

TIGER TRANSFER

From LFG Structures to the TIGER Treebank

Heike Zinsmeister, Jonas Kuhn, Bettina Schrader
& Stefanie Dipper

IMS, University of Stuttgart, Germany
www.ims.uni-stuttgart.de/projekte/TIGER

April 2001

Contents

1	Introduction	3
2	The LFG Analysis	4
2.1	German LFG Grammar	4
2.2	Disambiguation	5
3	From LFG to TIGER	8
3.1	LFG F-structure and Tiger Graphs	9
3.2	Criteria for implementing the conversion procedure	11
3.3	Preprocessing	12
3.4	The Transfer Grammar	14
3.5	Transfer Phenomena	18
3.6	Postprocessing Steps	20
4	Results	23
5	Future Work	23

1 Introduction

This paper reports on work done in the context of the TIGER project.¹ The TIGER project aims at creating a large German treebank, the TIGER treebank, and at developing search tools for exploiting the information encoded in the treebank (TIGERSearch). The annotation is very detailed in that it encodes information about part-of-speech, e.g.: NN, VMFIN (common noun, finite modal verb); morphology: Masc.Nom.Sg (planned); syntactic category: NP, PP (noun/prepositional phrase); and grammatical function: 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)). In this format, for instance, a nominal phrase forms a (discontinuous) constituent with an extraposed relative clause modifying 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 human 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, cf. fig. 1.

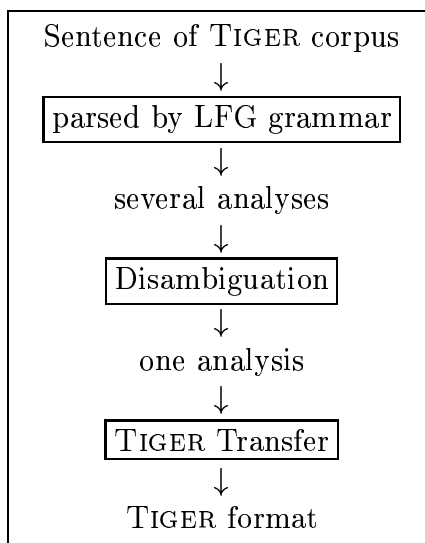


Figure 1: Scenario of annotation by LFG

Technique (i) is the main line in the annotation process; (ii) has a more experimental status. A motivation for the use of (ii) is to explore to what extent a pre-existing broad-

¹We would like to thank Anette Frank (DFKI Saarbrücken, formerly at XEROX Grenoble) for her great help with the transfer component; credit for the original idea of exploiting an existing transfer module goes to Martin Emele. Besides these two people, we would like to thank Stefan Evert, Jan Anderssen, Hannah Kermes, and the audience at the Workshop on “Syntactic Annotation of Electronic Corpora” (Tübingen, June 2000) for discussion and comments.

The TIGER project (URL: <http://www.coli.uni-sb.de/cl/projects/tiger>) is funded by the *Deutsche Forschungsgemeinschaft* (DFG).

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 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.

This report focusses on the grammar-based annotation approach (ii), and in particular on the transfer component within this method: TIGER Transfer. The paper is organized as follows: Sec. 2 gives a short introduction to the German LFG grammar used in annotation and addresses the disambiguation task. Sec. 3 presents the transformations that an LFG analysis of a sentence has to undergo on its way to the TIGER representation. Sec. 4 gives some statistics, sec. 5 an outlook on future work.

2 The LFG Analysis

This section gives a short introduction to the German LFG grammar used in annotation (2.1) and addresses the disambiguation task (2.2).

2.1 German LFG Grammar

The German grammar applied in parsing is a Lexical Functional Grammar (LFG, Kaplan and Bresnan (1982), Bresnan (2001)) and was developed in the PARGRAM project,² using the Xerox Linguistic Environment (XLE). The analysis an LFG grammar yields for a given sentence consists of 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.

Fig. 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 – the latter include a fine-grained differentiation of category symbols, among others so-called complex category symbols with category-level features in the squared brackets). For the present purposes, such details can be ignored. The German LFG grammar encodes a generalized CP analysis of German. The finite verb *herrscht* thus occupies the C position (= the so-called “Verb Second position”), preceded by the adverb phrase *hier* in the specifier position of CP the so-called “Vorfeld position”). The NP *Demokratie* is directly dominated by Cbar.³ Note finally, the ROOT node does

²URL: <http://www.parc.xerox.com/istl/groups/nlitt/pargram>

³For processing reasons, there is no VP projection covering the “Mittelfeld” and hence dominating the NP.

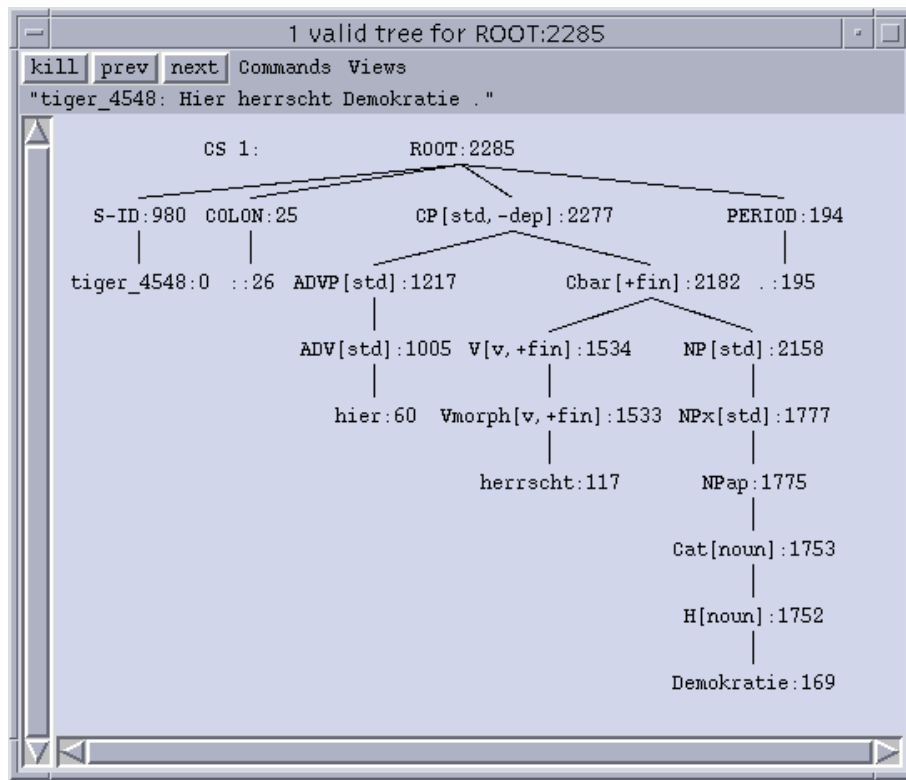


Figure 2: C-structure of tiger_4548: *Hier herrscht Demokratie*.

not only dominate the clausal projection of the sentence but also its identification and its final punctuation mark (S-ID + COLON).

Fig. 3 shows the f-structure of *Hier herrscht Demokratie*. 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), SUBJECT points to the predicate *Demokratie*. In addition the f-structures contain morphosyntactic information like TENSE, MOOD, CASE, GENDER, and so on.

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 fig. 2 and 3, for instance, the c-structure nodes representing the projections of the adverb *hier* are indexed by 60, 1005, and 1217 and are mapped to the feature structure that is the value of the (set-valued) feature ADJUNCT.

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 subcat-

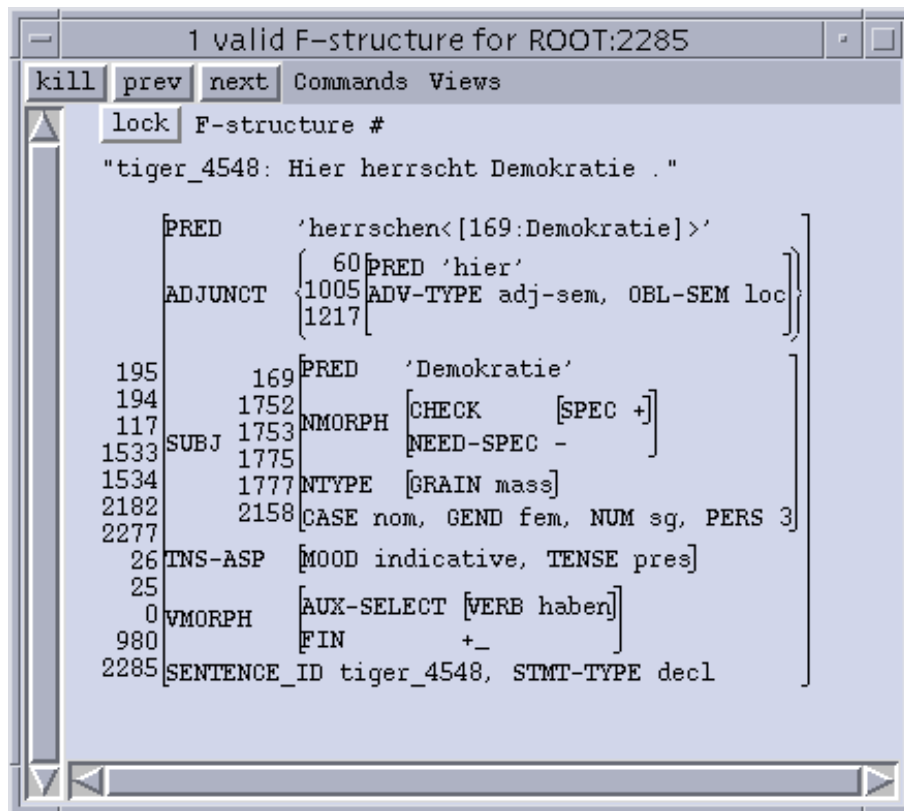


Figure 3: **F-structure** of tiger_4548: *Hier herrscht Demokratie*.

egorization. Hence the grammar output has to be disambiguated, i.e. a human annotator has to select the correct analysis.⁴ 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.

⁴Shallow 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 human annotator (this is only practical since the grammar includes many more explicit grammatical constraints than the grammars that are used in a shallow parser).

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 further supports manual disambiguation by various other browsing tools applied to c-structure as well as to f-structure (cf. King et al. (2000) where these tools are described in detail).

In the following example 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). In fig. 2.2, the f-structure alternatives restricted to the transitive reading are annotated by the variable a:1, the ditransitive ones by a:2. The ambiguity can only be resolved with knowledge of the context.

- (1) Die Stadt verkaufte das Haus der Stiftung .
- a. Die Stadt_{NOM} verkaufte [das Haus_{ACC}] [der_{DAT} Stiftung].
'The town sold the house to the foundation.'
 - b. Die Stadt_{NOM} verkaufte [das Haus_{ACC} [der_{GEN} Stiftung]].
'The town sold the house of the foundation.'

Automatic Disambiguation The example from the previous section, *Die Stadt verkaufte das Haus der Stiftung*, is even more ambiguous. Other ambiguities arise with respect to *das Haus* und *die Stadt*. Both can either be nominative or accusative, i.e., subject or direct object of the clause (cf. the additional readings (1c,d)). In this case one reading is rather improbable, namely that with the object occupying the first position (the so-called "Vorfeld"). In German there is a general dispreference for objects in the Vorfeld position. This as well as other biases are exploited by a (non-statistical) mechanism XLE provides for suppressing certain ambiguities automatically. The mechanism consists of a constraint ranking scheme inspired by Optimality Theory (OT) Frank et al. (2001). Each rule and each lexicon entry can be marked by so-called OT marks. When a sentence is parsed, each analysis is annotated by a multi-set of OT marks thus keeping a record of all OT-marked rules and lexicon entries 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.

- (1) c. Die Stadt_{ACC} verkaufte [das Haus_{NOM}] [der_{DAT} Stiftung].
'The house sold the town to the foundation.'
- d. Die Stadt_{ACC} verkaufte [das Haus_{NOM} [der_{GEN} Stiftung]].
'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 readings with *die Stadt_{ACC}* preceding

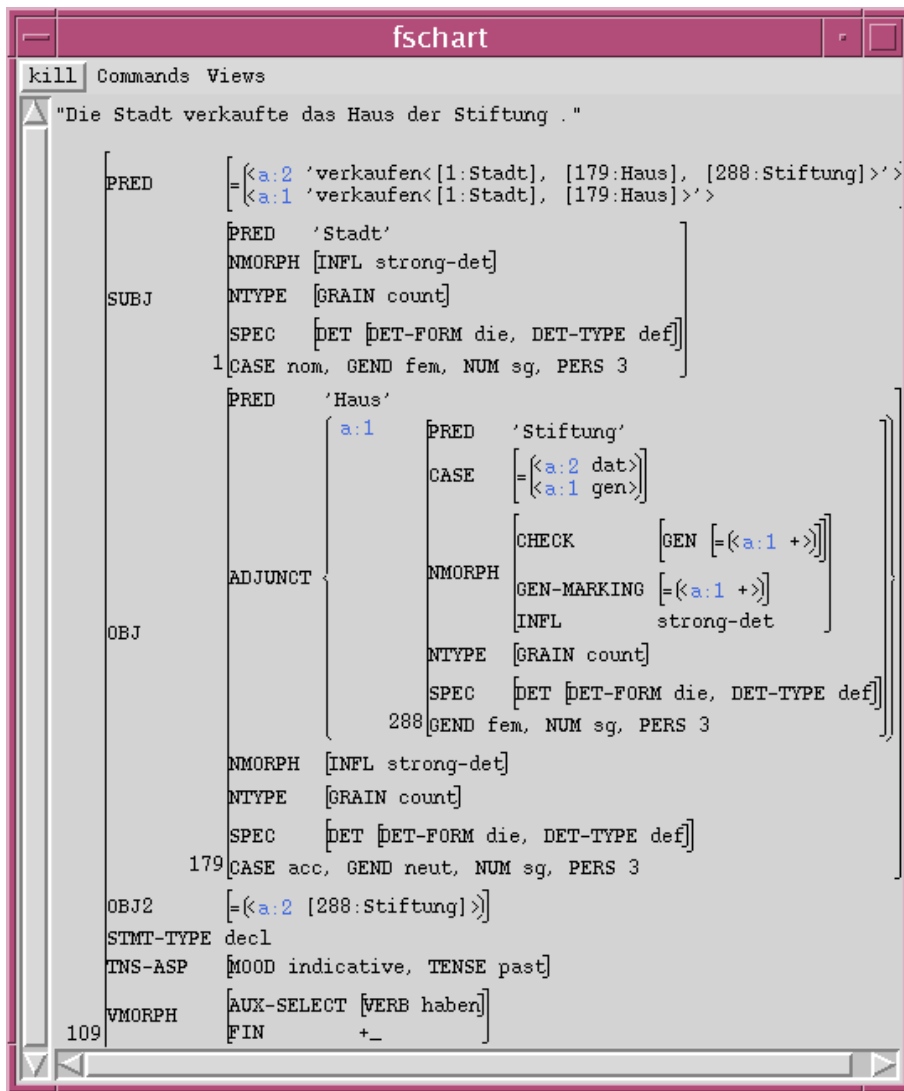


Figure 4: **Packed f-structure-chart** of *Die Stadt verkaufte das Haus der Stiftung* .

the finite verb (like the additional readings c. and d. of string (1)) are suppressed. For (1), the German LFG grammar reduces the total number of eight possible readings to two by means of the OT filter mechanism. In the future we plan to integrate a probabilistic disambiguation of a sentence analyses Riezler et al. (2000).

3 From LFG to TIGER

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

3.1 LFG F-structure and Tiger Graphs

Despite many differences in details, the syntactic analyses of the LFG grammar and the TIGER graph representation used for the treebanking are very similar at the level of functional/dependency structure. This provided the initial motivation for adopting the LFG-based annotation method. Both LFG's f-structure and the TIGER graph representation model a dependency structure at a comparable degree of granularity: If the simplified feature structure in fig. 5 is turned 90 degrees to the right it can almost directly be mapped on the TIGER graph in fig. 6 (with grammatical functions being marked at the square edge labels and syntactic categories encoded in circular nodes).

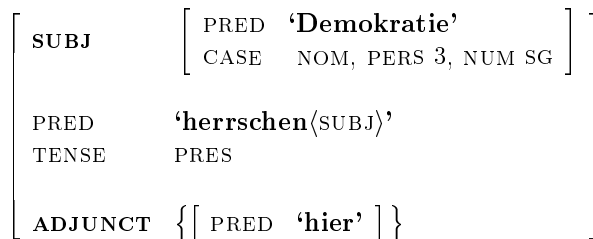


Figure 5: Simplified f-structure of tiger_4548

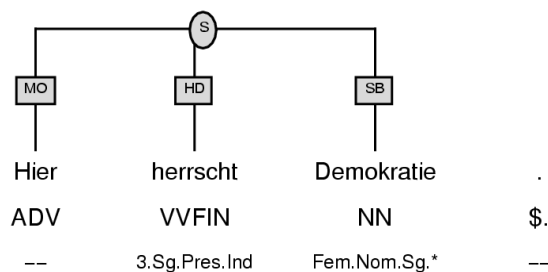
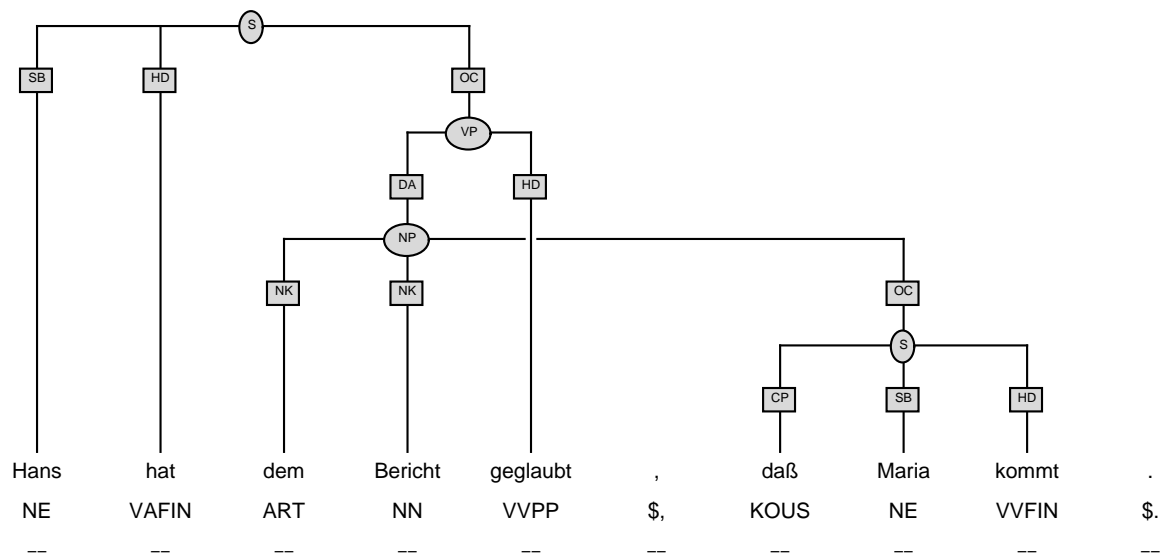


Figure 6: TIGER graph of tiger_4548

A further similarity between LFG and the TIGER format is that the dependency structure is explicitly related to the individual words forming the string underlying the analysis. In the case of LFG, this relation is mediated through the level of c-structure; in TIGER the relation is coded directly into the dependency graph, giving rise to the generalized tree graph (allowing for crossing branches) notation. Note the common strategy to give dependency structure priority in the structuring of the sentence and phrase structural constituency only a subordinate status. Consequently, in both schemes the situation can arise that a single f-structure/dependency-structural constituent corresponds to a set of terminal non-adjacent nodes, i.e., nodes with some intervening word material belonging to a higher-level f-structure/constituent. In fig. 7, the TIGER graph and LFG f-structure representation for *Hans hat dem Bericht geglaubt, daß Maria kommt.* ('Hans believed the report that Maria will come') are shown. The extraposed nominal complement clause *daß Maria kommt* yields a crossing branch in the TIGER annotation. In LFG the dependency is encoded in f-structure with the complement clause embedded in a COMP feature of the nominal predicate.



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

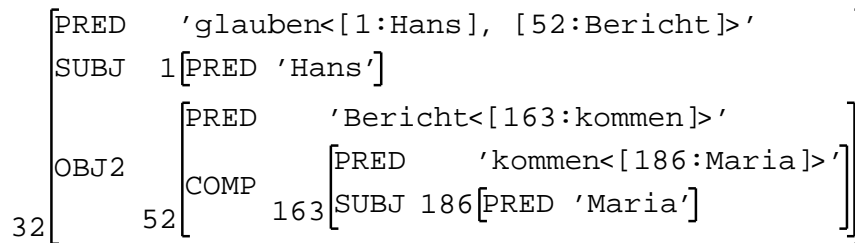


Figure 7: Long distance dependencies in TIGER graph and LFG f-structure representation

Along with these high-level similarities there are a number of differences in the representation conventions of TIGER vs. the German LFG grammar. One aspect can already be seen in fig. 6: in contrast to fig. 5, there are no phrasal projections NP, AVP dominating the words *Demokratie*, *hier*. In the TIGER format, redundant information is not encoded, cf. sec. 3.6.

Other discrepancies exist in the representational conventions for particular phenomena. They will be addressed in detail in section 3.5. For the moment, note as an example that in the TIGER representation, the verbal phrase *dem Bericht geglaubt* (containing the full verb) is embedded as a clausal object (OC) under the auxiliary *hat* in fig. 7. Contrary to this nesting analysis, in the LFG representation the auxiliary does not embed the full verb *glauben* – the verb and the auxiliary are 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.

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 procedure⁵ – until a final low-level conversion into the notational format of the TIGER treebank (this will be called the *postprocessing* below, cf. fig. 8). 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.

Besides the more or less notational conversion 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.

The resulting modular design shown in fig. 8 has some obvious advantages: Changes in the grammar or annotation scheme (unless radical) should only affect this “middle” step. Testing of the conversion steps can be performed separately, and it becomes an option simply to replace postprocessing step, for instance, if a different target format (e.g., XML) is desired.

⁵One advantage of this is that the 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. the figures 10 and 11 below).

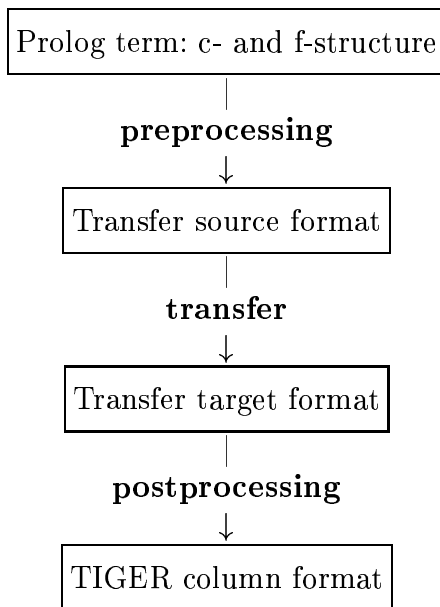


Figure 8: Subprocedures of the conversion routine

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 this order 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. fig. 9. 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

```

emacs@schwarzspecht
Buffers Files Tools Edit Search Prolog Help

structure('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,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(2285,'ROOT',2283,194)),
cf(1,phi(2285,var(0))),
cf(1,subtree(2283,'ROOT',982,2277)),
cf(1,phi(2283,var(0))),
cf(1,subtree(982,'ROOT',981,25)),
cf(1,phi(982,var(0))),
cf(1,subtree(981,'ROOT',-,980)),
cf(1,phi(981,var(0))),
cf(1,subtree(980,'S-ID',979,21)),
cf(1,phi(980,var(0))),
[...]
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))
])).

***-Emacs: tiger_4548-fschart.2.pl 12:10pm (Prolog[SICStus] Fill)--L2--A1
Find file: /projekte/pargram/tiger/report-new/

```

Figure 9: Prolog file of transfer input

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`.⁶

A detail of the enriched f-structure for our sample sentence `tiger_4548` is shown in fig. 10 (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 `TT_PHI` points to bracket 59, the feature structure of *'hier'*. This configuration allows one 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

⁶This part is in fact non-trivial since the 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.

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).

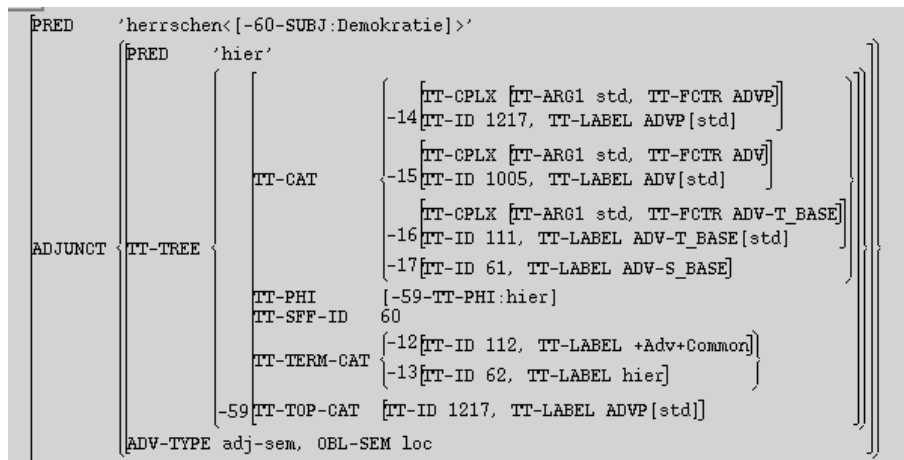


Figure 10: Detail of the enriched f-structure

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 sec. 3.1, differences in representation or linguistic analysis lead to structural 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 therefore lay on the interface routines and the specification of the mapping.

Syntax of the Rules The preprocessed LFG parse, a flat list of predicate-value pairs, is transformed step by step into the TIGER target structure. For example, the function SUBJ relating two feature structures is recoded as 2-place predicate taking the two f-structure indices as arguments: $\text{subj}(X,Y)$ ⁷. The rewriting rules apply in an ordered way to the gradually changing set of predicates. The output of a preceding rule is the input of the next rule following. The syntax of the rules is very transparent. The following will give a short introduction into it.

⁷As common in Prolog, constants begin with lower case, variables begin with upper case or an underscore.

Rules Input predicates are on the left-hand side of the rule. Output predicates are on the right-hand side of the rule. 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. In (2), 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 positions are not manipulated, i.e. the dependency structure is passed on unaltered.

```
(2)      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. A predicate on the output side of a rule is added to the output set of predicates. Rules can be contextually restricted. 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. In (3), the LFG function XCOMP is mapped to the TIGER function PD, predicative, only if there is a coindexed predicate XCOMP-TYPE(V1,'copula'). Otherwise, XCOMP is rewritten as the TIGER function OC, clausal object.

```
(3)      +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. All predicates on the left-hand side of the rule in (4) are deleted from the set of predicates without replacement.⁸

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

Macros and Templates The system allows for defining macros and templates, shorthand 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 input or output format. The following exemplifies the use of these devices.

The macro TT-VERB-MORPH in (5) is an abbreviation of the predicates encoding the information relevant for the mapping of verbal morphology tags. The predicates include the TENSE and MOOD features (embedded in the verbal TNS-ASP feature) and the corresponding person and number information encoded in the PERS and NUM feature of the subject (SB).

⁸The underscore is the common symbol for the anonymous variable in Prolog.

```
(5)   tt_verb_morph(V,Pers,Num,Tense,Mood) :=

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

      +sb(V,V2),
      +pers(V2,Pers),
      +num(V2,Num).
```

As macros are short-hand forms of predicates, templates are short-hand forms of rules. In (6), the template LABEL2HEAD expands to a rule that maps a c-structure label to a TIGER head relation. 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 to link the target predicates directly to the dependency structure.

```
(6)   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 the linking of the
                                  % surface 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 (7)⁹. Templates can also encode a sequence of rules. In this case, the grammar compiler expands the instantiated rule accordingly.

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

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

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

⁹LABEL2HEAD 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 structure of the transfer grammar mirrors the mapping process as the rules apply in the given order. Each compiled rule is called only once in the mapping process and applied to all predicates that match the rule input requirements. The grammar file starts with the definitions of macros and templates. Many of the actual transfer rules are specified using templates. The rules are organized in the following main parts.

(i) Redundant predicates are deleted. An example of a such a predicate is the subject feature that LFG assigns to adjectives which has no correspondence in the TIGER format.¹⁰

(ii) For more efficient processing, all set-valued features are rewritten as relational predicates. (8) 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.¹¹

```
(8)      +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 f-structure are mapped to target predicates. In many cases the predicate names are just rewritten and the functional structure is passed on unchanged, see (9) for sample rules applying in the mapping of tiger_4548: *Hier herrscht Demokratie* ('Democracy rules here').

```
(9)      x_adjunct(V,V1) ==> mo(V,V1).          % "hier"
          subj(V,V1)      ==> sb(V,V1).          % "herrscht"
                                          % "Demokratie"
                                          % "."
```

(iv) Syntactic categories and syntactic heads are mapped in combination with part-of-speech and morphology tags, see (10). Since order matters, more specific rules precede more general rules. If not all functional structure is given, the mapping also inserts structure, see sec. 3.5 for a more detailed discussion of this.

```
(10)     fctr2cat('ADVP', 'AVP').                % "hier"
          label2head('ADV', 'std', hd, 'ADV').
          vfin('Vmorph', 'v', '+fin', hd, 'VVFIN'). % "herrscht"
          fctr2cat('NP', 'NP').                  % "Demokratie"
          nn('H', 'noun', nk_hd, 'NN').
          punct_pos('PERIOD', 'Dollar.').        % "."
```

¹⁰In the case of attributive adjectives, for instance, the subject points to the modified head noun.

¹¹Doug Arnolds (p.c.) made us aware of the drawbacks temporary predicates might have if they are used in a large grammar.

(v) After the mapping proper follows a repair section. This includes rules that map temporary predicates on target predicates, for instance the specific head of finite auxiliaries HD_AUX on the general target head function HD. Other rules adopt specific target notations, see e.g. (11) for prepositional modifiers of nouns. In TIGER, they are labeled differently from other noun modifiers. They do not function as 'noun kernel element' (NK) but as 'modifier of the noun to the right' (MNR).

```
(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 conversion to the TIGER column format. If necessary a fragment relation is inserted. (vii) Finally, all non-target predicates are deleted.

The TIGER Transfer grammar expanded by the grammar compiler comprises 676 rules (without general deletion and robustness rules).

3.5 Transfer Phenomena

In contrast to natural language transfer, TIGER Transfer passes on the surface string. 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 so-called “transfer ambiguities” in natural language translation (for the latter see e.g. Kameyama et al. (1991), Emele et al. (2000)).

Ambiguous Predicates A simple example was introduced in 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 and the embedded predicate has a case feature with the value 'genitive' – unless it is an attributive adjective. Other ADJUNCTs within nouns are mapped to 'noun kernel element', NK. The overall default mapping for ADJUNCT is to the function 'modifier', MO.¹²

¹²ADJUNCT is mapped to the temporary predicate X_ADJUNCT, cf. 3.4

```

(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, in predicative constructions TIGER distinguishes two potential functions of prepositional phrases, see (3.5). PPs with an abstract meaning, i.e. idiomatic chunks, are analysed 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:
 Ich war auf der Hut ('I was on my guard')
- b. PP functions as modifier:
 Die Frau ist im Garten ('the woman is in the garden')

Head Switch The term refers to 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 (14), *like* is the matrix predicate which subcategorizes the infinitive *come*. In the corresponding German example in (15), the sentence features *komme* as main predicate which corresponds in meaning to the English embedded infinitive. The meaning of *like* is expressed by the adverb *gern* that modifies the main predicate.

(14) I like to come.

(15) Ich komme gerne.

In the mapping of LFG to TIGER representations, there are constellations that resemble head switch. Fig. 7 on page 10 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 temporal (or passive) auxiliaries do not have lexical meaning on their own, they coproject with the main verb on f-structure. In TIGER, on the other

hand, the finite verb is analysed as the head of the clause no matter 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.¹³ The mapping has to reorganize the verbal heads and insert additional structure. The restructuring is guided by the c-structural information.

Argument Switch The mapping of (14) and (15) above, not only exemplifies a head switch constellation but also an argument switch. The subject of *like* in English becomes the subject of *komme* in German.

Sometimes, arguments have to be rearranged in grammar mapping, as well. In fig. 7, the German LFG treats all arguments the same: they belong to the main predicate on f-structure. TIGER, in contrast, distinguishes between subjects and all other arguments. Only the subject is always co-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. Argument switch is encoded in the repair section of the transfer grammar. The encoding of argument switch is not entirely trivial since the transfer system does not allow for variables over predicates.

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 together in one string. This mapping could have been integrated in the transfer grammar, as well. But it would have slowed down the transfer procedure considerably.¹⁴

NEGRA Export Format The output of the transfer proper is a Prolog file. A canonical postprocessing step (implemented in Prolog) converts the Prolog file (fig. 11) into the NEGRA export format (used for TIGER) (fig. 12).

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.

¹³The main verb is the head of the 'clausal object', OC.

¹⁴Since the system does not allow for variables ranging over predicates as mentioned further above.

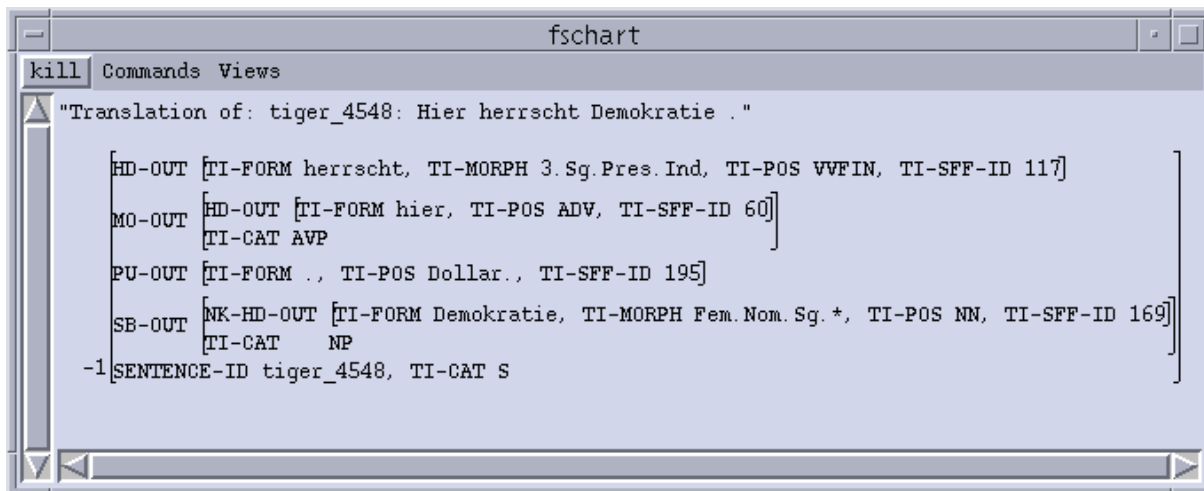


Figure 11: Transfer output of tiger_4548

Word	Category	Morphology	POS	TI-SFF-ID
#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 12: NEGRA export format of tiger_4548

Hyphenated compounds such as *CDU-Frauen* ('CDU women') are split into two tokens by the LFG tokenizer, and analysed 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 even can 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 Finally, as already mentioned in 3.1, certain syntactic projections are not represented in the TIGER format. This points to another peculiar 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 fig. 13, displayed by the annotation tool `annotate` (Plaehn and Brants (2000)). 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. Why this? First, the missing information is redundant, i.e. all missing nodes can be inserted automatically (such an insertion script has been written in the TIGER project). Second, the annotation task becomes easier since the structure is less complex and less nodes have to be checked and corrected by the human annotator. Third, 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.

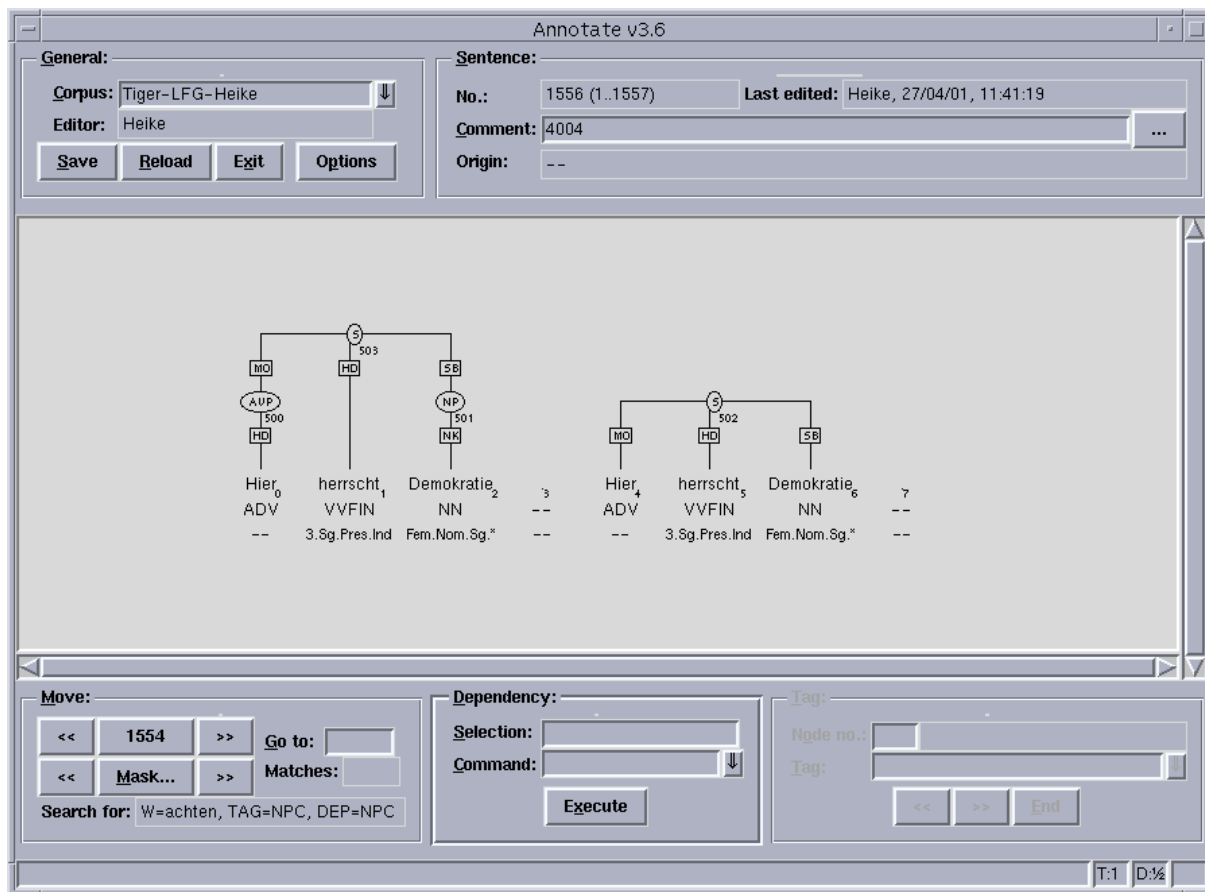


Figure 13: TIGER graph with and without unary branching nodes

4 Results

The German LFG currently parses 50.1% of the sentences in the TIGER corpus, roughly 70% thereof are assigned a correct parse. The sentences that are parsed contain 16.0 words on average, parsing time is 8.29 sec/sentence on average.

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)).

What do we gain from automatic corpus annotation by a large scale grammar? The main point is that a grammar annotates in a much more consistent way than any human annotator. Even the mistakes a grammar produces are much more systematic than the ones done by a human annotator. Accordingly, it is much easier to repair machine-caused errors.

5 Future Work

For the second project phase of TIGER we plan to make use of the LFG-based annotation in three new ways: (i) partial analyses (“fragments”); (ii) morphology and lemma annotation; (iii) consistency checks. All three application tasks will benefit from a planned coverage extension which we will discuss at the very end of this paper.

Partial analyses For the annotation of new sentences a new feature of the XLE system can now be exploited, the so-called “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. NP[std] and PP[std] were chosen as fragments. The evaluation was done based on 100 TIGER sentences that did not get any parse by the first parsing step. The fragment mechanism found 72.9% of the manually annotated NPs and PPs (ignoring PP attachment for the evaluation), the precision was 95.1%. During the second project phase we want to include other constituents as fragments. For each new fragment candidate, its effect and interaction with other fragments has to be evaluated carefully: larger parts of a sentence must be covered without losing good precision.

The partial analyses yielded by the fragment mechanism will then 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 also 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 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.¹⁵

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*. 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 it is our plan to exploit the annotated corpus for extending the coverage of the grammar – this will of course improve the results of the mentioned tasks performed with the grammar. With these applications in mind, it is justified to aim particularly at systematic extension of coverage with respect to the given corpus.

¹⁵Based on the already annotated corpus it is possible to enlarge the coverage of the grammar, s. below. Hence the LFG grammar will finally yield analyses for large parts of the 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 behaviour before and after a modification, only the annotation of the correct analysis will guarantee the desired grammar behaviour. (In very short, unambiguous sentences this is less of a problem, but when realistic sentences change from 80 to 60 readings, inspection of the solutions would be required to make sure that the 20 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 can be reparsed successfully, but hopefully also some significant proportion of previously uncovered sentences.

Ultimately, we plan to 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 can 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, Thorsten. 1999. *Tagging and Parsing with Cascaded Markov Models – Automation of Corpus Annotation*, volume 6 of *Saarbrücken Dissertations in Computational Linguistics and Language Technology*. Saarbrücken, Germany.
- Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Oxford: Blackwell.
- Dipper, Stefanie. 2000. Grammar-based corpus annotation. In *Proceedings of LINC-2000*, pp. 56–64, Luxembourg.
- Emele, M., M. Dorna, A. Lüdeling, H. Zinsmeister, and C. 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*. Stanford: CSLI Publications. To appear.
- 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.
- Kaplan, Ronald M., and Joan W. Bresnan. 1982. Lexical-functional grammar: a formal system for grammatical representation. In Joan W. Bresnan (ed.), *The Mental Representation of Grammatical Relations*, chapter 4, pp. 173–281. Cambridge, MA: MIT Press.
- King, Tracy Holloway, Stefanie Dipper, Annette Frank, Jonas Kuhn, and John Maxwell. 2000. Ambiguity management in grammar writing. In Erhard Hinrichs, Detmar Meurers, and Shuly Wintner (eds.), *Proceedings of the Workshop on Linguistic Theory and Grammar Implementation, ESSLLI-2000, Birmingham, UK*.
- 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.
- Maxwell, John, and R. 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.
- Skut, Wojciech, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of ANLP-97*.