TIGER TRANSFER
Utilizing LFG Parses for Treebank Annotation

Heike Zinsmeister

University of Stuttgart
Institute for Natural Language Processing
zinsmeis@ims.uni-stuttgart.de

Jonas Kuhn

Stanford University
Department of Linguistics[1]
jonask@mail.utexas.edu

Stefanie Dipper

University of Stuttgart
Institute for Natural Language Processing
dipper@ims.uni-stuttgart.de

Proceedings of the LFG02 Conference

National Technical University of Athens, Athens

Miriam Butt and Tracy Holloway King (Editors)

2002
CSLI Publications
http://csli-publications.stanford.edu/

[1]Jonas Kuhn was at the University of Stuttgart when the main work reported in this paper was performed.

Abstract

Creation of high-quality treebanks requires expert knowledge and is extremely time consuming. Hence applying an already existing grammar in treebanking is an interesting alternative. This approach has been pursued in the syntactic annotation of German newspaper text in the TIGER project. We utilized the large-scale German LFG grammar of the PARGRAM project for semi-automatic creation of TIGER treebank annotations. The symbolic LFG grammar is used for full parsing, followed by semi-automatic disambiguation and automatic transfer into the treebank format. The treebank annotation format is a 'hybrid' representation structure which combines constituent analysis and functional dependencies. Both types of information are provided by the LFG analyses.

Although the grammar and the treebank representations coincide in core aspects, e.g. the encoding of grammatical functions, there are mismatches in analysis details that are comparable to translation mismatches in natural language translation. This motivates the use of transfer technology from machine translation.

The German LFG grammar analyzes on average 50% of the sentences, roughly 70% thereof are assigned a correct parse; after OT-filtering, a sentence gets 16.5 analyses on average (median: 2). We argue that despite the limits in corpus coverage the applications of the grammar in treebanking is useful especially for reasons of consistency. Finally, we sketch future extensions and applications of this approach, which include partial analyses, coverage extension, annotation of morphology, and consistency checks.

1 Introduction

This paper reports on work done in the context of the TIGER project.[2] The project aims at creating a large German treebank, the TIGER treebank (Brants et al. 2002), and at developing search tools (TIGERSearch, Lezius (2002)) for exploiting the information encoded in the treebank. The annotation is very detailed in that it encodes information about part-of-speech: e.g. NN, VMFIN (common noun, finite modal verb); morphology: e.g. Masc.Nom.Sg; syntactic category: e.g. NP, PP (noun/prepositional phrase); and grammatical function: e.g. SB, OP (subject, prepositional object). In order to represent the functional dependency relations in a compact graph format with the sequential word string at the terminal nodes, a generalized tree format was adopted which includes crossing branches (cf. the NEGRA project, Skut et al. 1997; for an example, see Figure 5 below). In this format, for instance, a nominal phrase may form a discontinuous constituent with an extraposed relative clause that modifies it. (A more familiar phrase structure format involving traces can be obtained with a conversion routine.)

Two different annotation methods are used in the TIGER project: (i) an interactive combination of a cascaded probabilistic parser (Brants 1999) and manual annotation with the ANNOTATE tool (Plaehn and Brants 2000); (ii) parsing by a symbolic LFG grammar, followed by manual disambiguation and automatic transfer into the TIGER format (Dipper 2000). Technique (i) is the main line in the annotation process; (ii) has a more experimental status. A motivation for the use of (ii) was to explore to what extent a preexisting broad-coverage unification grammar of German can be exploited for an annotation project. Since the corpus is supposed to satisfy high standards of quality (in particular consistency), each sentence is generally annotated independently by two annotators. In cases of mismatch the annotators have to go over the sentence again in a discussion session. The use of two entirely independent methods

[2]The TIGER project (URL: `http://www.ims.uni-stuttgart.de/projekte/TIGER/`) is funded by the *Deutsche Forschungsgemeinschaft* (DFG).

We would like to thank Anette Frank (DFKI Saarbrücken, formerly at XEROX Grenoble) for her input and great help with the transfer component. Thanks also to Bettina Schrader who was not only responsible for the disambiguation of the LFG analyses but also implemented parts of TIGER transfer. Credit for the original idea of exploiting an existing transfer module goes to Martin Emele. Besides these three people, we would like to thank Stefan Evert, Jan Anderssen, Hannah Kermes, and the audiences at the Workshop on "Syntactic Annotation of Electronic Corpora" (Tübingen, 2000) and at the "Third Workshop on Linguistically Interpreted Corpora" (LINC – Leuven, 2001), for discussion and comments on previous versions of this paper.

is an additional way of ensuring consistency. With the more mechanical grammar-based approach, low-level mistakes resulting from carelessness are less likely to appear. To a certain degree this outweighs the problem that the grammar of course does not cover all the constructions appearing in a newspaper corpus.

sentence of TIGER corpus

parsed by LFG grammar
↓

several analyses

disambiguation
↓

one analysis

Tiger Transfer
↓

TIGER format

Figure 1: Scenario of annotation by LFG

This report focuses on the grammar-based annotation approach (ii), which is depicted in Figure 1, and in particular on its transfer component: TIGER Transfer. The paper is organized as follows: Section 2 gives a short introduction to the German LFG grammar that is used in the annotation and also addresses the disambiguation task. Section 3 presents the transformations that an LFG analysis of a sentence undergoes on its way to the TIGER representation. Section 4, then, gives some statistics. Finally, Section 5 concludes the paper with an outlook on future work.

2 The LFG Analysis

2.1 German LFG Grammar

The German LFG grammar (Dipper to appear) was developed in the PARGRAM project,[3] using the Xerox Linguistic Environment (XLE). Analyzing a given sentence with the LFG grammar yields two representations, the constituent structure (c-structure) and the functional structure (f-structure). C-structure encodes information about morphology, constituency, and linear ordering. F-structure represents information about predicate argument structure, about modification, and about tense, mood, etc.

Figure 2 shows the LFG c-structure for a simple sentence from the TIGER corpus: *Hier herrscht Demokratie* 'Democracy rules here'. The example demonstrates both, familiar aspects of German syntax and more technically motivated specialities of this particular grammar implementation.[4] The latter in-

[3]URL: `http://www.parc.com/istl/groups/nltt/pargram`

[4]The German LFG grammar encodes a generalized CP-analysis of German: The finite verb *herrscht* thus occupies the C-position, preceded by the adverb phrase *hier* in the specifier position of CP. The NP *Demokratie* is immediately dominated by Cbar. For processing reasons, there is no VP-projection covering the 'Mittelfeld' and hence dominating the NP. Note finally, the root node does not only dominate the clausal projection of the sentence but also its identification and its final punctuation

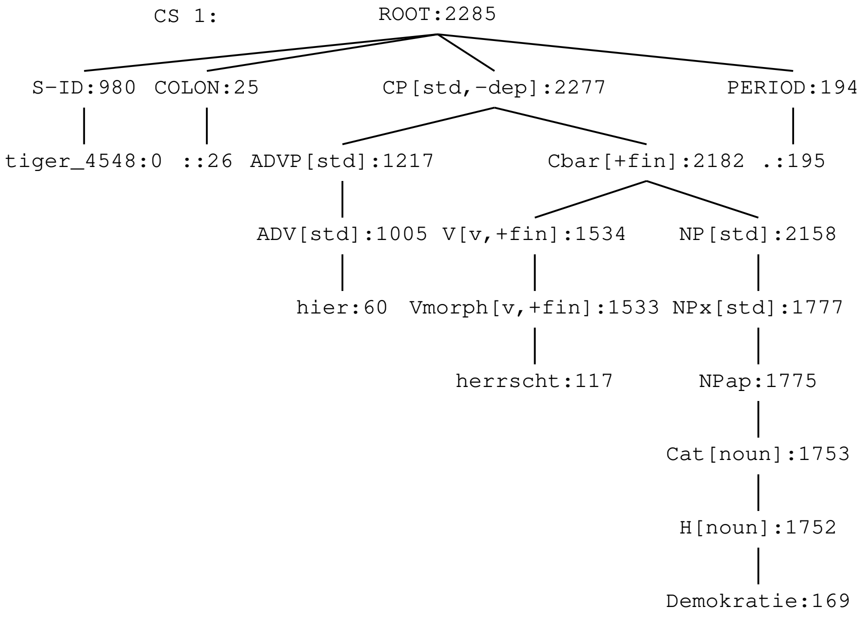


Figure 2: **C-structure** of tiger_4548: *Hier herrscht Demokratie.*

clude a fine-grained differentiation of category symbols, among others 'complex' category symbols with category-level features in squared brackets, like `V[v,+fin]`[5]. Figure 3 shows the f-structure of *Hier herrscht Demokratie* 'Democracy rules here'. The leftmost bracket opens the feature structure of the main predicate, the verbal predicate *herrscht*. This f-structure includes several grammatical functions, which have embedded f-structures as values: ADJUNCT points to the adverbial predicate *hier* (embedded in a set, since there can be several adjuncts), SUBJ(ect) points to the predicate *Demokratie*. In addition the f-structures contain morphosyntactic information like TENSE, MOOD, CASE, and GENDER. In LFG, the level of c-structure is related to the f-structure by the function ϕ which maps each c-structure node to a feature structure (a many-to-one mapping). The mapping relations between c-structure nodes and f-structures are indicated by indices. In Figure 2, for instance, the c-structure nodes representing the projections of the adverb *hier* are indexed with 60, 1005, and 1217, respectively. The function ϕ maps the nodes to the feature structure that is the value of the (set-valued) feature ADJUNCT on f-structure, cf. Figure 3.

2.2 Disambiguation

Almost every sentence of a newspaper corpus is syntactically ambiguous. There are structural ambiguities such as different attachment sites of adjuncts and word-level ambiguities due to ambiguous inflectional marking, homographic word forms or alternatives in subcategorization. Hence the grammar output has to be disambiguated, which normally means that a human annotator has to select the correct analysis.[6]

mark (S-ID and COLON).

[5] `V[v,+fin]` denotes a verbal category of the subtype 'finite full verb'.

[6] Shallow parsing approaches typically employ a more deterministic strategy, i.e., they combine parsing and disambiguation (from corpus-based training), producing just a single analysis. Quite obviously in the scenario of creating a high-quality treebank annotation, manual control is indispensable. In the approach using the ANNOTATE tool, this is ensured through the interactive cascaded procedure involving the human annotator at all levels. In our case, the entire analyses are presented to the

```
"tiger_4548: Hier herrscht Demokratie ."

      PRED      'herrschen<[169:Demokratie]>'
                    60 PRED 'hier'
      ADJUNCT     1005 ADV-TYPE adj-sem, OBL-SEM loc
                  1217
 195         169 PRED   'Demokratie'
 194        1752 NMORPH  CHECK      [SPEC +]
 117   SUBJ 1753         NEED-SPEC -
1533        1775
1534        1777 NTYPE   [GRAIN mass]
2182        2158 CASE nom, GEND fem, NUM sg, PERS 3
2277
  26  TNS-ASP   [MOOD indicative, TENSE pres]
  25
   0  VMORPH    AUX-SELECT [VERB haben]
 980            FIN        +_
2285  SENTENCE_ID tiger_4548, STMT-TYPE decl
```

Figure 3: **F-structure** of tiger_4548: *Hier herrscht Demokratie.*

XLE supports this disambiguation in so far as it packs all different readings into one complex representation that can easily be browsed by the human annotator. On average, however, a sentence of the TIGER corpus receives several thousands of LFG analyses. Obviously it is impossible to disambiguate those analyses manually. For that reason, XLE provides a (non-statistical) mechanism for suppressing certain ambiguities automatically. We illustrate both kinds of disambiguation in the following paragraphs.

Manual Disambiguation The parses of one sentence are represented in a packed feature structure chart, cf. Maxwell and Kaplan (1989): the features common to all readings of the sentence are represented a single time; feature constraints that do not hold in all readings are marked by context variables. The result is an f-structure that is annotated with variables to show where alternatives are possible. After some training, this representation is easily readable for the annotator. Manual selection of the correct analysis is done either by picking the corresponding c-structure tree or by clicking on the respective variables in the f-structure. XLE moreover supports manual disambiguation by various other browsing tools applied to c-structure as well as to f-structure (King et al. to appear describe these tools in detail).

In Example (1), ambiguity in case marking gives rise to two different predicate argument structures (and to two different constituent structures). *Der Stiftung* 'the foundation' can either be dative or genitive, i.e., it can either function as indirect object to the ditransitive verb *verkaufen* 'sell' or as genitive attribute to *Haus* 'house' (with a transitive version of *verkaufen*, which is likewise possible), cf. the readings in (2). In Figure 2.2, the f-structure alternatives that are restricted to the transitive reading are annotated with the variable `a:1`, the ditransitive ones with `a:2`, respectively. The resolution of this ambiguity requires context knowledge and has to be done manually.

(1)	Die	Stadt	verkaufte	das	Haus	der	Stiftung.
	The	town	sold	the	house	the	foundation

human annotator grammar includes many more explicit grammatical constraints than the grammars that are used in a shallow parser) .

(2) a. [Die Stadt]$_{NOM}$ verkaufte [das Haus]$_{ACC}$ [der Stiftung]$_{DAT}$.
‘The town sold the house to the foundation.’

b. [Die Stadt]$_{NOM}$ verkaufte [das Haus [der Stiftung]$_{GEN}$]$_{ACC}$.
‘The town sold the house of the foundation.’

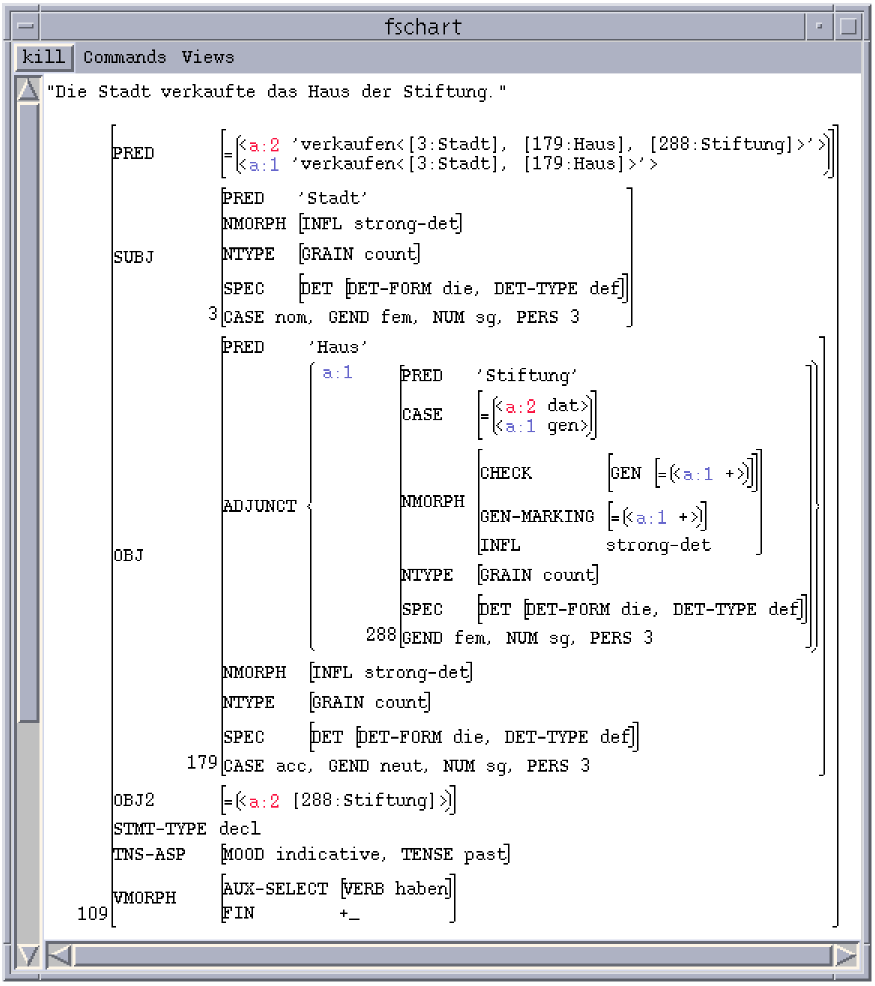


Figure 4: **Packed f-structure-chart** of *Die Stadt verkaufte das Haus der Stiftung.*

Automatic Disambiguation Example 1 is even more ambiguous. *Das Haus* and *die Stadt* can either be nominative or accusative, i.e. subject or direct object of the clause, cf. the readings in given in (3). In this case, one reading is rather improbable, namely that with the object occupying the first position (the ‘Vorfeld’). When like in this example, the case marking is ambiguous (but not when the accusative is overtly marked), then there is a strong dispreference against the reading with the object in the Vorfeld position. This phenomenon is sometimes called the *word order freezing effect* (compare (Kuhn 2001, sec. 4.2), Lee (2001)). This dispreference, as well as other biases, is exploited by a (non-statistical) mechanism that XLE provides for suppressing ambiguities automatically. The mechanism consists of a constraint ranking scheme inspired by Optimality Theory (OT), see Frank et al. (2001). Each rule and each lexicon entry may be marked by ‘OT marks’. When a sentence is parsed, each analysis is annotated by a multi-set of OT marks, thereby keeping a record of all OT-marked rules and lexicon entries that were used for the respective parse. The grammar contains a ranked list of all OT marks. When an ambiguous sentence is parsed, the OT mark multi-sets of all readings compete with each other. A multi-set containing a higher ranked OT mark than another multi-set is filtered out, thus suppressing highly improbable or marked readings, and reducing the number of ambiguities the human disambiguator has to deal with. (Note that no suppressing of parses happens in the absence of ambiguity.)

(3) a. [Die Stadt]$_{ACC}$ verkaufte [das Haus]$_{NOM}$ [der Stiftung]$_{DAT}$. [*improbable*]
'The house sold the town to the foundation.'

b. [Die Stadt]$_{ACC}$ verkaufte [das Haus [der Stiftung]$_{GEN}$]$_{NOM}$. [*improbable*]
'The house of the foundation sold the town.'

A mark 'ObjInVorfeld', for example, forces to disprefer a direct or indirect object in the 'Vorfeld' (rather than subject or adjunct). Thus the improbable readings in (3) are suppressed. For Sentence (1), the German LFG grammar reduces the eight possible readings[7] to two by means of the OT filter mechanism. A further option would be to integrate a probabilistic disambiguation of sentence analyses (cf. Riezler et al. 2000).

3 From LFG to TIGER

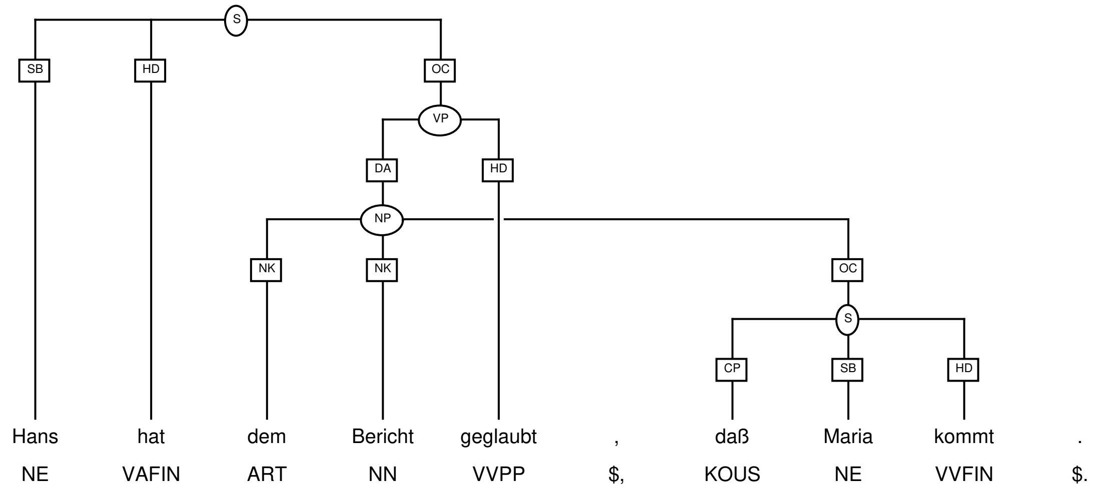
This section deals with the mapping of the LFG grammar output to the TIGER format. Section 3.1 illustrates the close correspondence between LFG f-structure and TIGER graphs. Section 3.2 gives criteria for implementing the conversion procedure and motivates the particular way of splitting up the task in subtasks. Section 3.3 presents preprocessing steps that make the LFG output suited for the actual transfer component. Sections 3.4 and 3.5 deal with the actual transfer. Finally, Section 3.6 concludes the section with the postprocessing steps that complete the mapping.

3.1 LFG F-structure and Tiger Graphs

The initial motivation for adopting the LFG-based annotation method was that despite many differences in details, the syntactic analyses of the LFG grammar and the TIGER graph representation are very similar at the level of functional/dependency structure: Both, LFG's f-structure and the TIGER graph representation, model a dependency structure at a comparable degree of granularity. A further similarity is that in both representations, the dependency structure is explicitly related to the word string. In the case of LFG, this relation is mediated through the level of c-structure; in TIGER it is coded directly into the dependency graph. The TIGER format gives dependency structure priority over phrase structural constituency. It makes use of a generalized tree graph notation which allows for crossing branches. In both schemes, the situation can arise that a single f-structure/dependency-structural constituent corresponds to a set of non-adjacent terminal nodes, i.e., nodes with some intervening word material belonging to a higher-level f-structure/constituent. For example, in *Hans hat dem Bericht geglaubt, daß Maria kommt* 'Hans believed the report that Maria will come', the embedded clause is a complement of the non-adjacent nominal *Bericht*, cf. the representations in Figure 5. The dependency which yields the crossing branch in the TIGER graph is encoded in the LFG f-structure as the complement clause embedded under a COMP feature of the nominal predicate.

Along with these high-level similarities there are a number of differences in the representation conventions of TIGER vs. the German LFG grammar. One main difference is due to the fact that redundant

[7]The discussed four readings are doubled by an additional ambiguity of the verb. It also allows a (dispreferred) subjunctive reading.




```
"Hans hat dem Bericht geglaubt, daß Maria kommt."

   [PRED   'glauben<[1:Hans], [52:Bericht]>'                     ]
   [SUBJ  1[PRED 'Hans']                                         ]
   [        [PRED    'Bericht<[163:kommen]>'                    ]]
   [OBJ2    [             [PRED    'kommen<[186:Maria]>'       ]]]
   [        [COMP         [SUBJ 186[PRED 'Maria']              ]]]
 32[      52[          163[                                    ]]]
```

Figure 5: Long distance dependencies in TIGER graph and LFG f-structure representation

information is not encoded in the TIGER format (which we will deal with in Section 3.6). For example, phrasal categories are always given in LFG c-structure, independent of the complexity of the phrase, whereas in TIGER only complex constituents are dominated by a phrasal node (e.g. VP), compare the representation of the intransitive *kommt* ‘comes’ vs. the transitive (i.e. complex) *geglaubt* ‘believed’ in the TIGER graph in Figure 5. Other discrepancies exist in the representational conventions for particular phenomena (dealt with in Section 3.5). For example, in the TIGER representation, the verbal phrase *dem Bericht geglaubt* ‘believed the report’ (containing the full verb participle *geglaubt*) is embedded as a clausal object (OC) under the auxiliary *hat*. The LFG grammar does not employ a nesting analysis on the functional level. The full verb and the auxiliary occur at the same f-structure level.

In summary, differences between the two representation schemes fall in two distinct categories: formalism-inherent differences and differences in representational convention or linguistic analysis. The latter could in principle be overcome *within* either of the two formal frameworks, e.g., by using the LFG formalism for writing a new grammar that uses category symbols according to the exact specifications of the TIGER annotation scheme (although this is certainly not a practical option, since the grammar has its independent motivation the way it is).

3.2 Criteria for implementing the conversion procedure

The observed systematic relation between the source and the target format of the required conversion makes it realistic to implement a mechanical routine for this conversion. At the same time, subtleties in the differences of the second kind (differences in convention or linguistic analysis) have to be approached with care. In particular, one has to be aware that neither the source nor the target format conventions are specified in all detail; they are not even fixed once and for all, but may undergo occasional changes. (Presumably they cannot be fixed in principle as long as one keeps applying the grammar and the annotation scheme to new corpus material.)

We put great emphasis on the criterion of flexibility in the specification of the conversion procedure in order to be able to react to modifications in the source and target format conventions. This excluded a monolithic implementation of the conversion step. Rather a design with a declarative specification of the convention-related conversion-steps was vital. As mentioned above, it is possible to make all conversions but the ones concerning formalism-inherent differences *within* either one of the formalisms. The decision we made was to exploit this fact and stay within the formal framework of LFG for most of the conversion procedure[8] – until a final low-level conversion into the notational format of the TIGER treebank (this will be called the *postprocessing* below, see Figure 6). The great advantage of this move was that we could exploit existing systems for modifying grammatical analyses within a linguistic formalism: transfer systems as used in machine translation. Although the present context of application is quite different, the task is very similar. In our case the source and target structures do not originate from grammars of different languages, but from different systems of representation for syntactic data of the same language.

In order to ensure a high-quality conversion, our goal has been to rely on *systematic* differences, which can be converted mechanically, as much as possible. Besides the more or less notational rewriting step that we perform towards the end of the conversion there are other highly systematic differences between the formalisms, concerning the relation between functional/dependency structure and the surface string. Basically, LFG's c-structural information has to be folded into the f-structure (and reduced to a subset of relevant categorial information). It turned out convenient to perform this conversion right at the beginning (as what we call *preprocessing* below). Having separated out these two formalism-inherent aspects of the conversion, the remaining step can focus on the more linguistically involved conversion aspects as part of the transfer proper.

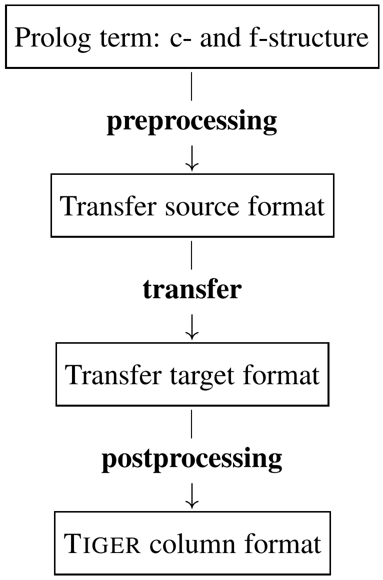


Figure 6: Subprocedures of the conversion routine

The resulting modular design shown in Figure 6 has some obvious advantages: Changes in the grammar or annotation scheme (unless radical) should only affect this 'middle' step. Testing of the conversion

[8]One advantage of this is that the XLE visualization tools for LFG structures can be used even at a stage in which the representation has already been converted quite a long way (cf. Figures 8 and 9 below).

steps can be performed separately, and it becomes an option simply to replace the postprocessing step, for instance, if a different target format (e.g. XML) is desired.

3.3 Preprocessing

Since the transfer step proper proceeds along the f-structures of the LFG representation, it has to be ensured that all the information required to construct the TIGER target structures is accessible from f-structure. This is the task of the preprocessing step. Note that information about linearization of the individual words – although not encoded at the level of f-structure – can be ignored in our context: the word string remains identical, therefore it would be redundant to keep track of word order information during transfer. As long as unique reference to the words is made, the linearization of the string can easily be overlaid over the target TIGER graph – crossing edges result automatically.

Nevertheless, reference to c-structure categories is required in order to construct the correct category labels in the target structure. Thus the preprocessing step implements a general format conversion of the LFG structures, folding the c-structural information into the f-structure. (Not all the c-structural information is required, but the transfer can easily eliminate irrelevant information.)

The LFG grammar development and parsing system XLE provides an export format for the syntactic analyses: a Prolog term, containing flat lists of f-structure and c-structure descriptions, cf. Figure 7. The preprocessing step was implemented as a Prolog program taking this format as input and producing a similar term with an 'enriched' f-structure as output. Following the modular philosophy argued for in the previous section, the preprocessing program performs only highly systematic, canonical modifications, leaving phenomenon-specific decisions to the transfer step.

What the program effectively does is traverse the c-structure from the root node, keeping track of the c-structure/f-structure correspondence. This leads to a set of partial subtrees consisting only of connected co-projecting c-structure nodes. A feature representation of such subtrees is then added to the respective f-structure under a special feature `TT_TREE`. Some special care has to be taken since a single f-structure may have several corresponding connected subtrees; furthermore, it has to be made sure that the connection to the words in the string remains recoverable, using a special feature `TT_TERM-CAT`.[9]

A detail of an enriched f-structure is shown in Figure 8 (here, not the Prolog term itself is shown, but the XLE display of it – the modified export format can be read in and displayed again). Note how the subtree projected by the adverb *hier* is merged into the feature structure of the corresponding predicate. The pointer feature `TT_PHI` (somewhat redundantly) has the feature structure of *hier* as its value. This configuration allows to test for features and values even in more complex structures without moving through the recursive tree structure. The integrated c-structure information does not only include information about the syntactic categories (e.g. `ADV[std]`, `AVDP[std]`) but also sublexical information like part of speech and lemma (e.g. `+Adv+Common`, `hier`), and the pointer to the surface token (`TT_SFF_ID`).

The preprocessing routine, in addition, splits parametrized category symbols like `ADVP[std]` into functor-argument lists. This is relevant for generalizations over parametrized features since the transfer system does not allow to use regular expressions on label names.

3.4 The Transfer Grammar

We made use of the transfer system of the XEROX Translation Environment by Martin Kay (XTE) which is part of the XLE development platform. The transfer component is a rule rewriting system based on Prolog. As mentioned in Section 3.1, differences in representation or linguistic analysis lead to structural

[9]This part is in fact non-trivial since the PARGRAM LFG grammar does not operate on a string of fullform words, but on the output of a morphological analyzer which adds branching to the c-structure that would not be recoverable in the target structure, given just the word string.

```
fstructure('tiger_4548:  Hier herrscht Demokratie .',
     [...]
     % Constraints:
     [
     cf(1,eq(attr(var(0),'PRED'),semform('herrschen',3,[var(1)],[]))),
     cf(1,eq(attr(var(0),'ADJUNCT'),var(2))),
     cf(1,eq(attr(var(0),'SENTENCE_ID'),'tiger_4548')),
     cf(1,eq(attr(var(0),'STMT-TYPE'),'decl')),
     cf(1,eq(attr(var(0),'SUBJ'),var(1))),
     cf(1,eq(attr(var(0),'TNS-ASP'),var(3))),
     cf(1,in_set(var(5),var(2))),
     cf(1,eq(attr(var(5),'PRED'),semform('hier',1,[],[]))),
     cf(1,eq(attr(var(5),'ADV-TYPE'),'adj-sem')),
     cf(1,eq(attr(var(1),'PRED'),semform('Demokratie',7,[],[]))),
     cf(1,eq(attr(var(1),'CASE'),'nom')),
     cf(1,eq(attr(var(1),'GEND'),'fem')),
     cf(1,eq(attr(var(1),'NUM'),'sg')),
     cf(1,eq(attr(var(1),'PERS'),'3')),
     cf(1,eq(attr(var(3),'MOOD'),'indicative')),
     cf(1,eq(attr(var(3),'TENSE'),'pres')),
     ],
     % C-Structure:
     [
     [...]
     cf(1,subtree(2273,'CP[std,-dep]',-,1217)),
     cf(1,phi(2273,var(0))),
     cf(1,subtree(1217,'ADVP[std]',-,1005)),
     cf(1,phi(1217,var(5))),
     cf(1,cproj(1217,var(10))),
     cf(1,subtree(1005,'ADV[std]',1004,111)),
     cf(1,phi(1005,var(5))),
     cf(1,subtree(1004,'ADV[std]',-,61)),
     cf(1,phi(1004,var(5))),
     cf(1,terminal(62,'hier',60)),
     cf(1,phi(62,var(5))),
     cf(1,terminal(112,'+Adv+Common',60)),
     cf(1,phi(112,var(5))),
     [...]
     cf(1,surfaceform(0,'tiger_4548',0,3)),
     cf(1,surfaceform(26,':',3,16)),
     cf(1,surfaceform(60,'hier',16,21)),
     cf(1,surfaceform(117,'herrscht',21,30)),
     cf(1,surfaceform(169,'Demokratie',30,36)),
     cf(1,surfaceform(195,'.',36,49))
     ]).
```

Figure 7: tiger_4548: Prolog term: c- and f-structure (details)

mismatches that are comparable to transfer ambiguities in natural language translation. It was reasonable, therefore, to develop the mapping in an actual transfer environment. Using a tested and integrated system had the additional advantage that neither a specification language nor the processing routines had to be developed. The main focus lay on the interface routines and the specification of the mapping.

The preprocessed LFG parse, cf. Figure 8, functions as transfer source format. It is a flat list of predicate-value pairs in which, for example, the LFG function SUBJ(ect) is represented as a two-place predicate that takes two f-structure indices as arguments: `subj(X,Y)` [10]. The predicate-value list is transformed step by step into the TIGER target format. The Transfer rewriting rules apply in an ordered way to the gradually changing set of predicates, which means that the output of a given rule is the input of the subsequently following rule. The architecture of the grammar file therefore mirrors the order of

[10] `Subj(X,Y)` reads as 'the predicate of f-structure Y is the subject of the predicate of f-structure X'. As common in Prolog, constants begin with lower case, variables begin with upper case or an underscore.

```
PRED      'herrschen<[-61-SUBJ:Demokratie]>'
          PRED    'hier'
                                   TT-CPLX [TT-ARG1 std, TT-FCTR ADVP]
                                -2 TT-ID 1217, TT-LABEL ADVP[std]
                                   TT-CPLX [TT-ARG1 std, TT-FCTR ADV]
                     TT-CAT     -3 TT-ID 1005, TT-LABEL ADV[std]
                                   TT-CPLX [TT-ARG1 std, TT-FCTR ADV-T_BASE]
ADJUNCT   TT-TREE               -4 TT-ID 111, TT-LABEL ADV-T_BASE[std]
                                -5 [TT-ID 61, TT-LABEL ADV-S_BASE]
                     TT-PHI     [-1-TT-PHI:hier]
                     TT-SFF-ID  60
                     TT-TERM-CAT -6 [TT-ID 112, TT-LABEL +Adv+Common]
                                 -7 [TT-ID 62, TT-LABEL hier]
                  -1 TT-TOP-CAT [TT-ID 1217, TT-LABEL ADVP[std]]
          ADV-TYPE adj-sem, OBL-SEM loc
```

Figure 8: tiger_4548: Detail of the enriched f-structure (Transfer target format)

potential rule application.

Rules Input predicates are on the left-hand side of a rule; output predicates are on its right-hand side. Input and output predicates are separated by a rewriting symbol, the operator '==>'. The most basic rules simply rewrite the name of the predicate and pass on the values of the arguments unchanged. For example, the LFG function SUBJ is mapped to the TIGER function SB, the function OBLAGT (the optional agent in a passive clause) to SBP, respectively. The argument slots are not manipulated, i.e. the dependency structure is passed on unaltered.

(4)
```
subj(X,Y)     ==>  sb(X,Y).   %  subject
oblagt(X,Y)   ==>  sbp(X,Y).  %  agent in passives ("von"-phrase)
```

A predicate on the input side of a rule is deleted from the input set of predicates, and a predicate on the output side of a rule is added to the output set of predicates. Rules may be contextually restricted by positive and negative tests. The operator '+' preceding a predicate indicates that the predicate is required in the input set for the rule to apply although the rule does not affect the predicate itself. A preceding '−' triggers a negative test: the rule is applied only if the predicate is absent. For example, in (5), the LFG function XCOMP is mapped to the TIGER function PD (predicative) only if there is a coindexed predicate XCOMP−TYPE(X,'copula'). Otherwise, XCOMP is rewritten as the TIGER function OC (clausal object).

(5)
```
+xcomp_type(Y,'copula'),  xcomp(X,Y)  ==>  pd(X,Y).  %  predicative
                          xcomp(X,Y)  ==>  oc(X,Y).  %  default
```

A zero on the right-hand side encodes the empty set, see e.g. (6). In this case, all predicates on the left-hand side of the rule are deleted from the set of predicates without replacement.[11]

(6)
```
tt_tree(_,_)  ==>  0.  %  deletion
```

[11]The underscore is the common symbol for the anonymous variable in Prolog.

Macros and Templates The system allows for the definition of macros and templates—short-hand notations of sets of predicates and transfer rules, respectively. They do not only facilitate rule development but also the adaptation to changes in either the source or target format. A format change, then, only requires to adapt a given macro or template but not all occurrences of it in the grammar.

The macro `TT_VERB_MORPH` in (7), for example, is an abbreviation of the predicates encoding the information relevant for the mapping of verbal morphology tags. The predicates include `TENSE` and `MOOD`, which are embedded in the verbal `TNS-ASP` feature; and the corresponding person and number information which LFG encodes as `PERS` and `NUM` embedded in the subject feature (`SB`).

(7)
```
tt_verb_morph(V,Person,Number,Tense,Mood):=

   +tns_asp(V,V1),
      +tense(V1,Tense),
      +mood(V1,Mood),

   +sb(V,V2),
      +pers(V2,Person),
      +num(V2,Number).
```

Just like macros are short-hand forms of predicates, templates are short-hand forms of rules. In (8), the template `LABEL2HEAD` expands to a rule that maps a c-structure label, `FCTR`, to a TIGER head relation, `HEAD`. The rule passes on the form and index of the surface token. In addition, it introduces a part-of-speech tag. The input predicates share their first argument, i.e., they are all connected to a specific feature structure in the input representation. `TT-PHI` is a pointer that relates c-structural information to the corresponding f-structure index. It allows one to link the target predicates directly to the dependency structure.

(8)
```
label2head(Fctr,Arg1,Head,Pos) ::

   cat_label(V,Fctr,Arg1),             %  macro for category label
   tt_phi(V,V0),                       %  pointer to f-structure
   sff_in(V,Id,Form)                   %  macro for linking of sur-
                                       %  face form and surface id
   ==>
   ti_terminal(V0,Head,Pos,Id,Form).   %  macro for TIGER heads
```

The actual transfer rules instantiate templates in that all argument slots are filled with constants, see (9)[12]. Templates can also encode a sequence of rules. In this case, the grammar compiler expands the instantiated rule accordingly.

(9)
```
label2head('ADV','std',hd,'ADV').      %  standard adverbs

fctr2head('ADV',hd,'PWAV').            %  interrogative and
                                       %  relative adverbs
label2head('PAdv','std',hd,'PROAV').   %  pronominal adverbs

fctr2head('PAdv',hd,'PWAV').           %  interrogative and
                                       %  relative pronominal
                                       %  adverbs
```

[12] `LABEL2HEAD` and `FCTR2HEAD` differ only with respect to the category label. `FCTR2HEAD` generalizes over the value of `Arg1`. `LABEL2HEAD` can express `FCTR2HEAD` if its second argument is instantiated with the anonymous variable. In this case any value of `Arg1` matches the input requirements. Since it turned out to be less efficiently processed, the grammar rules make no use of this encoding option.

Structure of the Grammar The transfer component applies the rules in the order of specification, i.e. the transfer grammar mirrors the mapping process. Each compiled rule is called only once in the mapping process and is then applied to all predicates that match the rule input requirements. The grammar is organized in seven main parts:

(i) Redundant predicates are deleted. E.g. the subject feature of adjectives in LFG has no correspondence in the TIGER format[13]. It is therefore deleted and will not interfere with the mapping process.

(ii) For more efficient processing, all set-valued features are rewritten as relational predicates. The rule in (10) introduces `X_ADJUNCT` which is a temporary predicate in the sense that it is neither present in the transfer input nor in the transfer output. It is both, introduced and subsequently deleted in the process of transfer.[14]

(10)
```
+adjunct(V,V1), in_set(V2,V1)    %  set-valued
==> x_adjunct(V,V2).             %  relational

adjunct(_,_) ==> 0.              %  deletion
```

(iii) Grammatical functions that are encoded in the LFG f-structure are mapped to target predicates. In many cases the predicate names are just rewritten and the functional structure is passed on unchanged, e.g. `subj(V,V1) ==> sb(V,V1).`

(iv) Syntactic categories and syntactic heads are mapped in combination with part-of-speech and morphology tags. More specific rules thereby precede more general rules. If not all functional structure is part of the input, the mapping inserts structure, see Section 3.5 for a more detailed discussion of this.

(v) A repair section follows after the mapping proper. It includes rules that map temporary predicates on target predicates. For instance, the temporary head of finite auxiliaries `HD_AUX` is mapped on the (more general) target head function `HD`. The section also includes rules that 'repair' target format notation. For example, the TIGER format distinguishes between prepositional modifiers of nouns and other noun modifiers. The former are labelled 'modifier of the noun to the right' (`MNR`) whereas the rest is uniformly labelled 'noun kernel element' (`NK`). It is much simpler to map all noun modifiers alike and only to check for the specific case at the end of the mapping process, cf. (11).

(11)
```
nk(V,V1),+ti_cat(V1,'PP') ==> mnr(V,V1).  %  prepositional
                                          %  modifier of noun
```

The mapping rules are followed by two further rule sections. (vi) Robustness rules check for all functional labels and terminals whether they are integrated in the dependency structure – which is a necessary prerequisite for the canonical postprocessing convertion to the TIGER column format. If necessary a fragment relation is inserted. (vii) Finally, all non-target predicates are deleted.

3.5 Transfer Phenomena

In contrast to natural language transfer, TIGER Transfer leaves the surface string unchanged. The task is to map a limited set of grammatical features into another limited set of grammatical features. Although there are many trivial cases, the format conversion is more complex than a simple mapping of two feature sets. Due to differences in representation chosen for particular linguistic phenomena, there are mismatches that are comparable to 'transfer ambiguities' in natural language translation (for the latter see e.g. Kameyama et al. 1991, Emele et al. 2000).

[13] In the case of attributive adjectives, for instance, the subject points to the modified head noun.

[14] Doug Arnolds (p.c.) made us aware of the drawbacks temporary predicates might have if they are used in a large grammar.

Ambiguous Predicates A simple example was introduced in Section 3.4, the mapping of XCOMP to PD or OC, depending on the value of the feature XCOMP. A more complex case is the translation of the predicate ADJUNCT. It is a very general function in the German LFG and corresponds to three different TIGER functions, i.e., it is three-fold ambiguous. The conditions for resolving the ambiguity are not encoded in a specific feature, but have to be found independently. In (12), ADJUNCT is mapped to the function 'genitive attribute' (AG) if the embedding predicate is a noun (marked by NTYPE) and the embedded predicate has a case feature with the value gen(itive) – unless it is an attributive adjective (ATYPE attributive). Other ADJUNCTs within nouns are mapped to 'noun kernel element' (NK). The default mapping of ADJUNCT is to the function 'modifier' (MO).[15]

(12)
```
+ntype(V,_), +case(V1,'gen'),     %  required context
-atype(V1,'attributive'),         %  negative condition
x_adjunct(V,V1),
==>
ag(V,V1).                         %  'genitive attribute'

+ntype(V,_), x_adjunct(V,V1)      %  other noun modifiers
==>
nk(V,V1).                         %  'noun kernel element'

x_adjunct(V,V1)                   %  default:  modifiers of
==>                               %  verbs and adjectives
mo(V,V1).                         %  'modifier'
```

Some ambiguities cannot be resolved by the transfer component. For example, TIGER distinguishes two potential functions of prepositional phrases in predicative constructions, see (13). PPs with an abstract meaning, i.e. idiomatic chunks, are analyzed as predicatives (PD), all other PPs as modifiers (MO). Transfer provides only the function MO here. The adaptation to the specific TIGER edge label has to be done manually after transfer, e.g. with the (semi-automatic) ANNOTATE tool.

(13) a. PP functions as predicative:
Sie ist auf der Hut ('She is on her guard')

b. PP functions as modifier:
Sie ist im Garten ('Sie is in the garden')

Structural Changes There are 'head switch' transformations in natural language translation in which the head of the source structure becomes a dependent element in the target structure and a former dependent element becomes the head of the constituent. In Example (14), *like* is the matrix predicate which subcategorizes the infinitive *come*. In the corresponding German example in (15), *komme* is the main predicate which corresponds in meaning to the English embedded infinitive, *come*. The meaning of *like* is expressed by the adverb *gern* 'gladly' that modifies *komme*.

(14) I like to come

(15) Ich komme gerne
I come gladly

In the mapping of LFG to TIGER representations, there are constellations that resemble head switch. Figure 5 in Section 3.1 shows how the two systems represent the concept 'head of a clause' differently. In the German LFG, on the one hand, the main verb is always the main predicate of the clause. Since

[15] ADJUNCT is mapped to the temporary predicate X_ADJUNCT, cf. Section 3.4

temporal (or passive) auxiliaries do not have lexical meaning on their own, they co-project with the main verb on f-structure. In TIGER, on the other hand, the finite verb is analyzed as the head of the clause, independent of whether the verb is a main verb or an auxiliary. Accordingly, in analytic tenses, like perfect, the finite auxiliary is the clausal head and the main verb becomes the head of an embedded function `OC` ('clausal object'). The mapping has to reorganize the verbal heads and insert additional structure. The restructuring is guided by c-structural information.

The natural language mapping of Example (14) to Example (15) does not only affect the hierarchical structure but also the distribution of the arguments: The subject of the predicate *like* in English becomes the subject of the predicate *komme* in German. A more explicit case of argument rearrangement is given in the mapping of Example (16) to Example (17). The German verb *kennenlernen* is translated compositionally into the expression *get to know* in English. The mapping splits the arguments of *kennenlernen* and relates the subject (*sie/they*) to the structurally higher verb *get* and the object (*sich/each other*) to the structurally embedded verb *know*.

(16)	(Er	glaubt,)	dass	sie	sich	kennenlernten
	he	thinks	that	they	REFL	got_to_know

(17) (He thinks) that they got to know each other

Sometimes, arguments have to be rearranged in grammar mapping, as well. If you go back to Figure 5, again, you see that the German LFG treats all arguments the same: both the subject (SUBJ) and the indirect object (OBJ2) belong to the main predicate on f-structure. TIGER, in contrast, distinguishes between subjects and all other arguments. Only the subject is always dependent of the finite head. The other arguments are dependent on the main predicate; if the main predicate is embedded, the arguments are embedded as well, cf. the main predicate *geglaubt* and the indirect object (DA). Argument rearrangement is encoded in the repair section of the transfer grammar.

3.6 Postprocessing Steps

After the renaming and restructuring procedures illustrated in the preceding sections, further small modifications complete the mapping LFG – TIGER.

Morphology Tags Currently the TIGER treebank format supports only one slot for morphology tags. In LFG, the morphological features are distributed on different predicates, e.g. `gend(V,'Masc')`, `case(V,'Nom')`. They are collected during the transfer process. A Perl script subsequently joins all morphological features belonging to one token into one string. The morphology mapping could have been integrated in the transfer grammar, as well. But it would have slowed down the transfer procedure considerably.[16]

TIGER Column Format The output of the transfer proper is a Prolog file. A canonical postprocessing step (implemented in Prolog) converts the Prolog file, cf. Figure 9, into the TIGER column format, cf. Figure 10.

Tokenizer Modifications Furthermore, certain string manipulations of the tokenizer used in LFG parsing have to be undone. These string manipulations involve certain punctuation marks (commas surrounding embedded clauses), upper and lower case (sentence-initial), hyphenated compounds, and quotation marks. Hyphenated compounds such as *CDU-Frauen* 'CDU women' are split into two tokens by the LFG

[16]Because the system does not allow for variables ranging over predicates.

```
[HD-OUT [TI-FORM herrscht, TI-MORPH 3.Sg.Pres.Ind, TI-POS VVFIN, TI-SFF-ID 117]                           ]
[MO-OUT [HD-OUT [TI-FORM hier, TI-POS ADV, TI-SFF-ID 60]]                                                  ]
[       [TI-CAT AVP                                     ]                                                  ]
[PU-OUT [TI-FORM ., TI-POS Dollar., TI-SFF-ID 195]                                                         ]
[SB-OUT [NK-HD-OUT [TI-FORM Demokratie, TI-MORPH Fem.Nom.Sg.*, TI-POS NN, TI-SFF-ID 169]]                  ]
[       [TI-CAT    NP                                                                  ]                   ]
-1[SENTENCE-ID tiger_4548, TI-CAT S                                                                         ]
```

Figure 9: Transfer target format of tiger_4548

```
#FORMAT 3
#BOS 4548 102 947689949 1%% LFG 4548
hier            ADV      -              HD      502
herrscht        VVFIN    3.Sg.Pres.Ind  HD      500
Demokratie      NN       Fem.Nom.Sg.*   NK      501
.               $.       -              -       0
#500            S        -              -       0
#501            NP       -              SB      500
#502            AVP      -              MO      500
#EOS 4548
```

Figure 10: TIGER column format of tiger_4548

tokenizer, and analyzed accordingly by both the LFG morphology component and the transfer algorithm. In contrast, the TIGER annotation scheme treats hyphenated compounds as one token, therefore requiring adjustments after transfer. Concerning punctuation, the tokenizer problems are more difficult to solve. Typically, commas surround embedded clauses, but the second comma is omitted if the embedded clause immediately precedes the sentence final punctuation. However, in order to avoid different rule versions of embedded clauses (with and without second comma), the LFG tokenizer provides for additional commas in front of the sentence final punctuation mark. These additional commas have to be deleted after transfer. Quotation marks in German may enclose phrases and chunks as well as arbitrary sequences of words (non-constituents). They can even cross sentence boundaries. Hence stating a rule that deals with at least the most common types of quotation is nearly impossible. Therefore the LFG tokenizer deletes quotation marks and the postprocessing procedure has to recover them. All those modifications are dealt with mainly by a string comparison done directly after transfer. It compares transfer output strings to the original input sentences before LFG parsing. In case of faulty output, missing elements are inserted or superfluous ones deleted as appropriate. The repaired structures are then checked and, if necessary, completed by the human annotator using the semi-automatic ANNOTATE tool.

Redundant Information As already mentioned in Section 3.1, certain non-branching category projections are not represented in the TIGER format. This points to another specific feature of the TIGER annotation format, namely the tendency to avoid redundancy. To illustrate this feature, see the graphs of the example tiger_4548 shown in two different versions in Figure 11. The first tree is the current output version of TIGER Transfer as presented so far. The second tree shows the actual TIGER version: all nodes dominating just one daughter node have been omitted. This is motivated by the fact that the missing information is redundant in the sense that it can be inserted automatically. Furthermore, the annotation task becomes easier if the structure is less complex and less nodes have to be checked and corrected by the human annotator. Finally, since the structure is flatter, bigger parts of a tree can be displayed on the screen. In TIGER Transfer the respective projections are deleted in the course of the postprocessing.

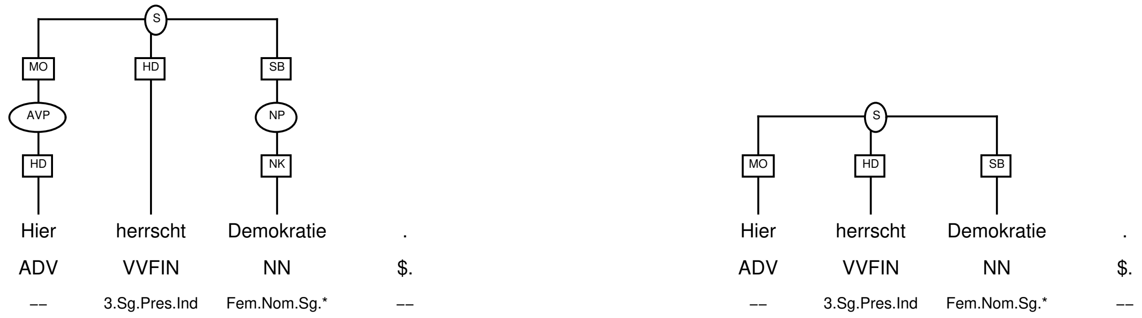


Figure 11: TIGER graph with and without non-branching nodes

4 Results

The German LFG Grammar analyzes on average 50.1% of the sentences, roughly 70% thereof are assigned a correct parse[17]. The sentences that are parsed contain 16.0 words on average and the average parsing time is 8.29 sec/sentence. Without using the OT filter mechanism a sentence gets 35,577 analyses on average (median: 20). After OT filtering, the average number of analyses drops to 16.5 (median: 2), cf. Dipper (2000). About 2,000 sentences of the TIGER corpus have been annotated by LFG parsing.

Although there are limits to the corpus coverage, the application of the grammar in annotation seems useful, in particular when combined with another annotation method. High quality treebanking requires at least two annotation passes for each sentence, and a comparison step. If one of the passes is classical manual annotation and the other pass is based on a full-depth grammar, the strength of the two techniques are combined, leading to a more consistent overall result.

5 Outlook

In the last section, we will discuss potential extensions to the LFG-based annotation: (i) partial analyses ('fragments'); (ii) morphology and lemma annotation; (iii) consistency checks. All three application tasks would benefit from a coverage extension of the grammar, an outline of which we will discuss at the very end of this paper.

Partial analyses For the annotation of new sentences a feature of the XLE system can be exploited, the 'Fragment mechanism'. This mechanism allows for every sentence to get at least a partial analysis. Hence the LFG-based annotation is no longer restricted to 35% of the corpus.

The fragment mechanism works as follows. A sentence is first parsed as usually; only if it does not get an analysis, XLE reparses the sentence. In this second parse XLE only tries to construct certain constituents, specified in advance by the grammar writer, e.g. `NP[std]`, `PP[std]` (ordinary NPs and PPs). Typically those constituents are maximal projections as defined in the grammar. Hence they are not chunks but may be complex and even recursive; an adjacent relative clause, e.g., is always part of the respective NP-fragment. Other constituents that are suited for fragments are subordinate clauses (adverbial and subcategorized), since their left and right boundaries are marked clearly.

First experiments with fragments were very promising. For instance, noun chunks were analyzed with a precision of 89% and a recall of 67%; prepositional chunks were found with a precision of 96%

[17] 10% of the sentences failed because of gaps in the morphological analyzer; 6% failed because of storage overflow or timeouts (with limits set to 100 MB storage and 100 seconds parsing time). 10% of the parsed sentences were not evaluated wrt. the correct analysis because they received more than 30 analysis after OT-filtering.

(ignoring PP-attachment for the evaluation) and a recall of 79%, cf. Schrader (2001).

The partial analyses yielded by the fragment mechanism would have to be mapped into partial analyses in the TIGER format and subsequently completed by human annotators supported by the tool ANNOTATE. In this scenario, TIGER Transfer has to be modified to deal with partial input and output.

Annotation of morphology It is planned in the TIGER project to add an annotation of morphology and lemma information to the sentences annotated syntactically so far; new sentences will be annotated with all information types. For this task the LFG grammar can be exploited easily. Each LFG analysis of a sentence automatically contains morphology and lemma information. The transfer already provides for a mapping of the respective tags.
There is is even a straightforward way to supply additional morphology and lemma information for sentences already annotated with syntactic structure, without having to disambiguate the parses again manually. Parts of the annotation of a sentence (e.g. predicate-argument structure, adjunct attachment) are transformed into Prolog terms. The sentence is parsed as usual and all analyses are stored in the Prolog export format. A test routine then picks out the analysis corresponding to the Prolog terms derived from the annotation. Thus the already existing annotation replaces the manual disambiguation step. Now the morphology and lemma information of the selected analysis is converted to the TIGER format. This way, large parts of the already annotated corpus can be automatically enriched by the LFG grammar with new information.

Consistency checks The method sketched in the preceding paragraph can also be used to perform consistency checks. Especially in the domain of part-of-speech tags, human annotators easily overlook errors. However, for a high quality corpus such as the TIGER treebank, correct part-of-speech tags are as important as correct structures. Likewise for application such as TIGERSearch (a query tool for the TIGER corpus), it is often necessary to rely on part-of-speech information, e.g. when looking for postnominal adverbs as in *Hans selbst* 'Hans himself'. The flat annotation style applied in TIGER makes this point even more important. In TIGER there is no structural property (node, label) in an NP differentiating between, e.g. a determiner, an attributive adjective, and the head noun. The only difference is their part-of-speech tags `ART`, `ADJA`, `NN/NE`, respectively.

As sketched above, information such as predicate-argument-structure of the existing annotation will be mapped to Prolog terms thus abstracting from part-of-speech tags. The sentence is parsed and disambiguated automatically, and the transfer generates the TIGER representation format. This way, part-of-speech tag errors will be detected automatically.

Coverage extension Furthermore the annotated corpus could be exploited for extending the coverage of the grammar – this would of course improve the results of the mentioned tasks performed with the grammar. With these applications in mind, it would be justified to aim particularly at systematic extension of coverage with respect to the given corpus.

There are two ways in which it is realistic to expect a possible coverage extension, now that the annotated TIGER corpus is available. The first strategy relies on 'classical' grammar writing, i.e., involving a linguist who identifies missing rule parts or lexicon entries. When this technique is applied, trying to extend a grammar that already has a relatively broad coverage, a predominant problem is the detection of unintended interactions. How can one exclude that a rule modification required for a given sentence cause the grammar to break down on a number of other sentences? Only if a sufficient amount of realistic sentences is used for a comparative test can modifications be accepted with reasonable confidence.

Here, the treebank can be used as a test suite in regression testing. While any collection of sentences can be used to compare the grammar behavior before and after a modification, only the annotation of the correct analysis will guarantee the desired grammar behavior. (In very short, unambiguous sentences this is less of a problem, but when realistic sentences change from eighty to sixty readings, inspection of the solutions would be required to make sure that the twenty readings lost are irrelevant.) Technically, the TIGER-derived test suite used in grammar development will be a list of strings annotated with target structures in a similar way as proposed in Kuhn (1998) (we use a slightly different scheme now, in which XLE's export representation is compared with the stored target representation). In essence, the annotation structures will specify predicate-argument structure and modifier attachment (as mentioned above), but leave further details open to the grammar.

Let us now turn to the second strategy we would like to experiment with for extending coverage with respect to the TIGER corpus. The basic idea turns on the fact that a high-quality symbolic grammar has to be highly restricted when it is applied to unseen text, in order to avoid that overgeneration leads to a proliferation of readings per sentence. For instance, the subcategorization frames listed in the verb lexicon are kept very restricted, although parsing failures are often due to missing frames for a known verb. But the alternative of relaxing such restriction would require some external control of the additional readings that would arise.

With the target annotation present in the TIGER corpus, this external control is actually in place. So, when the grammar is used to reparse the corpus, it makes sense to relax some of the strict conditions encoded in the lexicon and rules. The larger set of readings arising for each sentence will be cut down immediately by allowing only parses that meet the annotated predicate-argument structure (again using the test suite). Thus, not only sentences that are covered by the 'classical' grammars could be reparsed successfully, but hopefully also some significant proportion of previously uncovered sentences.

Ultimately, one could run training experiments with the statistical model of Riezler et al. (2000) over the relaxed grammar. The TIGER corpus would serve as 'complete data', i.e. supervised training material, so the relaxed grammar could then also be applied on unseen sentences with external control over the readings (in this case the external control is exerted by the statistical model). Evaluating such an experiment can be expected to be highly revealing about the information sources required for a large-scale unification grammar.

References

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Teebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol.

Brants, Thorsten. 1999. *Tagging and Parsing with Cascaded Markov Models – Automation of Corpus Annotation*, volume 6 of *Saarbrücken Dissertations in Computational Linguistics an d Language Technology*. Saarbrücken, Germany.

Dipper, Stefanie. 2000. Grammar-based Corpus Annotation. In *Proceedings of LINC-2000*, pp. 56–64, Luxembourg.

Dipper, Stefanie. to appear. *Implementing and Documenting Large-scale Grammars – German LFG (working title)*. PhD thesis, IMS, University of Stuttgart.

Emele, Martin, Michael Dorna, Anke Lüdeling, Heike Zinsmeister, and Christian Rohrer. 2000. Semantic-based Transfer. In *Verbmobil: Foundations of Speech-to-Speech Translation*, pp. 359–376.

Frank, Anette, Tracy H. King, Jonas Kuhn, and John Maxwell. 2001. Optimality Theory Style Constraint Ranking in Large-scale LFG Grammars. In Peter Sells (ed.), *Formal and Empirical Issues in Optimality-theoretic Syntax*, pp. 367–397. Stanford: CSLI Publications.

Kameyama, Megumi, Ryo Ochitani, and Stanley Peters. 1991. Resolving Translation Mismatches with Information Flow. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL'91)*, Berkeley, CA.

King, Tracy Holloway, Stefanie Dipper, Annette Frank, Jonas Kuhn, and John Maxwell. to appear. Ambiguity Management in Grammar Writing. *Journal of Language and Computation*.

Kuhn, Jonas. 1998. Towards Data-intensive Testing of a Broad-coverage LFG Grammar. In Bernhard Schröder, Winfried Lenders, Wolfgang Hess, and Thomas Portele (eds.), *Computers, Linguistics, and Phonetics between Language and Speech, Proceedings of the 4th Conference on Natural Language Processing – KONVENS-98*, pp. 43–56, Bonn. Peter Lang.

Kuhn, Jonas. 2001. Generation and parsing in Optimality Theoretic syntax – issues in the formalization of OT-LFG. In Peter Sells (ed.), *Formal and Empirical Issues in Optimality-theoretic Syntax*, pp. 313–366. Stanford: CSLI Publications.

Lee, Hanjung. 2001. Markedness and Word Order Freezing. In Peter Sells (ed.), *Formal and Empirical Issues in Optimality-theoretic Syntax*, pp. 63–128. Stanford: CSLI Publications.

Lezius, Wolfgang. 2002. *Ein Werkzeug zur Suche auf syntaktisch annotierten Textkorpora*. PhD thesis, IMS, University of Stuttgart.

Maxwell, John, and Ron Kaplan. 1989. An Overview of Disjunctive Constraint Satisfaction. In *Proceedings of the International Workshop on Parsing Technologies*, Pittsburgh, PA.

Plaehn, Oliver, and Thorsten Brants. 2000. Annotate – An Efficient Interactive Annotation Tool. In *Proceedings of ANLP-2000*, Seattle, WA.

Riezler, Stefan, Detlef Prescher, Jonas Kuhn, and Mark 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*, pp. 480–487.

Schrader, Bettina, 2001. Modifikation einer deutschen LFG-Grammatik für Partial Parsing. Studienarbeit.

Skut, Wojciech, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An Annotation Scheme for Free Word Order Languages. In *Proceedings of ANLP-97*.

Heike Zinsmeister
`zinsmeis@ims.uni-stuttgart.de`
IMS
University of Stuttgart
Azenbergstr.12
Postfach 10 60 37
D-70049 Stuttgart
Germany

Stefanie Dipper
`dipper@ims.uni-stuttgart.de`
IMS
University of Stuttgart
Azenbergstr.12
Postfach 10 60 37
D-70049 Stuttgart
Germany

Jonas Kuhn
`jonask@mail.utexas.edu`
Department of Linguistics
1 University Station, B5100
University of Texas at Austin
Austin, TX 78712-1196
USA