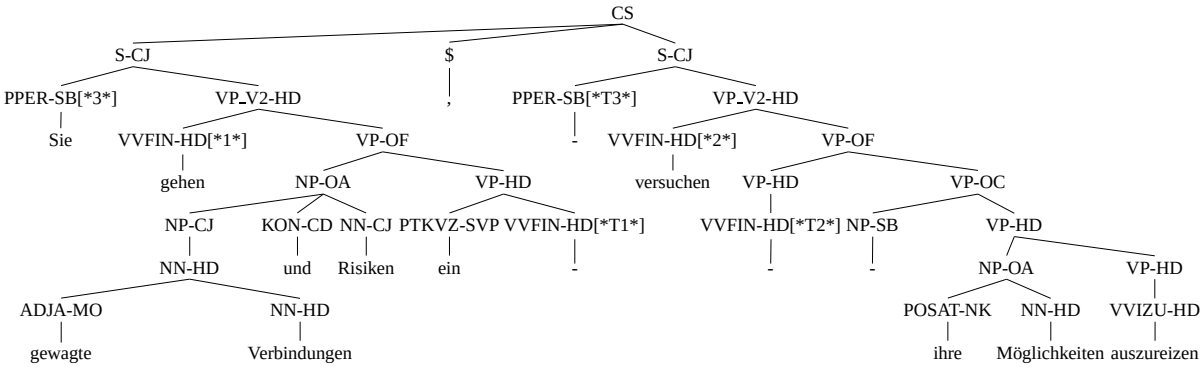


Figure 12: Converted coordination structure

	SB-VF	SGF	Vlast	other	all
# in corpus	84	56	46	289	475
# rule correct	73	48	-	-	121
# rule false	0	2	-	-	2
# not applied	11	6	-	-	17
Precision	100	96	-	-	
Recall	86,9	85,71	-	-	

Figure 13: Automatic coordination conversion[18]

what its syntactic or semantic type would be, is inevitably lost. A learning algorithm trained on this data will falsely deduce that *versuchen* does not take a subject.

Following linguistic insight,[15] we identify V2 + V1 conjuncts where the Vorfeld of the first (V2) conjunct is a subject, while the V1 conjunct doesn't contain a subject.[16] For such V1 conjuncts we introduce an empty Vorfeld constituent which we coindex with the Vorfeld subject of the V2 conjunct. Fig. 12 displays the converted structure resulting from Fig. 11.[17]

4.1.5 SGF or asymmetric coordination

SGF (Subject Gap in Fronted finite verb) coordination differs from the previous example in that in the first (V2) conjunct a modifier - as opposed to a subject - fills the Vorfeld position, with the subject realised in the Middlefield. Again, the second (V1) conjunct does not contain a subject, yet the first conjunct's subject is understood as a joint subject. Fig. 14 illustrates the NEGRA annotation for these examples. Again, we defined a conversion rule – constrained to apply to SGF structures as described above – which inserts a (coindexed) subject gap in the second conjunct (cf. Fig. 15).

4.1.6 Evaluation

On 10027 sentences we counted 267 rule applications for Subject-in-VF coordinations (Sec. 4.1.4), 143 for SGF-type coordinations. We evaluated 3 sections (406 sentences) of a randomly partitioned subcorpus consisting of the 1351 NEGRA sentences which display (any type of) S coordination. Fig. 13 gives results in terms of precision and recall. For V2 coordination cases (SB-VF and SGF) we obtain 100% and 96% precision and 86,9% and 85,71% recall, respectively. In 2 cases automatic annotation picked the wrong antecedent for the subject gap; in 17 cases (3,5% of all coordinations) the rules missed out an occurrence of the targeted phenomena in the corpus.[18] Vlast coordinations have not been treated yet.

These examples illustrate how information that is implicit or missing in treebank annotations can be induced, by rule-based linguistic knowledge, making use of categorial as well as higher-level functional annotations. Our coordination conversion rules could be used to (semi-)automatically port the existing NEGRA annotations to a more refined annotation scheme.[19] However, there are limits to rule-based structure conversion. Missing coindexation in right node raising constructions (see Fig. 16), cannot be inferred on the basis of structural or functional annotations alone. Still, our constraint language on trees could be used to extract candidate trees, based on structural descriptions and (possibly corpus-derived) subcategorisation constraints, and propose conversion towards RNR annotations, to be acknowledged or rejected by human annotators.

4.2 Generalisation and Grammar Induction

Section 2.2 motivated treebank conversion as a means to extract grammars with maximal generalisation and maximal coverage on unseen data, by optimal factorisation of corpus trees. Section 4.1 illustrated how rule-based linguistic knowledge can exploit corpus annotations at distinct - structural and functional - levels to guide conversion towards modular syntactic structures. Likewise, tree fragmentation rules are crucial for extraction of modular tree fragments.

The next level of generalisation is obtained by

[15] Our conversion rule is based on linguistic insight as to the ungrammaticality of coordinating subject-initial V2 and subjectless V1 sentences, as opposed to coordinating V2 sentences with joint Vorfeld subject.

[16] In our tree description language we induce predicates that encode verbal head projection lines. These make it easy to test for the arguments of verbs in binary branching trees.

[17] This structure is still non-standard, but a viable compromise in LTAG, to satisfy its locality assumptions (cf. Sarkar and Joshi (1996), but also Frank and van Genabith (2001)).

[18] Analysis of the missed-out cases yielded 3 cases where corpus annotations were faulty. In 4 cases tree conversion was not yet complete to provide the required input structure. For 6+2+1 occurrences we detected 3 types of structural variation that are not yet captured by the coordination rules. With corpus conversion and rule refinements completed, we can expect recall to improve to 96,43% and 100%.

[19] Such refinements are in fact planned in the TIGER project (see Dipper et al (2001)).

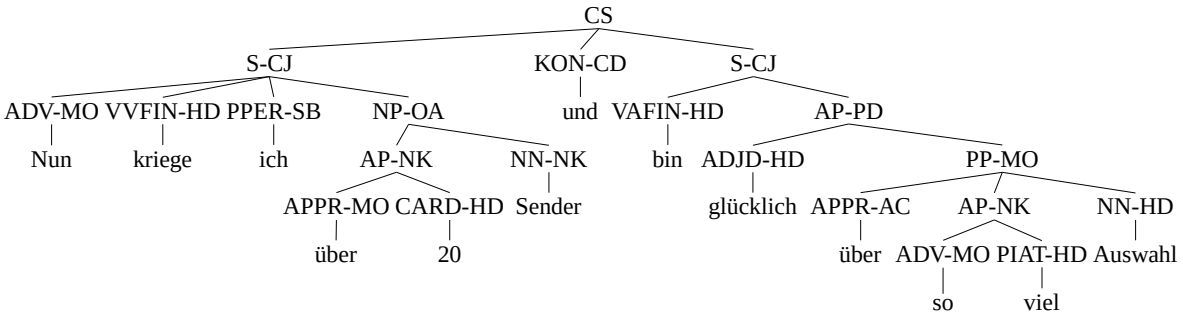


Figure 14: NEGRA coordination structure
(Now I receive more than 20 channels and [I] am happy about so much choice)

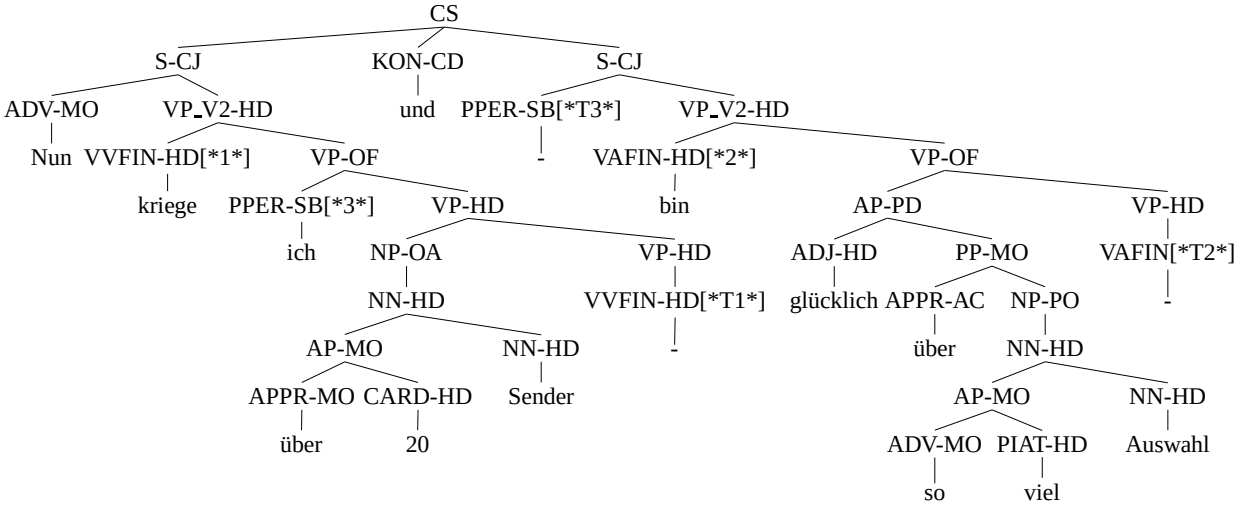


Figure 15: Converted asymmetric coordination structure

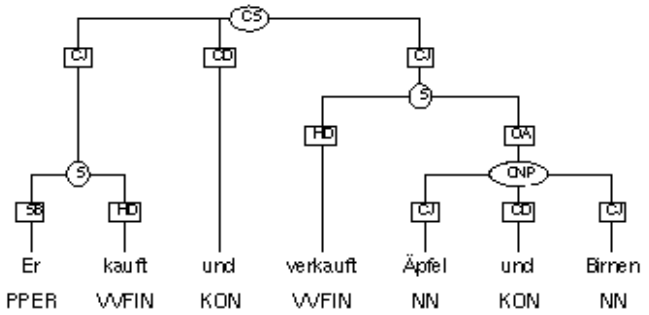


Figure 16: RNR in NEGRA annotation (He buys and sells apples and pears)

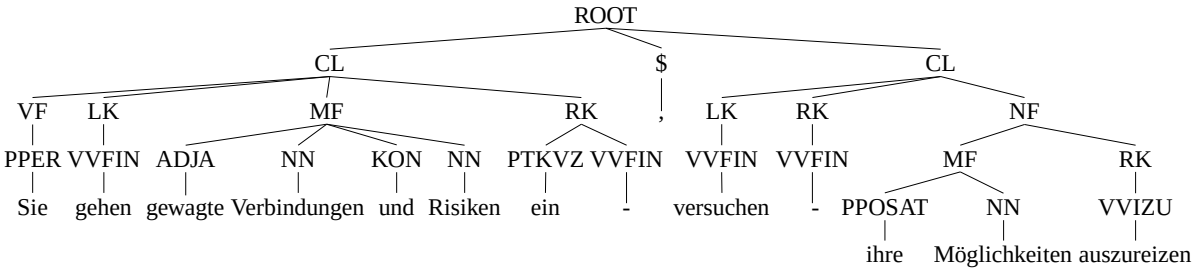


Figure 17: Topological Field Structures

grammar induction. In this step, we generalise over the existing corpus by inducing – from occurrences found in the (restructured) corpus – unseen tree fragments for alternative (unseen) realisations predicted by linguistic knowledge. This is illustrated by focussing on verb alternations.

4.2.1 Extraction of Subcat Frames

We defined templates for lexicalised tree *types*, with variables as placeholders for lexical anchors. Extracted fragments that match some tree type are marked well-formed, and are typed, accordingly.

For verbs, we automatically generate "families" of tree types from a single subcategorisation base frame. Basic subcat frame types are defined in bracketing format, as in (15) for intransitive verbs with indirect object. We compile such entries into our tree description language. 19 cascaded conversion rules rewrite basic frame entries into elementary tree types for various constructional variants, like V1/V2 clauses, compound tenses, scrambling, infinite and extraposed VPs, etc. (16) displays an example of the resulting tree type templates (recompiled to bracketing format), for a V2 auxiliary with infinitive construction (future composed tense).

(15)
```
frame(v_np1_np3,vletzt,
 ["VP",["Arg1","A1"],
       ["VP-HD",["Arg2","A2"],
                ["VP-HD",["VVFIN-HD","Verb"]]]]).
```
(16)
```
tree_type(v_np1_np3_aux_inf_v2,Verb,[Aux,Verb],
 ["S",[Arg1,[A1]],
   ["VP[V2]-HD",["VAFIN-HD[*I*]",[Aux]],
      ["VP-OF[VT]",["VP-OC",[Arg2,[A2]],
                            ["VP-HD",[VINF,[Verb]]]],
                   ["VP-HD",["VAFIN-HD[*TI*]",["-"]]]
]]]):-nominal_sb_da(Arg1,Arg2),
```

We are currently using 65 basic frames. From these we generate 2075 tree type templates (31.9 frames/subcat-type). From the currently extracted verb trees 11508 (66.76%) are successfully mapped to these types. 5730 (33.24%) are still untyped. In further work we plan to induce the distinction between argument and adjunct PPs,[20] using clustering techniques.

4.2.2 Tree Families for Grammar Induction

Once treebank conversion, fragmentation and type definitions are completed, the next step consists in grammar induction, using appropriately defined families of tree types. This, in conjunction with morphological lookup, will enable us to generate complete tree families, i.e. all constructional variants for verbs that have only been observed once, in a particular construction and subcategorisation reading. It should be evident that the resulting gain in coverage is considerable.

4.3 Where Corpus Linguistics meets Theoretical Linguistics

With continuous refinement of fragment extraction and fragment typing, an interesting grey-scale border-line between theoretical and corpus linguistics emerges in this approach to grammar extraction. The complement set of well-typed tree fragments, currently 27.649 non-typed trees, can be classified into (a) larger tree fragments which are not yet restructured to LTAG tree-adjunction structures, which therefore do not satisfy the existing fragmentation rules, and consequently cannot be typed; (b) fragments which are not cut down due to missing fragmentation rules; and (c) extracted fragments for which no tree templates have been defined yet. By inspection and classification of non-typed fragments we guide the development of conversion, fragmentation rules and tree type templates. The continuously reduced set of non-typed trees moves the border-line between non-classified corpus data and well-typed grammar more and more towards a growing, well-defined grammar and a remnant of non-classified corpus trees. Once we reach the limit where the remaining tree fragments cannot, or only with difficulties, be well-typed according to the coverage of theoretical syntax, we will have determined a border-line between corpus-based and theoretical syntax. The grammar's "complement set", the set of non-typed corpus-trees, could then be considered the target for corpus-based theoretical syntactic research. Note, however, that non-typed lexicalised tree fragments can be used as regular LTAG grammar components. Typed and non-typed tree fragments could therefore "meet" as theoretical-syntactic and corpus-derived syntactic components, in corpus-derived grammars.

5 Topological Field Marking

We finally illustrate the flexibility of our tree conversion method, which is easily adapted to derive

[20] These are not distinguished in the NEGRA corpus. Currently we treat PPs as modifiers.

topological structures from the NEGRA corpus.

We selected the 13 conversion rules which deal with sentential structure in the NEGRA corpus. We added 8 conversion rules which make use of structural clues in the restructured trees to introduce additional "topological category nodes" VF, MF, NF and LK, RK that dominate the respective "fields" in the restructured trees. In (17), finite verbs with an index (to a trace) are marked LK, finite verbs without trace index, or finite verb traces mark the right bracket RK. All constituents to the left of LK are subsumed under the VF node, those to the right of RK are collected under the NF node. The trees are then flattened by 2 conversion rules. Fig. 17 displays an example of a topological tree derived by this extension.

Topological structures are flat and easy to check. Since they are derived from the underlying LTAG conversion rules, they can be used for additional or partial evaluation of the first conversion approach. With confirmed results, the topological structures can be used for evaluation of existing topological parsers, and as a training corpus for stochastic topological parsing.

(17)

```
lk_v2::  arc(A,CA,_,B,V2,"HD"),
         CA ≠ "LK", v2_label(V2),
         lex(B,V2,"HD",_), index(B,I)
≫        lower_subtree(A,B,B,N,"LK","").

rk_vend::arc(A,CA,_,B,V2,"HD"), CA ≠ "RK",
         vletzt_label(V2), ~index(B,_)
≫        lower_subtree(A,B,B,N,"RK","").

rk_v2::  arc(A,CA,_,B,V2,"HD"), CA ≠ "RK",
         CA ≠ "LK", v2_label(V2),
         lex(B,V2,"HD","-"), index(B,T)
≫        lower_subtree(A,B,B,N,"RK","").

vf::     ( arc(A,_,_,B,"VP[V2]","HD")
         ∨ dom(A,B), compl_node(B,CP,FB)),
         ~arc(A,_,_,_,"VF",_),
         prec(Y,B), first_d(A,F)
≫        lower_subtree(A,F,Y,N,"VF","").

nf::     arc(A,"VP",_,B,"RK",""),
         prec(B,C), last_d(A,D)
≫        lower_subtree(A,C,D,_,"NF","").
```

6 Summary and Conclusion

We presented an approach for structure conversion of an existing, dependency- and phrase-structure-based treebank, the NEGRA corpus, towards a particular syntactic theory, LTAG syntax. LTAG is heavily phrase-structure oriented and lexicalised, and provides an inherent factorisation of optional and recursively embedded constituents, which ensures a high degree of generalisation to unseen structural contexts.

Our treebank conversion method is constraint-based, using a general tree description language. It allows for fine-grained, flexible definition of conversion and fragmentation rules. Treebank conversion and LTAG extraction are not yet completed, but >75% of the currently extracted fragments are well-typed. We hope to present more conclusive figures by the time of the workshop.

An important motivation for our work is the gap that is often perceived between theoretical syntax and corpus-linguistics. Our method for treebank conversion helps to bridge this gap, by mapping a "theory neutral" corpus annotation to theory-specific structural assumpions, to support extraction of grammar components for a German LTAG grammar. We argued that rule-based induction of external linguistic knowledge is essential for this structural conversion, and can to some extent be used to reconstruct missing, or implicit information from the original annotations.

Given the flat NEGRA annotations, the heavily structure-oriented LTAG formalism, and the complexities of German syntax, we can argue that our conversion method bridges a large gap. It does so by allowing for fine-grained encoding of rule-based linguistic knowledge, and on the basis of a highly informative annotation scheme, in particular functional annotations on top of a basic constituency encoding.

Future work will investigate how the extracted grammar behaves in stochastic parsing.

References

R. Bod and R. Kaplan. 1998. A Probabilistic Corpus-Driven Model for Lexical-Functional Analysis. In *Proceedings of COLING/ACL'98*, pages 145–151.

T. Brants, W. Skut, and B. Krenn. 1997. Tagging Grammatical Functions. In *Proceedings of EMNLP*, Providence, RI, USA.

N. Cancedda and C. Samuelsson. 2000. Experiments with Corpus-based LFG Specialization. In *Proceedings of ANLP-NAACL2000*, Seattle, Washington.

J. Carroll, G. Minnen, and T. Briscoe. 2000. Parser Evaluation: Using a Grammatical Relation Annotation Scheme. In A. Abeille, editor, *Building and*

using syntactically annotated corpora, Dordrecht. Kluwer.

E. Charniak. 1996. Tree-bank Grammars. In *AAAI-96. Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1031–1036. MIT Press.

E. Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of NAACL-2000*.

J. Chen, S. Bangalore, and K. Vijay-Shanker. 1999. New Models for Improving Supertag Disambiguation. In *Proceedings of EACL-99*.

M. Collins, J. Hajic, L. Ramshaw, and Ch. Tillman. 1999. A Statistical Parser for Czech. In *Proceedings of ACL 99*.

M. Collins. 1997. Three Generative Models for Statistical Parsing. In *Proceedings of the ACL-97*, pages 16–23.

S. Dipper, T. Brants, W. Lezius, O. Plaehn, and G. Smith. 2001. The TIGER Treebank. to be presented at *LINC 2001*.

M. Dorna and M.C. Emele. 1996. Efficient Implementation of a Semantic-based Transfer Approach. In *Proceedings of ECAI'96*, Budapest, Hungary, pages 134–142.

A. Frank and J. van Genabith. 2001. LL-based Semantics Construction for LTAG – and what it teaches us about the relation between LFG and LTAG. In M. Butt and T.H. King, editors, *Proceedings of the LFG01 Conference*, University of Hong Kong, CSLI Online Publications, Stanford, CA.

A. Frank, L. Sadler, J. van Genabith, and A. Way. 2001. From Treebank Resources to LFG F-Structures. Automatic F-Structure Annotation of Treebank Trees and CFGs Extracted from Treebanks. In A. Abeille, editor, *Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers, The Netherlands. to appear.

A. Frank. 2000. Automatic F-structure Annotation of Treebank Trees. In M. Butt and T.H. King, editors, *Proceedings of the LFG00 Conference*, University of California, Berkley, CSLI Online Publications, Stanford, CA.

M. Hepple and J. van Genabith. 2000. Experiments in Structure-Preserving Grammar Compaction. 1st Meeting on Speech Technology Transfer, Universidad de Sevilla and Universidad de Granada, Seville, Spain.

J. Hockenmaier, G. Bierner, and J. Baldridge. 2000. Providing Robustness for a CCG System. In *Proceedings of the Workshop on Linguistic Theory and Grammar Implementation*, Workshop held as part of ESSLLI-2000, Birmingham, Great Britain.

M. Johnson. 1999. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*.

A. Joshi and Y. Schabes. 1997. Tree Adjoining Grammars. In A. Salomma and G. Rosenberg, editors, *Handbook of Formal Languages and Automata*. Springer Verlag, Heidelberg.

A. Krotov, M. Hepple, R. Gaizauskas, and Y. Wilks. 1998. Compacting the Penn Treebank Grammar. In *Proceedings of COLING/ACL'98*, pages 699–703.

A. Krotov, M. Hepple, R. Gaizauskas, and Y. Wilks. 2000. Evaluating two Methods for Treebank Compaction. *Journal of Natural Language Engineering*, 5(4):377–394.

G. Neumann and D. Flickinger. 1999. Learning Stochastic Lexicalized Tree Grammars from HPSG. DFKI Technical Report, Saarbrücken.

G. Neumann. 1998. Automatic Extraction of Stochastic Lexicalized Tree Grammars from Treebanks. In *Proceedings of the 4th workshop on tree-adjoining grammars and related frameworks*, Philadelphia, PA, USA.

S. Riezler, D. Prescher, J. Kuhn, and M. Johnson. 2000. Lexicalized Stochastic Modeling of Constraint-Based Grammars using Log-Linear Measures and EM Training. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*, Hong Kong, China.

L. Sadler, J. van Genabith, and A. Way. 2000. Automatic F-Structure Annotation from the AP Treebank. In M. Butt and T.H. King, editors, *Proceedings of the LFG00 Conference*, University of California, Berkley, CSLI Online Publications, Stanford, CA.

A. Sarkar and A. Joshi. 1996. Coordination in Tree Adjoining Grammars: Formalization and Implementation. In *Proceedings of COLING'96*, Copenhagen, Denmark.

W. Skut, Th. Brants, and H. Uszkoreit. 1998. A Linguistically Interpreted Corpus of German Newspaper Text. In *Proceedings of the ESSLLI Workshop on Recent Advances in Corpus Annotation*, Saarbrücken, Germany.

F. Xia, M. Palmer, and A. Joshi. 2000. A Uniform Method of Grammar Extraction and Its Applications. In *Proceedings of EMNLP 2000*, Hong Kong.

F. Xia. 1999. Extracting Tree Adjoining Grammars from Bracketed Corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, Beijing, China.