AUTOMATIC F–STRUCTURE ANNOTATION OF TREEBANK TREES

Anette Frank

Xerox Research Centre Europe
6, chemin de Maupertuis
F-38000 Grenoble, France
Anette.Frank@xrce.xerox.com
tel: +33 (0)4.76.61.50.37

Proceedings of the LFG00 Conference

University of California, Berkeley

Miriam Butt and Tracy Holloway King (Editors)

2000

CSLI Publications

http://csli-publications.stanford.edu/

Automatic F–structure Annotation of Treebank Trees [1]

Abstract

We describe a method that automatically induces LFG f–structures from treebank tree representations, given a set of f–structure annotation principles that define partial, modular c– to f–structure correspondences in a linguistically informed, principle-based way.

This work extends the approach of van Genabith, Sadler and Way (1999a,b,c) where f–structure annotation of treebanks is driven by manual annotation of treebank-extracted PS rules. In this paper we present a method for automatic f–structure annotation of treebank trees, building on a correspondence-based view of the LFG architecture. A rule-based approach is explored, in parallel, by Sadler, van Genabith and Way (2000).

In our method we build on a correspondence-based view of the LFG architecture, where annotation principles define ϕ–correspondences directly in terms of ϕ–projection constraints, relating partial (possibly non-local) c–structure tree fragments to their corresponding partial f–structures. Application of the modular annotation principles to treebank trees directly induces the f–structure. Due to the disambiguated tree input, the resulting f–structures require only minimal manual disambiguation, and can be used to build large f–structure corpora as training data for stochastic NLP applications.

The f–structure annotation principles provide by themselves a principle-based, modular description of the LFG c–structure/f–structure interface. They define characteristic functional correspondences between partial c–structure configurations and their f–structure projections. By abstracting from away from irrelevant c–structure context, these principles are highly general and modular, and therefore apply to previously unseen tree configurations.

To define and process the annotation principles we make use of an existing term rewriting system, originally designed for transfer-based Machine Translation. The method is inherently robust. It yields partial, unconnected f–structures in the case of missing annotation rules.

We present the results of a first experiment where we apply this method to the Susanne treebank. The experiment is designed to measure to which extent the partial c– to f–structure correspondences encoded in annotation principles scale up and generalize, by applying them to previously unseen tree configurations. We then extend the model to *selective* filtering of ambiguities, using lexical subcategorization information in conjunction with an OT-based constraint ranking mechanism for ambiguity filtering and ranking (cf. Frank et al. 1998, 2000).

Finally we address some conceptual issues. The principle-based projection of f–structures from disambiguated tree input has interesting implications for the definition of grammatical constraints as compared to the classical LFG parsing architecture. We also discuss issues such as systematic modifications of given treebank encodings, and which types of treebank encodings should be expoited for different applications: the construction of f–structure banks, as opposed to more far-reaching goals, including rapid, corpus-based LFG grammar development, and robust parsing architectures.

1 Introduction

Methods and insights of corpus-based linguistics are now applied to practically all areas of natural language processing, ranging from morphological and syntactic analysis over terminology extraction, semantic disambiguation and machine translation to areas of categorization or summarization.

This paper addresses two issues in corpus-based linguistics. First, we describe and extend a method that allows us to build large LFG f–structure resources as training material for stochastic NLP applications, including but not restricted to LFG processing (see e.g. Bod and Kaplan (1998), Cormons (1999), Johnson et al. (1999), Way (1999), Eisele (2000), Cancedda and Samuelsson

[1] Thanks go to Josef van Genabith, Andy Way and Louisa Sadler, for many discussions on the ideas presented below, and comments on earlier versions of the paper. Fruitful feedback was provided in particular by Ron Kaplan and John Maxwell, as well as Christer Samuelsson, the members of the Pargram group at presentations given at IMS Stuttgart and Xerox PARC, the participants of the Tübingen Workshop "Syntactic Annotation of Electronic Corpora" and the LFG-HPSG2000 conference, as well as my colleagues at XRCE.

(2000), Johnson and Riezler (2000), Bod (2000a, 2000b), Riezler et al. (2000)). More importantly, this method is itself essentially corpus-based in that it exploits existing corpora of disambiguated c–structure representations, i.e., large treebanks, to derive such f–structure resources, and, ultimately, independent LFG grammar resources. The approach is also attractive in that it combines corpus-based methods with traditional rule-based techniques. It allows us to enrich the information extracted from corpora with higher-level syntactic information, which is captured in generalized descriptions, and applied automatically. The external linguistic knowledge encoded in annotation principles imports linguistic generalizations that cannot (easily) be extracted from treebanks, and represents a substantial gain in information, compared to a purely corpus-driven approach.

Creation of LFG-parsed corpora

The construction of LFG-parsed corpora traditionally proceeds by parsing sentences of a text corpus with an existing or adapted LFG grammar, and manually selecting the correct analysis from a set of alternatives proposed by the system. Depending on the size of the grammar and the availability of reliable ambiguity filtering mechanisms, this manual disambiguation task can be significant and cost-intensive: A linguistic expert is needed to choose from a considerable set of alternatives.[2] Moreover, existing unification-based grammars are usually restricted to a specific type of text or domain, and require considerable extension in order to process free text from a variety of domains. This factor further increases the cost of constructing LFG-parsed corpora on a large scale.

To date, there exist rather small LFG f–structure banks.[3] To extend the scope of such f–structure banks to broad-coverage corpora comparable to e.g. the Penn Treebank or the NEGRA corpus, the LFG analysis grammars need to be considerably scaled up.[4] This by itself constitutes a major effort not yet achieved. In addition, with increasing coverage, ambiguities proliferate, leading to increased overhead for disambiguation. Proposals have been made for filtering and ranking parsing ambiguities either by grammar-based preference marks (Frank et al. 1998, 2000) or by stochastic disambiguation methods (Eisele (2000), Bod (2000a, 2000b), Johnson and Riezler (2000), Riezler et al. (2000)) to reduce the search space for manual disambiguation – the latter relying, however, themselves on larger LFG-based training corpora or analysis grammars than currently available to yield reliable results in practice.

Semi–automatic generation of f–structures from treebanks

In a series of papers van Genabith et al. (1999a, 1999b, 1999c) introduced a new method for semi-automatic corpus-based construction of LFG f–structure banks, to address the need of broad-coverage training resources for statistical LFG processing. This method exploits existing treebanks by extracting the context-free PS grammar implicitly encoded by the individual tree assignments, following the method of Charniak (1996). The rules of the extracted "treebank grammar" are manually annotated with functional descriptions. Together with a set of macros for lexical entries, these rules are then used to deterministically "reparse" the original treebank entries by following the tree structure assigned by the annotators. In this reparsing process the f–structure annotations are resolved, and an f–structure is produced. This process is deterministic if the f–structure equations are, and manual inspection of candidate analyses can be significantly reduced.

While this method circumvents the coverage and ambiguity filtering problems of the classical approach, it still involves an important labour intensive component, namely the manual annotation of the grammar rules. This is particularly worrisome due to the fact that treebank grammars

[2] See King et al. (2000) for ambiguity managing techniques in the LFG grammar development platform XLE.

[3] Two corpora have been built at Xerox PARC and XRCE Grenoble, using the English and French grammars developed in the Pargram project: the "HomeCentre" corpus (approx. 1000 sentences for both English and French) and the VerbMobil corpus (540 sentences for English).

[4] See Dipper (2000) for grammar-based corpus annotation using an LFG grammar for German, which provides analyses for a restricted set of sentences within the corpus annotation project TIGER.

are very large (growing with the size of the treebank), and typically consist of flat PS rules with a significant amount of redundancy in their right-hand sides. Manual annotation of rules with functional descriptions is therefore very labour-intensive, can give rise to inconsistencies, and risks missing generalizations.

Automatic f–structure annotation of treebanks and CF grammars

In a collaborative effort the corpus-based approach of van Genabith et al. (1999a, 1999b, 1999c) was extended to two related methods for *automatic* f–structure annotation of treebanks (cf. Frank et al. (1999)). The two methods are based on a common underlying idea, but feature interesting differences. We present these alternative methods in two separate contributions, to allow for more in-depth discussion of the respective approaches.[5]

The key idea for automatic f–structure induction from treebanks is the observation that constituent- and higher-level feature structure representations stand in a systematic relationship. This insight is prevalent in theories like LFG and HPSG. In LFG c–structure and f–structure are independent levels of representation which are related in terms of a correspondence function ϕ. This correspondence follows linguistically determined principles which are partly universal and partly language specific (see in particular recent discussions of projection principles in Bresnan (2000) and Dalrymple (2000)). Following this idea, we propose two methods for automatic f–structure annotation of treebanks, driven by general annotation principles that describe systematic patterns, i.e. linguistic generalizations, in terms of partial c–structure/f–structure correspondences.

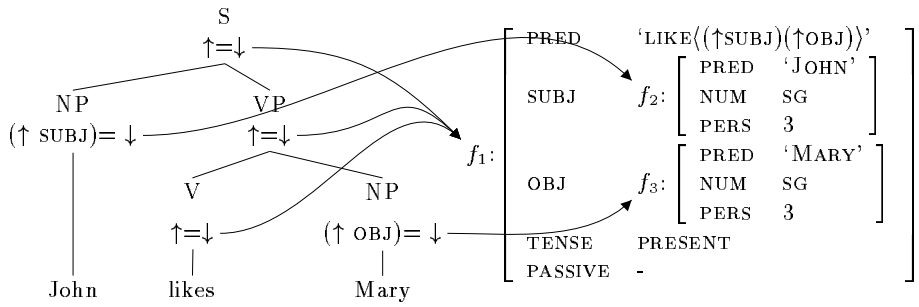
In one approach[5] we read off a CFG treebank grammar, develop annotation templates and compile the templates over the annotation grammar, augmenting it with f–descriptions. A corrected version of this grammar is then used to induce f–structure assignments for treebank trees following the reparsing method of van Genabith et al. (1999c).

In the alternative approach, described below, we operate directly on the PS trees from the treebank. Annotation rules define ϕ–correspondences between partial c– and f–structures. These rules are applied to the treebank tree structures, and annotate them directly with f–structures.

Both methods are partial: the first requires manual inspection and correction of the output produced by the automatic annotation process. The method described below is fully automatic and robust, and yields partial, unconnected f–structures in the case of missing annotation rules.

2 Principles for f–structure annotation of (treebank) trees

In the classical LFG architecture the ϕ–correspondence between c– and f–structure is defined in terms of functional annotations (f–descriptions) on the RHS categories in CFG rules. The f–structure is constructed in the parsing process by resolving the f–descriptions attached to the PS rules, with the meta variables $\uparrow$ and $\downarrow$ instantiated to f–structure nodes.



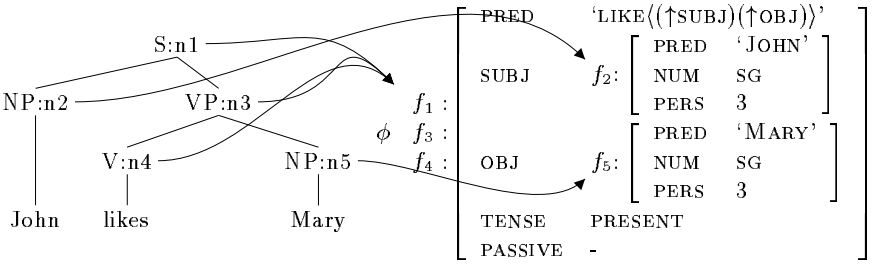
PS rules define f–structure via functional descriptions

$$S \rightarrow \begin{matrix} NP \\ (\uparrow \text{SUBJ}) = \downarrow \end{matrix} \quad \begin{matrix} VP \\ \uparrow = \downarrow \end{matrix} \qquad VP \rightarrow \begin{matrix} V \\ \uparrow = \downarrow \end{matrix} \quad \begin{matrix} NP \\ (\uparrow \text{OBJ}) = \downarrow \end{matrix}$$

[5] See the companion paper Sadler et al. (2000) in these proceedings.

Our first annotation method follows this classical LFG architecture, in that f–structure annotation principles apply to PS rules extracted from treebanks, and enrich them with f-descriptions. The resulting annotated rules are then used to "reparse" the assigned tree structure of treebank entries, thereby inducing the f–structure.

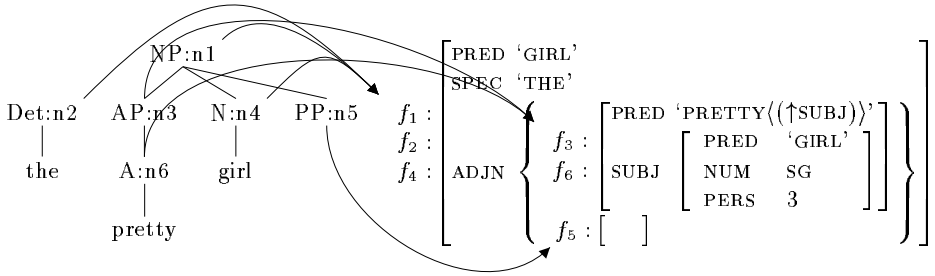
On a pure correspondence view of the LFG architecture we describe the correspondence between a given c–structure and its corresponding f–structure in terms of the projection function ϕ itself, as displayed below.[6]



ϕ–correspondence:		f–structure:	
$\phi(\text{n1}) = f_1$	$\phi(\text{n2}) = f_2$	$(f_1$ SUBJ$)= f_2$,	$(f_2$ PRED$)=$ 'John'
$\phi(\text{n3}) = f_3$	$\phi(\text{n5}) = f_5$	$(f_1$ OBJ$)= f_5$,	$(f_5$ PRED$)=$ 'Mary'
$\phi(\text{n4}) = f_4$		$(f_4$ PRED$=$ 'likes'	
$\phi(\text{n1}) = \phi(\text{n3}) = \phi(\text{n4})$		...	

The approach presented in this paper builds directly on this correspondence view of LFG, which leads us towards f–structure induction from c–structure *trees*, as opposed to PS rules. That is, annotation principles define ϕ–correspondences directly in terms of *ϕ–projection constraints*, relating partial c–structure configurations to their corresponding partial f–structures. Automatic application of these annotation principles to tree fragments directly induces the f–structure. This approach has two advantages. First, it can apply to non-local trees, while PS rules are restricted to trees of depth one. Second, by projecting f–structures from trees we skip the reparsing process for f–structure composition. While this purely correspondence-based f–structure annotation is decoupled from the parsing process, it can still be extended to a parsing architecture where a (probabilistic) PCFG grammar assigns a set (or parse forest) of best-ranked trees, which are then input to automatic f–structure annotation.[7]

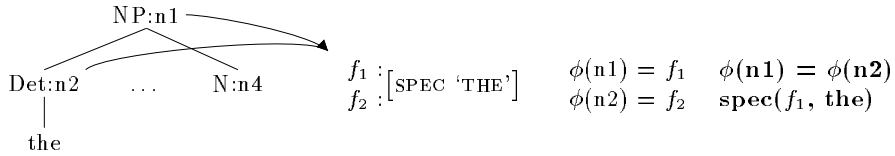
Partial correspondences, partial and non-local trees Before going into details, let us first illustrate the key ideas of automatic f–structure annotation based on modular, partial annotation principles. Below we display the complex c–structure/f–structure correspondence of an NP. This complex picture can be broken down into modular, piece-wise correspondences of *partial* c– and f–structures, which abstract away from irrelevant material in their surrounding context.



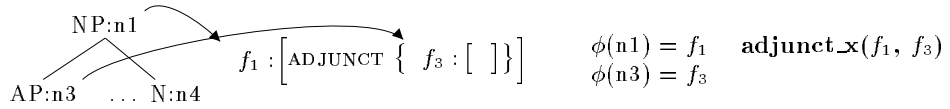
For instance, the functional contribution of the prenominal determiner *the* is independent from the presence of the AP or PP, and is captured by the following partial correspondence:

[6] See Dalrymple (2000) for a correspondence-based exposition of LFG syntax.

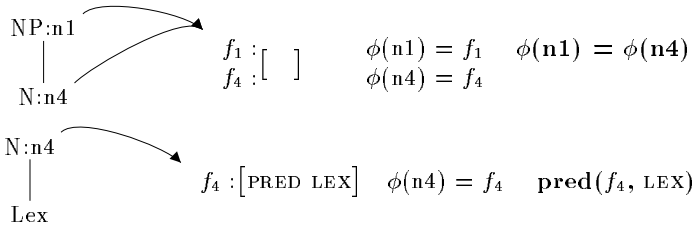
[7] See Section 6 for discussion.



Any AP daughter of NP is invariably analyzed as an ADJUNCT of the nominal head, unless the noun head N is omitted. The former generalization is captured in the following partial correspondence:

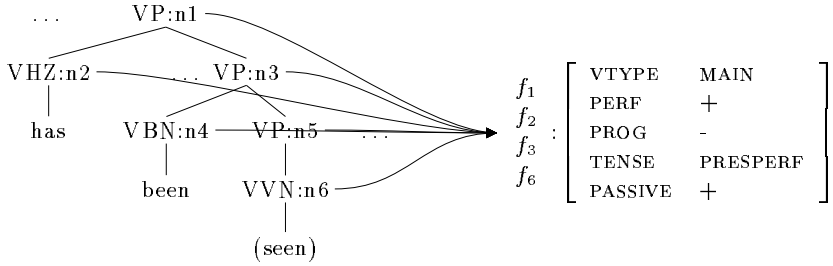


Projection principles for head categories and lexical nodes (here for nominal categories) are straightforward.[8]



Similar correspondences can be defined for the remaining c–structure fragments. By applying them all to the c–structure above, they define the corresponding complex ϕ–projection and f–structure in a modular, declarative way. Most importantly, due to the abstraction over immaterial c–structure context, these principles generalize over specific tree configurations, i.e. they can apply to fragments of unseen tree configurations.

Since the annotation principles apply to trees, they can be defined to involve partial, *non-local* tree fragments, while PS rules are restricted to trees of depth one. This allows us to define and constrain f–structure assignments in terms of complex non-local c–structure constraints. An example where this is fruitfully applied is the assignment of complex tense information in binary branching VPs.[9] By specifying characteristic partial, non-local c–structure contexts in a binary branching VP structure, we can capture the tense and active/passive distinctions of verbal constructions in a natural way. This is illustrated below for the characteristic complex construction indicative of present perfect tense.



In our approach, modular annotation principles establish correspondences between (possibly non-local) partial c–structure fragments and their corresponding partial f–structures. This is very much in the spirit of projection principles as proposed by Dalrymple (2000) and Bresnan (2000), and provides *a principle-based, modular c– to f–structure interface in the LFG architecture*.[10] The application of annotation principles to c–structure trees follows the traditional description-by-analysis (DBA) approach of Halvorsen and Kaplan (1995) in the c–structure/f–structure interface. Yet, while in the classical DBA approach *complete PS rules* are matched against the c–structure, in our approach *partial (non-local) c–structure fragments* are matched against the c–structure trees.

[8] See Sections 3.5 and 5 for the assignment of subcategorized grammatical functions.

[9] See also Section 3.5.

[10] It is also closely related to principle-based grammar description in HPSG.

3 Automatic f–structure annotation of trees

3.1 The XLE term rewriting component

To define and process f–structure annotation principles, we make use of an existing term rewriting system, originally designed as a rewriting component for transfer-based Machine Translation in the XLE (Xerox Linguistic Environment) system (see Kay (1999), Frank (1999)).

The system takes as input an unordered set of n-ary terms `p,q`, etc. and an ordered set of rewrite rules `p ==> q`.[11] If the LHS terms `p` of a rule `p ==> q` match the input, the matching terms `p` are eliminated from the input set, and the terms `q` are added to the input set. A rule applies to *each instantiation* of the LHS terms in the input. The LHS of a rule may contain positive `+p` and negative `-p` terms. A rule with positive constraint `+p` only applies if `p` matches some term in the input. Positive terms are not eliminated from the input set. A rule with negative constraint `-p` only applies if `p` does not match any term in the input set.

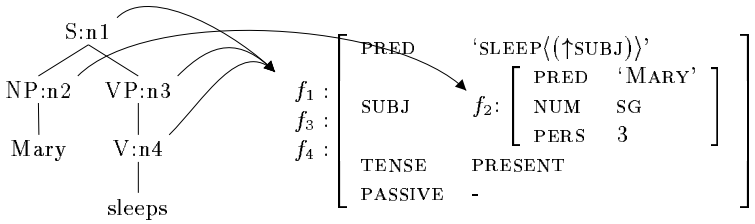
The order in which the rules are stated is crucial: each rule applies to the *current* input set, and yields an output set. The output set of a rule constitutes the input set for the next rule. Annotation principles will thus be interpreted as a cascade of rewrite rules that continuously transform, or enrich the input set of terms, to yield a final set of output terms.

3.2 A term representation of the LFG projection architecture

For the specific task of f–structure annotation of c–structure trees, using the XLE term rewriting component, we encode the LFG projection architecture in a term representation language as follows:[12]

imm. dominance	`arc(MNode, MLabel, DNode, DLabel)`
imm. precedence	`prec(CsNode_x, CsNode_y)`
lexical insertion	`lex(TerminalNode, Lex)`
ϕ-correspondence	`phi(CsNode, FsNode)`
	`equal(FsNode_x, FsNode_y)`
f–structure features[13]	`attr(FsNode_x, FsNode_y), attr(FsNode, Val)`

With this, the traditional representation



translates to the following set of terms:

```
arc(n1,s,n2,np), arc(n1,s,n3,vp), arc(n3,vp,n4,v),
prec(n2,n3),
lex(n2,mary), lex(n4,sleeps),
phi(n1,f1), phi(n2,f2), phi(n3,f3), phi(n4,f4), equal(f1,f3), equal(f3,f4),
pred(f1,sleep), subj_arg(f1), subj(f1,f2), pred(f2,mary), num(f2,sg), tense(f1,pres),..
```

3.3 Automatic annotation of trees with f–structures

The basic steps of f–structure annotation of treebank trees are illustrated below.

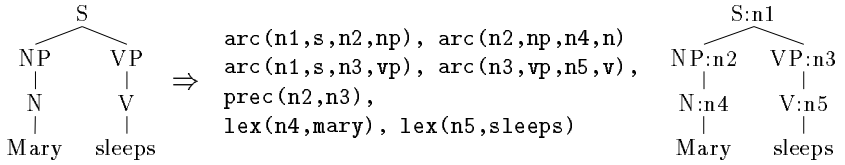
[11] There are obligatory (`==>`) and optional (`?=>`) rules. An optional rule creates two output sets, one where the rule applies, and one where it doesn't apply. All subsequent rules apply to all output sets of previous rule applications. The system operates on packed ("contexted") representations for efficient processing of ambiguities (see Kay (1999)).

[12] This term representation language can, in conjunction with the XLE term rewriting compenent, also be used for structure-based corpus queries, and is comparable in scope with the query tool presented in Kallmeyer (2000).

[13] with `attr` a variable over f–structure attributes

Preprocessing: converting bracket– to c–structure term representation In a preprocessing stage, we first compile the tree encodings of a given treebank to a canonical bracketing structure for PS trees. This bracket structure we convert to an equivalent term representation, where nodes are associated with unique node identifiers $\mathtt{n}_i$, besides their category labels.

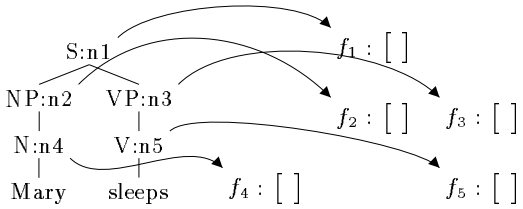
```
[s,[np,[n,Mary]],
   [vp,[v,sleeps]]].
```



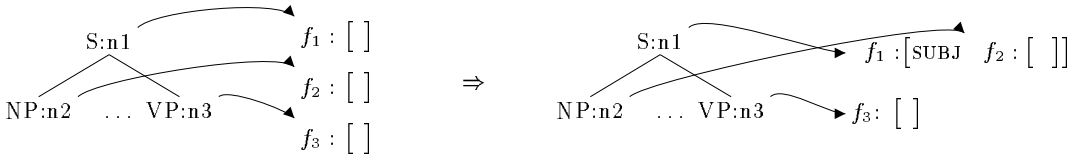
This c–structure term representation constitutes the input to automatic f–structure annotation with the term rewriting component of XLE. The task is to fill in the appropriate `phi` predicates and f–structure terms, i.e. to induce the ϕ–projection for the input c–structure.

Initialization: A trivial 1–1 ϕ–correspondence of c–structure nodes to f–structure nodes initializes, in a first step, a piece-wise partial mapping of c–structure nodes $\mathtt{n}_i$ to fully underspecified, empty f–structure nodes $\mathtt{f}_i$. This is defined by the following rules.[14]

```
+arc(n1,_,_,_) ==> phi(n1,f1).
+arc(_,_,CsNode,_) ==> phi(CsNode,FsNode).
```



F–structure annotation rules associate partial c–structure configurations with their corresponding ϕ–projected f–structure information, and further restrict the trivial 1–1 ϕ–correspondence via the predicate `equal(Fx,Fy)`. Below we give an example. The first rule defines the f–structure node f_2 for the VP-external NP:n2 as the SUBJ of the node f_1 that is projected from the S:n1 node.[15] [16]




```
+arc(A,s,B,np), +phi(A,FA), +phi(B,FB),
+arc(A,s,C,vp), +prec_x(B,C) ==> subj(FA,FB).
```

The second rule applies to the output resulting from the previous rule application. The predicate `equal(Fx,Fy)` restricts the ϕ–function to map the VP and S nodes to identical nodes in f–structure.


```
+arc(A,s,C,vp), +phi(A,FA), +phi(C,FC) ==> equal(FA,FC).
```

[14] $\mathtt{n}_1$ and $\mathtt{f}_1$ are designated constants for the tree and f–structure root nodes, respectively. The RHS term `phi(CsNode,FSNode)` in the second rule introduces new constants $\mathtt{f}_i$ for the non-instantiated variable `FsNode`.

[15] The graphical representations illustrate the effect of the rules, in terms of input and output set.

[16] Note the predicate `prec_x(B,C)`, which is defined (by use of macros) as a finitely constrained transitive closure over the precedence relation `prec`. This allows us to underspecify precedence constraints holding between nodes n_x and n_y to allow for an arbitrary or else a restricted sequence of intervening categories.

A set of annotation rules of this kind (which we call an *annotation grammar*)[17] is applied to any given c–structure term representation from the treebank. These rules continuously restrict and enrich the ϕ–projection. After resolution of f–structure node equalities the process yields, in the general case, a complete c–structure/f–structure projection architecture for the sentence at hand.

3.4 Formal restrictions and simplifications

Formal restrictions Annotation rules are subject to formal constraints in order to guarantee the functional property of the ϕ–correspondence, in particular:

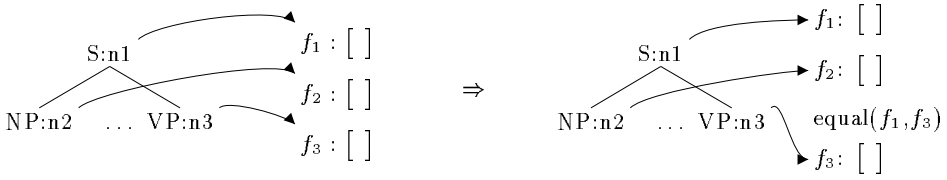
- `phi` predicates (`phi(C_Node,F_Node)`) are restricted to occur as *positive constraints*, i.e.
  - no `phi` predicate may be introduced (in RHS of rules)[18]
  - no `phi` predicate may be deleted (in LHS of rules), however,
  - new f–structure nodes may be introduced (in RHS of rules)

  Given the input specification of a 1–1 ϕ–projection, these constraints guarantee that the functional property of ϕ is preserved.

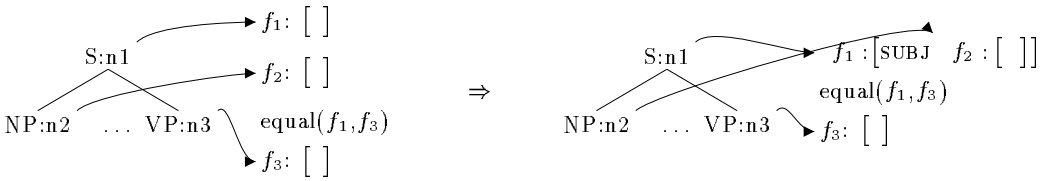
- `equal(F_Node_x,F_Node_y)` predicates *restrict* the ϕ–correspondence, while preserving its functional property. They may be introduced (in RHS), or used as constraints (in LHS of rules). `equal` predicates are resolved after the annotation process is completed.[19]

Order independence in a cascaded rewrite system Note that as long as annotation rules do not consume c–structure terms, or refer to f–structure terms introduced by other rules, the order in which the rules are stated, and thus applied by the system, is irrelevant.

This is illustrated by inverting the application order of the subject and VP head-projection rules from above: Below we first apply the VP head-projection rule. The subject annotation rule is then applied to the output of the first rule. Since we did not consume any c–structure terms, nor put constraints on f–structure terms, the rules yield the same result in any order of application.[20]




```
+arc(A,s,C,vp), +phi(A,FA), +phi(C,FC) ==> equal(FA,FC).
```




```
+arc(A,s,B,np), +phi(A,FA), +phi(B,FB),
+arc(A,s,C,vp), +prec_x(B,C) ==> subj(FA,FB).
```

[17] See Section 3.5 for more detail on the different types of rules, lexical, syntactic and morpho–syntactic, that make up such an annotation grammar.

[19] `equal` predicates could in principle also be resolved immediately during the annotation process.

[19] Except, of course, for the initialization rules above, which induce the 1–1 ϕ–correspondence.

[20] Had we consumed, for example, the term `arc(A,s,C,vp)` in the first rule, the second rule would not have applied. Nor would the first rule, if it had stated a positive constraint on the presence of a SUBJ function, which is introduced later in the annotation process.

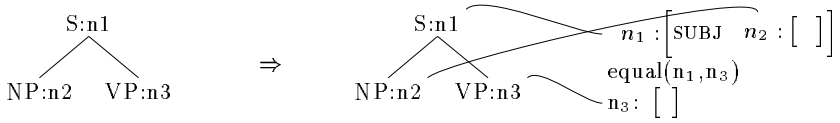
Order independence of f–structure annotation rules can be guaranteed by additional formal restrictions which prevent consumption of c–structure terms and constraints on (previously introduced) f–structure terms. This guarantees that *all* annotation rules have access to (a) the full information structure that constitutes the initial input, i.e. the c–structure plus the 1–1 ϕ-projection as defined via `arc, prec, lex`, and `phi` terms, and (b) no more than the initial information structure, i.e. no annotation rule is constrained in terms of f–structure information introduced by other rules:

- c–structure terms (`arc, prec, lex`) and `phi` terms only occur as positive constraints
- f–structure terms and `equal` terms only occur in RHSs of rules

These formal restrictions ensure that all f–structure annotation rules have access to the complete set of c–structure and `phi` terms, and no more than this, and thereby guarantee order-independence of rule application. The writer of annotation rules does not need to care about the order in which the rules are stated, and thus applied. However, the system is more powerful, and can be flexibly relaxed to allow for limited degrees of order-dependence where it appears to be useful.

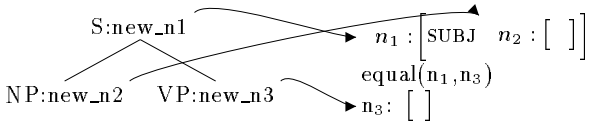
There is a trade-off between order-dependence and order-independence. Constraining rules to c–structure information only can require complex rule constraints in order to avoid application of different annotation rules to the same tree fragment, leading to inconsistencies. Moreover, access to f–structure information can be useful for generalizing annotation rules. If several c–structure configurations are indicative of, e.g. a subject function, or passive voice, it is possible to capture such diverse configurations in a more general way by referring to the more abstract f–structure information to guide further aspects of f–structure construction. In this case, the order of annotation rules needs to be respected, to make sure that the required f–structure information is being introduced by previously applied annotation rules. The term rewriting system is powerful enough to allow for flexible definition of annotation rules, providing much room for flexibility and experimentation in f–structure annotation.[21]

Simplifications To avoid cumbersome reference to `phi`-predicates in our annotation rules, we can exploit the fact that `equal`-predicates are resolved after the annotation process. Since the induced ϕ–projection is 1–1, we can effectively eliminate it during the annotation process, and attach f–descriptions directly to c–structure nodes $\mathtt{n}_i$.




```
+arc(A,s,B,np), +arc(A,s,C,vp), +prec_x(B,C) ==> subj(A,B), equal(A,C).
```

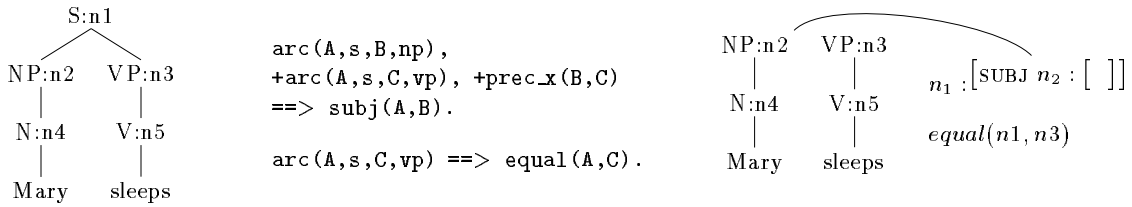
After the annotation process, and before resolution of equalities, we can restore the (implicit) 1–1 ϕ–correspondence between c–structure and f–structure nodes, which is of course essential to represent the mapping of distinct c–structure nodes to single (i.e. equated) f–structure nodes. This is performed by a set of rules which relabel the c–structure with new node identifiers, while encoding the 1–1 ϕ–correspondence between these new node identifiers and the original c–structure nodes, which are now playing the role of f–structure nodes.[22]



[21] See also Section 3.7. where order-dependence of rewrite rules is fruitfully applied for general c–structure transformations on existing treebank encodings, to facilitate principle-driven f–structure annotation.

[22] The rules rewrite the c–structure predicates `arc`, `lex` and `prec`, by replacing the node identifiers $\mathtt{n}_i$ with new constants $\mathtt{new\_n}_j$. Concurrently, a (new) phi-projection `phi` is defined between the new node identifiers $\mathtt{new\_n}_j$ and the corresponding nodes $\mathtt{n}_i$ that populate the f–structure space.

Alternative annotation schemes As an alternative to restoring the c–structure and ϕ–projection in this simplified annotation scheme, we can also choose to rewrite trees *into* f–structures, generating only *f–structures* instead of a full projection architecture. In this variant, we can *consume* c–structure predicates, stepwise, during the annotation process. This is illustrated below for the S – NP VP structure.



Consumption of c–structure induces a certain amount of order-dependence in the annotation process. Once some c–structure fragment is consumed, later rules cannot refer to this c–structure bit to constrain the application of another rule. Note, however, that the *f–structure* that corresponds to consumed c–structure fragments is available for reference at later stages of processing. An advantage of this annotation variant is that we can easily avoid inconsistent f–structure assignments by multiple rule applications. Annotation rules that would require complex application constraints to avoid such situations can in this variant be less constrained, by exploiting the order-dependence of rule application. An annotation grammar which makes use of c–structure consumption and reference to f–structure terms was developed for our experiment in Section 4. This grammar defines a natural order of part-of-speech related sections of annotation rules.

3.5 On the nature of f–structure annotation rules

Just like in an ordinary LFG grammar, we find, in our f–structure annotation approach, a division between lexical, morpho–syntactic, and phrasal annotation rules. However, in contrast to classical LFG PS rules, our syntactic rules describe *partial* c–structure/f–structure associations, which can extend to *non-local* configurations. In addition, our formalism allows us to define *underspecified* annotation rules, generalizing over classes of c–structure categories – both lexical and phrasal.

Lexical and morpho–syntactic rules Morpho–syntactic rules introduce morphological (and to some extent semantic) information encoded in lexical category labels into the f–structure space. This is illustrated below, for NUMber and NTYPE features. The example illustrates how highly specific category distinctions in treebank encodings can be neutralized: once NUMber is encoded in f–structure, based on the `nn1` vs. `nn2` distinction, this category distinction can be neutralized by mapping both lexical category labels to the generalized label `nn`.[23] This generalization is essential for compact rule definition. For example, below, the instantiation of the PRED–value of nouns is captured in a single lexical rule which applies to all "generalized" `nn`-daughters.

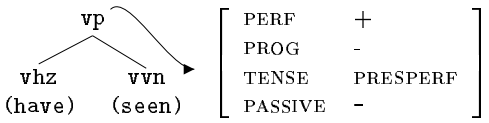
```
arc(A,RL,B,nn1) ==> num(B,sg), ntype(B,common), arc(A,RL,B,nn).
arc(A,RL,B,nn2) ==> num(B,pl), ntype(B,common), arc(A,RL,B,nn).

arc(A,RL,B,nnt1) ==> num(B,sg), ntype(B,temporal), arc(A,RL,B,nn).
arc(A,RL,B,nnt2) ==> num(B,pl), ntype(B,temporal), arc(A,RL,B,nn).

+arc(A,n,B,nn), +lex(B,Lex) ==> equal(A,B), pred(B,Lex), pers(B,'3').
```

Tense information as well as the active/passive distinction can be captured by stating constraints on the partial c–structure context of verbs, as illustrated below for active and passive present perfect tense in a flat VP structure, as it is assigned in the Susanne corpus.[24]

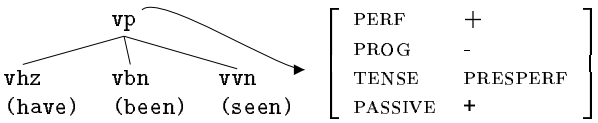
[23] See van Genabith et al. (1999b) for a similar approach.

[24] Since we assign a *flat* f–structure for complex tenses, we do not introduce PRED–values for lexical nodes of auxiliaries in the f–structure. See below for the assignment of appropriate subcategorization frames.

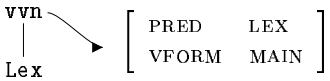
```
+arc(A,vp,B,vhz)   % have-aux
-arc(A,vp,D,vbn)   % no been-aux !
+arc(A,vp,C,vvn)   % main verb part.
==> perf(A,+), prog(A,-), tense(A,presperf),
    passive(A,-).
```




```
+arc(A,vp,B,vhz),  % have-aux
+arc(A,vp,C,vbn),  % been-aux
+arc(A,vp,D,vvn),  % main verb part.
==> perf(A,+), prog(A,+), tense(A,presperf),
    passive(A,+).
```

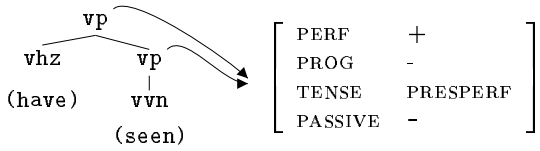



```
+arc(A,vp,D,vvn), +lex(D,Lex)
==> equal(A,D), pred(D,Lex), vform(D,main).
```

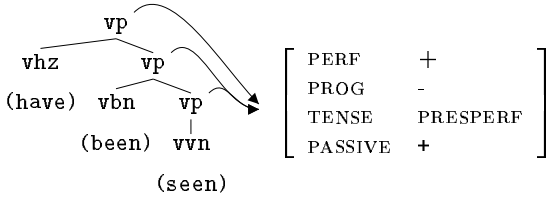


Given a treebank encoding that represents the verbal complex in terms of binary branching VP structures (as e.g. in the Penn Treebank, or the Lancester AP treebank), we can assign complex tense information in similar ways, by applying annotation rules to *non-local* tree fragments.

```
+arc(A,vp,B,vhz),                  % have-aux
+arc(A,vp,C,vp), +arc(C,vp,D,vvn)  % main verb part.
==> equal(A,C), perf(A,+), prog(A,-),
    tense(A,presperf), passive(A,-).
```




```
+arc(A,vp,B,vhz)                   % have-aux
+arc(A,vp,C,vp), +arc(C,vp,D,vbn),  % been-aux
+arc(C,vp,E,vp), +arc(E,vp,F,vvn)   % main vb part.
==> equal(A,C), equal(C,E), perf(A,+), prog(A,+),
    tense(A,presperf), passive(A,+).
```




```
+arc(A,vp,D,vvn), +lex(D,Lex)
==> equal(A,D), pred(D,Lex), vform(D,main).
```

Partial phrasal rules and underspecification As illustrated in Section 3.3, our annotation rules are designed to apply to linguistically motivated, i.e. modular, partial c–structure configurations, to define their corresponding functional projections. Even though treebanks do not encode classical X′ syntax, it still holds that specific types of tree branches correspond to functional dependencies in f–structure. Therefore, annotation rules apply, in the general case, to single tree branches, with some contextual constraints, and generalize to unseen tree configurations. Below, for example, *that*-clauses (with category label `f`) are associated with a function COMP in f–structure by referring to a single branch (`arc`) in the c–structure, abstracting away from irrelevant differences in the surrounding c–structure context.

The example also illustrates the effect of underspecification. *that*-clauses can appear in different syntactic contexts. By referring to an *underspecified* (variable) mother node label `RL`, we generalize over various possible mother labels (e.g. (in)finite, modal, nominal or adjective phrases).

```
+arc(A,RL,B,f), +comp_form(B,that) ==> comp(A,B).
```

Finer categorial restrictions can be captured by defining classes of category labels in (disjunctive) templates.[25] Below, the template `np_cat(X)` defines a class of category labels (`n, d, m`). The disjunctive template is called (by union) in the annotation rule for PPs (label `p`) to define this restricted class of alternative NP-types as complements (i.e., OBJ) of prepositions in a single rule.

[25] Disjunctive templates encode alternative rules, and can be unioned (`&&`) with annotation rules. While this does still involve disjunctive processing, the rules can be stated in a generalized, compact way.

```
template definition:   np_cat(X) ::  { X == n } ==> 0;   % n:  nominal phrase
                                     { X == d } ==> 0;   % d:  determiner phrase
                                     { X == m } ==> 0.   % m:  number phrase

annotation rule:       +arc(A,p,B,NP) ==> obj(A,B)
                       && np_cat(NP).
```

Grammatical function assignment In languages like English, grammatical function assignment relies heavily on c–structure configurations, while still not being fully deterministic. In case marking languages, morphological marking will be used to constrain grammatical function assignment. Below an example for the assignment of OBJ vs. OBJ2 functions for transitive and ditransitive verbs in English, which is determined by surface order.

```
+arc(A,vp,C,np), +arc(A,vp,D,np), +prec_x(C,D) ==> obj2(A,D).   % secondary OBJ of ditransitives

+arc(A,vp,C,np), +arc(A,vp,D,np), +prec_x(C,D) ==> obj(A,C).    % OBJ of ditransitives

-arc(A,vp,C,np), +arc(A,vp,D,np), {D \== C} ==> obj(A,D).       % OBJ of transitives
```

In various syntactic configurations we also need rules for non-local function distribution. Below, for example, a simple rule defines subject distribution in (the f–structure constructed from) VP-coordination.[26] Note that this rule refers uniquely to f–structure terms, since the generalization is easier to capture at the level of f–structure, rather than taking into account possible variations at the c–structure level.

```
+conj_form(A,_), +element(B,A), +subj(B,D),
                 +element(C,A), -subj(C,_) ==> subj(C,D).
```

Non-local dependencies, such as Wh-constructions can be captured by means of restricted path-equations `path(A,C)`. Below, we display an example from interrogative clauses, where the TOPIC clause, instantiated by the interrogative adverbial Wh-element (label `rrq`), is assigned the ADJUNCT function via restricted functional uncertainty embedding over the functions COMP and XCOMP (`comp_xcomp_path(A,C)`).[27]

```
+arc(A,f,B,r), +arc(B,r,C,rrq) ==> topic(A,B), comp_xcomp_path(A,C), adjunct_x(C,B).
```

Subcategorization assignment We induce subcategorization frames (so-called semantic forms) by collecting grammatical functions assigned by annotation rules into the predicate's semantic form, following van Genabith et al. (1999a)'s method.

Below we state the three rules that deal with OBJ assignment: an object `obj(A,B)` that is not yet assigned an argument position in the PRED's subcat list (`-arg(A,_,B)`) will be assigned the first argument position (`arg(A,1,B)`) in case of a coocurring non-thematic SUBJ (first clause), or if the head is a preposition (last clause); otherwise, it will be assigned the second argument position.

```
+obj(A,B), -arg(A,_,B), +subj(A,C), +nonarg(A,_,C) ==> arg(A,1,B).
+obj(A,B), -arg(A,_,B), +subj(A,C), -subj(A,2,C) ==> arg(A,2,B).
+obj(A,B), -arg(A,_,B), -subj(A,C) ==> arg(A,1,B).
```

[26] The rule is adjusted to the specificities of the Susanne corpus, where VP coordination is encoded as S-coordination, leaving the second conjunct without subject NP in the c–structure.

[27] In subcategorization filtering (see below Section 5.1) restricted functional uncertainty path equations are expanded by means of disjunctive templates, and the subcategorization constraints (with subscript _sc: e.g. `comp_sc(A,B)`) are checked against the f–structure.

```
path_exp(A,B) :: comp_xcomp_path(A,B) ?=> equal(A,B);
                 comp_xcomp_path(A,B) ?=> 0 && comp_or_xcomp(A,B);
                 comp_xcomp_path(A,B) ?=> 0 && comp_or_xcomp(A,C) && comp_or_xcomp(C,B);
                 comp_xcomp_path(A,B) ==> 0 && comp_or_xcomp(A,C) && comp_or_xcomp(C,D) &&
                                                                          comp_or_xcomp(D,B).

comp_or_xcomp(A,B) ::   0 ?=> comp_sc(A,B);
                        0 ==> xcomp_sc(A,B).
```

Obviously, pure c–structure information does not allow us to distinguish between NP/PP arguments vs. adjuncts, or infinitival complements vs. adjuncts. Similarly, lacking lexical information, raising and control constructions can only be represented as involving anaphoric control. In Section 5 we show how to integrate lexical subcategorization information, combined with strategies for ambiguity filtering (cf. Frank et al. (1998, 2000)).

3.6 Partial annotation and robustness

Our method for f–structure induction from trees embodies an important aspect of robustness. In cases of missing or incomplete annotation rules, the system does not fail, but partial trees are left without f–structure annotation. We obtain (typically large) partial, unconnected f–structures. See in particular the results and discussion of our Experiment in Section 4.

3.7 Moving treebanks

Finally, our framework can also be used to adjust particular treebank encodings, by *"moving"* treebanks to a different structural encoding, thereby facilitating principle-based f–structure induction. In our treatment of the Susanne corpus, we defined a set of general c–structure rewriting rules, which transform the encoding of coordination and flat modal VP structures into a more standard PS analysis, which lends itself to principle-driven f–structure annotation.[28]

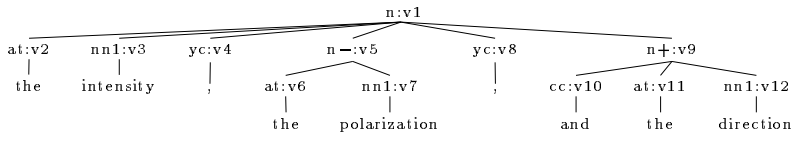
4 Experiment: An annotation grammar for the Susanne corpus

In a first controlled experiment, we applied our annotation method to the Susanne treebank. We developed an annotation grammar for two sections of the corpus, using an annotation scheme that consumes c–structure while building up the f–structure. For this approach, we set up a natural order for the various types of annotation rules presented in Section 3.5. In the following we summarize the basic results, and show, in Section 5, how the integration of lexical subcategorization information and preference ranking solves the obvious shortcoming of this first approach.

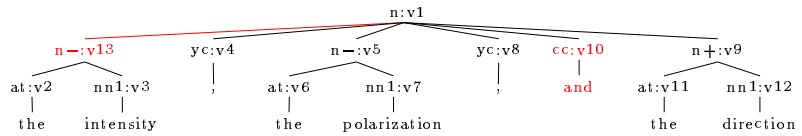
Data The Susanne treebank encodes labelled bracketed structures, where category labels are enriched with functional labels (subject, object). The trees contain traces to indicate certain control and long-distance dependencies. Functional labels and traces were eliminated in a preprocessing stage, to guarantee a non-biased evaluation scheme which takes conventional PS trees as input. In preprocessing we also collapse overspecific category labels, similar to van Genabith et al. (1999b). The treebank encoding was converted to a term representation as input for f–structure annotation.

Quite a few decisions on PS assignment in the Susanne corpus are debatable, and some make it difficult to assign f–structures in a principle-driven way. We therefore defined a set of initial (c–structure) rewriting rules, which transform the encoding of coordination and flat modal VP structures into a standard PS analysis. Finally, attachments in the NP are often flat, so that, for instance, complements to adjectives are not always correctly attached in the PS tree.

[28] In the Susanne corpus, coordination is encoded in a rather non-standard way:



A set of c–structure rewrite rules transforms these into classical coordination structures by moving and creating new nodes:



Experiment We chose two sections of the Susanne corpus, J01 and J02 (text type: learned writing). We ran an experiment in 3 steps. First, based on the 66 sentences of J01, we develop f–structure annotation rules to cover 50 sentences. In step 2 we apply the resulting annotation grammar AG1 to the unseen first 50 sentences of J02 (J02-1), and evaluate the results of f–structure annotation. Grammar AG1 is upgraded to AG2, which then covers these 50 sentences of J02-1. We record the number of rules that were added or modified. In step 3, AG2 is applied to the remaining unseen 46 sentences of J02 (J02-2). Again, we measure the results of f–structure annotation.

Evaluation and Results Table T1 provides basic data on the relevant subsections: the number of sentences processed and average sentence length (tokens, including punctuation); the number of phrasal and lexical categories and the number of distinct (i.e. types of) PS rules and PS branches encoded by the corpus trees (after categorial filtering in preprocessing).

Note that the percentage of new (unseen) tree branches in J02-1 and J02-2 is considerably lower than for new (unseen) PS rules.[29] This is not surprising, and supports our annotation scheme, where annotation involves underspecified, partial trees – often single branches.

T1	sentences	length	phrasal cat	lexical cat	PS rules	tree branches
J01	66	34.27 (max. 94)	32	73	430	281
J02-1	50 (1-50)	21.68 (max. 47)	25 (3 new)	64 (8 new)	249 (150=60.34% new)	172 (36=20.93% new)
J02-2	46 (51-96)	24.8 (max. 45)	24 (4 new)	57 (3 new)	212 (96=45.28% new)	163 (26=15.95% new)
	166		39	84	676	434

The results are summarized in table T2. Note that in this experiment correctness of f–structure assignment is measured modulo the PP argument/adjunct distinction, the missing assignment of control/raising equations, and systematic ambiguities as to the distinction between ADJUNCT vs. XCOMP readings of infinitives. Also, attachment or labelling mistakes in the treebank are not counted as annotation mistakes if the resulting f–structure is predicted from the given tree.

AG1 consists of 173 tag simplification and lexical rules, and 118 non-lexical f–structure assignment rules. AG1 covers 48% of the unseen sentences of J02-1. As expected from the above data, the upgrade from AG1 to AG2 required little effort: it involves 28 new and 5 modified rules, and required approx. 1 person day of work. AG2 applied to the unseen section J02-2 covers 76.09%.

T2	correct fs		partial fs		tag rules	lexical rules	non-lex rules	all rules
J01 with AG1	50	75.76%	16	24.24%	41	132	118	291
J02-1 with AG1	24	48%	26	52%	41	132	118	291
J02-1 with AG2	49	98%	1	2%	41+4	132+4 (2 mod)	118+20 (3 mod)	291+28 = 319
J02-2 with AG2	35	76.09%	11	23.91%	45	136	138	319

Discussion Our first experiment confirms the intuition that *partial c–structure/f–structure correspondences* generalize much stronger than full PS rules-to-f–structure correspondences, due to the fact that annotation principles typically select tree branches (plus some contextual constraints), which correspond to functional dependencies in f–structure. This is brought out by the ratio of new, unseen tree branches vs. PS rules in the corpus, and the corresponding ratio of added/modified annotation rules (table T3). The percentage of added/modified annotation rules corresponds almost exactly to the percentage of unseen PS branches. Our figures are of course not yet sufficient to judge whether the number of new annotation rules will decrease proportional to the number of new PS branches. But, since our rules make heavy use of category underspecification (see Section 3.5), we anticipate that the number of new annotation rules will show even stronger convergence.

[29] And this already as evaluated on types, not on tokens. With growing coverage, novelty percentages at the level of tokens should be lower.

T3	new PS rules	new PS branches	new phrasal annotation rules
J01 to J02-1	60.34%	20.93%	19.49%
J02-1 to J02-2	45.28%	15.95%	—

These results are promising, and particularly important for corpus-based approaches that rely on grammars (or trees) extracted from treebanks. Such grammars typically feature large, flat PS rules with repeated material in their right-hand sides. That is, they lack the information about optionality of daughter constituents, which in classical grammar writing is encoded by linguists. This linguistic knowledge captures generalizations that cannot easily be extracted from treebank grammars (and if so, only to a limited extent). The figures above confirm that our approach to f–structure induction compensates for this inherent lack of generalization by defining modular, principle-based annotation rules that apply to *partial c–structure configurations* of flat PS rules. The annotation principles import rule-based linguistic knowledge into the corpus-driven annotation process: they define how to generalize over varying syntactic contexts, abstracting away from linguistically irrelevant co-occurrences.

The amount of generalization captured by modular, partial annotation principles is also brought out by the high percentage of f–structures that could be correctly assigned for unseen portions of text: 48% and 76.09%, respectively.[30] These figures are extremely promising, but further work has to show how the novelty curve converges with repeated upgrades over new portions of text.

The robustness of our f–structure induction method – clearly indicated by the amount of large, partial f–structures in the case of missing annotation rules – relies on two main factors. One is the underlying rule compiler, the XLE term rewriting component. The rules are conditional, i.e., they only apply if their LHS terms match the input. They do not apply otherwise, hence the system doesn't fail in the case of missing rules. And since the principles specify small, modular c–structure configurations, they typically do not fail to apply in larger, unseen contexts. Second, in our application we rely on manually approved (i.e. deterministic) c–structure input, whereas in a classical LFG parsing architecture c– (and f-) structures are freely constructed from a set of PS rules, which therefore must encode numerous grammaticality constraints as a filter on a large set of possible structures assigned to an input string. Such grammars are by necessity more constrained, and therefore less robust. This burden is taken from annotation grammars that operate on disambiguated c–structure input.

5 Subcategorization and OT constraint ranking

The evaluation of our experiment in Section 4 does not measure the correctness of certain grammatical function assignments, in particular the argument/adjunct distinction for PPs and *to*-infinitives, and raising/control equations. In this first setup we focussed on the structural aspects of f–structure assignment, to test how well we can do without explicit lexical knowledge. Since we did not incorporate any lexical subcategorization knowledge, it is obvious that in some nondeterministic contexts grammatical function assignment cannot be correctly disambiguated, given only c–structure or lexical category information.

The next step is to integrate lexical subcategorization information in the annotation process. We first describe the basic formal account on how to integrate subcategorization information to resolve nondeterminism in grammatical function assignment. However, a simple, straightforward application of this scheme will involve considerable overhead in computation, and, in essence, compromise one attractive feature of our approach, its inherent robustness. We therefore propose a selective approach to subcategorization-based ambiguity filtering, combined with techniques for OT-based ambiguity ranking in large-scale LFG grammars (cf. Frank et al. (1998, 2000)).

[30] Even though section J01 features sentences with about 10 words in average more than in the later sections, this result is significant. Note also that these figures are conservative, in that they only record *completely* connected f–structures, whereas almost all remaining sentences yielded large pieces of partial, unconnected f–structures.

5.1 Lexical information for grammatical function assignment

Subcategorization ambiguities that cannot be predicted from ordinary c–structure encodings in treebanks are illustrated in (1) and (2) for argument/adjunct ambiguities of PPs and *to*-infinitives.[31]

(1) A plug and a tube with holes in its cylindrical walls divided the chamber above the porous plug into two parts.
[s, [n, a plug and a tube with holes in its cylindrical walls], [v,[vvd,divided]],[n, the chamber above the porous plug], [p, into two parts]]

(2) The high heat fluxes existing at the electrode surfaces of electric arcs necessitate extensive cooling to prevent electrode ablation.
[s,[n, the high heat fluxes existing at the electrode surfaces of electric arcs], [v,[vv0,necessitate]], [n, extensive cooling], [t, to prevent electrode ablation]]

In such nondeterministic contexts, annotation rules assign grammatical functions disjunctively, by use of optional rules. Below, the first rule applies optionally, defining the PP (`p`) as an ADJUNCT of the VP (`v`); the second rule (alternatively) assigns the PP the argument function OBLique.

```
arc(A,v,B,p) ?=> adjunct_x(A,B)
arc(A,v,B,p) ==> oblique(A,B)
```

Without any further lexical information about the governing verb, *divide* in (1), the ambiguity can only be resolved manually. Similar rules produce a systematic ambiguity for *to*-infinitives as in (2), which can be either ADJUNCTs or XCOMP complements.

In order to (partially) resolve such ambiguities, we integrate subcategorization knowledge, extracted from machine-readable dictionaries.[32] The subcategorization frames stated in the dictionary are compiled into the rule format illustrated below. Largely equivalent to subcategorization encoding in LFG semantic forms, the PRED value is expanded to a predicate `pred_x`, with an index `Id`, a unique identifier for the subcat-reading at hand (`pred_x(X,Lex,Id)`), together with a set of terms `GF_sc(X,Id)` for each grammatical function GF the lexical item subcategorizes for on reading `Id`.

```
pred(A,exist)        ==> pred_x(A,exist,1), subj_sc(A,1).
pred(A,necessitate)  ==> pred_x(A,necessitate,1), subj_sc(A,1), obj_sc(A,1).
pred(A,divide)       ?=> pred_x(A,divide,1), subj_sc(A,1), obj_sc(A,1).
pred(A,divide)       ==> pred_x(A,divide,2), subj_sc(A,2), obj_sc(A,2), obl_sc(A,2,into).
```

These lexical rules apply late in the annotation process, after the f–structure is completed through c–structure-guided, partially nondeterministic grammatical function assignment. With matching `pred` terms in the input, the rules introduce the respective subcategorization constraints into the f–structure space, where they can be used to check and filter the GF assignment ambiguities. This is done by encoding LFG completeness and coherence constraints in a straightforward way, as illustrated below for the SUBJ function. If completeness or coherence is violated, for any of the grammatical functions, we record the type of violation in a term `o_x(X,OTmark)`.

```
completeness constraints (subj):   +pred_x(A,_,X), +subj_sc(A,X), -subj(A,_) ==> o_x(A,incomplete).
coherence constraints (subj):      +pred_x(A,_,X), -subj_sc(A,X), +subj(A,_) ==> o_x(A,incoherent).
```

In this way we model the approach to OT constraint ranking for ambiguity filtering in LFG grammars proposed in Frank et al. (1998, 2000), extending it to well-formedness constraints of completeness and coherence. By declaring the "OT marks" `incomplete` and `incoherent` as `UNGRAMMATICAL`- or `NOGOOD`-marks in a grammar-specific constraint ranking hierarchy (see below), the f–structures marked with the respective OT marks will not be considered grammatical, and therefore effectively filtered from the set of grammatical f–structures.[33]

[31] Some treebanks, e.g. PennTreebank, encode subcategorization distinctions of this type. See Sec. 6 for discussion.

[32] The COMLEX dictionary, for example, (distributed by the Linguistic Data Consortium LDC at `http://www.ldc.upenn.edu`) provides extensive subcategorization information for English.

[33] For more detail on the interpretation of the various types of OT marks, see Frank et al. (1998, 2000). We implemented a corresponding constraint ranking algorithm which we apply to filter and rank ambiguities in the output of f–structure annotation.

```
Optimalityranking:  pp_obl inf_xcomp NEUTRAL UNGRAMMATICAL incomplete incoherent NOGOOD.
```

Given the lexical entry for *necessitate* above, the XCOMP reading for the *to*-infinitive in (2) is marked incoherent, the ambiguity is successfully resolved to the adjunct reading. On the other hand, *divide* in (1) *optionally* subcategorizes for an OBLique PP (*into*). Whereas an OBL function is penalized on the transitive reading 1, and vice versa for ADJUNCT assignment with reading 2, we cannot resolve the ambiguity between subcat reading 1 and 2 in terms of coherence and completeness constraints. However, in such cases of real syntactic ambiguity we can state a general preference for the OBLique reading of *optional* PP arguments, by assigning a preference mark `o_x(A,pp_obl)` in the corresponding lexical entry, and similarly for ambiguities involving *optional* XCOMP arguments.

```
pred(A,divide)   ?=> pred_x(A,divide,1), subj_sc(A,1), obj_sc(A,1).
pred(A,divide)   ==> pred_x(A,divide,2), subj_sc(A,2), obj_sc(A,2), obl_sc(A,2,into), o_x(A,pp_obl).
```

With the OT constraint ranking hierarchy given above, the (preferred) oblique reading will be determined as the winner in competition against the unmarked (`NEUTRAL`) ADJUNCT reading, which will be determined as suboptimal. While it will always be possible to construe counterexamples where such general preference constraints make wrong predictions, they prove very efficient and reliable in practice, and can be refined by further knowledge sources (cf. Frank et al. (1998, 2000)).

5.2 A selective approach to ambiguity filtering

With such a general approach for subcategorization-based ambiguity filtering and preference ranking, we could now compile large subcategorization dictionaries into lexical rules as illustrated above. Given the large amount of subcategorization frames that verbs, adjectives and also nouns can allow, this does not only imply a significant amount of processing, it also raises the issue of robustness. It is still easy to account for missing lexical entries in the subcategorization lexicon – here we exploit the conditional format of our lexical rules: if no entry is present for a given lexical form, no rule is applied. The worst that can happen in this case is that we do not filter any nondeterministic function assignments. We still get a set of (minimally) ambiguous f–structures for manual selection. However, the lexicon might contain a subcat entry for a lexical item which is missing a particular frame, the one that is crucial for the sentence at hand. In such a case, the corresponding (correct) function assignment will be judged incoherent or incomplete, the analysis will be rejected. One possible remedy is to declare completeness and coherence constraint violation as ungrammatical marks, as shown above. As in classical Optimality Theory, this type of constraint violation will only be considered "ungrammatical" if some other, less strongly marked competing analysis is available. That is, the analysis can surface as long as no other subcategorization frame fits the correct function assignment better, or no competing analyses can be validated. Yet, in such situations, unwarranted ambiguities may arise.[34]

But ...let us step back and reconsider our experiment above, where – without any lexical subcategorization information – most types of grammatical function assignments could be reliably stated by looking at c–structure configurations, lexical items, or lexical categories in treebank entries. For illustration, looking at (3), consider *that*-clauses adjacent to nouns, which could be either subcategorized COMPlements or relative clauses. Since the treebank uses distinct lexical categories for *that* as a complementizer, relative or demonstrative pronoun, the function assignment can be unambiguously determined by constraining the local syntactic context. Similarly, we can distinguish predicative *be* from the tense auxiliary in terms of its immediate c–structure context. And even though we generalize over the (original) distinct treebank categories for relative, adjunct, or complement clauses, the distinct f–structure contributions of these clauses can, again, be assigned by looking at a small class of lexical functional heads in the immediate syntactic context.

[34] As an example, if for some verb the lexicon specifies two subcat frames: ⟨SUBJ⟩ and ⟨SUBJ, OBL⟩, but misses a valid transitive frame ⟨SUBJ, OBJ⟩, a structure that introduces SUBJ and OBJ functions will be assigned an equal number of ungrammatical (completeness & coherence) violation marks for both incorrect readings.

(3) This result suggests a very high temperature at the solid surface of the planet, although there is the possibility that the observed radiation may be a combination of both thermal and non-thermal components and that the observed spectrum is that of a black body merely by coincidence.

A natural, selective approach to ambiguity filtering is thus to rely on grammatical function assignments *without lexical validation* in all those cases where the information can be gathered from the c–structure context (or morphological marking in case-marking languages), that is, by exploiting linguistic insights and generalizations encoded in the respective treebank annotations. Only those types of ambiguities that need to be validated or disambiguated in terms of lexical subcategorization properties will be checked by consulting lexical knowledge bases – which are by necessity error-prone and incomplete.[35] This *selective* approach to lexicon-based ambiguity filtering is an important move, in that it proposes a novel partition of ambiguity filtering knowledge in the LFG architecture, which is also essential to preserve the inherently robust architecture of our f–structure induction method. It is evident that this approach is strongly dependent on language-specific properties, the specific encoding schemes of the underlying treebanks, as well as, of course, our specific application scenario: f–structure annotation of *disambiguated* trees. Yet, the investigation of language-specific encoding strategies of grammatical functions constitutes an interesting research topic in itself.[36]

5.3 Future Work

We have implemented the basic and selective approach to subcategorization filtering and ambiguity ranking for a toy dictionary. In future work we will extract subcategorization frames for the 3 types of ambiguities discussed above from a large subcategorization dictionary, and test the lexicalized f–structure annotation grammar on the sections of the Susanne corpus analyzed in Section 4. The evaluation will then measure correct subcategorization assignment in *optimal* f–structures.

6 Annotating Treebanks – A Balancing Act

At this point, we need to raise an issue about varieties in treebank annotation schemes, and how best to exploit them for different applications. Some treebanks, such as the PennTreebank, encode a significant amount of subcategorization in complex category labels. The obvious question to ask then, is why not directly exploit these encodings. If our objective is simply to build f–structure banks, this is the best option. In our experiment we have shown that our approach is more flexible, in that it allows us to construct correct f–structures from poorer treebank encodings. Moreover, encoding grammatical functions in terms of complex category labels and coindexations in c–structure misses generalizations that appear, in a *normalized* representation, at f–structure (consider e.g. active/passive distinctions, extraposition, or topicalization). More importantly, our method can be extended to free parsing architectures, with c–structure filtering based on probabilistic CFGs, *trained on treebanks*. If such grammars are trained on highly complex PS categories, they will not generalize enough to deliver good statistics. Instead, treebank grammars are typically trained on suitably impoverished categorizations, stripping off special indexations and collapsing overspecific functional labels. The c–structures delivered by such a PCFG will miss some of the extra knowledge encoded in the treebank, but can be supplemented with the additional, much more general knowledge imported by annotation principles and lexical knowledge sources in subsequent f–structure projection. Finally, we can define *varying* annotation grammars for a validation scenario: one annotation grammar can exploit highly specific treebank encodings, i.e. functional labels and coindexations. The resulting f–structure bank can then be used as a reference

[35] In practice, this means that our subcategorization lexicon contains only a limited subset of lexical entries and subcat frame types, those related to the specific types of ambiguities we need to resolve – and these are limited. Lexicon lookup is then restricted to contexts where these respective types of ambiguities arise.

[36] See, for instance, the generalizations on morphological marking across languages in Berman (1999), as well as studies on head-marking languages (e.g., Nordlinger (1997), Bresnan (2000)).

corpus for evaluation of the second grammar, eventually used in free parsing, which operates on a coarser set of category labels (both in parsing and annotation), but exploits the complementary linguistic knowledge encoded in annotation principles and subcategorization lexica.

7 Conclusion

We presented one of two alternative methods to automate the approach of van Genabith et al. (1999a,b,c). Annotation principles define ϕ–correspondences from partial c–structure configurations to their corresponding partial f–structures. Our first experiment shows that this approach is a promising method for treebank annotation with f–structures. In particular, we show first promising figures for upgrading to extended fragments, which stem from generalizations captured in linguistically motivated, *modular and partial* annotation principles. Moreover, the method can be viewed as a technique for rapid, corpus-based grammar development. Our annotation grammar was set up in a time frame of about 3 weeks, and covers contiguous real-life text, including sentences with up to 94 words. The grammar comprises lexical and morphological rules ($\approx$ lexicon templates), tag compaction rules, as well as rules for virtually all core phenomena of syntax.[37]

In our approach – which we might call "corpus-based LFG" (CB-LFG) – we exploit the linguistic knowledge about c–structure organization that is implicit in the treebank entries, together with the fact that the assigned trees are disambiguated. In comparison, classical LFG grammars must not only define the context-free rule set for a particular language. Broad-coverage LFG grammars also have to cope with massive ambiguity arising from large sets of PS rules, and therefore state f–structure constraints to tame the amount of unwarranted ambiguities and ungrammatical analyses. In the context of f–structure induction from treebanks, we rely on c–structure disambiguation and, by and large, grammaticality of the input. The f–structure projection principles can be stated in a less constrained, declarative way, which naturally increases the robustness of such "grammars".

The annotation principles we use for f–structure induction from trees can be considered as a *modular, principle-based c–structure/f–structure interface for LFG grammar architectures*, and we apply it, here, to real-life language fragments, combining a corpus-driven approach with rule-based linguistic knowledge. There is, however, still some way to go to extend f–structure induction from treebank trees to a probabilistic robust parsing architecture for the analysis of new sentences.

References

Berman, J. (1999). Does German Satisfy the Subject Condition? In Butt, M. and King, T., editors, *Proceedings of the LFG99 Conference*, Manchester University, CSLI Online Publications, Stanford, CA.

Bod, R. (2000a). An Empirical Evaluation of LFG–DOP. In *Proceedings of COLING 2000*, Saarbrücken, Germany.

Bod, R. (2000b). An Improved Parser for Data-Oriented Lexical-Functional Analysis. In *Proceedings of ACL-2000*, Hong Kong, China.

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

Bresnan, J. (2000). Lexical-Functional Syntax. Blackwells Publishers, Oxford.

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

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

Cormons, B. (1999). *Analyse et désambiguïsation: Une approche à base de corpus (Data-Oriented Parsing) pour les représentations lexicales fonctionnelles.* PhD thesis, Université de Rennes, France.

[37] An example f–structure is given in the Appendix.

Dalrymple, M. (2000). Lexical-Functional Grammar. Manuscript, Xerox PARC.

Dipper, S. (2000). Grammar-based Corpus Annotation. In *LINC 2000*.

Eisele, A. (2000). *Representation and stochastic resolution of ambiguity in constraint-based parsing*. PhD thesis, Universität Stuttgart, Stuttgart.

Frank, A. (1999). From Parallel Grammar Development towards Machine Translation. A Project Overview. In *Proceedings of Machine Translation Summit VII "MT in the Great Translation Era"*, pages 134–142.

Frank, A., King, T., Kuhn, J., and Maxwell, J. (1998). Optimality Theory Style Constraint Ranking in Large-scale LFG Grammars. In Butt, M. and King, T., editors, *Proceedings of the LFG98 Conference*, University of Queensland, Brisbane, CSLI Online Publications, Stanford, CA. `http://www-csli.stanford.edu/publications/`.

Frank, A., King, T., Kuhn, J., and Maxwell, J. (2000). Optimality Theory Style Constraint Ranking in Large-scale LFG Grammars (revised and extended version). In Sells, P., editor, *Optimality Theoretic Syntax*. CSLI Publications.

Frank, A., van Genabith, J., Sadler, L., and Way, A. (1999). Automatic F-structure Annotation of Treebanks and CF Grammars. Unpublished Ms., Xerox Research Centre Europe, Dublin City University, University of Essex.

Halvorsen, P.-K. and Kaplan, R. (1995). Projections and Semantic Description in Lexical-Functional Grammar. In Dalrymple, M., Kaplan, R., Maxwell, J., and Zaenen, A., editors, *Formal Issues in Lexical-Functional Grammar*, pages 279–292. CSLI Lecture Notes, No.47.

Johnson, M., Geman, S., Canon, S., Chi, Z., and Riezler, S. (1999). Estimators for Stochastic "Unification-based" Grammars". In *Proceedings of the ACL*.

Johnson, M. and Riezler, S. (2000). Exploiting auxiliary distributions in stochastic unification-based grammars. In *Proceedings of the 1st NAACL conference*, Seattle, Washington.

Kallmeyer, L. (2000). A Query Tool for Syntactically Annotated Corpora. In *Proceedings of Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, Hong Kong.

Kay, M. (1999). Chart Translation. In *Proceedings of Machine Translation Summit VII "MT in the Great Translation Era"*, pages 9–14.

King, T.-H., Dipper, S., Frank, A., J.Kuhn, and Maxwell, J. (2000). Ambiguity Management in Grammar Writing. In *ESSLLI'2000 Workshop on Linguistic Theory and Grammar Implementation*, Birmingham, Great Britain.

Nordlinger, R. (1997). *Constructive Case. Dependent-Marking Nonconfigurality in Australia*. Ph.D. Thesis, Deperatment of Linguistics, Stanford University, Stanford.

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

Sadler, L., van Genabith, J., and Way, A. (2000). Automatic F-Structure Annotation from the AP Treebank. In Butt, M. and King, T., editors, *Proceedings of the LFG00 Conference*, University of California, Berkley, CSLI Online Publications, Stanford, CA. `http://www-csli.stanford.edu/publications/`.

van Genabith, J., Sadler, L., and Way, A. (1999a). Data-Driven Compilation of LFG Semantic Forms. In *EACL'99 Workshop on Linguistically Interpreted Corpora (LINC-99)*, pages 69–76, Bergen, Norway.

van Genabith, J., Sadler, L., and Way, A. (1999b). Structure Preserving CF-PSG Compaction, LFG and Treebanks. In *Proceedings ATALA Workshop - Treebanks*, Journées ATALA, Corpus annotés pour la syntaxe, Université Paris 7, France, 18-19 Juin 1999, pages 107–114.

van Genabith, J., Way, A., and Sadler, L. (1999c). Semi-Automatic Generation of F-Structures from Tree Banks. In Butt, M. and King, T., editors, *Proceedings of the LFG99 Conference*, Manchester University, 19-21 July, CSLI Online Publications, Stanford, CA. `http://www-csli.stanford.edu/publications/`.

Way, A. (1999). A Hybrid Architecture for Robust MT using LFG-DOP. In *Journal of Experimental and Theoretical Artificial Intelligence 11 (4) (Special Issue on Memory-Based Language Processing)*.

Appendix: An example f–structure from the Susanne Corpus

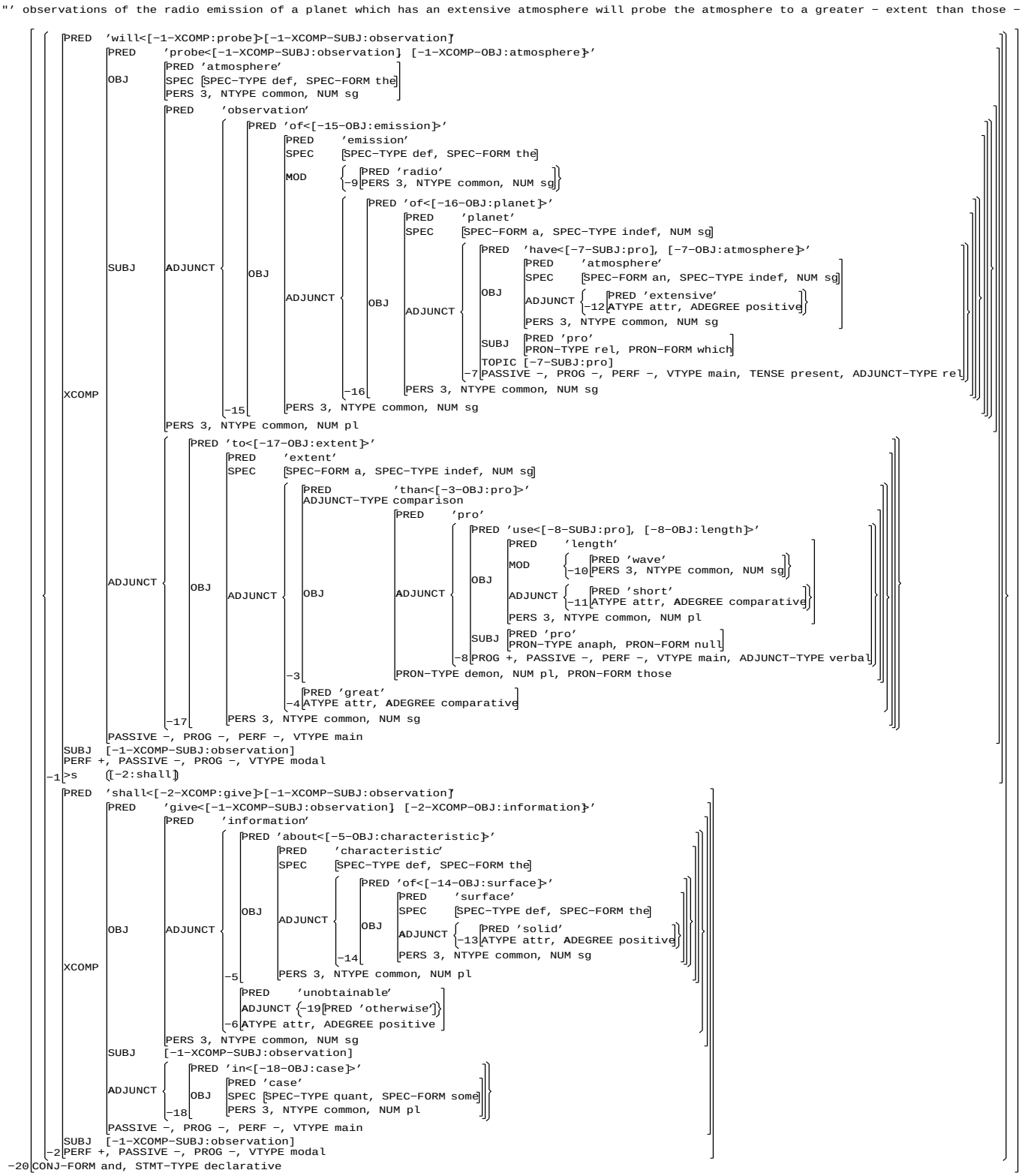


Figure 1: F–structure for: *Observations of the radio emission of a planet which has an extensive atmosphere will probe the atmosphere to a greater extent than those using shorter wave lengths and should in some cases give otherwise unobtainable information about the characteristics of the solid surface.*