Chapter 1

FROM TREEBANK RESOURCES TO LFG F-STRUCTURES

Automatic F-Structure Annotation of Treebank Trees and CFGs extracted from Treebanks

Anette Frank
Xerox Research Centre Europe
Anette.Frank@xrce.xerox.com

Louisa Sadler
University of Essex
louisa@essex.ac.uk

Josef van Genabith
Dublin City University
josef@compapp.dcu.ie

Andy Way
Dublin City University
away@compapp.dcu.ie

Abstract We present two methods for automatically annotating treebank resources with functional structures. Both methods define systematic patterns of correspondence between partial PS configurations and functional structures. These are applied to PS rules extracted from treebanks, or directly to constraint set encodings of treebank PS trees.

Keywords: Automatic annotation, higher-level syntax, corpus linguistics, robustness

1. INTRODUCTION

In this contribution we address two important concerns: automatic annotation of treebanks and CFGs extracted from such treebanks with LFG f(eature)-structures (Kaplan and Bresnan 1982), (Dalrymple et al. 1995).

Treebanks which encode higher-level functional structure, in addition to pure phrase structure information, are required as training resources for probabilistic unification grammars and data-driven parsing approaches, e.g. (Bod and Kaplan 1998). Manual construction of treebanks with feature structure annotations is very labour and cost intensive. So is the development of new or the scaling-up of existing unification grammars which can be used to analyse text corpora. What is more, even if a large-coverage unification grammar is available, typically, for each input string it would generate hundreds or thousands of candidate (constituent and feature structure) analyses from which a highly trained expert has to select. Although proposals have been made for filtering and ranking parsing ambiguities (e.g. (Charniak 1993), (Abney 1997), (Frank et al. 2000)), to date none is guaranteed to uniquely determine the best analysis. In order not to compromise the quality of the corpus under construction, a linguistic expert is required to find the best among a large number of candidate analyses.

Given this situation, is there a way to automate, or bootstrap, the construction of grammars and treebanks with feature structure annotations reusing existing resources?

In a number of papers van Genabith et al. (1999a,b,c) presented a new corpus based method. The basic idea is the following: take an existing treebank, read off the CFG following (Charniak 1996), *manually* annotate it with f-structure annotations, provide macros for the lexical entries and then "reparse" the treebank trees (not the strings) deterministically following the original tree structure annotations assigned in the treebank. During this "reparsing" process, the f-structure annotations are resolved, and an f-structure is produced. The entire process is deterministic if the feature structure annotations are, and to a considerable extent costly manual inspection of candidate analyses is avoided. The method is an improvement but still involves a large labour intensive component, namely *manual* annotation of the grammar rules.

Treebank grammars (CFGs extracted from treebanks) are large and grow with the size of the treebank (Charniak 1996), (Krotov et al. 1998). They feature rather flat rules, many of which share and/or repeat significant portions of their RHSs. This causes problems for manual rule annotation approaches such as the one described in (van Genabith et al. 1999a,b,c). Manual rule annotation is labour intensive, error prone, repetitive and risks missing generalisations.

In this paper we show how f-structure annotation of both grammar rules and tree fragments can (to a large extent) be *automated*.

The basic idea is simple: functional annotations follow systematic patterns. These systematic correspondences between constituent and higher level feature structure representations can be captured in general annotation principles, which are applied to either grammar rules extracted from a treebank or directly to treebank PS trees.

The observation that constituent and higher-level feature structure representations stand in a systematic relationship informs theoretical work in LFG (Kaplan and Bresnan 1982), (Dalrymple et al. 1995) and HPSG (Pollard and Sag 1994). In LFG c(onstituent)-structure and f-structure are independent levels of representation which are related in terms of a correspondence function ϕ. The correspondence follows linguistically determined principles which are partly universal, and partly language specific (Bresnan 2000), (Dalrymple 2000).

What is new in our approach is that (i) we employ *partial* and *underspecified* annotation principles in a principle based c- to f-structure interface for the LFG architecture; (ii) we use these to automate functional structure assignment to flat and "noisy" treebank trees and CFGs extracted from them; (iii) we reuse existing linguistic resources. In contrast to more theoretically informed work in LFG and HPSG, treebanks do not tend to follow highly abstract and general X′ architectural design principles. The challenge in our approach is to develop grammars and annotation rules for real text.

The potential benefits of automation are considerable: substantial reduction in development effort, hence savings in time and cost for treebank annotation and grammar development; the ability to tackle larger fragments in a shorter time, a considerable amount of flexibility for switching between different treebank annotation schemes, and a natural approach to robustness. Our methods can also be viewed as a new corpus- and data-driven approach to grammar development, an approach that as much as possible recycles existing resources.

In our work to date we have developed two related but interestingly different methods. Both methods define association principles as correspondences between *partial* and *underspecified* c- and f-structure configurations. In one approach we read off a CFG treebank grammar following the method of Charniak (1996) and then compile annotation principles over the treebank grammar. In our second approach we operate directly on constraint set encodings of PS treebank trees and rewrite or annotate them directly with f-structures.

Both methods are partial in the following further sense: the first requires manual inspection, completion and correction of the output (sets of annotated grammar rules) produced by the automatic annotation process. The second method is fully automatic and robust, and yields partial, unconnected f-structures in the case of missing annotation rules.

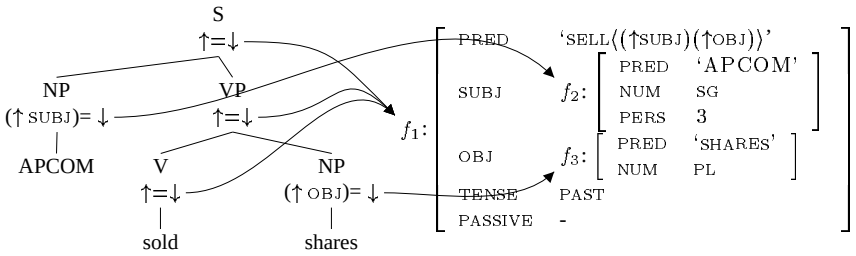
We describe two experiments, one for each method. In order to explore some of the possible architectures, for the first experiment we developed a regular

expression based annotation principle interpreter which operates on grammar rules with *order independent* and monotonic interpretation of principles. For the second experiment we employed a term rewriting system which operates on constraint set descriptions of LFG structures. The term rewriting system allows us to exploit both *order dependent*, cascaded and *order independent* formulations of annotation principles. In our first experiment we used the first 100 trees of the AP treebank (Leech and Garside 1991), in the second, experiment 166 trees of the Susanne treebank (Sampson 1993).

The paper is structured as follows: in Section 2 we motivate and describe our methods in more detail. In Sections 3 we report on our two experiments. For each experiment we explain the design, describe the data and evaluate the results. In Section 4 we compare the two methods and outline ongoing research. Section 5 concludes.

2. METHODS FOR AUTOMATIC F-STRUCTURE ANNOTATION

In LFG the correspondence between functional and phrasal structure is defined in terms of functional annotations of the RHS categories in CFG rules.



PS rules define f–structure via functional descriptions

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

Annotation follows universal and language specific principles. We define annotation principles as involving *partial* and *underspecified* phrase structure configurations and apply them to CFG rules or tree fragments that meet the relevant *partial* configuration. To illustrate the idea: a head principle assigns $\uparrow = \downarrow$ to the X daughter in all XP $\rightarrow \ldots \text{X} \ldots$ configurations, irrespective of the surrounding categorial context. For the example at hand, the challenge in our approach is to provide annotation principles that identify heads in the flat treebank tree and rule configurations which generally deviate signficantly from X′ design principles. Annotation principles capture generalisations and can be used to *automatically* annotate PS configurations with functional structures in a highly general and economical way. Both annotation methods are built on

this insight: in the first, correspondences are applied to CFG rules extracted from treebanks while in the second correspondences are applied directly to constraint set encodings of treebank trees and tree fragments.

2.1 F-STRUCTURE ANNOTATION OF CFGS EXTRACTED FROM TREEBANKS

Annotation Principle Interpreter. Our CFG rule annotation principles are of the form `L>R@A`. `L` and `R` are regular expressions (under)specifying LHSs and RHSs of CFG rules in terms of categorial and configurational constraints. The regular expressions provided include Kleene and positive Kleene (`*`, `+`), optionality (`?`), disjunction (`|`) and a limited form of complement (`~`). `A` is a set of attribute-value structure annotations (rule decorations). Given a grammar rule of the form `M>D` (expanding a mother category `M` into a sequence of daughter categories `D`) and an annotation principle `L>R@A`, if `L` matches `M` and `R` matches `D`, then `M>D` is annotated with `A`. A single grammar rule can match multiple principles and a single principle may match a given grammar rule in more than one way. The annotations resulting from all possible matches are collected and the grammar rule is annotated accordingly. More formally, let the denotation $[\![\mathtt{E}]\!]$ of a regular expression `E` be the set of strings denoted by `E`. Given a CFG rule `M>D` and a set of annotation principles `AP` of the form `L>R@A`, `M>D` is annotated with the set of feature structure annotations `F`:

$$\mathtt{M>D@F} \text{ iff } \mathtt{F} = \{\mathtt{A} | \exists \mathtt{P} \in \mathtt{AP} \text{ with } \mathtt{P} = \mathtt{L>R@A} \text{ and } \mathtt{M} \in [\![\mathtt{L}]\!] \text{ and } \mathtt{D} \in [\![\mathtt{R}]\!]\}$$

Annotation is monotonic and order independent.

Example Principles. In our Prolog implementation, CFG grammar rules extracted from the treebank are represented as

```
C:F > C1:F1, ... , Cn:Fn.
```

where syntactic categories `C` and logical variables `F` representing feature-structure information are paired `C:F`. Annotation principles can underspecify the LHS and RHS of grammar rules. To give a simple example, the following annotation principle[1] states that infinitival phrases `infp` following the final `v0` in `vp` rules are open complements (`xcomp`) controlled by the subject of the final `v0`:

```
vp > *  v0:V0  *(~v0)  infp:I  *
   @ [ V0:xcomp = I, V0:subj = I:subj ].
```

The next principle states that in non-conjunctive contexts[2] `v0` sequences, possibly separated by adverbials `adv`, form open complement sequences where the subject of the preceding `v0` controls that of the following:

```
vp > *(~conj)  v0:V1  ?adv  v0:V2  *(~conj)
   @ [ V1:xcomp = V2, V1:subj = V2:subj ].
```

Note that the principle applies twice to a ...`v0:V1, v0:V2, v0:V3` ... RHS rule configuration with `[ V1:xcomp = V2, V1:subj = V2:subj, V2:xcomp = V3, V2:subj = V3:subj ]` as the resulting annotation. Finally observe that the formalism supports the statement of generalisations over LHSs of CFG rules:

```
( fn:X|infp:X|tgp:X|si:X|vp:X )
   > *(~(v0|conj))  v0:V0  *(~conj)
   @ [ X = V0 ].
```

This principle states that for a variety of constructions including verbal (`vp`) and infinitival (`infp`) phrases in non-conjunctive contexts the initial `v0` is the head of the clause.

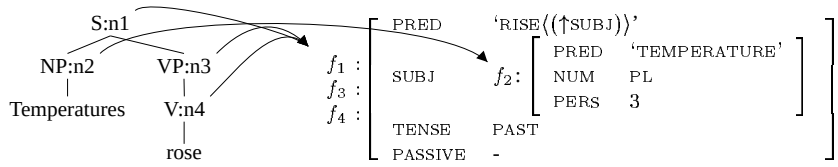
Example output (automatically annotated grammar rules) is shown below:[3]

```
vp:A > v0:B,v0:C,v0:D,np:E,fa:F
     @ [A===B,D:obj===E,C:xcomp===D,C:subj===D:subj,
        B:xcomp===C,B:subj===C:subj,A:vp_adjunct:1===F].

vp:A > v0:B,v0:C,v0:D,rp:E,pp:F
     @ [(D:obl===F;D:vp_adjunct:1===F),A===B,D:part===E,
        C:xcomp===D,C:subj===D:subj,B:xcomp===C,
        B:subj===C:subj].

vp:A > vp:B,pnct:_,vp:C,pnct:_,conj:D,vp:E
     @ [A:conj:3===C,A===D,A:conj:2===B,A:conj:1===E]).

vp:A > vp:B,conj:C,vp:D,pp:E,fa:F
     @ [(D:obl===E;D:vp_adjunct:1===E),A===C,A:conj:2===B,
        A:conj:1===D,A:vp_adjunct:1===F]).
```

In the first and in the second rule the leftmost `v0` is identified as the head of the construction. In `v0,v0` sequences the second `v0` provides an open complement `xcomp` to the first with the subject of the second controlled by the subject of the first. The `np` in the first rule is analysed as the object of the rightmost `v0`, while the `pp` in the second rule is either an adjunct or an oblique argument to the `vp`. The last two example rules show conjunctive structures. Note that in the final rule the `pp` is analysed as oblique or as an adjunct to the rightmost `vp`. Here our current annotation principles miss a possible attachment of the `pp` to the mother `vp`.

2.2 F-STRUCTURE ANNOTATION OF TREEBANK TREES

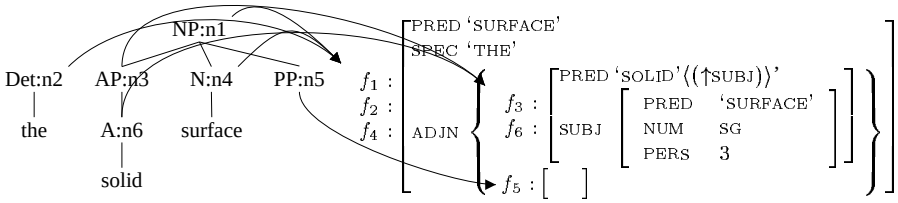
In our second approach we build on a pure correspondence view of the LFG architecture, where the mapping from c- to f-structure is encoded by the projection function ϕ. Annotation principles define ϕ-projection constraints which associate partial c-structures with their corresponding partial f-structures. Application of annotation principles to constraint set encodings of treebank trees directly induces the f-structure, allowing us to skip the (re)parsing process for f-structure composition. The principles can apply to non-local tree fragments, as opposed to local CFG rules.



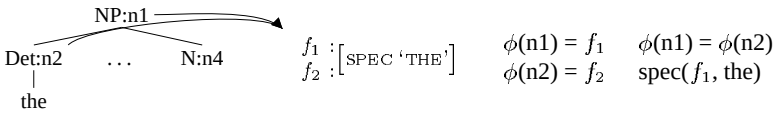
ϕ–correspondence:
ϕ(n1) = f_1 ϕ(n2) = f_2
ϕ(n3) = f_3 ϕ(n4) = f_4 ϕ(n1) = ϕ(n3) = ϕ(n4)

f–structure:
(f_1 SUBJ)= f_2, (f_2 PRED)= 'temperature'
(f_4 PRED= 'rise' ...

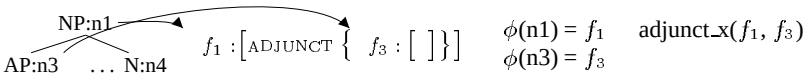
Modular projection principles for f-structure annotation of tree fragments. To illustrate the key idea of partial f-structure annotation principles, below we display the representation of a complex NP. This complex configuration can be broken down into modular, piece-wise correspondences of *partial* c- and f-structures, abstracting away from irrelevant material in the surrounding context.



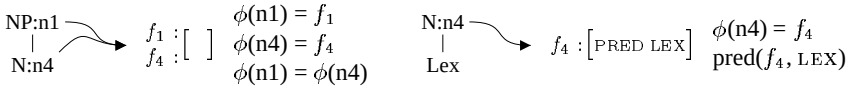
The functional contribution of the prenominal determiner *the* is independent of the presence of AP or PP, and is captured by the partial correspondence constraints stated on the right hand side.



An AP daughter of NP is analysed as an ADJUNCT of the nominal head, unless the N head is omitted. The former generalisation is captured below.

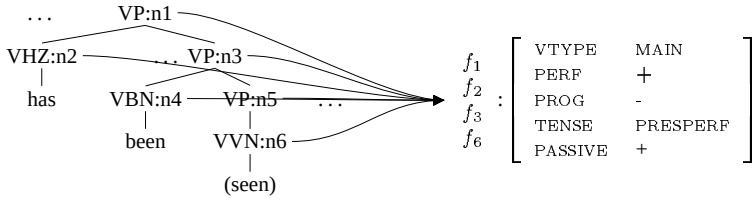


Projection principles for head categories and lexical nodes (here for nominal categories) are straightforward:



Similar correspondences are defined for the remaining c-structure fragments. These correspondences all apply to the complex NP structure above, conspiring to define the ϕ-projection and f-structure in a modular, declarative way. By dint of abstracting away from immaterial c-structure context, the principles generalise over specific tree configurations, and therefore apply to fragments of unseen trees.

In the correspondence-based approach annotation principles can apply to *non-local* tree fragments. This allows us to associate partial f-structures with complex c-structure fragments. For example, by specifying non-local c-structure fragments in binary branching VPs, we capture tense and active/passive distinctions of the verbal complex in a natural way. This is illustrated for the characteristic construction indicative of present perfect tense.



The idea of modular annotation principles is much in the spirit of projection principles as proposed by (Dalrymple 2000) and (Bresnan 2000), and provides a principle-based c- to f-structure interface in the LFG architecture.[4] Application of annotation principles to c-structure trees follows the description-by-analysis (DBA) approach of (Halvorsen and Kaplan 1995) in the c-to-f-structure interface. 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.

A term rewriting system for f-structure annotation. To define and process annotation principles we make use of an existing term rewriting system, originally designed for transfer-based Machine Translation (Kay 1999), (Frank 1999).

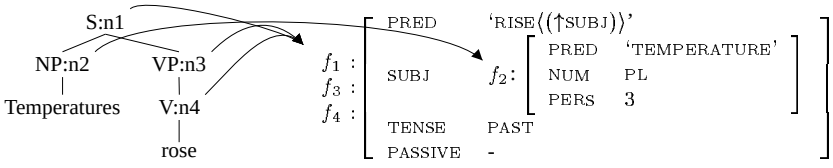
The system takes as input an unordered set of n-ary terms `p,q`, and an ordered set of rewrite rules `p` $\Rightarrow$ `q`.[5] If the LHS terms `p` match the input, the

matching terms `p` are eliminated from the input set, and the terms `q` are added to the output set. A rule applies to *each instantiation* of the LHS terms in the input. Besides terms `p` that are to be eliminated from the input, the LHS may state positive `+p` and negative `-p` terms. A rule with positive term `+p` only applies if `p` matches some term in the input. Positive terms are not eliminated from the input set. A rule with negative term `-p` only applies if `p` does not match any term in the input. 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.

A term representation of the LFG architecture We encode the LFG projection architecture in a term representation language as follows:

immediate dominance:	`arc(MNode, MLabel, DNode, DLabel)`
immediate precedence:	`prec(CsNode_x, CsNode_y)`
lexical insertion:	`lex(TerminalNode, Lex)`
ϕ-correspondence:	`phi(CsNode, FsNode)`, `equal(FsNode_x, FsNode_y)`
f-structure attributes:	`attr(FsNode_x, FsNode_y)`, `attr(FsNode, Val)`

With this, the traditional representation



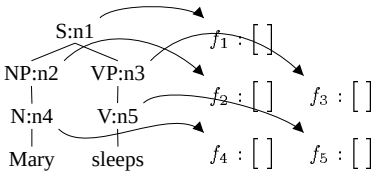
is translated into the following set of terms:

```
arc(n1,s,n2,np), arc(n1,s,n3,vp), arc(n3,vp,n4,v), prec(n2,n3),
lex(n2,Temperatures), lex(n4,rose),
phi(n1,f1), phi(n2,f2), phi(n3,f3), phi(n4,f4),
equal(f1,f3), equal(f3,f4),
pred(f1,rise), subj(f1,f2), pred(f2,temp.), num(f2,pl), tense(f1,past),..
```

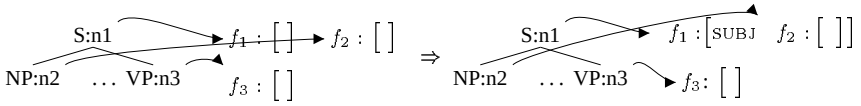
2.2.1 Automatic annotation of trees with f-structures.

Initialisation Starting from the c-structure term representation, we induce a 1-1 ϕ-correspondence from c-structure nodes to empty f-structure nodes.

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

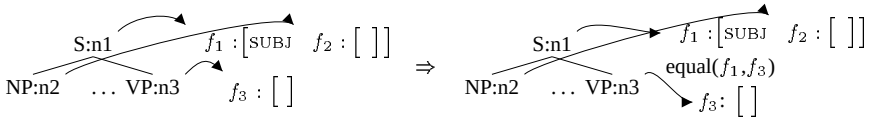


Annotation rules associate partial c-structure configurations with their corresponding partial f-structures, and further restrict the trivial 1-1 ϕ-correspondence via the predicate `equal(Fx,Fy)`. The rule below defines the external NP as the SUBJ of f_1, the f-structure projected from the S node. The predicate `prec_x(B,C)` is defined (by use of macros) as a finitely constrained transitive closure over the precedence relation `prec`. It can be used to underspecify precedence constraints holding between nodes n_x and n_y, allowing for an arbitrary or else a restricted sequence of intervening categories.




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

The following 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).
```

Formal restrictions We restrict `phi` predicates to only occur in LHSs of rules as *positive constraints*. Given the input specification of a 1-1 ϕ-projection, this guarantees that the functional property of the ϕ-correspondence is preserved. `equal` predicates *restrict* the ϕ-correspondence, while preserving its functional property.

Order independence in a cascaded rewrite system Although annotation rules operate in a cascaded, order dependent way, order independence can be obtained by requiring that no annotation rule refers to f-structure information introduced by other rules, and no rule consumes (or adds) any c-structure information referred to by other rules. These constraints ensure that annotation rules have access to the full initial input structure, and no more than this, and thereby guarantee order independence of annotation, irrespective of the order in which the rules are stated and applied. The effect of order independence can be observed by inverting the application order of the subject and head-projection rules above: while the intermediate term set will be different, the final output set will be identical.

There is a trade-off between order dependence and independence. Constraining rules to c-structure information only can require complex rule constraints to

avoid application of different annotation rules to the same tree fragment, leading to inconsistencies. Reference to f-structure information can also be used to generalise annotation rules. If several PS configurations are indicative of e.g. a subject, or passive voice, such diverse configurations can be captured by referring to the more abstract f-structure information to further guide f-structure construction. The order of annotation rules must then ensure that the required f-structure information is introduced by previous annotation rules.

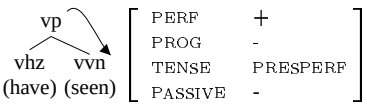
An annotation grammar consists, just like an ordinary LFG grammar, of different types of annotation rules: lexical, morphosyntactic, and phrasal.

Lexical and morphosyntactic rules Morphosyntactic rules introduce morphological (and some semantic) information encoded in lexical category labels into the f-structure space. The example given below illustrates how highly specific category distinctions in treebank encodings can be neutralised: once NUMber is encoded in f-structure, based on the `nn1` vs. `nn2` distinction, the distinction can be neutralised by mapping both lexical category labels to the generalised label `nn` (see (van Genabith et al. 1999b) for a similar approach). Such generalisations are 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 "generalised" `nn`-daughters.

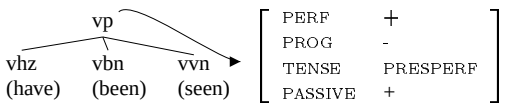
```
arc(A,ML,B,nn1) ==> num(B,sg), ntype(B,common), arc(A,ML,B,nn).
arc(A,ML,B,nn2) ==> num(B,pl), ntype(B,common), arc(A,ML,B,nn).

+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 present perfect tense in a flat VP, as it is assigned in the Susanne corpus. For binary branching VPs (as assigned in the Penn-II Treebank), we can define complex tense information in similar ways, by extending annotation rules to *non-local* tree fragments (see above and (Frank 2000)).

```
+arc(A,vp,B,vhz)     % have-aux
-arc(A,vp,D,vbn)     % no been-aux !
+arc(A,vp,C,vvn)     % main verb participle
⇒ 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,+).
```



Partial phrasal rules and underspecification Annotation rules are designed to apply to modular, partial c-structure configurations, to define their corre-

sponding functional projections. Even though treebanks do not tend to follow classical X′ syntax, specific types of tree branches correspond to functional dependencies in f-structure. Annotation rules apply, in the general case, to single tree branches, with some contextual constraints, and generalise to unseen tree configurations. Below, *that*-clauses (category `f`) are associated with a function COMP in f-structure by referring to a single branch (`arc`) in c-structure, abstracting away from irrelevant co-occurrences in the 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 `ML`, we generalise over various possible mother labels (e.g. (in)finite, modal, nominal or adjective phrases).

+arc(A,ML,B,f), +comp_form(B,that) ⇒ comp(A,B).

Finer categorial restrictions can be captured by defining classes of category labels in disjunctive templates.[6] Below, the disjunctive template `np_cat(X)` defines a class of category labels (`n`, `d`, `m`). The template is called (by logical "and" &&) in the annotation rule for PPs (`p`) to define this restricted class of alternative NP-types as complements (i.e., OBJ) of prepositions in a single rule.

```
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 we give an example for the assignment of OBJ vs. OBJ2 functions for transitive and ditransitive verbs in English, which is determined by surface order. Long-distance phenomena are captured by path expressions (see (Frank 2000)).

```
+arc(A,vp,C,np), +arc(A,vp,D,np), +prec_x(C,D) ==> obj2(A,D).   % OBJ2 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,C).[7]    % OBJ of transitives
```

Subcategorisation assignment We induce subcategorisation frames (the semantic forms) by collecting grammatical functions assigned by annotation rules into the predicate's semantic form, following the method of (van Genabith et al. 1999a).

Obviously, pure c-structure information does not allow us to distinguish between NP, PP, or infinitival arguments vs. adjuncts. Similarly, lacking lexical information, raising and control constructions can only be represented as involving anaphoric control. In (Frank 2000) we show how to extend this

model by integration of lexical subcategorisation information, combined with strategies for OT-based ambiguity ranking and filtering (cf. Frank et al. (2000)).

Partial annotation and robustness Our second f-structure annotation method embodies an important aspect of robustness. In the case of missing annotation rules the system does not fail, but partial trees are left without f-structure annotation. We obtain (typically large) partial, unconnected f-structures.

Moving treebanks 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 c-structure rewriting rules to transform the encoding of coordination and flat modal VP structures into more standard PS analyses, which lend themselves to principle-driven f-structure annotation.

3. TWO EXPERIMENTS

3.1 EXPERIMENT I

Experiment Design. Our first experiment involves the first 100 trees of the AP treebank (Leech and Garside 1991). We refer to this subsection as AP01. We preprocess the treebank using the structure preserving grammar compaction method reported in (van Genabith et al. 1999b) preserving as much categorial fine-grainedness as is required to guide annotation. From this we extract a treebank grammar following (Charniak 1996). We develop a set of feature structure annotation principles. The regular expression based interpreter described in section 2.1 compiles the principles over the rules extrated from the AP01 treebank fragment. The results obtained are compared against a manually annotated "gold standard" reference grammar and precision and recall measures are reported.[8]

Data. The AP treebank annotation schema employs 183 lexical tag types and 53 non-terminal category types, with tree structure encoded in terms of labelled bracketing. The corpus is 'skeletally parsed', that is, it contains some unlabelled brackets. We remove these in an automatic pre-editing step. The sentences in the AP01 fragment range from 4 to 50 leaf tokens (including punctuation symbols). The AP01 section of the corpus attests 94 of the 183 lexical tag types and 25 of the 53 phrasal tag types. The large number of highly discriminating terminal and non-terminal categories results in a large number of flat and often very specific rules. To facilitate annotation we use the structure preserving grammar compaction method presented in (van Genabith et al. 1999b) to compact the grammar into a more general one that still preserves important categorial information to drive automatic annotation. Compaction

works by generalising tags, i.e. collapsing tags (and categories) into supertags. This reduces the number of CFG rule types from 511 to 330. AP01 and the compacted AP01^{C} are summarised in table T1 below:

T1	sentences	average lengh	phrasal types	lexical types	CFG rule types
AP01	100	20	25	94	511
AP01^{C}	100	20	12	28	330

Manually Annotated Reference Grammar. In order to evaluate Experiment I we manually constructed a "gold standard" reference grammar following (van Genabith et al.1999a,b,c). The grammar features 1143 annotations, on average 3.46 annotations per rule.

Automatic Annotation and Evaluation. For the experiment we constructed 119 annotation principles, this against 330 CFG rules resulting in a template/rule ratio of 0.36. We expect the ratio to skew more in favour of templates as we proceed to larger fragments. Automatic annotation generates 1029 annotations, on average 3.12 annotations per rule. Experiment I is evaluated in terms of *precision* and *recall* measures:

$$\text{precision} = \frac{\#\text{ generated annotations also in reference}}{\#\text{ generated annotations}}$$

$$\text{recall} = \frac{\#\text{ reference annotations also generated}}{\#\text{ reference annotations}}$$

The results are summarised in table T2:[9]

T2	Experiment I
precision	87.9
recall	83.7

The numbers are conservative: *precision* and *recall* are computed automatically for a first pass encoding of annotation principles as regular expressions. The results are encouraging and indicate that automatic annotation is more often partial than incorrect.

3.2 EXPERIMENT II

Our method for f-structure annotation of trees in section 2.2 is evaluated in Experiment II, this time based on the Susanne corpus (Sampson 1993).

Data The Susanne treebank encodes labelled bracketed structures with surface form and lemmatised lexical entries. Functional category labels (subj, obj) and traces indicating control or long-distance dependencies are eliminated in preprocessing, to guarantee a non-biased evaluation with conventional PS trees as input. In preprocessing we also collapse overspecific phrasal categories.

Some decisions on PS assignment in the Susanne corpus are debatable. We defined a set of c-structure rewriting rules that transform the encoding of coordination and flat modal VP structures into a standard PS analysis.

Experiment Design We chose two sections of the Susanne corpus, J01 and J02 (text type J: learned writing). On these, we ran an experiment in 3 steps:

First, based on the first 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 first 50 sentences of J02, and measure the annotation results. Grammar AG1 is then upgraded to AG2, which covers these 50 sentences. We record the number of rules that were added or modified. In step 3, AG2 is applied to the remaining 46 sentences of J02. Again, we measure the results. In this experiment we applied an order dependent annotation scheme that consumes c-structure terms while building up the f-structure (cf. (Frank 2000)). We established a natural order for the different types of annotation principles discussed in section 2.2.1.

Evaluation and Results Table 3 provides basic data on these subsections: the number of sentences and average sentence length; the number of phrasal and lexical categories and the number of distinct PS rules and PS branches encoded by the corpus trees. Note that the percentage of new (unseen) PS rules in J02-1 and J02-2 is considerably higher than for new (unseen) tree branches. This is not surprising, and supports our annotation scheme, where annotation involves underspecified, partial trees (often single branches).

Table 3	sent.	length	phrasal cat	lexical cat	PS rules	tree branches
J01	66	34.27	32	73	430	281
J02-1	50	21.68	25 (3 new)	64 (8 new)	249 (60.34% new)	172 (20.93% new)
J02-2	46	24.8	24 (4 new)	57 (3 new)	212 (45.28% new)	163 (15.95% new)

The results are summarised in Table 4. We measured correctness of f-structure assignment modulo the argument/adjunct distinction for PPs and infinitival VPs, and the missing assignment of control/raising equations. 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 features 118 non-lexical (phrasal) annotation rules and assigns correct f-structures to 48% of the unseen section J02-1. As expected, the upgrade from

AG1 to AG2 required little effort: it involves 28 new and 5 modified rules and required approx. one person day of work. AG2 applied to the unseen section J02-2 yields 76.09% of correct f-structures.

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

Although small scale, we consider these results as promising. Upgrading to larger fragments takes little effort due to the generalisation capacity of annotation principles. This is also brought out by the increasing percentage of correct f-structure assignments to unseen trees, and the fact that partial f-structure assignments generally consist of large pieces of partial f-structures.

4. DISCUSSION AND CURRENT RESEARCH

We have presented two automatic f-structure annotation methods for treebanks and grammars. Both methods and the experiments show considerable overlap and several interesting differences.

Annotation principles can apply to extracted PS rules or to PS tree fragments encoded as constraint sets. Our second method can be specialised to PS rules by restricting trees to depth one. The first method generates an annotated grammar, which can be used to reparse treebank trees or serve as a basis for developing a stand-alone LFG resource. In the second approach an f-structure is built during the annotation process. In order to parse free text, this method can be applied to the output of (P)CFG parsing. The same architecture can be implemented using the principles designed in the first approach. Our second approach can be modified to annotate (non-local) tree fragments with f-descriptions for the reparsing scenario applied in the first method. Both our methods use compaction techniques for generalising overspecific categorisation. In the first experiment the structure of treebank entries remains unchanged while in the second certain structures are transformed to conventional PS analyses to support principle-based annotation. For our first method, we implemented an order independent and monotonic annotation principle interpreter. For the second, a more general term rewriting system was used. The term rewriting system allows us to exploit an order dependent, cascaded statement and processing of annotation principles. Alternatively, the term rewriting system can implement order independent annotation without consumption of input constraints. Order independence can sometimes ease maintenance of annotation principles, but

requires more complex and verbose constraints in order to avoid inconsistent annotations. By contrast, order dependent cascaded rewriting allows for a compact representation of annotation rules. The extra power of an order dependent system can be useful in category generalisation and subcategorisation induction during the annotation process. Experiment I uses a manually constructed "gold standard" reference grammar for evaluation, experiment II is evaluated with respect to how it performs on extending the treebank fragment. For larger fragments, clearly this is the only possible evaluation method.

Output of the first method (a set of annotated rules) can be manually corrected and completed, while the second is automatic and robust by returning partial and unconnected f-structures in case of missing or conflicting annotation rules.

Robustness is an inherent property of the approaches presented here. It resides in a number of levels: First, our principles are partial and underspecified and will match new, as yet unseen configurations. Second, the principles are conditional. If a certain context (a regular expression or a constraint set) is met, a principle applies. Even if only few principles apply, the system will not fail but deliver partial annotations. Third, the constraint solver employed in our second method can cope with partial, unconnected or even conflicting information. A constraint solver of this type can also be imported into the processing of rules annotated by our first method.

Both approaches factor out information spread over CFG grammar rules into modular and general principles. To a first approximation, the reason why our principles allow a compact representation of grammatical knowledge is the following: by and large the annotation principles capture statements about single mother – daughter relationships in CFG rules or local trees of depth one. This means that the principles are essentially about single branches in local configurations. Given a treebank (grammar) with n distinct categories the worst case number of distinct branches is n^2. Contrast this with the worst case number of possible grammar rules:

$$\begin{array}{lcl} x \rightarrow y_1 & \hat{=} & n^2 \\ x \rightarrow y_1 y_2 & \hat{=} & n^3 \\ \ldots & \ldots & \ldots \\ x \rightarrow y_1 \ldots y_m & \hat{=} & n^{m+1} \end{array}$$

Clearly, given a grammar with n categories and a RHS rule length of at most m, the worst case number of different grammar rules

$$\sum_{i=1}^{m} n^{i+1} \gg n^2$$

for $m > 2$ is much higher than the worst case number n^2 of distinct branches.

In our current research we are working with the Penn-II treebank resource. Compared to our AP and Susanne experiments we are applying our methods to a treebank fragment larger by an order of magnitude.

In order to develop stand-alone LFG grammars we need semantic forms (subcategorisation lists) to enforce subcategorisation requirements. We are currently exploring a number of ways of semi-automatically compiling these from machine readable dictionaries and the f-structure annotated corpus resources produced.

We expect that our approach can also feed into grammar development efforts. To be sure, because treebank grammars are large and flat, automatically annotated treebank grammars are less maintainable than the more compact, linguistically designed grammars which follow X′ design principles. However, as pointed out above, our approaches allow for a novel grammar design and processing architecture: given a treebank, a PCFG compiled from the treebank parses new text. For each input string, the (possibly n-) best parse trees are passed on to the annotation interpreters which annotate or rewrite the parse trees and induce f-structures. We consider this a promising new approach to large scale and corpus based grammar developement with applications in areas such as information retrieval.

5. SUMMARY

We have presented two automatic f-structure annotation methods for treebanks and grammars. The approaches make use of a corpus-based strategy that takes disambiguated tree structures as input, and combine them with traditional rule based techniques in the form of (linguistically motivated) annotation principles. The principles are used to automatically enrich treebanks or extracted treebank grammars with higher-level functional information not present the original corpora. Automatic annotation holds considerable potential in curtailing development costs and opens up the possibility of tackling large fragments. To date, our experiments are relatively small-scale. Still, we have presented an grammar development and treebank annotation methodology which is data-driven, semi-automatic, reuses existing resources and covers real text. We found the LFG framework very conducive to our experiments. We do believe, however, that the methods can be generalised, and we intend to apply them in an HPSG scenario and to semantic representation based annotations. Our second method could be applied to work in a TAG scenario (see also the closely related work in Neumann(1998,2000) and Neumann and Flickinger(1999)). In our work to date, contrary to an often perceived view, we have found that treebanks encode highly useful linguistic information, albeit often in rather flat representations.

Our methods encourage work in the best linguistic tradition as (i) they are concerned with real language and (ii) they enforce generalisations in the form of annotation principles. Our methods factor out information spread over CFG rules into modular and general principles. What is new in our approach is that (i) the principles state partial and underspecified correspondences between c- and f-structure configurations and (ii) they are applied to flat and noisy treebank representations that do not follow general X′ design principles. Our experiments show how theoretical work and ideas on principles can translate into grammar development for real texts. In this sense the methods may help to bridge the often perceived gap between theoretically motivated views of grammar as a set of principles vs. grammars for "real" text.

Acknowledgements

The authors wish to thank the members of the Pargram group, in particular Ron Kaplan, Mary Dalrymple and John Maxwell as well as Joan Bresnan, for helpful discussions and feedback.

Appendix: Example of an Automatically Generated F-Structure

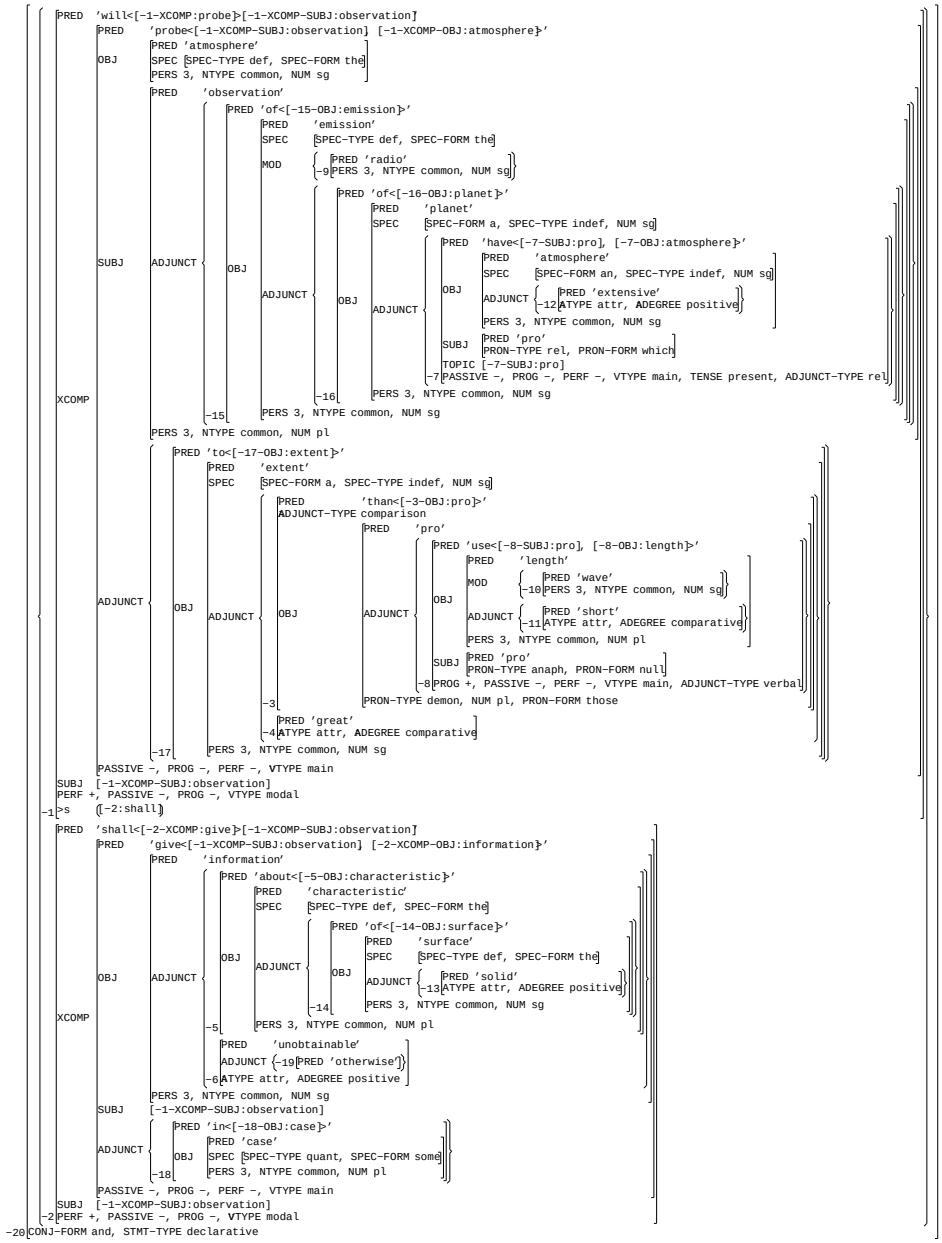


Figure 1.A.1 F–structure for: “Observations of the radio emission of a planet which has an extensive atmosphere will probe the atmosphere to a greater extent than those using shorter wave lengths and should in some cases give otherwise unobtainable information about the characteristics of the solid surface.”

Notes

1. For expository purposes, these are slightly simplified principles from our annotation grammar.

2. The annotation principles have to take into consideration that, in many cases, the representation of coordination in treebank rules is overly flat.

3. The annotation process itself is fast: in our experiments the interpreter annotates about 40 treebank CFG rules per second (Sparc 400Mhz).

4. It is also closely related to principle-based grammar description in HPSG.

5. There are obligatory ($\Rightarrow$) and optional ($?\Rightarrow$) rewrite rules.

6. Disjunctive templates encode alternative rewrite rules, and can be unioned (by logical and `&&`) with annotation rules. While this does still involve disjunctive processing, the rules can be stated in a generalised, compact way.

7. We require B and C to be distinct variables through inequality constraints (in curly brackets).

8. Templates, grammars and f-structures generated are available at: `http://www.compapp.dcu.ie/~away/Treebank/treebank.html`.

9. In earlier work (Sadler et al. 2000) we were able to report precision and recall results of 93.4% and 91.6%, respectively. These results were achieved with our previous Prolog list constraint based formulation of annotation principles and the corresponding interpreter. In moving to the new regular expression based format presented in this paper we have not yet been able to undertake the all important fine tuning of principles required to achieve precision and recall results above 90%. Fine tuning is currently under way and we hope to report the final results (rather than the preliminary first pass results) before the present volume goes to print.

References

S. Abney. 1997. Stochastic Attribute-Value Grammars. In *Computational Linguistics*, 23(4), pages 597–618.

R. Bod and R. Kaplan. 1998. A probabilistic corpus-driven model for lexical-functional analysis. In *Proceedings of COLING/ACL'98*, pages 145–151.

J. Bresnan. 2000. Lexical-functional syntax. Blackwells Publishers, Oxford.

E. Charniak. 1993. Statistical Language Learning. MIT Press, Cambridge MA.

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

M. Dalrymple, R.M Kaplan, J.T. Maxwell III, and A. Zaenen, editors. 1995. *Formal Issues in Lexical-Functional Grammar*. CSLI Lecture Notes, No. 47. CSLI Publications.

M. Dalrymple. 2000. *Lexical-functional grammar*. Manuscript, Xerox PARC.

A. Frank. 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.

A. Frank. 2000. Automatic F-Structure Annotation of Treebank Trees. in: M. Butt and T.H. King editors, *Proceedings of the LFG00 Conference*, 19 - 20 July 2000, University of California at Berkeley, CSLI Online Publications, Stanford, CA, `http://www-csli.stanford.edu/ publications/`.

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

P.-K. Halvorsen and R. Kaplan. 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, Stanford, CA.

R.M. Kaplan and J. Bresnan, 1982. *Lexical Functional Grammar*, pages 173–281. MIT Press, Cambridge, Mass.

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

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

G. Leech and R. Garside, 1991. *Running a Grammar Factory: On the Compilation of Parsed Corpora, or 'Treebanks'*, pages 15–32. Mouton de Gruyter, Berlin.

G. Neumann. 2000. Automatic Extraction of Stochastic Lexicalized Tree Grammars from Treebanks. In A. Abeille (ed) Treebanks: building and using syntactically annotated corpora. Kluwer

G. Neumann and D. Flickinger. 1998. Learning Stochastic Lexicalized Tree Grammars from HPSG. DFKI Technical Report, Saarbrücken, 1999.

G. Neumann. 2000. Automatic Extraction of Stochastic Lexicalized Tree Grammars from Treebanks. In Proceedings of the 4th workshop on tree-adjoining grammars and related frameworks. Philadelphia, PA, USA, August, 1998.

C. Pollard and I. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago, Illinois.

L. Sadler, J. van Genabith and A. Way. 2000. Automatic F-Structure Annotation from the AP Treebank In *Proceedings of the LFG 2000 Conference*, The University of California at Berkeley, 19 July - 20 July 2000, CSLI Publications, Stanford, CA, `http://www-csli.stanford.edu/ publications/`

G. Sampson, 1993. *The Susanne Corpus*. Release 2.

J. van Genabith, L. Sadler, and A. Way. 1999a. Data-driven compilation of LFG semantic forms. In *EACL'99 Workshop on Linguistically Interpreted Corpora (LINC-99), Bergen, Norway, June 12th*, pages 69–76.

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

J. van Genabith, A. Way, and L. Sadler. 1999c. Semi-Automatic Generation of f-Structures from Tree Banks. In M. Butt and T.H. King, editors, *Proceedings of the LFG99 Conference*, Manchester University, 19-21 July, CSLI Online

Publications, Stanford, CA. http://www-csli.stanford.edu/publications/.