Design and Evaluation of the Linguistic Basis of an Automatic F-Structure Annotation Algorithm for the Penn-II Treebank

Mairéad McCarthy
M Sc in Computer Applications,
School of Computing,
Dublin City University

Supervisors
Dr Andy Way
Dr Josef van Genabith

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of M.Sc. in Computer Applications is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Mairéad McCarthy

Mairéad McCarthy

Date: August 30, 2003

Contents

List of Figures

List of Tables

Abstract

In this thesis, we describe the design and evaluation of the linguistic basis of an automatic f-structure annotation algorithm for the Wall Street Journal (WSJ) section of the Penn-II Treebank, which consists of more than 1,000,000 words, tagged for part-of-speech information, in about 50,000 sentences and trees. We discuss the background and some of the main principles of Lexical-Functional Grammar (LFG), which is the theory of language used to represent the predicate-argument-modifier structure of a sentence by us in our application.

We then present the guidelines for the tagging of the Penn-II Treebank, followed by a description of how the linguistics of the Penn-II Treebank relate to LFG. The automatic annotation of such Treebank grammars is difficult as annotation rules often need to identify sub-sequences in the right-hand-sides of (often) flat Treebank rules as they explicitly encode head, complement and modifier relations. The algorithm we have developed is designed to handle these flat grammar rules. We describe the methodology used to encode the linguistic generalisations needed to annotate Treebank resources with LFG f-structure information, which, unlike previous approaches to this problem, scales up to the size of the WSJ section of the Penn-II Treebank.

Finally, we present and assess a number of automatic evaluation methodologies for assessing the effectiveness of the techniques we have developed. We first employ a quantitative evaluation, whcih measures the coverage of our annotation algorithm with respect to rule types and tokens, and calculates the degree of fragmentation of the automatically generated f-structure. Secondly, we present a qualitative evaluation, which measures the quality of the f-structures produced against a manually constructed 'gold standard' set of f-structures. Finally, we summarise our work to date, and outline possibilities for further work.

Chapter 1

Introduction

1.1 Overview

In this thesis, we describe the design and evaluation of the linguistic basis of an automatic f-structure annotation algorithm, using Lexical-Functional Grammar (LFG), for the Wall Street Journal (WSJ) section of the Penn-II Treebank, which consists of more than 1,000,000 words, tagged for part-of-speech information, in about 50,000 sentences and trees.

The algorithm automatically generates a large-scale LFG grammar. The Penn-II Treebank is automatically annotated with LFG f-structure equations, from which we can automatically extract a large-coverage unification grammar, thereby generating a new linguistic resource from the Treebank.[1] Such large-coverage unification grammars are difficult to obtain, and extremely time-consuming and expensive to construct by hand.

The background and motivations for this research, together with an outline of the structure of the thesis, are provided in this chapter.

1.2 Background

1.2.1 Lexical-Functional Grammar

We have chosen to represent the basic predicate-argument-modifier structure of a sentence using LFG f-structures. LFG ([Kaplan and Bresnan, 1982],

[1]This author's work mainly entailed providing a specification for the treatment of the linguistic information, and the implementation of the algorithm was undertaken by [Cahill, forthcoming].

[Bresnan, 2001], [Falk, 2001], [Dalrymple, 2002]) is a non-transformational, constraint-based theory of language, which evolved from research within the transformational framework [Bresnan, 1978] and from earlier computational and psycholinguistic investigations. LFG is "mathematically well defined and simple, (and is, therefore,) easy to implement." ([Bresnan, 2001], p.iii). It is, therefore, particularly well suited to the construction of automatic f-structure annotation principles as contained in our algorithm. LFG consists of two basic levels of representation, c(constituent)- and f(functional)- structure. C-structure is the organisation of overt surface phrasal syntactic representation ([Dalrymple, 2002], p. 45). It represents surface grammatical configurations such as word order, most often shown in a conventional phrase structure tree format. F-structure represents abstract syntactic functions such as subject, object, predicate, etc. These grammatical functions are represented as a set of ordered pairs in the form of an attribute-value matrix, which is a hierarchy of attribute-value pairs, and are useful for abstracting away from c-structure, which can vary greatly across languages.

1.2.2 The Penn-II Treebank

The Penn-II Treebank[2] is a large corpus of newspaper text, consisting of extracts from the WSJ corpus, the ATIS corpus (a database of flight detail queries), and the Brown corpus. The Penn-II Treebank contains, in total, over 4.5 million words of American English. A treebank is a database of linguistic trees, which show the syntactic structure of sentences. Many Natural Language Processing (NLP) applications require high quality training corpora, which have to provide tree structures with meaning representations. Treebank corpora which have been syntactically annotated enable researchers to pursue projects that would otherwise be a lot more demanding. Examples of such projects are parsers which use machine learning—treebanks are required as training resources for probabilistic unification grammars and data-driven parsing approaches [Bod and Kaplan, 1998]—and automatic grammar development [Frank et al., 2001].

The WSJ section of the Penn-II Treebank consists of more than 1,000,000 words, tagged for part-of-speech information, in about 50,000 sentences and trees, but contains no information regarding the meaning of these sentences/trees. Treebank grammars typically involve large sets of lexical tags and

[2]Available from `http://www.ldc.upenn.edu/ldc/online/treebank/`

non-lexical categories, as syntactic information tends to be encoded in monadic category symbols. They often feature flat rules in trees that do not express linguistic generalisations. The WSJ section of the Penn-II Treebank contains a very large Context-Free Grammar (CFG) rule base—17,000 rules, about 1 rule type for every three sentences/trees.

1.2.3 Automatic Annotation

Automatic annotation of these treebank grammars is difficult as annotation rules often need to identify subsequences in the right-hand-sides of flat treebank rules as they explicitly encode head, complement and modifier relations.

[Frank, 2000], [Sadler et al., 2000]) and [Frank et al, 2003] have developed methods for automatically annotating treebank resources with LFG f-structure information, which approximate to basic predicate-argument structures. They are fine-grained and produce detailed f-structure representations, but have only been developed to date using the Susanne Corpus[3], which comprises an approximately 130,000-word subset of the Brown Corpus of American English, and the publicly available subset of the AP Treebank[4], which consists of 100 sentences of newswire reports, with a CFG rule base of about 500 CFG rules. [Sadler et al., 2000] use a regular expression-based, indirect automatic f-structure methodology. [Frank, 2000] developed a method that is in many ways a generalisation of the regular expression-based annotation method. This method can access arbitrary tree fragments and can be order-dependent or order-independent (cf. [Frank et al, 2003] for more detail on these approaches), whereas the algorithm described in this thesis is order-dependent and can only access local subtrees of depth one, i.e. CFG rules. These methods, however, have not yet been tested to scale up to larger corpora, of the size of the Penn-II Treebank.

[Forst, 2003] reports on the TIGER treebank, which consists of 36,000 syntactically annotated German newspaper sentences, in which dependency information is expressed explicitly in the edge labels, so it does not need to be annotated with functional equations, which are used in LFG as an intermediary between c- and f-structure, cf. Section 2.3.3. This would seem to make it easier to convert the trees into f-structures than from the Penn-II Treebank, as a great deal of information is already explicitly marked in the TIGER graphs.

[3] Available from `http://www.grsampson.net/RSue.html`

[4] Available from `http://www.comp.lancs.ac.uk/computing/research/ucrel/corpora.html`

Previous research by ([Cahill et al., 2002a], [Cahill et al., 2002b], [Cahill et al., 2002c]) led to the design of a coarse-grained automatic annotation algorithm, which produced 'proto' f-structures, encoding basic predicate-argument-modifier structures (although some f-structures were partial or unconnected, associating trees with two or more unconnected f-structure fragments). These proto f-structures did not use the traces and empty productions provided by the Treebank to encode 'moved' or 'extraposed' material (long distance dependencies), to reflect them as corresponding re-entrancies in the f-structure. This thesis expands previous work to develop an automatic f-structure annotation algorithm which produces more detailed f-structure representations, encoding long distance dependencies such as topicalisation and wh-movement, as well as marking passive constructions.

1.3 Motivation

Significant, rapid progress can be made in both text and spoken language understanding by investigating those phenomena that occur most often in naturally occurring unconstrained material and by attempting to automatically extract information about language from very large corpora. Annotated corpora serve as an important research tool for investigators in NLP, speech recognition, and integrated spoken language systems, as well as theoretical linguistics.

The automatic f-structure annotation algorithm described in this thesis generates a large-scale LFG grammar. The resources produced are useful as a linguistic resource. Languages may differ with respect to surface representation, but they may encode the same (or similar) grammatical functions—hence the need for f-structure which abstracts away from differences at sentence level.

The c- and f-structure pairs generated can be used for parsing, [Riezler et al., 2002], i.e. the determination of syntactic structure, which is an important step in natural language processing as syntactic structure determines semantic interpretation in the form of predicate-argument structure, dependency structure or logical form. We can automatically extract large-scale, wide-coverage LFG grammars from the Penn-II Treebank resource. These grammars are then used to parse text into LFG f-structures. Data-Oriented Parsing (DOP) [Bod, 1998], comprises an experience-based approach to parsing in that new input is analysed by referencing grammatical analyses of previous language experiences. However, context free representations limit DOP models. When

LFG representations, which are context-sensitive, are allied to techniques of DOP, the resulting model is robust and capable of handling linguistic phenomena which appear below surface level [Bod and Kaplan, 1998].

The resources produced can also be used for machine translation. The Data-Oriented Translation (DOT) framework provides us with a powerful, probabilistic model of translation which is sensitive to context. Again, context free representations preclude the handling of certain linguistic phenomena. ([Way, 1999], [Way, 2003]) proposes the use of LFG-DOP as the basis for an innovative MT system which describes the translation relation as required and overcomes the restrictions of the DOT translation model. Both the LFG-DOP and LFG-DOT models require the c- and f-structure pairs generated from the LFG grammar and the Treebank.

LFG is the formalism used in our automatic f-structure annotation algorithm. However, the methodology described in this thesis can be also be applied to HPSG typed feature-structures [Pollard and Sag, 1994], dependency structures, or Quasi-Logical Form (QLF) [Liakata and Pulman, 2002] annotations.

1.4 Structure of the Thesis

Chapter 2 outlines the background and some of the main principles of LFG, which is the theory of language used in our methodology to represent the predicate-argument-modifier structure of a sentence. This chapter gives a brief introduction to LFG, and some of the main principles of the theory, such as the Lexical Integrity Principle and the Principle of Economy of Expression, referring to the three recent major works on LFG by [Bresnan, 2001], [Falk, 2001], and [Dalrymple, 2002]. An introduction to c- and f-structure is given, with a clear step-by-step example of how to construct an f-structure from a simple English sentence. This is followed by sections on grammaticality checks, Lexical Mapping Theory, and an exposition of how LFG deals with linguistic phenomena such as passivisation, coordination and long distance dependencies, including topicalisation, wh-questions, and relative clauses.

Chapter 3 gives a review of the linguistic design embodied in the Penn-II Treebank as detailed in [Santorini, 1991], [Marcus et al., 1994], and [Bies et al., 1995]. The tagset and notation used in coding the Penn-II Treebank, the basic clause structure, subordinate clauses, and wh-phrases are described in the following sections. Coordinate structures, the modification of

noun phrases and the handling of null elements in the Penn-II Treebank are then described in individual sections as they can pose particular problems for automatic annotation, due to the often flat treebank analyses provided. Selected examples and references are made to our annotation algorithm, where relevant, in order to illustrate some examples of mistagging within the Penn-II Treebank, and to give some indication of the problems this causes to our automatic f-structure annotation algorithm, which is described in detail in Chapter 4. The linguistic assumptions of the Penn-II Treebank in relation to LFG theory are discussed at the end of the chapter, followed by a summary of the contents of the chapter.

Chapter 4 describes the methodology used to encode the linguistic generalisations needed to annotate treebank resources with LFG f-structure information, which can scale to the size of the WSJ section of the Penn-II Treebank. The automatic f-structure annotation algorithm is implemented in Java, and takes the form of a recursive procedure, which traverses the Penn-II treebank top-down and annotates the nodes with f-structure information. The algorithm is designed in terms of four main sequential components: Left/Right context, Coordination, Traces, and 'Catch-all and Clean-up' principles. The four components support a clean annotation methodology and documentation of the linguistic basis of the annotation principles to facilitate updating, maintenance and re-use of the annotation procedure and the linguistic information encoded within. In our initial research ([Cahill et al., 2002a], [Cahill et al., 2002b], [Cahill et al., 2002c]), we designed a coarse-grained automatic annotation algorithm, which produced 'proto' f-structures, which encoded basic predicate-argument-modifier structure (with some f-structures which were partial or unconnected, i.e. the trees were associated with two or more unconnected f-structure fragments), but did not use the traces and empty productions provided by the Treebank to encode 'moved' or 'extraposed' material (long distance dependencies) to reflect them as corresponding re-entrancies in the f-structure. We have since refined the algorithm to produce more detailed f-structure representations, encoding re-entrancies for topicalisation and wh-movement, as well as annotating passive, and it is this improved methodology which is described in this chapter.

Chapter 5 presents a number of automatic evaluation methodologies for assessing the effectiveness of the techniques we have developed. We first employ a quantitative evaluation, which measures the coverage of our annotation algorithm with respect to rule types and tokens, and calculates the degree of fragmentation of the automatically generated f-structure, i.e. the number of

unconnected f-structures produced by each sentence. It is necessary for the annotation algorithm to be robust and encode linguistic generalisations while maintaining quality. Secondly, we present a qualitative evaluation, which measures the quality of the f-structure annotations automatically generated by the annotation algorithm against a manually constructed 'gold standard' set of f-structures produced by this author. We first use the EVALB[5] test on the annotated trees to compare the automatically generated annotations on the trees against the gold standard annotated trees, and, as a further measure of comparison, we calculate precision and recall on flat sets of term-based descriptions of the f-structures generated. This is done for all annotations. Then, as a further measure, coordination is evaluated in isolation, using the same qualitative measures, for a more in-depth analysis of the often complex coordinate structures.

Chapter 6 concludes the thesis, summarising our work to date, and analysing the progress made. Avenues for future work are also outlined.

Appendices containing the lexical macros and the annotation matrices are provided following the conclusion.

[5] Available from `http://www.cs.nyu.edu/cs/projects/proteus/evalb/`

Chapter 2

Lexical-Functional Grammar

2.1 Introduction

Lexical-Functional Grammar (LFG: [Kaplan and Bresnan, 1982], [Bresnan, 2001]) is a non-transformational, constraint-based theory of language, which evolved from research within the transformational framework [Bresnan, 1978] and from earlier computational and psycholinguistic investigations. LFG consists of two (basic) levels of representation, c(constituent)- and f(functional)-structure. This chapter gives a brief introduction to the background of LFG, and some of the main principles of the theory, such as the Lexical Integrity Principle (LIP) and the Principle of Economy of Expression, referring to the three recent major works on LFG by [Bresnan, 2001], [Falk, 2001], [Dalrymple, 2002]. An introduction to c- and f-structure is given, with a clear step-by-step example of how to construct an f-structure from a simple English sentence. This is followed by sections on grammaticality checks, Lexical Mapping Theory, and an exposition of how LFG deals with linguistic phenomena such as passivisation, coordination and long distance dependencies, including topicalisation, wh-questions, and relative clauses, in terms of functional uncertainty equations.

2.2 LFG Background

The origins of the theory of LFG are in the field of generative linguistics. However, LFG differs from another well known model of syntax, 'transformational grammar', in the sense that LFG is a non-transformational, constraint-based theory of language. This means that there is no distinction between 'deep' (underlying) structure, and surface structure (there is just one level of constituent structure in LFG, cf. Section 2.3.1), and no transformations need be applied to relate deep and surface structure. The *constraint-based* aspect of the theory means that grammaticality is determined by the satisfaction of simultaneous static constraints.

LFG is a *lexical* theory, one in which words in the lexicon play a major role. In transformational grammar, the role of the lexicon is seen as relatively limited. In LFG, however, the lexical elements are as important, if not more so, as syntactic rules in expressing grammatical information, and relations among different types of verbs, such as complementation patterns (whether a verb is intransitive, transitive, or ditransitive), are stated in the lexicon rather than by means of transformations.

The *functional* aspect of the theory states that abstract grammatical functions can be the same, or similar, across languages, despite the obvious differences in phrase structure configurations. These abstract grammatical functions are primitives of the theory, and are not defined in terms of phrase structure configurations or semantic or argument structure relations. The abstract syntactic structure of languages, LFG claims, obeys universal principles of functional organisation.

Unlike other theories, LFG was designed to be implementable in computational systems from the beginning, and is therefore particularly well suited to the construction of automatic f-structure annotation principles as contained in our algorithm (cf. Section 4.3).

2.2.1 Lexical Integrity Principle

The *lexical hypothesis* was introduced as early as 1970 by [Chomsky, 1970], and developed by [Bresnan, 1978], who claimed that syntactic transformations do not play a role in word formation.

This led to the introduction of the **Lexical Integrity Principle (LIP)** by [Lapointe, 1980], who stated that "No syntactic rule can refer to elements of

morphological structure." This principle was explicit in LFG from the beginning ([Dalrymple, 2002], p.84), and states that syntax cannot see the internal structure of words ([Falk, 2001], p.4). The component of grammar responsible for phrase and sentence structure is distinct from the one responsible for word structure. Lexical integrity, as understood in LFG, is limited to c-structure. The LIP states that words are the atoms out of which 'syntactic structure' is built (in LFG there are two levels of syntactic structure: c-structure and f-structure, which are described in more detail in Section 2.3.1 and Section 2.3.2 respectively), where morphologically complete words are the leaves of the c-structure tree and each leaf corresponds to one and only one c-structure node ([Bresnan, 2001], p.105f.). Therefore, the accuracy of the tagging of the Penn-II Treebank (cf. Section 3.2.1) is very important for the accuracy of our lexical macros (cf. Appendix A), which are used to encode lexical information in our automatic f-structure annotation algorithm (cf. Section 4.2.2).

Lexical integrity within LFG keeps the internal structural formation of words invisible to c-structure principles and therefore independent of the structural formation of phrases, but it allows the f-structures specified by the words to unify with the f-structures of the syntactic contexts. While words and syntactic phrases are different types of forms of expression in c-structure, they may carry the same types of information in f-structure, i.e. words and phrases may be functionally equivalent.

2.2.2 Principle of Economy of Expression

C-structure is subject to the **Principle of Economy of Expression**, which rates words to be of higher importance than syntactic phrase structure nodes, and requires the choice of the simplest and smallest phrase structure tree that allows for the satisfaction of f-structural constraints, while expressing the intended meaning. ([Falk, 2001], p.34) states that "all syntactic phrase structure nodes are optional and not used unless required to create a well-formed f-structure or to add semantic content". The same principle is stated by ([Bresnan, 2001], p.103) as "all syntactic phrase structure nodes are optional and not used unless required by independent principles (completeness and coherence)", which are discussed in detail in Section 2.4. The principle refers to non-terminal nodes which do not immediately dominate a lexical element. It limits the use of empty elements in c-structure, and disallows the use of syntactic phrase structure nodes that provide only redundant information. However, empty nodes appear in the

Penn-II Treebank and can be used to our advantage in our automatic annotation algorithm to provide more information in the f-structures generated. Null elements in the Penn-II Treebank are described in Section 3.8.

2.3 Grammatical Representations

The basic architecture of LFG crucially separates the three notions of structure: (i) **c**(constituent)- structure, (ii) structural description: **f**(functional)- structure, and (iii) structural correspondence: **f**-description. C-structures are formally quite different from f-structures, in that c-structures are defined in terms of syntactic categories, terminal strings, and their dominance and precedence relations, whereas f-structures are composed of grammatical function names, semantic forms, and feature symbols.

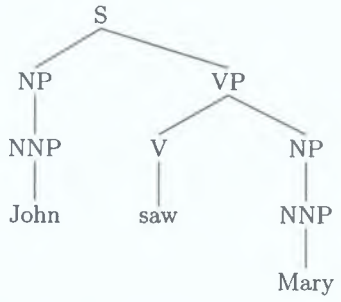
2.3.1 C-structure

C-structure is an organisation of words that make up a sentence into successively larger units, where each unit (**constituent**) belongs to a category. It represents surface grammatical configurations such as word order, most often shown in a conventional phrase structure tree format.

(1) John saw Mary.

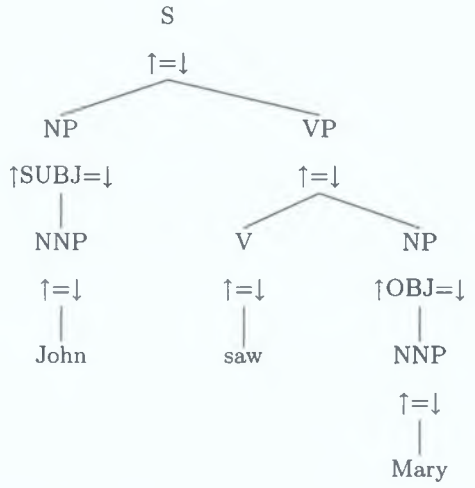
A simple English sentence such as (1) is associated with the c-structure tree shown in (2):

(2)



Functional annotations are then inserted into the tree to produce an LFG-annotated c-structure tree as shown in (3):

(3)



The '↑' meta-variable refers to the functional projection of the mother c-structure node, and the '↓' meta-variable refers to the functional projection of the annotated local node itself. The '↑SUBJ=↓' annotation in (3) indicates that the f-structure for S has a subject attribute whose value consists of the f-structure associated with the NP node itself. The '↑=↓' annotation on the VP node indicates that the verb phrase is the head of the sentence, and the '↑OBJ=↓' annotation on the second NP node indicates that it is the object of the VP, and therefore the object of the entire sentence.

The c-structure in (3) can only be derived by c-structure rules with functional annotations such as (4):

(4)

S	→	NP	VP
		↑SUBJ=↓	↑=↓
NP	→	NNP	
		↑=↓	
VP	→	V	NP
		↑=↓	↑OBJ=↓

Lexical entries also use the '↑' meta-variable to encode information about the f-structures of the preterminal nodes that immediately dominate them. A partial lexical entry for the subject in this example, *John*, is shown in (5):

(5)
NNP → John
(↑PRED) = 'John'
(↑NUM) = SG
(↑PERS) = 3

This encodes information stating that John is a 3rd person singular proper noun. This information is encoded in our automatic f-structure annotation algorithm using lexical macros, which are described in detail in Section 4.2.2.

2.3.1.1 Xbar Theory

Xbar (X′) theory is a theory of c-structure. ([Falk, 2001], p.35) states that groups of words form constituents, or phrases, which can be identified by their ability to occur in different places in the sentence. Lexical categories, the categories of words that carry meaning, mainly N(noun), V(verb), A(adjective), and P(preposition), are the heads of these phrases.

The basic principle of X′ theory states that the lexical category X is related to the projections of the phrase XP. For example, the category NP is said to be the phrasal projection of N. An example of N′ is given in (6):

(6) a. The love of my life and mother of my children would never do such a thing.

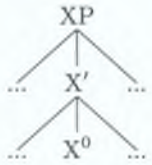
b. The museum displayed no painting by Picasso or drawing by Da Vinci.

Note that neither *mother of my children* nor *drawing by Da Vinci* are full NPs. They cannot occur without determiners. It was deemed necessary to have an intermediary category, as N′ is neither a full NP nor is it a lexical item.

In LFG, a lexical head and its phrasal projection correspond to the same piece of f-structure, so their features unify. Constituents that function as arguments are sisters of the head, i.e. in complement position, while adjuncts are adjoined to the phrasal node. Different categories have different properties, e.g. adjectives only modify noun phrases; other categories, such as verb phrases, are modified by adverbs and prepositional phrases.

The **endocentric** property of X′ theory states that all phrases have heads of the same category (e.g. NP headed by N), as shown in the general schema in (7):

(7)



XP is the maximal projection of the category X. A phrase of category XP dominates a non-maximal projection of category X′, which in turn dominates a lexical item of category X^0, for any lexical or functional category X^0, as shown in (7). There is usually some consistency in the order of the heads and their complements, e.g. in English, the head precedes its complement(s) ([Falk, 2001], p.36). (This can be made use of when annotating rules, cf. Section 4.2.1, for details on the head rules used in our automatic annotation algorithm.)

It is not always correct to say that all phrasal categories are projected from heads, or lexical items, i.e. not all categories are endocentric. A phrasal category with no c-structure, or lexical head, is said to be **lexocentric**. S is assumed to be the only exocentric category, and can dominate a series of either lexical or phrasal constituents, containing a predicate with any, or all of its arguments ([Dalrymple, 2002], p.64), including the subject. It is not an X′-theoretic category and does not obey the X′-theoretic generalisations governing the relation between c-structure positions and f-structure functions. This enriches the theory of c-structure. ([Falk, 2001], p.50) states that languages may have combinations of *endocentric* (grammatical functions encoded in c-structure configurations) and *lexocentric* (grammatical functions encoded by lexical means, such as case and agreement morphology) structures.

2.3.2 F-structure

F-structure represents abstract syntactic functions such as subject, object, predicate, etc. These grammatical functions are represented as a set of ordered pairs in the form of an attribute-value matrix, a hierarchy of attribute-value pairs. An attribute is a grammatical feature or function name, and the name precedes the value. The f-structure for the sentence in (1) is shown in (8):

(8)
$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'John'} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{'Mary'} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRED} & \text{'see}\langle(\uparrow\text{SUBJ})(\uparrow\text{OBJ})\rangle\text{'} \\ \text{TENSE} & \text{PAST} \end{bmatrix}$$

In this structure, the TENSE attribute has the simple symbol value PAST; pairs with this kind of value represent syntactic features. Grammatical functions have subsidiary f-structure values, as illustrated by the subject and object functions. The attributes NUM(ber) and PERS(on) mark embedded features with symbol values SG and 3, respectively. The f-structure indicates that *John* is the grammatical subject, *Mary* is the grammatical object, and *see*$\langle(\uparrow\text{SUBJ})(\uparrow\text{OBJ})\rangle$ is a predicate-argument expression containing the predicate name, *see* followed by an argument list specification enclosed in angle brackets. It contains a subcategorisation list, i.e. the verb *see* subcategorises for a subject and an object. Predicates specify a list of the governable grammatical functions that they require in this way. The verb *see* lexically specifies that it requires both a subject and an object, and the grammatical function of each argument is determined by the phrase structure position.

2.3.2.1 Grammatical Functions

Grammatical functions are encoded in different ways in different languages. English makes use of a *configurational encoding* ([Kaplan and Bresnan, 1982], p.296f.), in that phrase structure positions are associated with grammatical functions by means of annotations on the phrase structure rules, e.g. a noun phrase occurring to the right of a verb in a verb phrase is normally annotated as an object '$\uparrow$OBJ=$\downarrow$'. In English, subject, object, and oblique arguments receive annotations in this way. C-structure rules such as those given in (4) annotate the head of the phrase as '$\uparrow$=$\downarrow$', in order to ensure that the phrase and its head correspond to the same f-structure.

Grammatical functions (subject, object, etc) are useful for abstracting away from c-structure, which can vary greatly across languages. ([Bresnan, 2001], p.109) states that subject and object functions have no single universal structural form. They are **core functions**, formally distinguished from **non-core functions**, such as obliques, predicate complements, and adjuncts.

2.3.2.2 Open vs Closed Functions

Open functions are generally non-finite complements. However, there are also occurences where an adjective phrase subcategorises for a subject, e.g. in a sentence such as *John is old.* In the Penn-II Treebank data, such occurrences of adjectival phrases are labelled as ADJP-PRD, and will then be annotated as XCOMPs by our automatic f-structure annotation algorithm (cf. Section 4.3).

XCOMP is an 'open' function, in which the embedded subject must be controlled by an argument in the root (matrix) clause. This exemplifies **functional control**. Functional control verbs require an XCOMP as an argument. The subject of the so-called "raising" verb *functionally controls* the subject of the subordinate XCOMP ([Dalrymple, 2002], p.314). The 'raised' argument, the subject in (12), *the driver*, is not a semantic argument of the raising verb, *seem.* The subject of *seem* is required to be the same f-structure as the subject of the XCOMP. In (12), the *driver* and the 'starter of the tractor' must be one and the same person, as shown in the f-structure in (9):

(9)
$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{SPEC} & \text{'the'} \\ \text{PRED} & \text{'driver'} \end{bmatrix} \\ \text{XCOMP} & \begin{bmatrix} \text{SUBJ} & [\ ] \\ \text{OBJ} & \begin{bmatrix} \text{SPEC} & \text{'the'} \\ \text{PRED} & \text{'tractor'} \end{bmatrix} \\ \text{PRED} & \text{'start}\langle(\uparrow\text{SUBJ})(\uparrow\text{OBJ})\rangle\text{'} \end{bmatrix} \\ \text{PRED} & \text{'seem}\langle(\uparrow\text{XCOMP})(\uparrow\text{SUBJ})\rangle\text{'} \\ \text{TENSE} & \text{PRES} \end{bmatrix}$$

Closed functions generally have finite complements. COMP is a 'closed' function, which either displays an overt subject of its own, or can have an anaphorically controlled PRO subject. The subject of COMP does not have to be identical with an argument of the matrix clause. This exemplifies **anaphoric control**. The subordinate complement in anaphoric control is the closed function COMP. The nature of control is different to functional control, as the subject of the COMP is syntactically independent from the matrix controller. The sentential complement *that she has started the tractor* bears the grammatical function COMP in (10):

(10) The driver thinks [(that) she has started the tractor].

The verb *think* subcategorises for two arguments: a subject, *the driver*,

and an embedded *that*-clause. As shown in (11), *the driver* and *she* are not connected in the f-structure, and therefore need not be the same person.

(11)
$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{SPEC} & \text{'the'} \\ \text{PRED} & \text{'driver'} \end{bmatrix} \\ \text{COMP} & \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'she'} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{SPEC} & \text{'the'} \\ \text{PRED} & \text{'tractor'} \end{bmatrix} \\ \text{PRED} & \text{'start}\langle(\uparrow\text{SUBJ})(\uparrow\text{OBJ})\rangle\text{'} \\ \text{that} & + \end{bmatrix} \\ \text{PRED} & \text{'think}\langle(\uparrow\text{SUBJ})(\uparrow\text{COMP})\rangle\text{'} \\ \text{TENSE} & \text{PRES} \end{bmatrix}$$

However, it is also possible that *the driver* and *she* are the same person, in which case they would be co-indexed, illustrated by a re-entrancy in the f-structure (the SUBJ inside the COMP would be linked to the main SUBJ).

Arguments of a predicate are distinguished from modifiers in that arguments are *governable grammatical functions* ([Dalrymple, 2002], p.11)—they are subcategorised for, or *governed* by the predicate, while modifiers are not governed, but rather modify the phrase with which they appear. These functions are not defined in terms of phrase structure configurations or in terms of semantic or argument structure relations.

In LFG, the standard grammatical functions are categorised as follows:

- **Core arguments**: SUBJ, OBJ and the family of thematically restricted objects OBJ_θ.

  These arguments are more strictly *grammatical functions* than the non-core arguments. Core arguments in English have canonical c-structure positions which can be occupied only by NPs/DPs, and are specified by the predicate of the verb, as in the entry for *see* in the f-structure in (8). The attribute SUBJ, which in English fills the first argument position of the verb, is required for all verbs. OBJ is required for transitive verbs, OBJ_θ is associated with a particular semantic role, such as benefactor (tagged -BNF in the Penn-II Treebank, cf. Section 3.2.3, for more detail).[1]

- **Non-core arguments**: COMP, XCOMP and the family of oblique functions OBL_θ.

[1] More information on semantically restricted (SUBJ and OBJ) and unrestricted (OBJ_θ and OBL_θ) functions can be found in ([Dalrymple, 2002], pp.15–16).

Non-core functions are more closely tied to semantics than core functions. These arguments are generally expressed by other c-structure categories (OBL by PPs, COMPs and XCOMPs by VPs, APs, or CPs, etc.). COMP is the grammatical function of clausal complements. The sentential complement *that she has started the tractor* bears the grammatical function COMP in (10).

XCOMP is also a clausal function. The infinitive phrase *to start the tractor* in (12) bears the grammatical function XCOMP:

(12) The driver seems to start the tractor.

OBL_θ is a thematic role which is identified morphologically (by a preposition in English such as *to* in the example in (13)):

(13) The old man gave a present to his wife.

- **Non-argument functions**: ADJ, FOCUS, TOPIC.

  Adjuncts (ADJ) are modifiers, not governable grammatical functions, and therefore multiple instances of these are allowed without violating the uniqueness condition on f-structures (cf. Section 2.4.2). If they are not included, the sentence is still grammatically correct. In (14), *beautiful* and *old* are both members of the adjunct set:

  (14) The beautiful old woman likes the man.

  This adjunct set is included in the f-structure for the subject noun phrase, as shown in the partial f-structure in (15):

(15)
$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{SPEC} & \text{'the'} \\ \text{PRED} & \text{'woman'} \\ \text{ADJ} & \{\begin{bmatrix} \text{PRED 'old'} \end{bmatrix} \\ & \begin{bmatrix} \text{PRED 'beautiful'} \end{bmatrix}\} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{SPEC} & \text{'the'} \\ \text{PRED} & \text{'man'} \end{bmatrix} \\ \text{PRED} & \text{'like}\langle(\uparrow\text{SUBJ})(\uparrow\text{OBJ})\rangle\text{'} \\ \text{TENSE} & \text{PRES} \end{bmatrix}$$

Non-argument functions do not map directly to **a**(argument)- structure roles (cf. Section 2.5 for more detail on a-structure). ADJ binds to the

predicate itself ([Bresnan, 2001], p.111), while TOPIC and FOCUS bind to a-structure through other syntactic functions. English topicalisations—constructions in which a displaced constituent bears two roles, one of which is associated with some other position in the sentence—can have either FOCUS or TOPIC functions.[2] An example of a topicalised construction is given in (16):

(16) The child, the man saw.

In this construction, *the child* bears the TOPIC function, and also the OBJ function.[3]

The FOCUS function is a discourse function, often a wh-phrase linked to the OBJ in the f-structure, as in (17):

(17) What is the man doing?

In this construction, *what* bears the FOCUS function, and is linked to the OBJ function in the f-structure. (More information on the TOPIC and FOCUS functions can be found in Section 2.7.1 and Section 2.7.2 respectively.)

We also use the following grammatical functions in our annotations: RELMOD (to mark relative clauses), SPEC (to mark the specifier), PART (to mark the particle of the verb), POS (to mark the possessive), APP (to mark apposition as part of a set), and CONJ (to mark conjunction; also used as a set feature). *who John saw* is a relative clause in (18), *the* is the specifier (also known as the determiner), and *up* is the particle of the verb *look*.

(18) The man who John saw looked up the information in the library.

The phrase, *John's mother and father*, includes the possessive marker *'s*, and the conjunction, *and*. The function SPEC used in our algorithm is often included in LFG theory as a feature, 'DEF=+ / -'. The SPEC function which we use allows us to include additional information, such as ↑SPEC:DET=↓ or ↑SPEC:QUANT=↓, depending on whether the specifier consists of a determiner or a quantifier, which is useful in complex determiners, cf. (142), p.90. A list of all the grammatical functions used by the automatic annotation algorithm is given in Section 4.3.

[2]We also include a TOPICREL function in our automatic f-structure annotation algorithm to annotate topic in a relative clause, cf. Section 4.5.2.

[3]The functional uncertainty equation associated with (16) is given in (47), p.34.

2.3.2.3 Unification

A motivation for representing grammatical functions at a level distinct from c-structure is that features cannot always be associated with the c-structure constituents that they describe ([Falk, 2001], p.16f.). For example, a noun may be unspecified for number (as in the noun *sheep* in English, which may be singular or plural), and receive it from the lexical entry of the verb (i.e. from a different element of the c-structure). The merging of features is called **unification**, and it is a central concept of feature-based approaches to syntax. Because the grammatical features in LFG are represented independently of c-structure, there is no need for feature inheritance.

Unification is an operation that combines consistent feature structures into a new feature structure by taking the union of all the attribute-value pairs in the original structures ([Dalrymple, 2002], p.103). It accounts for the ungrammaticality of a sentence such as (19):

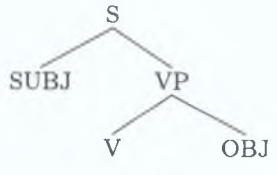
(19) * These boys likes girls.

The noun phrase, *these boys*, is plural, while the verb phrase is singular, due to the number value for the verb, *likes*. This causes a feature structure clash.

2.3.2.4 Differences in Word Order

Another reason for representing grammatical functions at a level distinct from c-structure is that grammatical functions cannot be universally dependent on c-structure. English, which is a configurational language, characterises subject and object in c-structure (a VP constituent distinguishes subject from object), as in (20):

(20)



However, this is not a universal distinction, hence the need for f-structure, which abstracts away from differences at sentence level. Languages may differ with respect to surface representation, but they may encode the same (or very similar) grammatical functions. English has a rigid word order—subject-verb-object (SVO)—as it is a (relatively) morphologically impoverished language. In

a language such as German, however, the word order is not so strict. It contains a richer morphological component which characterises subject and object. Regular (monoclausal) German word order is SVO, as in (21):

(21) SVO: Der Mann jagte den Hund.
'The man hunted the dog'

However, German also allows subject-object-verb (SOV), and object-subject-verb (OSV) for emphasis, as in (22), which means the same as (21), but places the emphasis on the fact that it is *the dog* that *the man* hunted, rather than something else.

(22) OSV: Den Hund der Mann jagte.
'The dog the man hunted'

By abstracting away from the different possible c-structure configurations in German, translation to and from English becomes more straightforward using f-structures.

This can also be shown for the example sentence in (1), and its Irish translation, *Chonaic Séan Marie.* Figure 2.1 shows the c-structure tree and f-structure for the English sentence in (1), with the equivalent c-structure tree and f-structure for the translation of the sentence in Irish.

In the English c-structure tree in Figure 2.1, the verb is preceded by the subject, *John*, and followed by the object, *Mary*, illustrating SVO word order, whereas in the Irish c-structure tree, the verb precedes its complements, illustrating VSO word order, as Irish is a verb-initial language. Although the sentences and c-structure trees differ, the f-structures for both languages are basically the same apart from the lexical translational differences, both encoding the same grammatical information, containing a predicate, a subject, and an object.

2.3.3 F-Descriptions

F-descriptions are the intermediary between c- and f-structures. They resemble a set of simultaneous equations in algebra, and can be used constructively: the statements support a set of inferences for which an f-structure satisfying the f-descriptions may be generated (cf. Chapter 4).

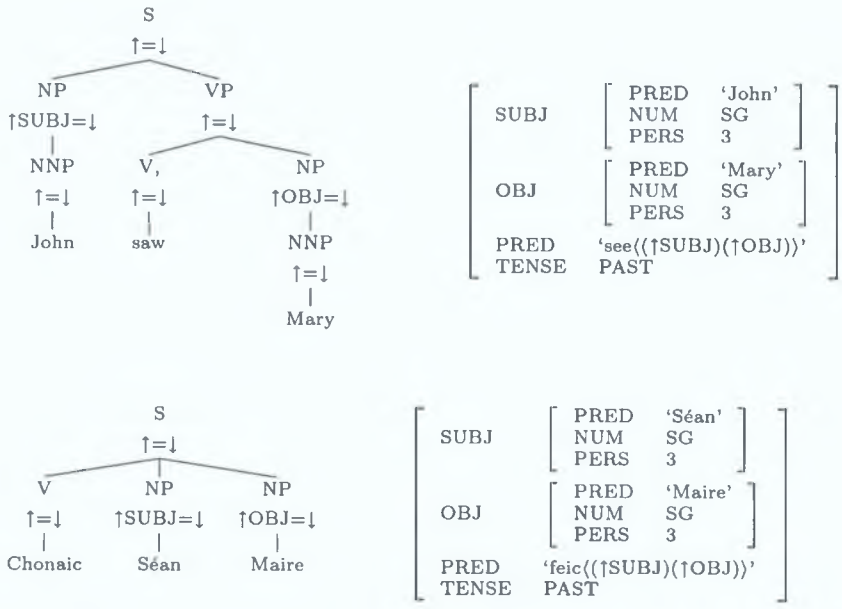


Figure 2.1: English c- and f-structure for the string *John saw Mary*, with the equivalent c- and f-structure for the Irish string *Chonaic Séan Maire*

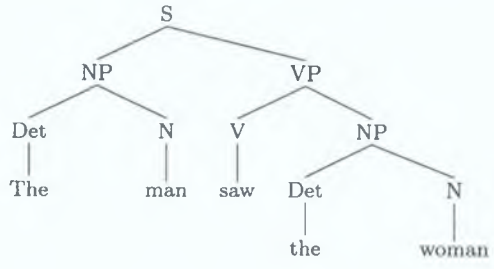
2.3.4 An Example

Consider the English sentence in (23):

(23) The man saw the woman

The c-structure tree corresponding to (23) is (24):

(24)



The c-structure rules with functional annotations associated with the tree in (24) are given in (25):

(25)

S	→	NP	VP
		↑SUBJ=↓	↑=↓
NP	→	Det	N
		↑SPEC=↓	↑=↓
VP	→	V	NP
		↑=↓	↑OBJ=↓

These functional annotations are then inserted into the tree, as shown in Figure 2.2.

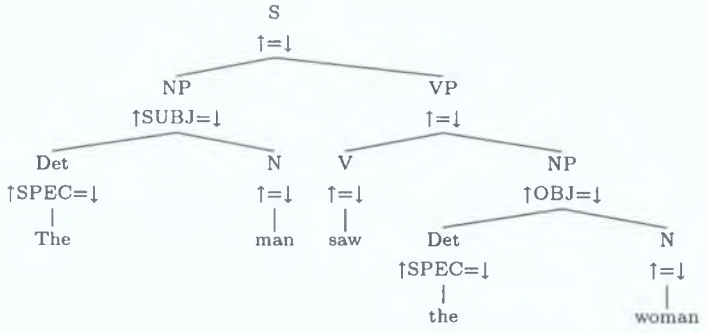


Figure 2.2: C-structure tree with structural annotations for the string *The man saw the woman*

The functional annotations on the tree are the same as those described in (3). Once we have a c-structure tree with functional annotations, it is possible to produce an f-description by inserting actual variables at the root node and at each node containing the '↓' meta-variable, i.e. those nodes whose value is a smaller f-structure (cf. the 'f' numbers in the f-structures in Figure 2.3). The meta-variables in c-structure rules are replaced with actual variables to produce f-descriptions (equations which link the nodes in the tree to functional attributes).

These actual variables replace the '↑' and '↓' meta-variables, and produce the f-description in (26):

(26) Syntactically determined equations:
$f_1(\text{SUBJ}) = f_2$
$f_1 = f_3$

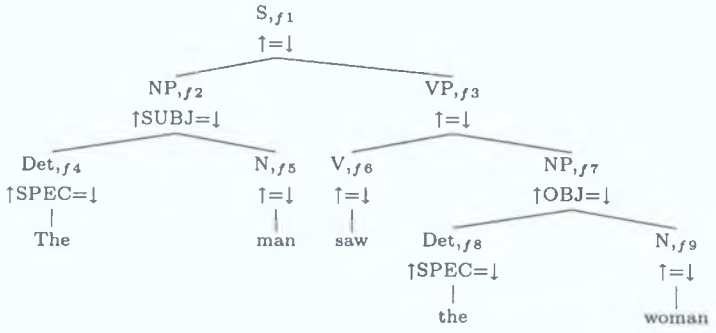


Figure 2.3: C-structure tree with f-variables for the string *The man saw the woman*

$f_2(SPEC) = f_4$

$f_2 = f_5$

$f_3 = f_6$

$f_3(OBJ) = f_7$

$f_7(SPEC) = f_8$

$f_7 = f_9$

Lexically determined equations:

$f_4(PRED) =$ 'the'

$f_4(PRED) =$ 'man'

$f_4(NUM) = 3$

$f_4(PERS) = sg$

$f_6(PRED) =$ 'see⟨(↑SUBJ)(↑OBJ)(↑TENSE=PAST)⟩'

$f_8(PRED) =$ 'the'

$f_9(PRED) =$ 'woman'

$f_9(NUM) = 3$

$f_9(PERS) = sg$

The lexical information about number and person comes from the lexical entries. A partial lexical entry for the subject in this example, *the man*, is shown in (27):

(27)

Det → the	N → man
(↑PRED) = 'the'	(↑PRED) = 'man'
	(↑NUM) = sg
	(↑PERS) = 3

This encodes information stating that *the man* is a 3rd person singular noun. This information is encoded in our automatic f-structure annotation algorithm using lexical macros, which are described in detail in Section 4.2.2.

An f-structure for the string in (23) is produced by solving the equations in the f-description. The variables can be deleted once they have all been instantiated. "The f-structure for an utterance is the minimal solution satisfying the constraints introduced by the words and phrase structure of the utterance" ([Dalrymple, 2002], p.101). The notion of a minimal model is important in LFG—extra features in the f-structure which are not necessary in order to make it complete and coherent should not be included. This ties in with the Principle of Economy of Expression (Section 2.2.2). The f-structure for the string in (23) is shown in (28):

(28)
$$f_1, f_3, f_6 \left[\begin{array}{lll} \text{SUBJ} & f_2, f_4, f_5 & \left[\begin{array}{ll} \text{SPEC} & \text{'the'} \\ \text{PRED} & \text{'man'} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{array}\right] \\ \text{OBJ} & f_7, f_8, f_9 & \left[\begin{array}{ll} \text{SPEC} & \text{'the'} \\ \text{PRED} & \text{'woman'} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{array}\right] \\ \text{PRED} & \text{'see}\langle(\uparrow\text{SUBJ})(\uparrow\text{OBJ})\rangle\text{'} & \\ \text{TENSE} & \text{PAST} & \end{array}\right]$$

2.4 Grammaticality Constraints

Once the f-structure is produced it is necessary to run grammaticality checks. **Grammaticality** states that a sentence is grammatical if and only if it is assigned a complete and coherent f-structure.

2.4.1 Completeness and Coherence

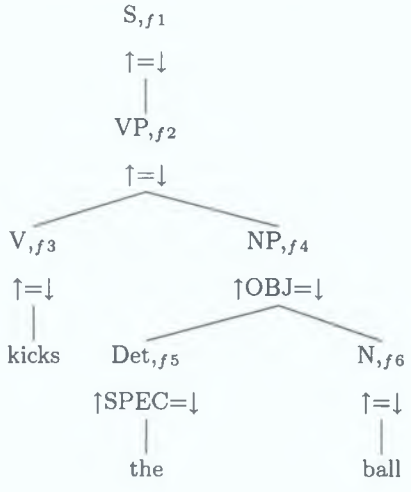
Completeness and Coherence guarantee that grammatical functions and lexical predicates appear in mutually compatible f-structure configurations ([Kaplan and Bresnan, 1982], p.204).

A f-structure is **complete** if and only if it contains all the governable grammatical functions that its predicate governs. The string in (29) is ungrammatical:

(29) * kicks the ball.

This string is associated with the c-structure in (30):

(30)



C-structure is defined in terms of syntactic categories, and is determined by grammar rules. Therefore, the c-structure in (30) is valid, but the completeness condition causes the string in (29) to be marked ungrammatical. As can be seen from the f-structure in (31), the predicate, *kick*, requires two grammatical functions, a subject and an object, but the f-structure only contains an object. The SUBJ attribute is not assigned a value, so the f-structure is incomplete.

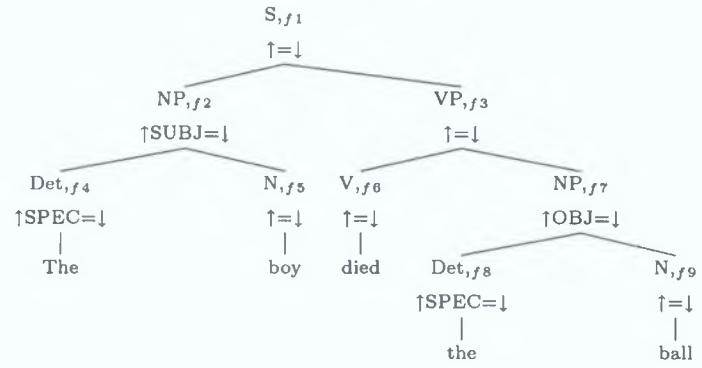
(31)
$$f_1, f_2 \begin{bmatrix} \text{OBJ} & f_4 \begin{bmatrix} \text{SPEC} & \text{'the'} \\ \text{PRED} & \text{'ball'} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{bmatrix} \\ \text{PRED} & \text{'kick}\langle(\uparrow\text{SUBJ})(\uparrow\text{OBJ})\rangle\text{'} \\ \text{TENSE} & \text{PRES} \end{bmatrix}$$

A f-structure is **coherent** if and only if all the governable grammatical functions the f-structure contains are governed by a local predicate. All items with a PRED value must be assigned to a grammatical function. The string in (32) is ungrammatical:

(32) *The boy died the ball.

This string is associated with the c-structure in (33):

(33)



Again, the c-structure in (33) is valid, but the coherence condition causes the string in (32) to be marked ungrammatical. The f-structure produced is shown in (34):

(34)

$$f_1, f_3, f_6 \left[\begin{array}{lll} \text{SUBJ} & f_2, f_4, f_5 & \left[\begin{array}{ll} \text{SPEC} & \text{'the'} \\ \text{PRED} & \text{'boy'} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{array}\right] \\ \text{OBJ} & f_7, f_8, f_9 & \left[\begin{array}{ll} \text{SPEC} & \text{'the'} \\ \text{PRED} & \text{'ball'} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{array}\right] \\ \text{PRED} & \text{'die}\langle(\uparrow\text{SUBJ})\rangle\text{'} & \\ \text{TENSE} & \text{PAST} & \end{array}\right]$$

The predicate for *die* requires only a subject. However, it must require both a subject and an object in order for the f-structure in (34) to be coherent. *The ball* is not associated with any argument of the verb, nor can it be interpreted as an adjunct, so it should not appear in the f-structure. Accordingly, the grammaticality requirements of LFG show (32) to be ill-formed.

2.4.2 Uniqueness

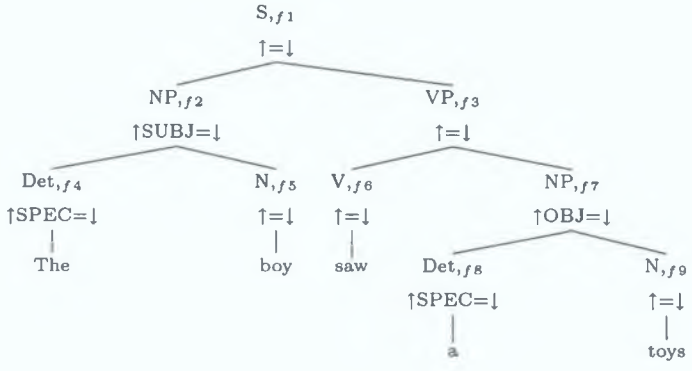
The **uniqueness**, or consistency, condition states that in a given f-structure, a particular attribute may have at most one value. Violations on the uniqueness

condition enforce many co-occurrence restrictions besides those that are normally thought of as agreements, cf. ([Kaplan and Bresnan, 1982], p. 205) for more detail on these restrictions. (19) gives an example of an agreement violation, where the subject noun phrase is plural which clashes with the singular verb phrase. The properties of a node's f-structure must be compatible if that f-structure is to exist. The uniqueness condition will disallow a sentence such as (35), as the attribute for NUM in the object NP has more than one value.

(35) *The boy saw a toys.

The c-structure for the sentence is shown in (36):

(36)



The sentence is ungrammatical because the number of the final determiner and noun disagree, i.e. the determiner is singular, while the noun is plural, as shown in the partial lexical entries, annotated with the relevant f numbers, in (37):

(37) *a*: f_8(NUM) = SG
toys: f_9(NUM) = PL

The NP node cannot be both singular and plural, so a conflict arises. In this case the unification operation (cf. Section 2.3.2.3) will fail, and no f-structure will be produced.

2.5 Argument Structure

A(rgument)- structure, which represents the syntactic arguments of a predicate, maps between thematic roles and grammatical functions. Arguments can

be identified by their role in the meaning of the predicate. These roles are known as thematic or theta (θ) roles, which illustrate semantic, or conceptual, relations by means of labels such as Agent, Patient, Theme, Goal, etc. Thematic relations are not primitives of linguistic theory (unlike grammatical functions).[4]

A-structure determines the syntactic functions of the semantic arguments of a predicate ([Dalrymple, 2002], p.195f). In order to consider the nature of argument structure and the mapping between the lexicon and the syntax, it is necessary to consider a function-changing process such as the active-passive alteration. The input and output of relation-changing rules like passive can be considered to be a good test for grammatical functions.

2.5.1 Passive

In LFG, passivisation is taken to be a lexical process. Passivisation is one class of constructions that can be classified as function-changing, and can be viewed as an alternative means of linking grammatical functions to semantic arguments.

Active and passive verb forms can both be listed in the lexicon. Both express the same predicate, but with different mappings of the arguments. In English, all transitive verbs which have an OBJ as the second argument may be passivised. The rule of passivisation applies uniformly to 'transform' an object into a subject ([Dalrymple, 2002], p.22). The object, *him*, in (38) becomes the subject of (39).

(38) She gave him a present.

(39) He was given a present.

Traditional grammatical descriptions classify *a present* in the example sentence in (38) as the direct object, and also in the example sentence in (40).

A present can be passivised in (38), i.e. in the case of ditransitive verbs, two passives can be formed:

(40) She gave a present to him.

(41) A present was given to him (by her).

The object, *present*, in (40), becomes the subject of (41).

In (38), *him* is classified as the indirect object. In English verb phrases, the primary, or direct, object, *him*, immediately follows the verb, with the secondary,

[4]This is according to a theory of thematic roles, based on [Jackendoff, 1990]. Thematic relations are descriptions of certain aspects of cognitive conceptualisation ([Falk, 2001], p.101).

or indirect, object, *a present*, following it as in (38). The traditional grammatical view assumes that a phrase must bear the same grammatical function if its semantic role does not change. However, the LFG view differs to this, and classifies the first noun phrase following the verb as the object, and the second noun phrase bears some other function. In (38), *him* is classified as the object, whereas in (40), *a present* is classified as the object. This makes it easier to state a generalisation for the passive: that the object in the active sentence becomes the subject in the passive.

Generally, passivisation involves the demotion of a subject to either an unexpressed argument or an adjunct or an oblique (often with a restricted form, e.g. with the preposition *by* in English, as in (41), *by her*). This captures the fact that the OBL plays a special role with respect to the verb in passive sentences, i.e. that it is the logical subject.

Using the mathematical symbol $\mapsto$, 'maps into', and the emptyset, $\emptyset$, it is possible to characterise passivisation in English as (42), as given in ([Falk, 2001], p.94).

(42) $(\uparrow \text{SUBJ}) \mapsto \emptyset$
$(\uparrow \text{OBJ}) \mapsto (\uparrow \text{SUBJ})$
Morphology: participle

The passive mapping rule is a function-changing construction, i.e. it takes existing information and changes it. The lexical rules represent regularities which store lexical items, so the usual LFG constraint against changing information is inapplicable. Different assignments of grammatical functions to the same thematic roles (as in the active and passive versions of a verb) are treated using *lexical redundancy rules*, which encode a regular lexical relation between different forms. In order to move away from a transformationally oriented view of grammatical function alternations, such as the active/passive relation, work has been done on lexical rules and relations, forming a richly structured lexicon and an articulated theory of relations among lexical forms ([Dalrymple, 2002], p.201). However, there were some problems with this mapping. Complex predicates, which are characterised by the fact that two verbs appear to act as a single predicate, are an example of a problem with this mapping. Complex predicate formation, and therefore argument linking, can not be defined by reference to argument roles of a single word or a phrase structure constituent. Given problems such as this and the need for a mapping between grammatical function and thematic roles, the Lexical Mapping Theory was formulated.

2.5.2 Lexical Mapping Theory

In more recent work within LFG, the emphasis has moved from such lexical rules to a more general theory of *linking* or *mapping* between the predicate-argument structure and the grammatical functions. Lexical Mapping Theory (LMT) maps from a semantic (or conceptual) relationship of thematic roles, which can be known as 'θ-structure', to f-structure, via a-structure, which, as discussed in Section 2.5, is the intermediate representation between the two, as shown in Figure 2.4.

Figure 2.4: Mapping from θ-structure to f-structure

Subcategorisation frame alternations such as the passive are seen as different possibilities of mapping from the predicate-argument structure to the grammatical functions of a predicate, where the *agent* argument of a verb is suppressed and the other argument, usually a *theme*, is then mapped to a SUBJ.

Research on argument mapping led to the conclusion that grammatical functions like SUBJ and OBJ can be grouped into natural classes, and thematic roles can be associated with these classes rather than specific roles ([Dalrymple, 2002], p.203). SUBJ and OBJ are semantically unrestricted functions, i.e. they can be filled by an argument with any thematic role.

LMT provides a principled account of the mapping from thematic roles to syntax, one in which there is no changing of grammatical functions in constructions like the passive. It provides a classifactory system for argument grammatical functions which is the basis for syntactic mapping of thematic roles.

There are certain conditions to restrict the range of possible mappings; that (i) every verb must have a subject, and (ii) each a-structure role must correspond to a unique f-structure function, and vice versa.

2.6 Coordination

In this section we discuss the standard views on coordinate constructions in LFG as handled by ([Butt et al., 1999], p.139f.), ([Dalrymple, 2002], p.361f.).

A c-structure rule such as (43) allows any like categories, e.g. S and S, VP and VP, PP and PP, etc., to be coordinated. The conjuncts form a set via the

'$\downarrow \in \uparrow$' annotation:

(43)

SCCOORD (CAT) =	CAT	CONJ	CAT
	$\downarrow \in \uparrow$	$\downarrow = \uparrow$	$\downarrow \in \uparrow$

Coordination must allow for more than two conjuncts[5] and differs according to whether the non-final conjuncts are separated from one another only by commas, as shown in Figure 2.5, or whether a conjunction is also required ([Butt et al., 1999], p.142).

SCCOORD (CAT) =	CAT	([COMMA CAT]+	(COMMA))	CONJ	CAT
	$\downarrow \in \uparrow$	$\downarrow \in \uparrow$		$\uparrow = \downarrow$	$\downarrow \in \uparrow$

Figure 2.5: LFG rule for conjuncts separated by commas

An example of this from the Penn-II Treebank is given in Section 4.4, Figure 4.11, p.105, where the LFG approach to handling coordination is discussed in relation to real data.

So far, we have been dealing with coordination involving one main conjunct, but it is also necessary to consider cases with two-part conjunctions, such as *both...and*, etc. ([Butt et al., 1999], p.142) suggest the following for such two-part conjunctions: the first half of the conjunction is annotated as a 'PRECONJ', which is constrained to occur with a particular 'CONJ-FORM', provided by its paired conjunction. In ([Dalrymple, 2002], p.367), the same idea is proposed. The features PRECONJ and CONJ are classified as non-distributive features, i.e. the feature and its value become a property of the set as a whole. The constraining equation for *both* is given in (44):

(44) *both*
PreCnj ($\uparrow$ PRECONJ) = BOTH
Cnj ($\uparrow$ CONJ) $=_c$ AND

This ensures that *both* can occur only with the CONJ value *and*, thus preventing an ungrammatical phrase such as **both walked or ran.* The f-structure for (44) would then appear as shown in (45):

[5]When working with data from the Penn-II Treebank, we also must allow for cases containing more than one conjunction, cf. Figure 4.14, p.106. And, sometimes, there is only one conjunct, as in (146), p.98.

(45)
$$\begin{bmatrix} \text{PRECONJ} & \text{both} \\ \text{CONJ} & \text{and} \\ \{\begin{bmatrix} \text{PRED} & \ldots \end{bmatrix} & \\ \begin{bmatrix} \text{PRED} & \ldots \end{bmatrix}\} & \end{bmatrix}$$

Nested coordinate structures are discussed in ([Dalrymple, 2002], p.367), but do not extend to multiple conjunctions such as Figure 4.14, p.106, which shows an example of real data from the Penn-II Treebank.

Cases also occur where only a punctuation marker, separates two complete sentences, as in (46), from WSJ 0596, tree 34, which contains the rule S → S : S, where the 'lexical entry' for the colon is '- -'.

(46) His answer reveals his vulnerability - - it also draws the line that Soviet society must cross to enter the normal dialogue of Western culture.

One LFG approach to handling such structures is to have a special coordination rule for the highest category under ROOT, which allows certain types of punctuation in place of a conjunction ([Butt et al., 1999], p.143). Therefore, the punctuation is analysed as the head of the phrase (in these special cases). We have implemented this in our algorithm. The two conjoined sentences are members of the conjunct set, each conjunct containing the complete f-structure for each sentence; and the '- -' is annotated as the main pred, as shown in the partial f-structure in Figure 2.6.

Coordinate structures with more complex data from the Penn-II Treebank are given in Section 3.6. Coordinate structures following the linguistic assumptions of the Penn-II Treebank in relation to LFG are discussed in Section 3.9.2, with a description of the implementation of the coordinate structures given in Section 4.4.

2.7 Long Distance Dependencies

Long distance dependencies are constructions in which a displaced constituent bears a syntactic function usually associated with some other position in the sentence ([Dalrymple, 2002], p.389f.). These long distance dependencies, such as topicalisation, relative clauses, and wh-questions in English, must be licensed

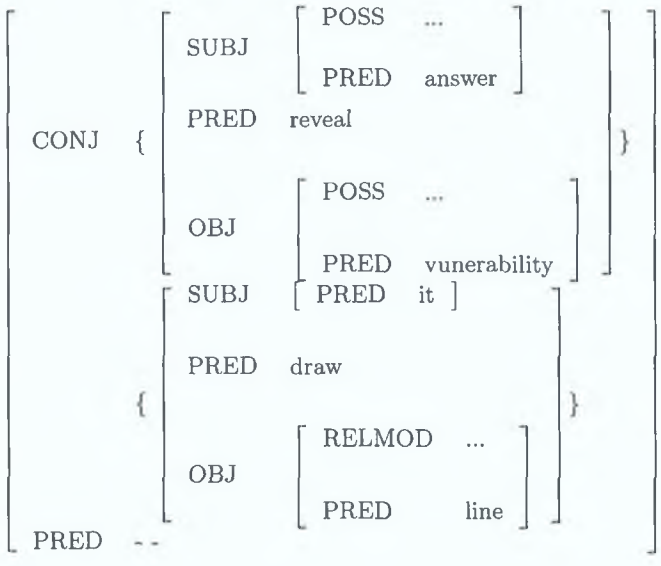


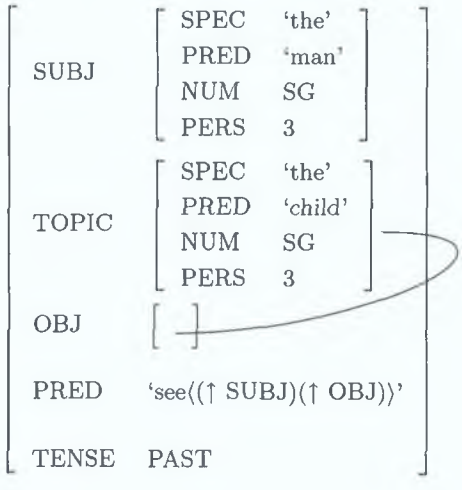
Figure 2.6: Partial f-structure where punctuation is the head for the string *His answer reveals his vulnerability - - it also draws the line that Soviet society must cross to enter the normal dialogue of Western culture.*

by functional control equations, an example of which is shown in (47) for the sentence in (16), *The child, the man saw.*

(47) $(\uparrow \text{TOPIC}) = (\uparrow \text{OBJ})$

The f-structure for (16) is given in (48):

(48)

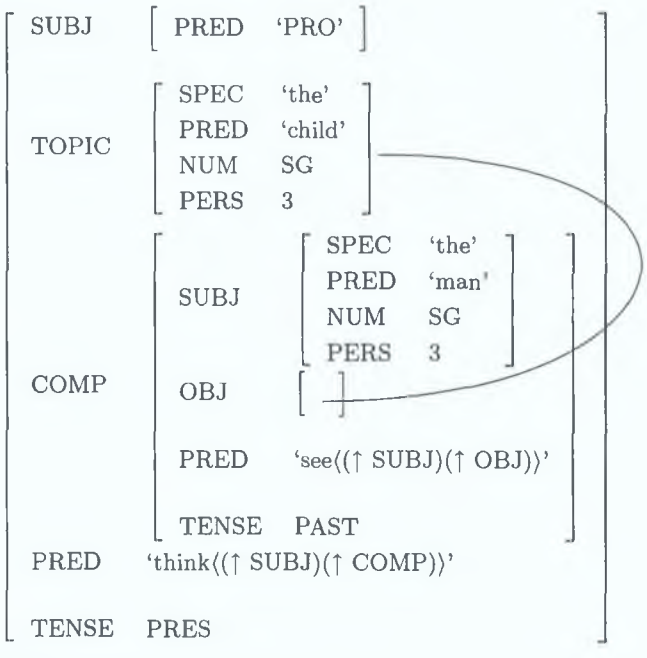


The number of clauses between two positions is unlimited. In (16), the TOPIC is linked directly to the OBJ of the sentence. However, longer paths, which occur in a sentence such as (49), are also allowed, where the TOPIC is linked to the OBJ within the COMP function.

(49) The child, we think that the man saw___.

The f-structure for (49) is given in (50):

(50)



The TOPIC in this sentence, *the child*, is linked to the empty OBJ of *saw* within the COMP function. The functional equation is '↑TOPIC = ↑COMP OBJ'.

Many syntactic constraints on long distance dependency constructions are definable in terms of the grammatical function of the displaced phrase. The primary constraints on long distance dependencies are mainly functional in nature, and best expressed within f-structure terms, rather than in c-structure ([Dalrymple, 2002], p.411).

The element at the front of the clause, called the **filler**, is understood as filling a particular grammatical role within the clause. The lower end, called the **gap**, involves a missing element, the grammatical role of which is determined by the arrangement of the c-structure nodes inside the clause. In (51), the gap is associated with the subject role, whereas in (52), it is associated with the object of the phrase.

(51) The girl promised the boy to go.

(52) The girl persuaded the boy to go.

The functional control equation associated with (51) is '↑SUBJ=↑XCOMP SUBJ'. The functional control equation associated with (52) is '↑OBJ=↑XCOMP SUBJ'. The content of the filler has two grammatical functions, one expected of the position in which the filler is located, and one expected of the gap. The filler and the gap need not be the same grammatical function. This can be seen by revisiting (16), *The child, the man saw*, where the expected functional category of the filler, *the child*, is TOPIC, and the expected functional category of the gap is OBJ. Assigning the filler position a discourse function is the LFG equivalent of calling it a non-argument position in structural theories. There are two structural positions for fillers in English, [SPEC, CP (complement phrase)] for *wh* phrases, and adjoined to IP (which corresponds to a sentence) for topicalised phrases. Elements in either position can have the function TOPIC, representing old information, or FOCUS, representing contrast, and thus new information ([Falk, 2001], p.105). However, ([Dalrymple, 2002], p.395) states that the English TOPIC phrase does not appear in specifier position, and this is shown in ([Bresnan, 2001], p.212f.), where it is stated that the topic must instead be adjoined to IP.

2.7.1 Topicalisation

The displaced TOPIC constituent is often referred to as having been 'fronted' or 'extracted'. The TOPIC phrase adheres to the Extended Coherence Condition (the Coherence condition is explained in Section 2.4.1), which states that all functions in an f-structure must be incorporated into the semantics. The TOPIC of a sentence could appear in the specifier position of IP and also bear a grammatical function (such as OBJ) inside the same sentence, as shown in (16). In this sentence, the displaced constituent, *the child*, bears the TOPIC and OBJ functions. The f-structure in (48) shows the relation between the TOPIC and the OBJ. The fronted phrase in a topicalised construction is not restricted to the category NP, but may also be PP, AP, CP, and even VP depending on the speaker ([Dalrymple, 2002], p.391), as in (53):

(53) To leave, we convinced Chris.

Certain constraints occur regarding which grammatical functions of the fronted phrase can be related to the TOPIC. TOPIC can fill the role of OBJ, as

in (16), and can also be the OBJ of the subordinate COMP, as in (49), which is shown in the f-structure in (50).

However, there is a further restriction here that TOPIC can only be linked to the OBJ within the COMP of a so-called 'bridge verb', like *think*, and not within the COMP of a non-bridge verb, such as *whisper*, as in (54) (cf. ([Dalrymple, 2002], p.392) for more detail):

(54) *Chris, we whispered that David saw

Another restriction involves the SUBJ function, which, for English, is not permitted on the path which links the TOPIC to an OBJ, as in (55):

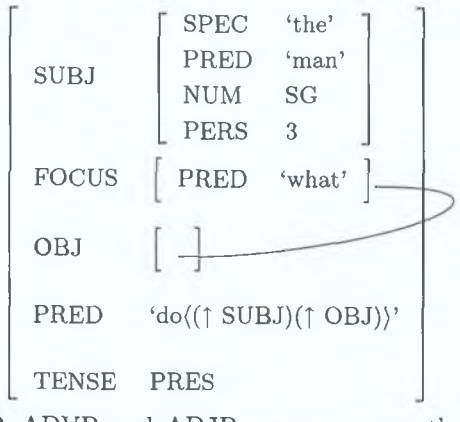
(55) The child that the man saw___ surprised us.

In this example, *the child* is the subject of the sentence, and *that the man saw surprised us* is a relative clause. Further discussion on topicalisation is given in ([Bresnan, 2001], p.212f.).

2.7.2 Wh-Questions

The question word in an English wh-question appearing in the initial position in the sentence (as the specifier of the CP) receives the FOCUS annotation. The FOCUS function is a discourse function, often a wh-phrase linked to the OBJ in the f-structure, as in (17), *What is the man doing?*. In this construction, *what* bears the FOCUS function, and is linked to the OBJ function in the f-structure by the functional equation '↑FOCUS = ↑OBJ', as shown in (56):

(56)



NP, PP, ADVP and ADJP can appear as the FOCUS constituent in the specifier of CP. Constraints on the FOCUS path are similar to those on the TOPIC path, i.e. it is not restricted to be of the category NP, it may also be PP, AP, CP, or VP.

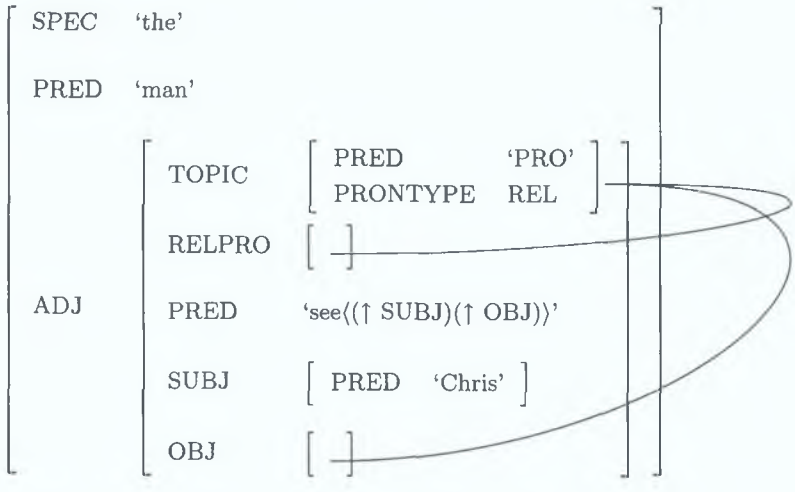
2.7.3 Relative Clauses

Relative clauses in English may involve long distance dependencies, as in (57):

(57) The man who Chris saw.

Two long distance dependencies are involved in a relative clause construction ([Dalrymple, 2002], p.400f.). The first dependency holds between the fronted phrase and the within-clause grammatical function that it fills. The TOPIC function is linked to the OBJ within the clause (*Chris* is the SUBJ). The second dependency involves the relative pronoun and its position, possibly embedded, within the fronted phrase. *who Chris saw* appears in the f-structure as a member of the adjunct set, as shown in the f-structure in (58):

(58)



The f-structure of the relative pronoun appears as the value of the feature RELPRO, in the initial position in the relative clause, and its f-structure is both the TOPIC and the RELPRO of the relative clause.

2.7.4 Functional Uncertainty

If there is more than one possible f-structure configuration that can satisfy a particular functional equation, **functional uncertainty** is the tool used to describe such phenomena. It is used as a means of extending functional descriptions to allow for variable chains of attributes that define possible paths through f-structures which may be unknown to us in advance ([Bresnan, 2001], p.73). The path through the f-structure that relates TOPIC to its within-clause function illustrates functional uncertainty, i.e. the TOPIC function could bear

OBJ, SUBJ, or OBL function. Functional uncertainty formalises long distance dependencies as a chain of strictly local relationships ([Falk, 2001], p.152f.). By analysing the long distance dependencies in terms of functional uncertainty, LFG claims that c-structure properties are irrelevant to the behaviour of the construction.

In the examples illustrating long distance dependencies given previously (in (16), (17), etc.), some clause-internal function is identified with the TOPIC or FOCUS. This is similar to functional control, as it involves feature sharing, but the relation between the two functions cannot be expressed as a finite expression. Therefore, the treatment of long distance dependencies such as topicalisation, question formation, and relative clause formation in English is handled by means of functional uncertainty.

Two types of functional equations are needed to express different grammatical functions, outside-in and inside-out.

Outside-in functional uncertainty, which is often known as regular functional uncertainty, is used to define constraints on more deeply embedded structures. Outside-in functional designators allow us to specify an f-structure embedded at an arbitrary depth inside another f-structure. They start from the clause of the filler, and have a long distance path to the grammatical function to which they are linked. An example of an outside-in functional uncertainty equation is given in (59):

(59) $(\uparrow \text{FOCUS}) = (\uparrow \text{COMP}^* \text{OBJ})$

An example of this is *Which book do you think I put on the shelf* ([Falk, 2001], p.154), in which the FOCUS is linked to the missing OBJ of *put*, by means of the equation '$(\uparrow$ FOCUS$) = (\uparrow$ COMP OBJ)'. With outside-in functional uncertainty there is no c-structure gap (cf. the Principle of Economy of Expression, as described in Section 2.2.2, which limits the use of empty elements in c-structure). There are no constraints on identifying the gap.

Inside-out functional uncertainty involves f-structures that enclose an f-structure at an arbitrary level of distance, and is used to define constraints on enclosing structures. Inside-out functional designators involve situations where the word imposes a constraint on the larger structure in which it is found. They are used primarily in the analysis of long-distance dependencies and anaphora. They start from the clause of the gap, and involve an empty category in c-structure. Inside-out functional uncertainty has to be associated with the gap.

An example of an inside-out functional uncertainty equation is given in (60):

(60) $((\text{COMP } *\uparrow) \text{ OBJ}) \simeq (\uparrow \text{ TOPIC})$

An example of this is the string *Ann, I think he likes*, in which the functional equation '((COMP ↑) OBJ) = (↑ TOPIC)' links the TOPIC, *Ann*, to the OBJ of the COMP.

2.8 Summary

LFG was designed to be implementable in computational systems from the beginning, and is therefore particularly well suited to the construction of automatic f-structure annotation principles as we have done in our automatic f-structure annotation algorithm. This chapter introduced LFG as a theory of linguistic description, giving a brief description of the background to the theory, and detailing some of the main principles, such as the Lexical Integrity Principle and the Principle of Economy of Expression. The grammatical representations used (c- and f-structure) are described, including, where necessary, some detail on X′ theory and grammatical functions. We also provided a detailed step-by-step example of how to construct an f-structure from a simple sentence of English. We discussed the constraints used and the way in which LFG deals with linguistic phenomena such as passivisation and long distance dependencies, examining topicalisation, wh-questions, relative clauses, and functional uncertainty. Passivisation and long distance dependencies are encoded in the Penn-II Treebank by means of trace and index information on null elements (cf. Section 3.8), differing to theories of LFG which limit the use of empty elements in c-structure. However, use can be made of this in our automatic f-structure annotation algorithm (cf. Section 4.5), so we extend the theory of LFG to incorporate the linguistic assumptions of the Penn-II Treebank. In Section 3.9, we an exposition of these assumptions is given in relation to LFG theory, describing some of the difficulties encountered when applying the theory to real data.

Chapter 3

Linguistic Encoding of the Sentences in the Penn-II Treebank

3.1 Introduction

The Wall Street Journal (WSJ) section of the Penn-II Treebank consists of more than 1,000,000 words, tagged for part-of-speech (POS) information, in about 50,000 sentences and trees. It provides information to construct a representation of predicate-argument structure [Marcus et al., 1994]. This chapter gives a review of the linguistic design embodied in the Penn-II Treebank as detailed in [Santorini, 1991], [Marcus et al., 1994], and [Bies et al., 1995]. The tagset and notation used in coding the Penn-II Treebank, the basic clause structure, subordinate clauses, and wh-phrases are described in the following sections. Coordinate structures, the modification of noun phrases and the handling of null elements in the Penn-II Treebank are then described in individual sections as they can pose particular problems for automatic annotation, due to the often flat treebank analyses provided. Selected examples are provided and references are made to our annotation algorithm, where relevant, in order to illustrate some examples of mistagging within the Penn-II Treebank, and to give some indication of the implementation of our automatic f-structure annotation algorithm, which is described in detail in Chapter 4. The linguistic assumptions of the Penn-II Treebank in relation to LFG theory are discussed at the end of the

chapter, followed by a summary of the contents of the chapter.

3.2 Coding the Penn-II Treebank

This section discusses the motivations for the original tagging and bracketing design decisions of the Penn-II taggers for the Penn-II Treebank (henceforth referred to as the Treebank), together with difficulties encountered when encoding information. The basic clause structure of the Treebank is described in Section 3.3. Subordinate clauses, wh-phrases, coordinate structures, the modification of noun phrases and the handling of null elements are discussed in separate sections, as they can pose particular problems for tagging the Treebank and also for the encoding of our algorithm.

3.2.1 Tagset

In the development of the Treebank, raw text was first assigned POS tags automatically, using [Church, 1988]'s PARTS part-of-speech labeller, and then corrected by human annotators. PARTS uses a modified version of the Brown Corpus tagset and assigns POS tags with an error rate of 3-5%. The size of the tagset was made as small as possible in order to minimize the chances of tagging inconsistencies. Of course, the smaller the tagset, the harder it can be in certain cases to decide how a node should be tagged. One of the main steps taken to minimize the size of the tagset was to eliminate redundancy by taking into account lexical and syntactic information ([Marcus et al., 1993], p.3). Use is made of lexical recoverability when distinguishing between verbs, i.e. it is clear what type of word it is from the tag, and any further information needed to distinguish between words can be found in the lexicon (by means of lexical macros used in our algorithm, which associate lexical information with each word, cf. Section 4.2.2). Lexical recoverability is also used with words that can precede articles in NPs (cf. Appendix A)—there is no distinction made between pre-qualifiers, such as *quite, rather, such*, and pre-quantifiers, such as *all, half, many*; all are assigned to a single category, PDT (predeterminer).

The Treebank distinguishes 36 POS tags and 12 other tags (for punctuation and currency symbols). They are listed in alphabetical order in Table 3.1, with a description of the POS or the symbol corresponding to them also given. The tags used are very important for the accuracy of our lexical macros (cf. Section 4.2.2), as they will be incorrect if the correct tag is not used, cf. (61). As stated

in Section 2.2.2, in LFG words are as important, if not more so, as syntactic rules in expressing grammatical information—words are the atoms out of which syntactic structure is built.

The Treebank tagset does not distinguish between subject and object pronouns, even in cases where the distinction is not lexically recoverable, as with *you*, which does not change its form depending on whether it is nominative or accusative (this can be distinguished based on the position of the pronoun in the c-structure tree), unlike the male singular pronoun, which takes the form *he* when it is in nominative case (subject position), and *him* when it is in accusative case (object position).

The Treebank tags both subordinating conjunctions (e.g. *that*, *as*, etc., discussed in more detail in Section 3.4), and prepositions as IN. By looking at syntactic context, it is still possible to distinguish between the two, as subordinating conjunctions precede clauses, and prepositions precede noun phrases or prepositional phrases. However, the phrase *that man* can be ambiguous, and such mistagging can cause problems for our automatic annotation algorithm. The tag TO is used for the preposition *to* and for the modal *to*, which leads to difficulties for automatic annotation, cf. Section 4.2.2. However, the tagging guidelines are not always strictly adhered to as there are some instances of mistagging where *to* is tagged as IN in a PP. It does not cause us any problems as it is a preposition in these cases, but it does indicate inconsistencies in following the tagging guidelines.

POS is the tag used for the possessive ending on nouns (cf. Section 3.7.1). However, it has also been used incorrectly in verb phrases to tag a third person singular present verb *'s*, as in (61), from WSJ 0071, tree 56:

(61) By January it should be fairly clear what **'s** hot – and what **'s** not

VBZ is the tag assigned for a third person singular present verb. This mistagging is taken into account by our head rules (cf. Section 4.2.1), so POS within a verb phrase will be annotated as the head (as a verb in this case). The complete guidelines for tagging are given in [Santorini, 1991].

3.2.2 Bracketing

Bracketing the Treebank was also partially automated—the output of the POS tagging process was parsed automatically, using the Fidditch parser [Hindle, 1989], and simplified to yield a skeletal syntactic representation, which

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition / subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol (mathematical or scientific)
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund / present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person sing present
WDT	wh-determiner
WP	wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb
#	Pound sign
$	Dollar sign
.	Sentence-final punctuation
,	Comma
:	Colon, semi-colon
(	Left bracket character
)	Right bracket character
‘	Left open single quote
“	Left open double quote
’	Right close single quote
”	Right close double quote

Table 3.1: List of Lexical Tags used in the Penn-II Treebank

was then corrected by human annotators, their main task being to combine the syntactic chunks produced by the parser to produce one complete parse tree, i.e. an unambiguous solution, for each sentence. Extra tags were added in cases in which complete disambiguation was not possible (cf. description of *PPA* tag in Section 3.8.4). The simplification process removes POS tags, non-branching lexical nodes and certain phrasal nodes, notably NBAR. A table showing the complete syntactic tagset is given in Table 3.2 (*cf.* [Marcus et al., 1993], p.10).

Syntactic Tag	**Description**
ADJP	Adjective phrase
ADVP	Adverb phrase
NP	Noun phrase
PP	Prepositional phrase
S	Simple declarative clause
SBAR	Clause introduced by subordinating conjunction or 0
SBARQ	Direct question introduced by wh-word or wh-phrase
SINV	Declarative sentence with subject-aux inversion
SQ	Subconstituent of SBARQ excluding wh-word or wh-phrase
VP	Verb phrase
WHADVP	Wh-adverb phrase
WHNP	Wh-noun phrase
WHPP	Wh-prepositional phrase
X	Unknown constituent
Null elements	**Description**
*	"Understood" subject of infinitive or imperative
0	Zero variant of *that* in subordinate clauses
T	Trace—marks position where moved wh-constituent is interpreted
NIL	Marks position where preposition is interpreted in pied-piping contexts

Table 3.2: List of Syntactic Tags used in the Penn-II Treebank

The bracketing labels used at phrase level are, for the most part, widely used (such as ADJP for **adj**ective **p**hrases, ADVP for **adv**erbial **p**hrases, etc.), with some lesser known labels such as FRAG (**frag**ment), INTJ (**int**er**j**ection), LST (**list** marker), NAC (**n**ot **a** **c**onstituent, which is used to show the scope of certain prenominal modifiers within a noun phrase), NX (used within certain complex noun phrases to mark the head of the noun phrase), PRN (**paren**thetical material), UCP (**u**nlike **c**oordinated **p**hrase, see Section 3.6.2 on UCP coordination rules for more detail), and X (unknown, uncertain, or unbracketable) ([Bies et al., 1995], p.35f.). FRAG and X are the only categories not taken into account in our automatic annotation algorithm when calculating results (explained in Section 4.3.1).

Given a simple example English sentence, *The man likes the woman*, the

corresponding bracketing, following the tagging and bracketing guidelines given for the Treebank, is shown in (62):

(62)
```
(S   (NP  (DT  the)
          (NN  man))
     (VP  (VBZ likes)
          (NP  (DT the)
               (NN woman))))
```

3.2.3 Function Tags

The Treebank tagset distinguishes a handful of semantic roles, as well as the syntactic roles of surface subject and logical subject. First and second verbal objects can be identified by the syntactic structure, and are not distinguished within the tagset.

The Treebank lists twenty functional tags, given in the form: CAT-TAG, where CAT is a CFG category and TAG is used to tag the category with additional functional information. Two tags are used to indicate form / function discrepancies: -ADV (adverbial) and -NOM (nominal).

Seven tags are used to indicate grammatical roles: -DTV (dative), -LGS (logical subject), -PRD (predicate), -PUT (locative complement of *put*), -SBJ (surface subject), -TPC (topicalised), -VOC (vocative).

Seven are associated with more specific adverbials than -ADV (which are generally VP adjuncts): -BNF (benefactive), -DIR (direction), -EXT (extent), -LOC (locative), -MNR (manner), -PRP (purpose or reason), -TMP (temporal). The final four tags are put under the 'Miscellaneous' heading: -CLR (closely related), -CLF (cleft), -HLN (headline) and -TTL (title).

We make use of the following functional tags in the Treebank in our automatic f-structure annotation algorithm to provide more detailed f-structures and improve the accuracy of our annotations (cf. Section 4.5):

- -ADV marks a constituent other than ADVP or PP when it is used adverbially, as in (63) from WSJ 0120, tree 47:

  (63) **Also spurring the move to cloth** : diaper covers with Velcro fasteners that eliminate the need for safety pins

  This sentence contains the rule FRAG → S-ADV : NP. The text dominated by S-ADV is *highlighted in bold*.

This functional tag is used in the 'Catch-all and Clean-up' component of our algorithm, cf. Section 4.6 to annotate any nodes with -ADV which have not already received an annotation as '$\downarrow \in \uparrow$ADJ'.

- -BNF marks the benefactive of the sentence, and attaches to an NP or a PP node. It is used only when the verb can undergo dative shift, as in (64) from WSJ 0576, tree 20:

  (64) We may all hope that California 's voters will heed the scientific realities that their own university 's renowned Prof. Tom Jukes provides **them** and ignore the charlatanry profferred by their " wealthy Hollywood weepers

  This sentence contains the rule VP → VBZ NP-BNF NP, where VBZ contains the lexical entry *provides*, and the NP node following the NP-BNF is empty. The text dominated by NP-BNF is highlighted in bold. The tag is also used when the prepositional phrase uses *for*, as in (65) from WSJ 0576, tree 20:

  (65) Jayark , New York , distributes and rents audio-visual equipment and prints promotional ads **for retailers**

  This sentence contains the rule VP → VBZ NP PP-BNF. The text dominated by PP-BNF is highlighted in bold. Prepositional objects of dative-shifting verbs with other prepositions than *for* (such as *to* or *of*) are tagged with the -DTV functional tag, cf. (68). The -BNF functional tag is used in the 'Catch-all and Clean-up' component of our algorithm to annotate any nodes with -BNF as '$\uparrow$OBL=$\downarrow$', as the benefactor in a sentence will always be an oblique argument.

- -CLR ('**CL**osely **R**elated' to the predicate) is used to mark constituents that are not strictly complements, but are treated as complements whenever it makes a bracketing difference. For example, **PP**-CLR is used to mark a PP as an oblique argument to the verb, as in (66) from WSJ 0096, tree 50:

  (66) It was just a stupid mistake to get the license , " he said , adding , " I 'd just as soon not get **into " details of the settlement**

  This sentence contains the rules VP → VB PP-CLR and PP-CLR → IN " NP. The text dominated by PP-CLR is highlighted in bold.

In the 'Catch-all and Clean-up' component of our algorithm, as described in Section 4.5, we annotate any PP-CLRs which have not already received an annotation as '↑OBL=↓', unless it is preceded by a comma, in which case it is annotated as '↓∈↑ADJ', as in (67) from WSJ 0515, tree 19:

(67) Psyllium is an annual herb , Plantago ovata , that has been used for centuries by folk doctors here , **mainly as a laxative and anti-diarrheal**

This sentence contains the rules VP → VBN NP PP-TMP PP , PP-CLR and PP-CLR → ADVP IN NP. The text dominated by PP-CLR is highlighted in bold. In this case, PP-CLR is not an oblique argument, instead it is (correctly) annotated as an adjunct, cf. Section 4.5.

The only case in which a PP-CLR may have already have received an annotation is if it is annotated as the head of a rule, which may occur under the mother category PRN, or if PP-CLR is the only category on the RHS, in which case it will automatically be annotated as head.

- -DTV marks the dative object in the unshifted form of the double object construction, as in (68) from WSJ 0041, tree 40:

(68) Then , just as an image of the statue of Thomas Jefferson dissolves from the screen , the announcer continues : " On the issue of abortion , Marshall Coleman wants to take away your right to choose and give *it* ***to the politicians*** .

This sentence contains the rule VP → VB NP PP-DTV. The text dominated by PP-DTV, which marks the dative object, is highlighted in bold. This functional tag is used in the 'Catch-all and Clean-up' component of our algorithm to annotate any nodes with -DTV as '↑OBL=↓' (only PPs are tagged with this functional tag in the Treebank).

- -LGS marks the logical subject in the passive. It attaches to the NP object of *by* and not to the PP node itself, as in (69) from WSJ 0130, tree 33:

(69) They must figure that justice has to get done by **somebody** , but know it wo n't be done by **Congress**

This sentence contains the rule PP → IN NP-LGS, which occurs twice in the sentence. The text dominated by NP-LGS is highlighted in bold. This

functional tag is used as part of the module in our algorithm for analysing passive sentences and capturing traces within such sentences (cf. Section 4.5.1).

- -NOM marks free relatives and gerunds when they act nominally, as shown in (70) from WSJ 0113, tree 07:

  (70) With **prices soaring** , they were able to sell the reclaimed commodities at " considerable profit , " the agency 's 240-page report said .

  This sentence contains the rule PP → IN S-NOM. The text dominated by S-NOM is highlighted in bold. The S-NOM is annotated '↑OBJ=↓' as stated in the annotation matrices (cf. Appendix B).

- When there is no VP, the predicate is labelled -PRD, as in small clauses, as in (71) from WSJ 0018, tree 11:

  (71) But Mr. Barnum called that **" a worst-case " scenario**

  This sentence contains the rule S → NP-SBJ NP-PRD. The text dominated by NP-PRD is highlighted in bold. In (71), the NP-PRD would be annotated as head even if it did not have the -PRD functional tag, as the first NP is clearly the subject (labelled NP-SBJ, and in initial position in the sentence). However, the -PRD tag is useful for determining the head of the phrase in cases where the -SBJ tag is not included and it might otherwise be unclear which noun phrase would be the head, as the mother node is S.

- -PUT marks the locative complement of *put*, as shown in the sentence in (72) from WSJ 0937, tree 13:

  (72) All of this is what history will note , assuming that events do n't make it seem a bad joke , when the record of this time is put **down**

  This sentence contains the rule VP → VBN NP ADVP-PUT ADVP-TMP. The text dominated by ADVP-PUT is highlighted in bold. (ADVP-TMP in this case is empty—cf. Section 3.8 for the handling of null elements in the Treebank.) This is used in our algorithm to annotate the node with

'↑PART=↓', with the provision that it has not already been annotated as the head. Initially, we annotated the node as an oblique, which is sometimes the case, but it is more often the case that it should be the particle.

- -SBJ marks the structural surface subject including those with null subjects. This is particularly useful when the subject appears in non-initial position, as shown in (73) from WSJ 2200, tree 17:

  (73) But Congress did n't anticipate or intend more public debt , say **opponents of the RTC 's working-capital plan**...

  This sentence contains the rule SINV → S , VP NP-SBJ. The text dominated by NP-SBJ is highlighted in bold. The subject is usually identified structurally in English (the subject of the sentence is usually the noun phrase preceding the main verb). However, this tag can be useful in cases such as (73), where the noun phrase occurs after the verb phrase. It is used in (71), where it is already in intial position in the sentence; however, this could be to distinguish between the NP-SBJ and NP-PRD in case of any confusion.

- -TPC marks elements that appear before the subject in a declarative sentences, as in (74) from WSJ 0015, tree 05:

  (74) **The refund pool ... may not be held hostage through another round of appeals**, Judge Curry said *T*

  This sentence contains the rule S → S-TPC ," NP-SBJ VP. The text dominated by S-TPC is highlighted in bold. Our algorithm annotates any elements tagged -TPC with '↑TOPIC=↓'. The fronted element is associated with *T* in the position of the gap. More discussion on fronted elements is given in Section 3.8.1.3.

The Treebank also includes adverbial function tags which we make use of in the 'Catch-all and Clean-up' component of our annotation algorithm to annotate any nodes with these tags as adjuncts ('↓∈↑ADJ') if they have not already received an annotation. The tags are: -DIR (direction), -EXT (extent), -LOC (locative), -MNR (manner), -PRP (purpose or reason), and -TMP (temporal). These tags may be used in future work to include more detail in the f-structures generated automatically by the f-structure annotation algorithm, e.g. to distinguish locative from temporal adjuncts, should this be required.

3.3 Basic Clause Structure

This section describes the guidelines for clause structure, as given in ([Bies et al., 1995], p.11f.). The basic structure of a simple sentence in the Treebank grammar is given in (75):

(75)
```
(S    (NP-SBJ  The boy)
      (VP      throws
               (NP  the ball)))
```

The predicate of the sentence is either the lowest (right-most branching) VP or the phrasal structure immediately under copular *be*. The predicate phrase is tagged -PRD in cases where the predicate cannot be identified by either of the above criteria, so use is made of the functional tag described in (71). However, due to inconsistencies in the tagging of the Treebank, the predicate is not always labelled -PRD if there is no VP, making it more difficult to identify the head of the phrase, as in (76) from WSJ 2008, tree 12:

(76) If you could say their business in the U.S. was mediocre , but **great everywhere else** , that would be fine , " says Bonita Austin , an analyst with Wertheim Schroder & Co

This sentence contains the rule S → ADJP ADVP. The text dominated by S is highlighted in bold. Moved predicates leave a coindexed trace *T* in VP (as explained in (74) and discussed in detail in Section 3.8.1).

Direct object NPs occur after the verb, and are not followed by another NP. Indirect object NPs can occur between the verb and its direct object, as in dative shift constructions such as (77), which contains the rule VP → ADVP VBD NP NP (the text relevant to the rule is highlighted in bold):

(77) "When Mr. Green won a $ 240,000 verdict in a land condemnation case against the state in June 1983 , he says Judge O'Kicki **unexpectedly awarded him an additional $ 100,000**"

Indirect objects can also occur in dative PPs, which are tagged -DTV, as shown previously in (68). Subject NPs, the highest VP, fronted constituents, initial and final punctuation, and most modifiers that precede the verb phrase are attached at S-level.

3.3.1 Clause types

The basic clause types in the Treebank, as outlined in ([Bies et al., 1995], pp.15–25), are S, SINV, SBAR, RRC, SBARQ, SQ, S-CLF, *it*-extraposition, and FRAG.

- The S label is used for simple declarative sentences, passives, imperatives, questions with declarative word order, infinitives, participial and gerund clauses, as in (78) from ([Bies et al., 1995], p.18):

```
(78)    (S  (NP-SBJ-1  The audience)
            (VP        keeps
                       (S  (NP-SBJ  *-1)
                           (VP      leaving
                                    (ADVP-TMP  early)))))
```

  The *-1 indicates an empty subject NP traced to the NP-SBJ-1. LFG limits the use of empty elements in c-structure. However, empty nodes appearing in the Penn-II Treebank can be used to our advantage in our automatic annotation algorithm to provide more information in the f-structures generated. Null elements and trace information in the Penn-II Treebank are described in Section 3.8.

- SINV is the label used for inverted declarative sentences, where the subject follows the tensed verb or modal, as in (79) from ([Bies et al., 1995], p.19):

```
(79)    (SINV  (ADVP-TMP  Never)
               had
               (NP-SBJ  I)
               (VP      seen
                        (NP  such a place)))
```

- SBAR is used for relative clauses and subordinate clauses, including indirect questions, as in (80) from ([Bies et al., 1995], p.20) (cf. Section 3.4 for more detail on subordinate clauses):

```
(80)    (S  (NP-SBJ  Willie)
            (VP      knew
                     (SBAR  that
                            (S  (NP-SBJ  Casey)
                                (VP      threw
                                         (NP  the ball))))))
```

- RRCs (reduced relative clauses) are adjoined to the NP they modify. It is only used if there is no VP and an extra level is needed for proper attachment of sentential modifiers, as in (81) from ([Bies et al., 1995], p.21):

```
(81)    (NP  (NP   110 titles)
             (RRC  not
                   (ADVP-TMP  presently)
                   (PP-LOC    in
                              (NP  the collection))))
```

- The SBARQ label marks direct questions introduced by a wh-word or phrase (i.e. those that contain a gap and therefore require a trace), as in (82) from ([Bies et al., 1995], p.22):

```
(82)    (SBARQ  (WHNP-3  Who)
                (SQ      (NP-SBJ  *T*-3)
                         (VP      will
                                  (VP  throw
                                       (NP  the ball))))
```

- Questions missing both subject and auxiliary are labelled SQ. The label is also used for yes/no questions (including inverted) and inside SBARQ, as in (82), if there is an inverted auxiliary, SQ contains it and the rest of the sentence.

- *it*-extraposition is used for clauses that are extraposed from subject position, which are labelled S or SBAR. The extraposed clause is attached at VP level and adjoined to *it* with *EXP*-attach (cf. (129)). The NP containing *it* and *EXP* is tagged -SBJ, as in (83) from ([Bies et al., 1995], p.25):

```
(83)    (S  (NP-SBJ  (NP  It)
                     (S  *EXP*-1))
            (VP       is
                      (NP-PRD  a pleasure)
                      (S-1     (NP-SBJ  *)
                               (VP      to
                                        (VP  teach
                                             (NP  her))))))
```

- FRAG marks those portions of text that appear to be clauses, but lack too many essential elements for the exact structure to be easily determined, as in (84) from ([Bies et al., 1995], p.26):

  (84)
  ```
  (SBAR-ADV  if)
             (FRAG  (ADJP  possible)))
  ```

  In our automatic annotation algorithm we discount FRAG when obtaining results, as it would be extremely difficult to extract predicate-argument structure (cf. Section 5.2.2).

3.4 Subordinate Clauses

This section deals with clauses introduced by subordinating conjunctions such as *after, while, before*, etc. Such conjunctions introduce finite clauses, past participle clauses, and sentence fragments ([Bies et al., 1995], p.172f.). They can appear as sentential or verbal adjuncts, adjuncts or complements of the noun, predicates, objects of a PP, or in the phrase *so...that.*

The label SBAR is used for subordinate clauses. Relative clauses are adjoined to the NP they modify, as in (85), from WSJ 0162, tree 18:

(85) Prudential-Bache Securities boosted the stock 's short-term investment rating in response to the departure ; analyst John McMillin said he believes the company will turn to new management " **that 's more financially oriented.**

This sentence contains the rule NP → NP " SBAR. The SBAR, highlighted in bold, is a complement to the noun, and is analysed as a RELMOD by our automatic annotation algorithm.

SBARs can be sentential or verbal complements, as shown in (86), from WSJ 0294, tree 08:

(86) That may leave a lot of leeway for U.S. Bankruptcy Judge Burton R. Lifland to decide **what , if anything , the pilots actually collect**

This sentence contains the rule VP → VB SBAR ADVP-LOC. The text dominated by SBAR is highlighted in bold. ADVP-LOC contains an empty node.

Conditional, temporal, and other such adverbial SBARs are attached under either S or VP, depending on whether they precede or follow the main clause, and given the appropriate adverbial function tag (-TMP, -ADV, etc.), see Section 3.2.3. A temporal SBAR attached under a VP is shown in (87), from WSJ 0003, tree 02:

(87) The asbestos fiber , crocidolite , is unusually resilient **once it enters the lungs** , with even brief exposures to it causing symptoms that show up decades later , researchers said

This sentence contains the rule VP → VBZ ADJP-PRD SBAR-TMP , PP. The VBZ tag is used for 3rd person singular present verbs. The text dominated by SBAR-TMP is highlighted in bold. This can then be annotated as '$\downarrow \in \uparrow$ADJ' by the automatic annotation algorithm.

3.5 Wh-Phrases

The bracketing labels used in direct and indirect questions are WHNP, WHADVP, WHADJP, and WHPP ([Bies et al., 1995], p.161*f.*).

- WHNP is used when *what, who,* and *which* stand alone, as shown in the bracketing in (88), from ([Bies et al., 1995], p.161):

(88)
```
(SBARQ  (WHNP-1   Who)
        (SQ       are
                  (NP-SBJ  you)
                  (VP      thinking
                           (PP-CLR  about
                                    (NP      *T*-1)))))
        ?)
```

Who is tagged WHNP-1 here as it is coindexed to the NP trace (*T*-1) in the PP-CLR (an index trace -1 is included in the annotation; traces are discussed in Section 3.8.1). WHNP is also used for *how many*, and as a phrase containing single-word wh-premodifiers, such as *what/whose/which*, as in the phrase *which outfit.* In a relative clause, WHNP can appear as a single word, or as a pre- or post-modified WHNP, as shown in the bracketing in (89), from ([Bies et al., 1995], p.167):

(89)
```
(NP  (NP      the teacher)
     (SBARQ   (WHNP-1  whose scarf)
              (S       (NP-SBJ  I)
                       (VP      admired
                                (NP  *T*-1)))))
```

- WHADVP is used when *why, when, where,* and *how* stand alone, as shown in the bracketing in (90), from ([Bies et al., 1995], p.163):

(90)
```
(SBARQ  (WHADVP-42   How)
        (SQ          did
                     (NP-SBJ  you)
                     (VP      fix
                              (NP         the car)
                              (ADVP-MNR   *T*-42)))
        ?)
```

- WHADJP is used when *how many* modifies a nominal head, and also to bracket phrases consisting of a wh-adverb modifier and an adjectival head, such as *how sweet*, as shown in the partial bracketing in (91) from WSJ 0214, tree 65:

(91)
```
(SBAR-TPC    (WHADJP-1  How sweet)
             (S         (NP-SBJ  it)
                        (VP      (VBZ       is)
                                 (ADJP-PRD  *T*-1))))
```

- WHPP is used with a preposition and a wh-word, such as *At which*, as shown in the bracketing in (92) from WSJ 0044, tree 114:

(92)
```
(SBAR-LOC  (WHPP-1  At which)
           (S       (NP-SBJ  she)
                    (VP      (VBZ       was)
                             (VBN       dismissed)
                             (PP-LOC-1  *T*-1))))
```

3.6 Coordination

Coordinating constructions in the Treebank come in two forms: like and unlike constituent coordinations. Section 3.6.1 deals with the coordination of categories, phrases, and clauses for like constituent coordination, while Section

3.6.2 describes unlike constituent coordination, which has its own category label, UCP (Unlike Constituent Phrase).

In the Treebank, the category for coordinating conjunction, CC, includes *and, '&', but, nor, or, either, neither, both, whether, yet* (as in *It's cheap, cheap yet good*), as well as the mathematical operators *plus, minus, less, times* (i.e., "multiplied by") and *over* (i.e., "divided by"), when they are spelled out ([Santorini, 1991], p.2).

CONJP is used for multi-word conjunctions, such as *as well as* and *if not.* However, mistagging in the Treebank means that it has also been used to tag *if* in the sentence in (93) from WSJ 0359, tree 40. *not* in this case is tagged as RB.

(93) "I think we could very well have an economic slowdown , beginning very soon *if* not already , " he says.

if is fine as a conjunction in this sentence, however, CONJP should be used to tag a phrase, not specific lexical items.

Coordination in the Treebank is a problem for an automatic f-structure annotation process due to the often flat context free rules encountered. Nevertheless, in our automatic annotation procedure, we try to impose a hierarchical structure even on these cases. Initially, coordination was causing a problem in our automatic annotation algorithm, but as it is dealt with in a separate component, it was possible to isolate certain difficulties, and now correctly annotates most of the trees. Section 4.4 gives a description of the coordination component in our automatic f-structure annotation algorithm, and Section 5.3.3 evaluates this component.

3.6.1 Like Category Coordination

Coordination of phrases is represented in the annotation at the lowest level possible. Single word elements of the same syntactic category are coordinated at word-level and are annotated with flat structure ([Bies et al., 1995], p.117f.), as shown in (94):

(94)
```
(S (NP-SBJ  Girls and boys)
   (VP      throw and catch
            (NP balls)))
```

There are cases where different syntactic categories are conjoined, and therefore should be tagged as unlike coordination, but do not appear under the UCP

label, as shown in (95) from WSJ 0426, tree 12:

(95) "In 1972 , the high court swept aside all capital-punishment laws – **federal and state alike** – as unconstitutional."

This sentence contains the rule NP → JJ CC NN. The text dominated by NP is highlighted in bold.

There are also many examples where the mother category differs from that of the conjunct daughters, as in (96) from WSJ 0186, tree 26, which contains the rule NP→ JJ CC JJ coordinating like non-nominal categories (the text dominated by NP is highlighted in bold):

(96) "Among the lot of them , not one is wrestling with **good and evil** , or especially intelligent or even temporarily insane . "

This is a misparse within the Treebank, as the daughter nodes here *good* and *evil* should bear a nominal tag. This causes difficulties for our automatic f-structure annotation algorithm.

Constituents that function as arguments are sisters of the head, i.e. they appear in complement position, as in (97), from WSJ 0045, tree 38, which contains the rule VP→ VB CC VB NP (the text dominated by VP is highlighted in bold), similar to (94), where the NP object is an argument.

(97) 'Messrs. Brownell and Kean say they are unaware of any efforts by McGraw-Hill to **modify or discontinue Scoring High** . "

Such complements present scoping problems for our algorithm, as it can be difficult to ascertain which VB the NP is the object of, or if it should be the object for both. The *RNR* pseudo-attachment notation (cf. Section 3.8.4, (127)) is an avenue planned for further work which may reduce some of the ambiguity.

Adjuncts are adjoined to the phrasal node, as shown in the bracketing in (98), where the ADVP-MNR is attached to the lower verb phrase, rather than the mother VP, which contains the coordinating conjunction.

```
(98)        (S  (NP-SBJ   (NP  These girls)
                          and
                          (NP  those boys))
                (VP       (VP  throw
                               (ADVP-MNR well))
                          and
                          (VP  catch
                               (ADVP-MNR badly))))
```

Again, this can causes scoping problems for an automatic procedure, as it can be difficult to determine the correct scoping of modifiers.

When like clauses are coordinated, the coordinate structure as a whole is meant to have the same label as the coordinated clauses[1], as shown in the bracketing in (99), which contains the rule S → S CC S:

```
(99)        (S   (S   (NP-SBJ  Casey)
                      (VP      threw
                               (NP  the ball)))
                     and
                     (S       (NP-SBJ  Willie)
                              (VP      caught
                                       (NP  it))))
```

There are of course exceptions to this, as can be seen by revisiting sentence (76): "If you could say **their business in the U.S. was mediocre, but great everywhere else,** that would be fine", which contains the rule SBAR → S , CC S (the text dominated by SBAR is highlighted in bold).

It is hard to analyse "great everywhere else" as a sentence (again a misparse in the Treebank tagging), but what is relevant here, in relation to the coding of the Treebank, and for the accuracy of our annotation algorithm, is that the mother node (SBAR) differs from the daughters in the coordinate set. However, our algorithm is implemented in such a way that it is able to handle these exceptions (described in detail in Section 4.4).

3.6.2 Unlike Category Coordination

As mentioned previously, the coordination of unlike phrases has its own category label, UCP, and this is dealt with in a separate section within the coordination

[1]This is similar to the example shown in (96), where the mother category differs from the conjunct daughters.

component of our f-structure annotation algorithm (described in Section 4.4.4). [Cahill et al., 2002c] gave results which showed a decline in results which included coordinate structures. As a result, we decided to deal with UCP and CC in separate sections within the coordination component in order to keep the algorithm as modular as possible, so that additions, changes, and (hopefully) improvements can easily be made.

As stated in the Treebank manual ([Bies et al., 1995], p.120), a coordinate structure joining two or more elements which are typologically different is dominated by UCP. The label UCP is meant to be used in all cases where the constituents to the left and right of the CC are unlike, as shown in the rule in (100), from WSJ 0110, tree 02:

(100)

UCP →	NNP	CC	JJ
	U.S.	and	foreign

However, the coordination of unlike categories occurs, as shown in the rule in (101), where the categories to either side of the conjunction contain no similarity, but the construction is not labelled UCP, from WSJ 1632, tree 07:

(101)

NP→	NP ,	CC	S-NOM
	no alternative	but	to go along

Rules like (101) are not always problematic, if the conjuncts have different, but similar tags, as in (102), from WSJ 0117, tree 03, which contains the rule NP-SBJ → QP NNP CC JJ NNS:

(102) **As many as 70 U.K. and international banks** stand to lose several hundred million pounds should the decision be upheld and set a precedent for other municipalities

The text dominated by NP-SBJ is highlighted in bold. As NNP and NNS are both nominals, it makes the automatic annotation of the rule more straightforward, as a provision has been made in our algorithm to take this into account, described in more detail in Section 4.4.2.

Examples also occur where the UCP label is used but does not actually contain a coordinate structure, as shown in the rule in (103) from WSJ 2045, tree 26:

(103)
UCP →		PP	NP :
	In other commodity markets		yesterday :

In such a case it is difficult to distinguish what should be the head of the phrase. Our algorithm searches for the UCP head from left to right so in this case the PP is annotated as the head.

3.7 Noun Phrases

This section deals with the modification of noun phrases, as detailed in ([Bies et al., 1995], p.179f.). Null elements in noun phrases are described in Section 3.8.

3.7.1 Premodifiers

Single word ADJPs are not labelled as such, as in (104), from WSJ 1131, tree 50:

(104)
```
(NP  (DT  the)
     (JJ  discounted)
     (NN  value))
```

discounted is tagged as JJ within the NP. Hyphenated adjectives are considered to be single words. However, multi-word ADJPs are labelled ADJP within the NP, as in (105), from WSJ 0003, tree 11:

(105)
```
(NP  (ADJP  (RB  very)
            (JJ  modest))
     (NNS   amounts))
```

Nominal modifiers are not labelled, as it can be very difficult to determine the scope (which in turn makes it more difficult for our annotation algorithm to ensure the correct scoping of modifiers, see Section 3.9). Such modifiers just receive their POS tag (e.g. NN, NNS, etc.) within the NP label, as in (106), from WSJ 0324, tree 02, where *share*, is the head of the phrase:

(106)
```
(NP  (NN  one-eighth)
     (NNP QVC)
     (NN  share))
```

Nominal modifiers containing PPs are labelled NAC. PPs within NAC are fully annotated, as in (107), from ([Bies et al., 1995], p.180):

(107)
```
(NP  (NAC   (NN  sale))
            (PP    (IN  of)
                   (NP  (NNS  firecrackers))))
     (NN    law))
```

NAC in this instance will be annotated as an adjunct by our automatic f-structure annotation algorithm, as the NN, *law*, will be annotated as the head of the NP.

The possessive marker is treated as an individual token and tagged 'POS'. However, possessive pronouns are tagged PRP$. The phrase containing a possessive is always labelled NP, even if the possessor is a single word, because the possessive marker is a separate token, as in (108), from ([Bies et al., 1995], p.181):

(108)
```
(NP  (NP  (NNP  Sharon)
          (POS  's))
     (NN   bananas))
```

Nominal modifiers that are expressions of measure or amount, dates, or places, are treated as adjectives if simple (and just receive their POS tag, cf. single word adjectives).

QP is used for multi-word numerical expressions. The determiners *a* and *an* are included in QP where the appropriate interpretation is *one*, as shown in (109), from ([Bies et al., 1995], p.193):

(109)
```
(NP  (QP  less than a)
     (NN  year))
```

Participial modifiers are bracketed like adjectival modifiers. Therefore, the head of an ADJP may be tagged as VBG (a gerund or present participle verb) or VBN, as in (110), from WSJ 0032, tree 4, where *announced* is the head of the ADJP phrase:

```
(110)    (NP  (DT    a)
              (ADJP  (RB  previously)
                     (VBN announced))
              (NN    agreement))
```

Gerundive modifiers are bracketed like nominal modifiers, as in *developing* in (111), from WSJ 0021, tree 01:

```
(111)    (NP (DT  the)
             (VBG developing)
             (NN  world))
```

3.7.2 Postmodifiers

Postmodifiers are adjoined to the phrase they modify, with the exception of clausal complements of certain nouns (e.g. deverbal nouns) ([Bies et al., 1995], p.184f.). Only clausal complements of NP, such as SBAR, are placed inside NP, as shown in the bracketing in (112), from WSJ 0331, tree 29:

```
(112)      (NP  the belief
                (SBAR   that he will find `` values '' of 30 a share ))
```

Consecutive unrelated adjuncts are non-recursively attached to the NP they modify, as shown in the bracketing in (113), from WSJ 0559, tree 03:

```
(113)      (NP   (NP       the first ANC rally)
                 (PP-LOC   inside South Africa)
                 (SBAR     since the black liberation movement...))
```

This results in a flat RHS, which is the general approach taken in the Treebank. It causes problems for the automatic f-structure annotation algorithm when trying to encode generalisations, and when using the resource for parsing, more difficulties are encountered as the number of rules expands, producing a less accurate result.

3.8 Handling of Null Elements in the Treebank

Empty nodes are often used to encode additional information about non-local dependencies and displacement phenomena between words and phrases, and are

important for the interpretation of constructions such as passive, wh-movement, and relative clauses [Johnson, 2002]. Along with the pseudo-attachment notation described in Section 3.8.4, the Treebank annotation scheme provides a set of coindexed null elements, which mark phenomena such as wh-movement, passive, and the subjects of infinitival constructions. The aim is to provide some non-context free annotational mechanism to allow the structure of discontinuous constituents to be easily recovered, and allow for a clear, concise tagging system for some semantic roles. According to ([Marcus et al., 1994], p.1f.), these null elements must be co-indexed with the appropriate lexical material. If this were done consistently, we would have a good chance of capturing all these non-local dependencies. Unfortunately, inconsistency in tagging is a problem when handling data from the Treebank, making the task of annotating it correctly with functional information considerably more problematic.

Co-indexing of null elements is done by adding an integer as a suffix to non-terminal categories, as in S-TPC-2 in Figure 3.1. This integer serves as an id number for the constituent. A null element itself is followed by the id number of the constituent with which it is co-indexed, as in *T*-2 in Figure 3.1. Indices are used only when they can be used to indicate a relationship that would otherwise not be unambiguously retrievable from the bracketing ([Bies et al., 1995]). The predicate argument structure can then be recovered.

In the following sections, we discuss the most frequent and (often, therefore) the most useful types of null elements in the Treebank.

3.8.1 *T*

T is used to mark wh-movement and topicalisation. The trace *T* bears a referential index that corresponds to the identity index of some other constituent in the sentence (moved wh-word, topicalised NP or ADVP, etc.), as shown in the c-structure tree in Figure 3.1, which is associated with the sentence given previously in (74).

3.8.1.1 Wh-Questions

Wh-moved phrases are labelled WHNP and put inside SBARQ (as shown in examples (88)–(89)). They bear an index that matches the reference index on the *T* in the position of the gap, as in (114), from WSJ 0041, tree 62:

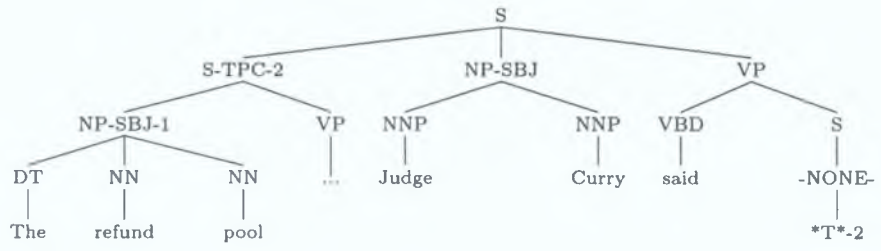


Figure 3.1: Partial tree with -TPC for the string *The refund pool ... may not be held hostage through another round of appeals, Judge Curry said*

```
(114)   (SBARQ-TPC   (WHNP-1  (WP      Who)
                     (S       (NP-SBJ  *T*-1)
                              (VP      (VBZ  's)
                                       (ADVP really)
                                       (VP   lying))))
```

The empty node's preterminal label is NP-SBJ. The bracketing for the SBARQ phrase is shown in (114), which illustrates that the WHNP is coindexed with the empty NP-SBJ node.

3.8.1.2 Relative Clauses

Relative clauses are adjoined to the head noun phrase. As with wh-moved phrases, they bear an index that matches the reference index on the *T* in the position of the gap. The relative pronoun is given the appropriate wh-label, put inside the SBAR level and coindexed with a *T* in the position of the gap. An example sentence is given in (115), from WSJ 0003, tree 08:

(115) Neither Lorillard nor **the researchers who studied the workers** were aware of any research on smokers of the Kent cigarettes

The empty node's preterminal label is NP-SBJ. The bracketing for the NP phrase is shown in (116), which illustrates a trace linking the WHNP node to the NP-SBJ node by means of the reference indices:

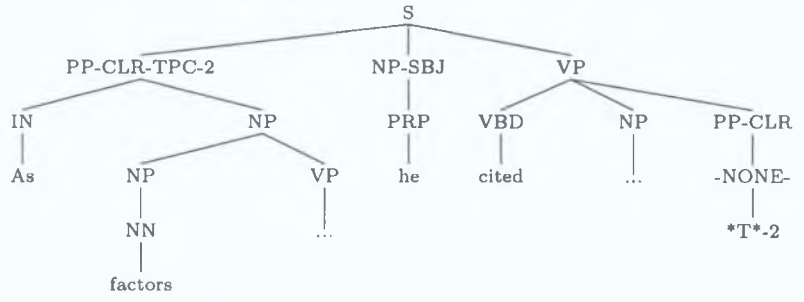


Figure 3.2: Partial tree associated with the string *As factors contributing to the temporary slowdown, he cited ...*

```
(116)   (NP   (NP    the researchers)
              (SBAR  (WHNP-3  (WP      who)
                     (S       (NP-SBJ  *T*-3)
                              (VP      (VBD  studied)
                                       (NP   the workers)))))
```

Relative clauses introduced by a wh-word or by *that* are annotated in the same way; *that* is given the appropriate wh-label, put inside an SBAR level, and coindexed with the *T* in the position of the gap.

3.8.1.3 Fronted Elements

Fronted elements are those that appear before the subject in a declarative sentence. Fronted arguments are placed inside the top clause level (e.g. the main VP, a predicate, the locative complement of *put*, NP, S, SINV, SQ, SBAR), always leave a trace (*T*) and are tagged -TPC, and their identity index matches the reference index on the *T* inserted in the position of the gap, as shown in Figure 3.1. Any constituent tagged -CLR is considered an argument for these purposes: it leaves a *T* and receives the -TPC tag if fronted, as in Figure 3.2, from WSJ 0494, tree 04.

The Treebank manual states that "Fronted arguments are attached under the main clause level ... This holds whether the argument is fronted within a single clause or crosses more than one clause boundary" ([Bies et al., 1995], p.29). However, fronted arguments should not receive the -TPC label in cases where they are associated with a *T* in a clause contained inside the fronted construction, as in (117), from WSJ 0293, tree 27:

(117) THE MILEAGE RATE allowed for business use of a car in 1989 has risen to 25.5 cents a mile for the first 15,000 from 24 cents in 1988 , **the IRS says** ; the rate stays 11 cents for each added mile

The text dominated by S-TPC is highlighted in bold, and the bracketing for which is shown in (118):

(118)
```
(S-TPC-1   (NP-SBJ  the IRS)
           (VP      (VBZ   says)
                    (SBAR  *T*-1)))
```

The topic here is linked to a complement within itself, the reason for which is difficult to understand, and appears to be mistagged by the Treebank taggers. If our algorithm tries to link the topic and complement using functional equations, it will get caught up in an infinite loop, therefore necessitating a provision in our algorithm to ignore topics that are associated with a *T* in a clause inside the fronted construction.

Fronted adjuncts receive the -TPC label only in cases where they are associated with a *T* in a lower clause. A *T* only appears in the annotation if the adjunct is fronted over more than one clause boundary, as in (119), from WSJ 0089, tree 04:

(119) **Of all scenes that evoke rural England , this is one of the loveliest** : An ancient stone church stands amid the fields , the sound of bells cascading from its tower , calling the faithful to evensong

This sentence contains the rule S → PP-TPC-2 , NP-SBJ VP. The text dominated by PP-TPC-2 is highlighted in bold. The index 2 associates the topicalised PP with a PP within the VP, as shown in Figure 3.3.

However, TOPIC in LFG is always associated with a grammatical function, while in our annotation algorithm these adjuncts are annotated as TOPIC (adjuncts are optional elements in an f-structure—they are not necessary in order for the f-structure to be complete and coherent). This is done as a general rule in our algorithm states that any node with the -TPC tag receives the TOPIC annotation. This needs to be investigated in more detail to decide the most appropriate annotations for such cases.

Fronted adjuncts receive function tags, such as -TMP, which marks temporal adjuncts, and -ADV, which marks adverbial adjuncts, as shown in Figure 3.4, the tree associated with the sentence given previously in (63).

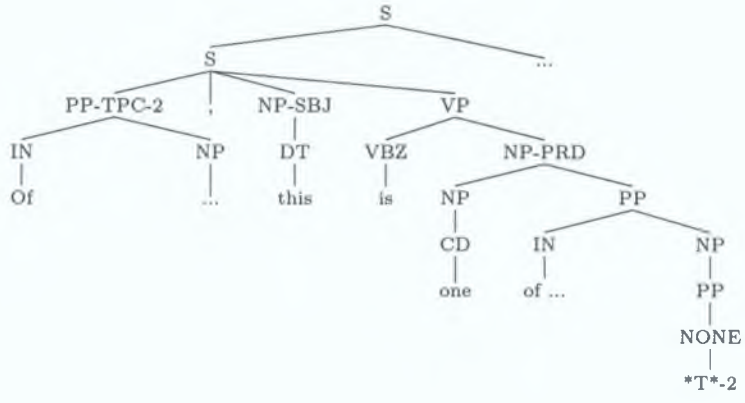


Figure 3.3: Partial tree associated with the string *Of all scenes that evoke rural England , this is one of the loveliest ...*

3.8.2 (NP *)

* is used with the trace of NP-movement, controlled PRO, or arbitrary PRO. In controlled PRO the reference is known, as opposed to arbitrary PRO in which the reference is undetermined. * always appears inside an NP, i.e. (NP *).

In the Treebank, (NP *) bears a reference index whenever it is fairly clear what nominal it is controlled by, corresponding roughly to controlled PRO and the passive trace ([Bies et al., 1995], p.68). Figure 3.5 shows the partial tree containing controlled PRO, associated with the sentence from WSJ 0332, tree 07, shown in (120):

(120) Orkem said **it eventually would seek to** make a public share offering in its U.K. business

The reference number *-1 associates the empty NP-SBJ node with NP-SBJ-1 containing *it*. Unlike *T*, * may appear without a reference index. Unindexed (NP *) corresponds roughly to arbitrary PRO. Figure 3.6 shows the partial tree containing arbitrary PRO, associated with the sentence from WSJ 0333, tree 05, shown in (121):

(121) Mr. Edelman said the decision has **nothing to do** with Marty Ackerman

In passives, the surface subject is tagged -SBJ, a passive trace is inserted after the verb, indicated by (NP *), and co-indexed to the surface subject (i.e.

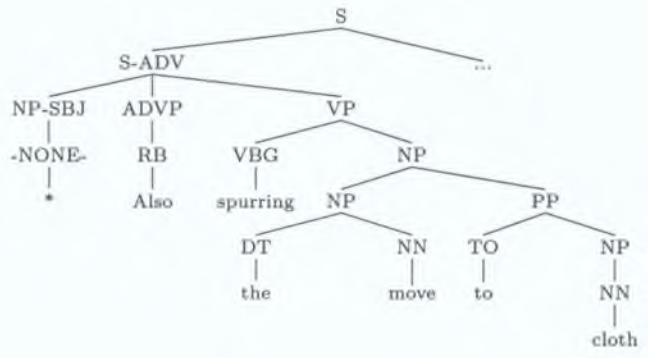


Figure 3.4: Partial tree associated with the string *Also spurring the move to cloth ...*

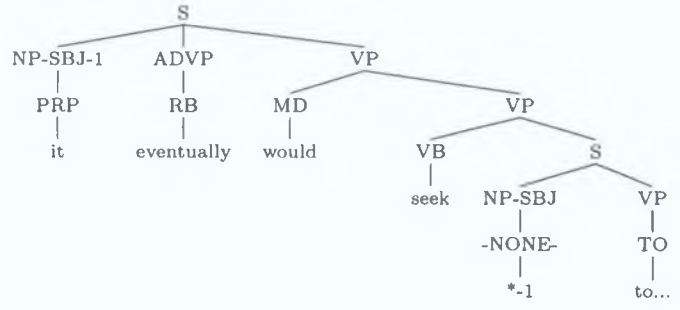


Figure 3.5: Partial tree containing controlled PRO for the string *Orkem said it eventually would seek to make a public share offering in its U.K. business*

the logical object). If present, the logical subject (within the PP containing *by*), is a child of the VP, and is tagged -LGS ([Marcus et al., 1994], p.3), as shown in Figure 3.7, associated with the sentence given previously in (69).

The NP marked with -LGS, *somebody*, can be treated as the subject of the sentence when building the predicate argument structure. We make use of this passive trace NP *, together with the occurrence of the -LGS tag to annotate sentences as 'passive = +'. This is described in more detail in Section 4.5.1.

3.8.3 The Null Complementiser

0 is the null complementiser. It is labelled 0 if it corresponds to *who, which, that*, etc. inside zero relative clauses, (i.e. where there is no overt wh-element or *that*), as in (122), from WSJ 0018, tree 16:

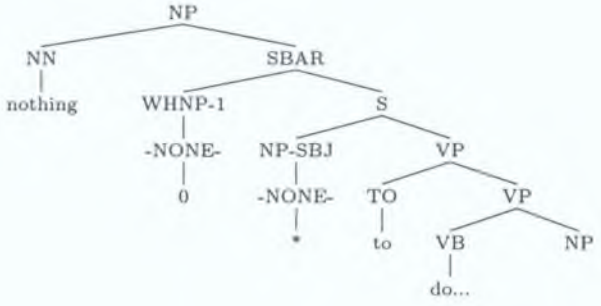


Figure 3.6: Partial tree containing arbitrary PRO for the string *Mr. Edelman said the decision has nothing to do with Marty Ackerman*

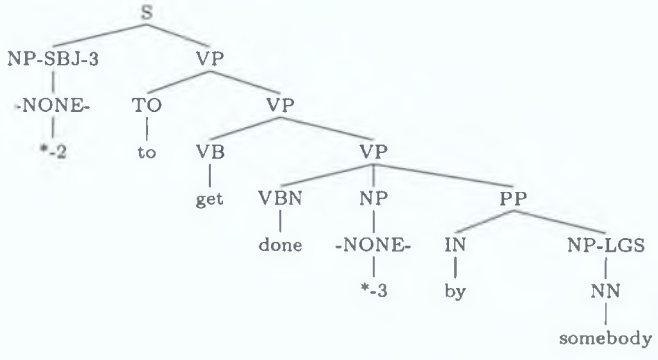


Figure 3.7: Passive tree containing (NP *) trace for the string *They must figure that justice has to get done by somebody, but know it wo n't be done by Congress*

(122) You either believe Seymour can do it again or you do n't

The partial tree associated with (122) is shown in Figure 3.8. There is no overt *that*, but -NONE- inside the SBAR could be replaced with *that*.

The null complementiser is also used inside infinitival relative clauses for NP objects and subjects. (WHNP 0) is used in the case where the missing element can be paraphrased as *in which*, *at which*, *for which*, etc. However, in Figure 3.6, this does not appear to be the case, illustrating a contradiction between what is stated in the Treebank manual, and what has been tagged. The null complementiser introduces most tensed complement clauses.

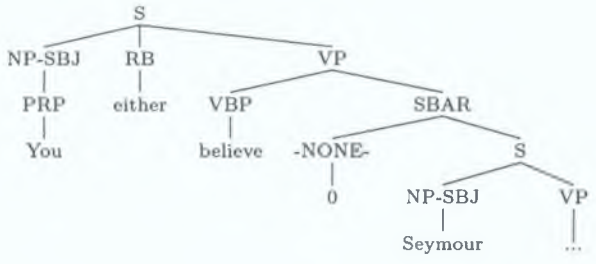


Figure 3.8: Partial tree containing null complementiser for the string *You either believe Seymour can do it again or you do n't*

3.8.4 Pseudo-attachment

There are many limitations using the bracketing schema described in Section 3.2.2, as is discussed in ([Marcus et al., 1993], p.19), including the fact that many otherwise clear adjunct/argument relations in the corpus are not indicated, due to the flat context free representation.

However, the pseudo-attachment notation, which represents the "underlying" position of extraposed elements, can be extended to indicate a variety of crossing dependencies. In addition to being attached in its superficial position in the tree, the extraposed constituent is pseudo-attached within the constituent to which it is semantically related ([Marcus et al., 1993], p.17). There are four different forms of pseudo-attachment in the Treebank:

- *ICH* (**I**nterpret **C**onstituent **H**ere) pseudo-attach is used for simple extraposition, using co-indexation to indicate discontinuous structures, as shown in (123) from WSJ 0113, tree 12:

  (123) It also helps explain **the reluctance of the major farm lobbies and many lawmakers to make any significant changes in the 1985 farm program next year**

  This sentence contains the rule NP → NP PP S-1 (highlighted in bold), and is associated with the bracketing shown in (124), which ensures that the meaning of the sentence is understood correctly, as the S-1 is coindexed with a slot inside the daughter NP:

(124)
```
(NP   (NP  (DT the)
           (NN reluctance)
           (S  *ICH*-1))
      (PP  ...)
      (S-1 to make...))
```

- The *PPA* (**P**ermanent **P**redictable **A**mbiguity) tag is used in cases in which one cannot tell where a constituent should be attached, even given context, as shown in (125) from WSJ 0375, tree 1:

  (125) Deere & Co. said it reached a tentative agreement with the machinists ' union at its Horicon , Wis. , plant , ending **a month-old strike by workers at the facility**

  This sentence contains the rule NP → NP PP PP-LOC-1 (highlighted in bold), and is associated with the bracketing shown in (126), indicating that *at the facility* can be attached to the mother NP node, or to the daughter NP node containing *workers*:

(126)
```
(NP   (NP  a month-old strike)
      (PP  (IN  by)
           (NP  (NNS     workers))
                (PP-LOC  *PPA*-1))
      (PP-LOC-1 (IN      at)
                (NP      the facility))))
```

- The *RNR* (**R**ight **N**ode **R**aising) tag is used in conjunctions where the same constituent appears to have been shifted out of both conjuncts, as shown in (127) from WSJ 0562, tree 7:

  (127) Companies that actually market speed as part of their service train their managers to **lead and participate in teams that increase speed and improve quality in everyday operations**

  This sentence contains the rule VP → VP CC VP NP (highlighted in bold), and is associated with the partial bracketing shown in (128), indicating that the NP constituent *teams that increase speed and improve*

quality in everyday operations appears to have been shifted out of both VP conjuncts:

(128)
```
(VP   (VP  (VB  lead)
           (NP  *RNR*-1)))
      and
      (VP  (VB  participate)
           (PP-CLR  (IN  in)
                    (NP  *RNR*-1)))
      (NP-1  teams that...))
```

- The *EXP* (**EXP**letive) tag is used with extraposed sentences which leave behind a semantically null *it*, as shown in (129) from WSJ 0585, tree 27:

  (129) It 's harder **to sell stocks when the sell programs come in because some market makers do n't want to take the orders**

  This sentence contains the extraposed sentence (highlighted in bold), and is associated with the syntactic structure shown in (130), showing the semantically null *it* left behind in the NP-SBJ, which is coindexed to S-1:

(130)
```
(S   (NP-SBJ  (NP-SBJ  It)
              (S       *EXP*-1))
     (VP   's
           (ADJP-PRD  harder))
           (S-1       (NP-SBJ  *)
                      (VP      to
                               (VP  sell stocks...))))
```

To date, we have not made use of the pseudo-attachment provided by the Treebank syntactic annotators. However, it is an avenue for future work, and as they appear to be used consistently and correctly, should prove useful to include more detail in the f-structures generated by the automatic annotation algorithm.

3.9 Linguistics of the Treebank in relation to LFG

"Treebank grammars typically involve large sets of lexical tags and non-lexical categories, as syntactic information tends to be encoded in monadic category symbols" [Frank et al., 2001]. They often feature flat rules in trees that do not express linguistic generalisations. In this section, we discuss how published LFG analysis and theory differ from the annotation principles in the Treebank, and how this, along with the tagging and syntactic annotation inconsistencies in the Treebank, causes problems for our automatic f-structure annotation algorithm. The implementation of this algorithm is described in detail in Chapter 4.

3.9.1 Phenomena in the Treebank vs. LFG Theory

The linguistics of the Treebank violates the endocentric property of X′ theory (which states that all phrases have heads of the same category, cf. Section 2.3.1.1), as has been shown in numerous examples throughout this chapter, e.g. (76), and also in coordinate structures, where the mother category differs from that of the conjunct daughters, e.g. (96). In LFG, S is assumed to be the only exocentric category, i.e. the only category with no c-structure, or lexical head. The following sections discuss further problems encountered when trying to apply theoretical LFG to the practical data found in the Treebank.

3.9.1.1 Empty Nodes

Standard LFG theory limits the use of empty elements in c-structure (in fact some theories of LFG state that there are no empty elements in c-structure), and disallows the use of syntactic phrase structure nodes that provide only redundant information ([Falk, 2001], p.34). Empty nodes are included in the trees in the Treebank to mark re-entrancy, as discussed in Section 3.8. This can be made use of in our algorithm to capture passives, long distance dependencies and wh-movement (use of this trace and index information is described in detail in Section 4.5).

3.9.1.2 Topic

TOPIC in LFG is always associated with a grammatical function, but in the Treebank tagging, fronted adjuncts receive the -TPC label in cases where they

are associated with a *T* in a clause contained inside the fronted construction, as described previously in Section 3.8.1.3, example (119). As our algorithm is implemented using generalisations, these adjuncts are annotated as TOPIC. This is done as a general rule in our algorithm states that any node with the -TPC tag receives the TOPIC annotation. This needs to be investigated in more detail to decide the most appropriate annotations for such cases.

3.9.2 Coordination in the Treebank in relation to LFG

LFG analysis of simple coordination can be shown by revisiting (43), p.32, ([Butt et al., 1999], p.142):

SCCOORD (CAT) =	CAT	CONJ	CAT
	$\downarrow \in \uparrow$	$\downarrow = \uparrow$	$\downarrow \in \uparrow$

This c-structure rule allows any category, e.g. S, VP, PP, to be coordinated. The conjuncts form a set via the '$\downarrow \in \uparrow$' annotation. However, when working with real data from the Treebank, it is not always the case that the categories on either side of the conjunction are the same, as in the rule in Figure 3.9,[2] which requires the annotations as shown.

NP →					
	NN ,	NN ,	NN	CC	NNS
	$\downarrow \in \uparrow$CONJ	$\downarrow \in \uparrow$CONJ	$\downarrow \in \uparrow$CONJ	$\uparrow = \downarrow$	$\downarrow \in \uparrow$CONJ
	sex ,	class ,	race	and	politics

Figure 3.9: NP Coordination for the string *sex, class, race and politics*

sex, class, race and *politics* are all equal conjuncts, but *politics* receives a different tag to the others as it is a plural noun (cf. Section 3.2.1 for the complete list of tags used). NP coordination causes particular difficulties, due to the scoping of modifiers, as discussed previously in Section 2.6 and Section 3.7. An example of this is shown in the sentence in (131), from WSJ 1405, tree 28:

[2]Revisiting Figure 4.13, p.105.

(131) Gen. Scowcroft knows as well as anyone that one of the biggest dangers he faces is that NSC staffers working in relative anonymity will take over policy-making and operational tasks that are best left to **bigger and more experienced State Department and Pentagon bureaus**

This sentence contains the rule NP → ADJP NNP NNP CC NNP NNS, with the text dominated by the NP highlighted in bold. The rightmost NNS, *bureaus*, is the head of this phrase , CC is the conjunction, and the NNPs to the immediate left and right, *Department* and *Pentagon* of the CC are the conjuncts. The NNP to the left of the left conjunct, *State*, modifies the left conjunct, not the entire coordinate structure or the head of the phrase. Due to the flat analysis by the Treebank taggers, this is a very complex phrase to analyse. The scoping of the modifier, *State*, is wrong in our algorithm as it modifies the head of the phrase. Named entity recognition could help disambiguate complex coordinate phrases such as is—we have not yet integrated a named entity recogniser with our automatic f-structure annotation algorithm, but it is an avenue for futher work.

It is problematic to encode these complex coordinate phrases in an automatic procedure, as the Treebank also contains instances of a similar nature where the noun to the left of the left conjunct does modify the coordinate structure, as shown in (132), from WSJ 1722, tree 07:

(132) It also licenses **optically based data storage and retrieval devices**

This sentence contains the rule NP → ADJP NNS NN CC NN NNS, with the text dominated by the NP highlighted in bold. In this case *data* modifies the coordinate structure, not just the conjunct. The NNS, *devices* is annotated as the head of the phrase, but the scoping of the modifier, *data*, is wrong in our algorithm as it modifies the head of the phrase, not the coordinate structure. It is hard to generalise in such cases. This becomes more complicated still with the inclusion of additional adjectival modifiers and adjectives which are part of the conjunct set, as shown in (133), from WSJ 1317, tree 41:

(133) He also will sit on the company 's corporate planning and policy committee, made up of **the top corporate and operating executives**

This sentence contains the rule NP → DT JJ JJ CC NN NNS, with the text

dominated by the NP highlighted in bold. The JJ to the left of the conjunction, *corporate*, is a conjunct, as is the NN, *operating*, to the right. The JJ to the left of that conjunct, *top*, modifies the head of the phrase, the NN, *executives*. However, in (134), from WSJ 2428, tree 55, the JJ to the left of the conjunct modifies the conjunct rather than the head of the phrase.

(134) In July , Southmark Corp. , **the Dallas-based real estate and financial services company** with about $ 1.3 billion of junk bonds , voluntarily filed for protection under U.S. bankruptcy law .

This sentence contains the rule NP → DT JJ JJ NN CC JJ NNS NN, with the text dominated by the NP highlighted in bold. The JJ to the immediate right of the conjunction, *financial*, is not a conjunct, but rather modifies the NNS to its right, *services*, which is the right conjunct.

As can be seen from the above examples, NP coordination is not a trivial matter, hence the reason that we have tried to generalise our algorithm to get the most accurate possible annotations, as it is extremely difficult to automatically annotate each and every case 100% correctly.

([Butt et al., 1999], p.142) suggest that for two-part conjunctions, such as *either... or*, the first half of the conjunction is annotated as a 'PRECONJ', which is constrained to occur with a particular 'CONJ-FORM', provided by its paired conjunction. In ([Dalrymple, 2002], p.367), the same idea is proposed. The features PRECONJ and CONJ are classified as non-distributive features. The constraining equation for *both* is given in (135):

(135) *both*

PreCnj (↑ PRECONJ) = BOTH

Cnj (↑ CONJ) $=_c$ AND

This ensures that *both* can only occur with the CONJ value *and*, thus preventing an ungrammatical phrase such as **both walked or ran.* No special rules for two-part conjunctions have yet been implemented in our automatic f-structure annotation algorithm, but are planned for future work. At the moment, a rule containing the structure *either... or*, receives the annotations as shown in (136) from WSJ 2321, tree 20:

(136)

ADJP →	CC	JJ	CC	JJ
	↓∈↑ADJ	↓∈↑CONJ	↑=↓	↓∈↑CONJ

"I do n't feel either hard or soft"

Cases occur in the Penn-II Treebank where only a punctuation marker separates two complete sentences, as in (46), p.33, which contains the rule S → S : S, where the 'lexical entry' for the colon is '- -'.

One LFG approach to handling structures where a punctuation marker is analysed as the head is to have a special coordination rule for the highest category under ROOT, which allows certain types of punctuation in place of a conjunction ([Butt et al., 1999], p.143). Our approach is to allow certain types of punctuation in place of a conjunction and to annotate the punctuation accordingly as head.[3] The two conjoined sentences are members of the conjunct set, each conjunct containing the complete f-structure for each sentence; and the '- -' is annotated as the head (the f-structure is shown in Figure 4.10, p.104, and the annotation method is discussed in Section 4.4.2).

3.10 Summary

This chapter outlines the tagging and bracketing principles in the Treebank, with particular attention paid to coordinate structures, the modification of noun phrases, and the handling of null elements. Treebank grammars typically involve large sets of lexical tags and non-lexical categories, as syntactic information tends to be encoded in monadic category symbols. They often feature flat

[3]Punctuation is annotated as the head only in these special cases, as it usually receives no annotation.

rules in trees that do not express linguistic generalisations, making the task of automatic annotation more complex, as annotation principles have to identify sub-sequences on the RHS of the corresponding CFG rule for annotation. Passivisation and long distance dependencies are encoded in the Penn-II Treebank by means of trace and index information on null elements, differing to theories of LFG which limit the use of empty elements in c-structure. However, use can be made of this in our automatic f-structure annotation algorithm, so this chapter looks at the linguistics employed in the Treebank in relation to published LFG analyses and relates it to our algorithm, illustrating cases where the tagging guidelines of the Treebank are not followed correctly, and describing how that can make the encoding of generalisations in an automatic process more difficult.

Chapter 4

The Linguistic Information encoded in the Automatic F-Structure Annotation Algorithm for the Penn-II Treebank

4.1 Introduction

The linguistic design embodied in the Penn-II Treebank (as described in Chapter 3) is used in conjunction with the theory of LFG (as described in Chapter 2) to construct an automatic f-structure annotation algorithm for the WSJ section of the Treebank. This chapter is divided into five main sections—describing briefly some background to automatic annotation, and then detailing the methodology used to construct each of the four components (Left/Right (L/R) context, Coordination, Traces, and 'Catch-all and Clean-up' principles) in the automatic f-structure annotation algorithm. This author's work mainly entailed providing a specification for the treatment of the linguistic information, and the implementation of the algorithm was undertaken by [Cahill, forthcoming]. In our initial research ([Cahill et al., 2002a], [Cahill et al., 2002b], [Cahill et al., 2002c]), we designed a coarse-grained automatic annotation algorithm. This generated

'proto' f-structures, which encoded basic predicate-argument-modifier structure (with some f-structures which were partial or unconnected, i.e. the trees were associated with two or more f-structures). This version of the algorithm did not use the traces and empty productions provided by the Treebank to encode "moved" or "extraposed" material (long distance dependencies) to reflect them as corresponding re-entrancies in the f-structure. However, the algorithm has since been refined to produce more detailed f-structure representations, encoding re-entrancies for topicalisation and wh-movement, as well as annotating passive, and it is this improved methodology which is described in this chapter. Evaluation of the automatic procedure is given in Chapter 5.

4.2 Automatic Annotation

Methods have been developed for automatically annotating treebank resources with f-structure information ([Frank, 2000], [Sadler et al., 2000], ([Frank et al, 2003]). They are fine-grained and produce detailed f-structure representations, but have only been developed to date using the Susanne Corpus[1], which comprises an approximately 130,000-word subset of the Brown Corpus of American English, and the publicly available subset of the AP Treebank[2], which consists of 100 sentences of newswire reports, with a CFG rule base of about 500 CFG rules. [Sadler et al., 2000] use a regular expression-based, indirect automatic f-structure methodology. [Frank, 2000] developed a method that is in many ways a generalisation of the regular expression-based annotation method. This method can access arbitrary tree fragments and can be order-dependent or order-independent (cf. [Frank et al, 2003] for more detail on these approaches), whereas the algorithm described in this thesis is order-dependent and can only access local subtrees of depth one, i.e. CFG rules. These methods, however, have not yet been tested to see whether they scale up to larger corpora, of the size of the Penn-II Treebank.

[Forst, 2003] reports on the TIGER treebank, which consists of 36,000 syntactically annotated German newspaper sentences, in which the annotation consists of generalised graphs, i.e. trees which may contain crossing and secondary edges. Dependency information is expressed explicitly in the edge labels, so it does not need to be annotated, with f-descriptions (cf. Section 2.3.3). This makes it easier to convert the trees into f-structures than from the Penn-II

[1] Available from `http://www.grsampson.net/RSue.html`

[2] Available from `http://www.comp.lancs.ac.uk/computing/research/ucrel/corpora.html`

Treebank, as a lot of information is already explicitly marked in the TIGER graphs.

We have designed an automatic f-structure annotation algorithm, which scales up to the 50,000 sentences in the Wall Street Journal (WSJ) section of the Penn-II Treebank. In our initial research ([Cahill et al., 2002a], [Cahill et al., 2002b], [Cahill et al., 2002c]), we designed a coarse-grained automatic annotation algorithm, which produced 'proto' f-structures, which encoded predicate-argument-modifier structures (although some f-structures were partial or unconnected, associating trees with two or more f-structures). This version of the algorithm did not use the traces and empty productions provided by the Treebank to encode 'moved' or 'extraposed' material (long distance dependencies) to reflect them as corresponding re-entrancies in the f-structure. However, the algorithm has since been refined to produce more detailed f-structure representations, encoding long distance dependencies such as topicalisation and wh-movement, as well as marking passive constructions.

The automatic f-structure annotation algorithm is implemented in Java and recursively traverses a tree in top-down, mostly left-to-right fashion. Complete annotation of the WSJ section of the Penn-II Treebank takes less than 30 minutes on a Pentium IV PC. Once annotated, the feature structure annotations for each tree are collected and fed into a constraint solver implemented in Prolog, which can cope with equality constraints, disjunction and simple set-valued feature constraints. In order to keep the main algorithm modular so that additions and changes can easily be made, it is divided into four components: Left/Right (L/R) context, Coordination, Traces, and 'Catch-all and Clean-up' principles, as shown in the flow diagram in Figure 4.1, which are described in separate sections in this chapter.

Figure 4.1: Four stages of the Automatic Annotation Algorithm

4.2.1 Head Rules

The annotation procedure begins by locating the head daughter, i.e. a basic lexical category such as noun, verb, or adjective, is said to be the head of a noun, verb or adjective phrase, respectively. In order to compute the head, we initially used [Collins, 1996] scheme, but found [Magerman, 1994] gave improved

LHS	Direction	RHS
ADJP	right	% QP JJ VBN VBG ADJP $ JJR JJS DT FW **** RBR RBS RB
ADVP	left	RBR RB RBS FW ADVP CD **** JJR JJS JJ
CONJP	left	CC RB IN
FRAG	left	
INTJ	right	
LST	left	LS
NAC	right	NN NNS NNP NNPS NP NAC EX $ CD QP PRP VBG JJ JJS JJR ADJP FW
NP	right	EX $ CD QP PRP VBG JJ JJS JJR ADJP DT FW RB SYM PRP$
PP	left	IN TO FW
PRN	left	
PRT	left	RP
QP	right	CD NCD % QP JJ JJR JJS DT
RRC	left	VP NP ADVP ADJP PP
S	right	VP SBAR ADJP UCP NP
SBAR	right	S SQ SINV SBAR FRAG X
SBARQ	right	SQ S SINV SBARQ FRAG X
SINV	right	S VP VBZ VBD VBP VB SINV ADJP NP
SQ	right	VP VBZ VBD VBP VB MD SQ
UCP	left	
VP	left	VBD VBN MD VBZ TO VB VP VBG VBP ADJP NP
WHADJP	right	JJ ADJP
WHADVP	left	WRB
WHNP	right	WDT WP WP$ WHADJP WHPP WHNP
WHPP	left	IN TO FW
X	left	

Table 4.1: Magerman's Head Rules

results. An example of this can be seen in the order given in which categories are evaluated as the most likely heads for WHADVP. Collins lists CC as the most likely head, followed by WRB. Magerman looks for WRB as the most likely head, which, as a wh-adverb, is more likely to be head of a wh-adverb phrase than CC, the coordinating conjunction. This also integrates better with our algorithm, in which coordination is handled in a separate component and it is only within coordinate structures that CC is most likely to be head. The full version of Magerman's head lexicalised grammar annotation scheme is given in Table 4.1.

For each category on the left-hand-side (LHS), Magerman gives a list of categories (excluding functional tags) in the order in which they are most likely to appear as head on the right-hand-side (RHS), stating the direction in which the head is searched for (right indicates that the head is searched for from right to left). For example, WHADJP on the LHS, with JJ ADJP as possible heads

(in that order of likelihood), searches the RHS from right to left, looking first for a JJ as the head of the phrase, then ADJP. If neither are found, the rightmost non-punctuation node will be annotated as the head.

As stated previously, this author's work mainly entailed providing a specification for the treatment of the linguistic information, and the actual implementation of the algorithm was undertaken by [Cahill, forthcoming]. Analysing the data in the Treebank allowed the estimation of the most likely heads. Magerman's order did not give us optimal results (although still better than Collins'), so we amended the NP, QP, S, SBAR, SINV, SQ, UCP, VP and WHNP categories to the order given in Table 4.2.

LHS	Direction	RHS
NP	right	EX $ CD QP PRP VBG JJ JJS JJR ADJP DT FW RB SYM PRP$ **** POS PRN
QP	right	$ % CD NCD QP JJ JJR JJS DT
S	right	TO VP SBAR ADJP UCP NP
SBAR	right	IN S SQ SINV SBAR FRAG X
SINV	right	MD IN VBZ VBD VBP VB VP S SINV ADJP NP
SQ	right	MD VBZ VBD VBP VB VP SQ
UCP	left	CC S **** ADVP RB PRN
VP	left	MD VBD VBN VBZ VB VBG VBP POS VP TO ADJP JJ NP
WHNP	right	NN NNS NNP NNPS NP WDT WP WP$ WHADJP WHPP WHNP

Table 4.2: Our (Differing) Head Rules

The NP rule has been extended to include '**** POS PRN', which indicates that any other category should be the head sooner than PRN or POS. POS or PRN can occur as the only category on the RHS in an NP, in which case they will be annotated as the head. $, followed by % are the most likely heads in a QP, so we changed the order of the list to search for them first. The modal *to* is the more likely head in a sentence than S, so we included TO at the beginning of the list. IN (tagging a preposition, or a subordinating conjunction, such as *that*, *as*, etc., cf. Section 3.2.1 Table 3.1, p.44, for a complete listing of the tags used in the Treebank with a description) is the most likely head of an SBAR, so it has been included at the beginning of the list. Inspecting extracted rule frequencies from the Treebank allowed us to research the most likely heads.

A finite verb is always more likely than a verb phrase to occur as head of inverted sentences (SINV), or sentences involving questions (SQ), so the order has been re-arranged for both categories. In Magerman's list, no head is specified in a UCP to be the most likely. Following the analysis on the UCP rules, we

found CC to be the most likely head, followed by S. Any category other than ADVP, RB, or PRN is then most likely to be head. A finite verb is also more likely than a verb phrase to occur as head of a VP rule, as in (137), from WSJ *0004*, *tree 06*, which contains the rule VP → VBP VP.

(137) Shorter maturities **are** considered a sign of rising rates because portfolio managers **can** capture higher rates sooner.

are in this sentence is tagged as a VBP. In a rule such as this, the VBP, under our analysis, is the head of the phrase,[3] and the VP can then be annotated as an XCOMP. This, however, was not stated in Magerman's ordering; neither did he allow for a modal verb to be head of a SINV, SQ or VP rule. (137) also contains the rule VP → MD VP. *can* is tagged as an MD. In a rule such as this, the MD under our analysis is the head of the phrase, and the VP can again be annotated as an XCOMP.

The tag POS has been used incorrectly in verb phrases to tag a third person singular present verb *'s* (as described in Section 3.2.1), as in (138), from WSJ *0071*, tree 56:

(138) By January it should be fairly clear what **'s** hot – and what **'s** not

This mistagging is taken into account by our head rules, so POS (as a verb in the case) will be annotated as the head of the verb phrase.

We have included nominals as the most likely heads in a WHNP, which would seem likely as WHNP is a wh-noun phrase; WP pronouns and wh-phrases are annotated as the possessive marker with '↑POSS=↓'. This is differentiated from possessives in NPs (e.g. PRP$, the possessive pronoun) in which the mother NP node is annotated with '↑POSS=↓', and the POS-tagged node receives no annotation.[4] The changes were implemented following further analysis of the grammar rules and the ordering of the head rules, and gave improved results (evaluation and results are described in Chapter 5).

4.2.2 Lexical Macros

At the lexical level, we create a lexical macro which associates each preterminal category type in the Treebank with the required f-structure information. We

[3] Under a different analysis, *consider* could be the head as this is a passive construction.

[4] Note that this is done to indicate that it is the mother node, the NP, which is the possessive phrase. Here, we deviate from our annotation principles and standard LFG theory, in which no mother (i.e. LHS category) receives an annotation. This may need to be revisited in further work to determine if it is the best annotation method for possessive noun phrases.

use the XLE morphological component to lemmatise the Treebank strings, i.e. the words are returned to their base forms.The XLE component provides pred values—all other morpho-syntactic information comes from the lexical macros. Using the part-of-speech (POS) tag information provided by the Treebank, we can automatically extract the lemmas and use them in our lexical macros to include number, person and predicate information. The lexical macros themselves were constructed manually, based on those used in the AP Treebank [Sadler et al., 2000]. They are looked up by the algorithm and are included in the program in the form of a hash table. An example entry for a plural common noun is given in (139):

(139) NNS: ↑PRED=lemma, ↑NUM=pl, ↑PERS=3, ↑CAT=nns

This includes the word in its root form (lemma), and person, number, and category information for that noun, which is the same for all plural common nouns. A lexical macro for a past tense verb is given in (140):

(140) VBD: ↑PRED=lemma, ↑TENSE=past, ↑CAT=vbd

The lexical macro for WP (wh-pronoun) and WP$ (possessive wh-pronoun) includes '↑wh=+'; this feature is then included in the f-structure for any strings containing wh-words.

The tag TO is used for the preposition *to* and for the modal *to* (cf. Section 3.2.1). As the modal *to* occurs more frequently in the Treebank than the preposition *to*, we include the lexical macro '↑to=+,↑inf=+', which is wrong for the preposition. To correct this, the algorithm checks (in the Catch-all and Clean-up component, cf. Section 4.6) to ensure that TO appears under a VP, if this is not the case, the lexical macro '↑pred='to' is included instead. A full list of lexical macros is given in the appendix.

4.3 L/R Context Principles

L/R Context principles are formulated in terms of a simple partitioning of local trees of depth one (context-free rules) into head, left context and right context on immediate daughters, and are applied only if there is no CC or CONJP present on the RHS (cf. Section 4.4.1 and Section 4.4.2), or if the mother category is not a UCP (cf. Section 4.4.4). In these two cases, the coordination component is applied instead.

Grammatical Function	Description
ADJ	Adjunct, or Modifier
APP	Apposition
COMP	Complement
CONJ	Conjunction
FOCUS	Focus
OBJ	Object
OBJ2	Object
OBL	Oblique
PART	Particle of the verb
POSS	Possessive marker
RELMOD	Relative clause
SPEC	Specifier
SUBJ	Subject
TOPIC	Topic
TOPICREL	Topic in a relative clause
XCOMP	Complement

Table 4.3: Grammatical Functions in LFG

Section 3.2 discusses the categories and tags used in the Treebank. As mentioned in Section 4.2, we distinguish (using the traditional LFG distinction) between arguments (subcategorisable grammatical functions such as SUBJ, OBJ, OBJ2, OBL, XCOMP, COMP, POSS) and adjuncts (non-subcategorisable grammatical functions such as ADJ, FOCUS, TOPIC) in left and right contexts, and annotate tree nodes accordingly with functional annotations. The grammatical functions used in our algorithm are given in Table 4.3, and are described in more detail in Section 2.3.2.1.

ADJ, APP, and CONJ are always used in a set (cf. Annotation Matrices in Appendix B and C), therefore allowing multiple occurrences of adjuncts, appositions, and conjuncts. In the f-structure we also include the attributes given in Table 4.4.

Using grammatical functions given in Table 4.3, we can construct an 'annotation matrix' for each LHS category in the Treebank grammar.

4.3.1 Annotation Matrices

An interesting property of treebanks is that there is a small number of frequently occurring rules which, for each rule LHS category, expand those categories, followed by a large number of less frequent rules, many of which occur only once or twice in the whole treebank.

The annotation matrices we have constructed state generalisations in the form of a tripartition of local trees (of depth one, i.e. CFG rules). This allows

Attribute	Description
adegree	Used with adjectives and adverbs to measure the degree of comparison (e.g. comparative or superlative)
case	Indicates case (e.g. dative or genative)
if	no pred is included for the word *if*, atrribute takes the form "↑if=+'
inf	Marks infinitives
lgs	Marks the logical subject in passives
modal	Marks the modal *to*
num	Includes number information
participle	Marks participles
passive	Marks passives
pers	Includes person information
tense	Includes tense information
that	no pred is included for the word *that*, atrribute takes the form "↑that=+'
to	no pred is included for the modal *to*, atrribute takes the form "↑to=+', this ensures no clashes occurs when *to* is annotated as a co-head in a VP
wh	no pred is included for wh-words, atrribute takes the form "↑wh=+'

Table 4.4: Grammatical Attributes used in our Automatic F-structure Annotation Algorithm

a compact encoding of linguistic generalisations, with unfortunately some lack of detail and some mistakes. However, this gives us a methodology that can be easily scaled up to all of the Penn-II Treebank. Most errors are corrected in the Catch-all and Clean-up component of the algorithm, which allows the L/R context principles to be simple, clean, perspicuous, and maintainable. Coordinating constructions are treated in a separate component, as they present particular problems in our automatic annotation algorithm, due to the often flat treebank analyses provided (cf. Section 4.4).

To construct the annotation matrices, we analysed the most frequently occuring rule types, such that we get 85% coverage of token occurrences of these rules, and provided linguistic generalisations, i.e. in an English sentence, the first noun phrase to the left of a verb phrase is likely to be the subject of the sentence.

This is done for each of the main categories: ADJP, ADVP, NP, PP, PRN (parenthesis), PRT (particle), QP (quantifier phrase), RRC (reduced relative clause), S, SBAR, SBARQ, SINV (inverted sentence), SQ, VP, WHADJP, WHADVP, WHNP, WHPP. As mentioned in Section 4.3, we distinguish between arguments (SUBCAT functions) and adjuncts (NON-SUBCAT functions) to the left and right of the head. The head receives an '↑=↓' annotation.

The annotation matrices are a core part of our annotation algorithm, used not only in the L/R context component, but also in the component used to

annotate coordinate structures (cf. Section 4.4.1 and Section 4.4.2). We only assign subcategorisable grammatical functions where we are sure, in order to minimise errors, e.g. no PP is assigned an oblique function in an NP, as it is often the case that it is an adjunct, and is difficult to distinguish automatically. However, for prepositional phrases, use can be made of the -CLR functional tag (in the 'Catch-all and Clean-up' component, see Section 4.6), which indicates that the PP is closely related to the verb, and is therefore most likely a complement, and so we annotate PP-CLR with an oblique function. There are instances in the Treebank where PP is tagged with the -CLR tag such as the rule in (141), from WSJ 0576, tree 19, in which it is incorrect to annotate PP-CLR as an oblique. However, since our algorithm is based on generalisations, it is difficult to capture specific cases correctly all of the time.

(141)

NP →	NP	PP	PP-CLR
	↑=↓	↓∈↑ADJ	↑OBL=↓
	your characterization	of California's Greens in particular	as la-la activists

A simplified matrix for NP rules is shown in Table 4.5. For rules expanding English NPs, the rightmost nominal (e.g. N, NN, NNS) on the RHS is usually the head, which receives an '↑=↓' annotation. This is the case unless the rightmost nominal occurs after a comma, in which case it is annotated as part of an apposition set, '↓∈↑APP', cf. Section 4.6. A determiner occurring in the left context is annotated '↑SPEC:DET=↓', in order to allow an analysis of complex determiners, cf. (142), which include a quantifier phrase, rather than just the annotation '↑SPEC=↓' mentioned in Chapter 2. Any adjective phrase or noun is annotated as an adjunct, '↓∈↑ADJ'.[5] Any prepositional phrase in the right context is annotated as an adjunct, '↓∈↑ADJ'; a relative clause (RRC: reduced relative clause) is annotated as a relative modifier, '↑RELMOD=↓', and any nominal to the right of the head is annotated as an apposition, '↓∈↑APP'. Apposition is annotated as part of a set to allow multiple occurrences without clashes.

In LFG, both the determiner and the head noun in a noun phrase are annotated as '↑=↓', i.e. as co-heads, and in the lexicon the entry for determiner is '↑SPEC=the'. However, we annotate DT as '↑SPEC:DET=↓' and in the lexicon an example entry for determiner is '↑PRED=the', so the f-structure entry

[5]We have not yet analysed adjectival modifiers as HEADMOD, however, this is an avenue planned for further work.

NP	Left context	Head	Right context
SUBCAT	DT: ↑SPEC:DET=↓	N, NN, NNS: ↑=↓	
NON-SUBCAT	ADJP: ↓∈↑ADJ		RRC: ↑RELMOD=↓
	N, NN, NNS: ↓∈↑ADJ		PP: ↓∈↑ADJ
			N, NN, NNS: ↓∈↑APP

Table 4.5: Simplified Annotation Matrix for NP rules

for determiner is 'spec:det:pred:the'. Initially, this was done as our algorithm could not deal with co-heads, but this is no longer a problem (cf. annotating modal TO as a co-head, as described below—provided that two preds do not occur at the same level there is no clash). Further analysis of NPs in coordinate structures is needed[6] before changing the specifier to be a co-head. It would also mean a change for complex determiners, which at the moment are annotated as shown in (142), from WSJ 1139, tree 08, with the QP quantifier phrase highlighted in bold:

	NP →	DT	QP	NN
(142)		↑SPEC:DET=↓	↑SPEC:QUANT = ↓	↑=↓
		the	**5.8 million**	rate

The analysis of a complex specifier as shown in (142), '↑SPEC:QUANT = ↓' follows the analysis in ([Butt et al., 1999], p.102). In order to construct the NP matrix, we only had to analyse 102 out of the 6595 NP rule types (approximately 1.5% of the total NP rule types). The full matrix for NP rules is given in Table 4.6 and Table 4.7.

These tables state the category that appears to the LHS of the head and the annotation that it receives. It also states the category that appears to the RHS of the head and the annotation that it receives. # occurring to the left of the head is annotated with '↓∈↑ADJ', and # occurring to the right of the head receives the same annotation. Similarily, an ADJP occurring to the left of the head, is annotated with '↓∈↑ADJ', and an ADJP occurring to the right of the head receives the same annotation. A determiner (DET), predeterminer (PDT), or wh-determiner (WDT) occurring to the left or the right of the head is annotated with '↑SPEC:DET=↓'. A quantifier phrase (QP) occurring to the left or the right of the head is annotated with '↑SPEC:QUANT=↓'. A reduced relative clause (RRC) occurring to the left or the right of the head is

[6]Named entity recognition could help disambiguate complex coordinate phrases; we have not yet integrated a named entity recogniser with our automatic f-structure annotation algorithm, but it is an avenue for futher work.

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
#	↓ ∈ ↑ ADJ	#	↓ ∈ ↑ ADJ
ADJP	↓ ∈ ↑ ADJ	ADJP	↓ ∈ ↑ ADJ
ADVP	↓ ∈ ↑ ADJ	ADVP	↓ ∈ ↑ ADJ
CD	↓ ∈ ↑ ADJ	CD	↓ ∈ ↑ ADJ
DT	↑SPEC:DET=↓	DT	↑SPEC:DET=↓
FW	↓ ∈ ↑ ADJ	FW	↓ ∈ ↑ ADJ
IN	↓ ∈ ↑ ADJ	IN	↓ ∈ ↑ ADJ
INTJ	↓ ∈ ↑ ADJ	INTJ	↓ ∈ ↑ ADJ
JJ	↓ ∈ ↑ ADJ	JJ	↓ ∈ ↑ ADJ
JJR	↓ ∈ ↑ ADJ	JJR	↓ ∈ ↑ ADJ
JJS	↓ ∈ ↑ ADJ	JJS	↓ ∈ ↑ ADJ
LST	↓ ∈ ↑ ADJ	LST	↓ ∈ ↑ ADJ
MD	↓ ∈ ↑ ADJ	MD	↓ ∈ ↑ ADJ
NAC	↓ ∈ ↑ ADJ	NAC	↓ ∈ ↑ ADJ
NN	↓ ∈ ↑ ADJ	NN	↓ ∈ ↑ ADJ
NNP	↓ ∈ ↑ ADJ	NNP	↓ ∈ ↑ ADJ
NNPS	↓ ∈ ↑ ADJ	NNPS	↓ ∈ ↑ ADJ
NNS	↓ ∈ ↑ ADJ	NNS	↓ ∈ ↑ ADJ
NP	↓ ∈ ↑ ADJ	NP	↓ ∈ ↑ APP
NX	↓ ∈ ↑ ADJ	NX	↓ ∈ ↑ ADJ
PDT	↑SPEC:DET=↓	PDT	↑SPEC:DET=↓
PP	↓ ∈ ↑ ADJ	PP	↓ ∈ ↑ ADJ
PRN	↓ ∈ ↑ ADJ	PP-TMP	↓ ∈ ↑ ADJ
PRP	↓ ∈ ↑ ADJ	PRN	↓ ∈ ↑ ADJ
PRP$	↑POS=↓	PRP	↓ ∈ ↑ ADJ
PRT	↓ ∈ ↑ ADJ	PRP$	↓ ∈ ↑ ADJ
QP	↑SPEC:QUANT=↓	PRT	↓ ∈ ↑ ADJ
RB	↓ ∈ ↑ ADJ	QP	↑SPEC:QUANT=↓
RBR	↓ ∈ ↑ ADJ	RB	↓ ∈ ↑ ADJ
RBS	↓ ∈ ↑ ADJ	RBR	↓ ∈ ↑ ADJ
RP	↓ ∈ ↑ ADJ	RBS	↓ ∈ ↑ ADJ
RRC	↑RELMOD=↓	RP	↓ ∈ ↑ ADJ
S	↓ ∈ ↑ ADJ	RRC	↑RELMOD=↓
SBAR	↑COMP=↓	S	↑XCOMP=↓, ↑SUBJ=↓SUBJ
SBARQ	↑COMP=↓	SBAR	↑RELMOD=↓
SYM	↓ ∈ ↑ ADJ	SBARQ	↑COMP=↓
TO	↓ ∈ ↑ ADJ	SINV	↑COMP=↓
UCP	↓ ∈ ↑ ADJ	SQ	↑RELMOD=↓
UH	↓ ∈ ↑ ADJ	SYM	↓ ∈ ↑ ADJ

Table 4.6: Annotation Matrix for NP rules

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
VB	↓∈↑ ADJ	TO	↓∈↑ ADJ
VBD	↓∈↑ ADJ	UCP	↓∈↑ ADJ
VBG	↓∈↑ ADJ	UH	↓∈↑ ADJ
VBN	↓∈↑ ADJ	VB	↓∈↑ ADJ
VBP	↓∈↑ ADJ	VBD	↓∈↑ ADJ
VBZ	↓∈↑ ADJ	VBG	↓∈↑ ADJ
VP	↓∈↑ ADJ	VBN	↓∈↑ ADJ
WDT	↑SPEC:DET=↓	VBP	↓∈↑ ADJ
WHNP	↓∈↑ ADJ	VBZ	↓∈↑ ADJ
WHPP	↑RELMOD=↓	VP	↑RELMOD=↓
WP	↑SPEC:DET=↓	WDT	↑SPEC:DET=↓
WP$	↓∈↑ ADJ	WHNP	↓∈↑ ADJ
WRB	↓∈↑ ADJ	WHPP	↑RELMOD=↓
		WP	↑SPEC:DET=↓
		WP$	↓∈↑ ADJ
		WRB	↓∈↑ ADJ

Table 4.7: Annotation Matrix for NP rules (continued)

annotated with '↑RELMOD=↓'. S occurring to the left of the head is annotated with '↓∈↑ADJ', whereas an S occurring to the right of the head is annotated with '↑XCOMP=↓,↑SUBJ=↓SUBJ'. SBAR occurring to the left of the head is annotated with '↑COMP=↓', whereas an SBAR occurring to the right of the head is annotated with '↑RELMOD=↓'. SBARQ or VP occurring to the left or the right of the head are annotated with '↑COMP=↓'. SINV occurring to the right of the head is annotated with '↑RELMOD=↓'; SINV does not occur to the left of the head in an NP. WHPP occurring to the left or the right of the head is annotated with '↑RELMOD=↓'. PRP$ is annotated with '↑POS=↓'. Possessive NPs (tagged POS) are treated as an exception to the usual annotation method, as the mother node of a POS tag receives the annotation '↑POSS=↓', cf. Section 4.2.2.

We have constructed matrices for all the main categories, a full listing of which is given in Appendix B and C. We initially tagged S to the right of the head in a VP rule as UCOMP, as it was unclear whether it should be COMP or XCOMP. This was done as a temporary measure—there is no such grammatical function as UCOMP in LFG, and when evaluating our automatic annotations against the manually constructed 'gold-standard' (cf. Section 5.3), it would always be incorrect. An example of the difficulty first encountered can be seen with the rule VP → VB S. In (143), from WSJ 0039, tree 39, where *deserve* is tagged as VB, S should definitely be annotated as an XCOMP:

(143) But she did n't **deserve** to have her head chopped off .

However, the same rule is used in (144), from WSJ 0003, tree 09, where *said* is tagged as VB, and the empty S is linked to the sentence highlighted in bold, and should be annotated as a COMP.

(144) **We have no useful information on whether users are at risk** , " said James A. Talcott of Boston 's Dana-Farber Cancer Institute

Following further analysis we were able to analyse S as an XCOMP, i.e. the embedded subject must be controlled by an argument in the root (matrix) clause, exemplifying functional control, generally containing non-finite complements, cf. Section 2.3.2.2. Note that this is still a generalisation, so in some cases, we provide wrong annotations.

TO occurring to the left of the head in a VP rule is also annotated as a co-head. The VP was initially annotated as the head, according to the head rules in Section 4.2.1; TO receives an annotation from the annotation matrices, as shown in Figure 4.2.

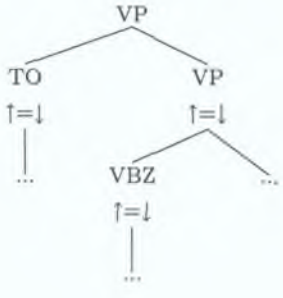


Figure 4.2: Partial tree containing TO and VP as co-heads

As mentioned previously, this does not cause a clash as the lexical entry for TO does not contain a pred (cf. Section 4.2.2). The tag POS also appears in VPs—it has also been used incorrectly in verb phrases to tag a third person singular present verb *'s*. This mistagging (described in Section 3.2.1) is taken into account by our head rules (cf. Section 4.2.1). In this case the POS node receives the annotation '↑=↓'.

If PRN is the mother category (i.e. occurs on the LHS of the rule), the algorithm then looks to the mother of PRN, and annotates the daughters of

PRN according to the annotation matrices for that mother category. The PRN node is annotated as an adjunct, '↓∈↑ADJ', as in Figure 4.3, from WSJ 1574, tree 56.

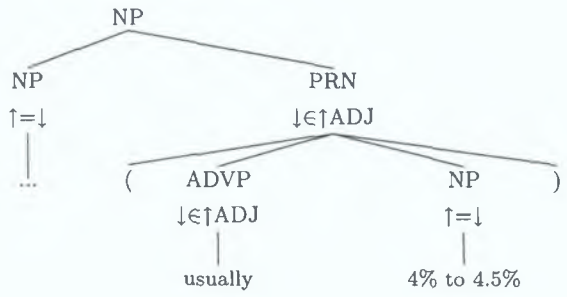


Figure 4.3: Partial tree illustrating annotations for PRN for the string *They commonly give two scenarios : One is based on interest rates that the company guarantees (usually 4 % to 4.5 %) and the other on the rate it is currently getting on investment , often 8.5 % or more*

In Figure 4.3, the mother node of the PRN is an NP, so the daughters of the PRN are annotated according to the NP annotation matrix.

FRAG(ment) and **X**(unknown constituents) are the only 'constituents' that we have not covered to date. FRAG marks chunks of text that appear to be clauses, but lack too many essential elements for the exact structure to be easily determined, such as WSJ 0214, tree 06, *Old-time kiddies , " he says*, where FRAG is the label used for the answer to a question, the tree for which is shown in Figure 4.4.

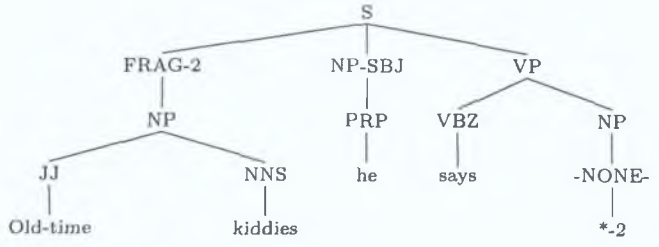


Figure 4.4: Partial tree containing FRAG for the string *Old-time kiddies , " he says*

In our automatic annotation algorithm we discount FRAG when obtaining

results, as it would be extremely difficult to extract predicate-argument structure (cf. Section 5.2.2). X is the label used for unknown constituents, such as WSJ 0239, tree 44, *You bet attention , " I yelled back , leaping atop the propane tanks , " I 'm wearing alligator loafers !* , which is shown in Figure 4.5.

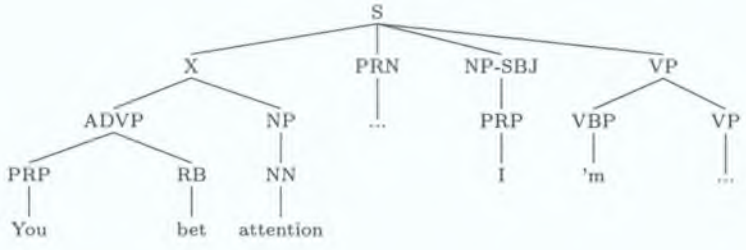


Figure 4.5: Partial tree containing X for the string *You bet attention , " I yelled back , leaping atop the propane tanks , " I 'm wearing alligator loafers !*

Note that in this tree *bet*, which is a verb is tagged as an RB within an ADVP. If this tagging was done correctly, it may have been possible to determine the constituent which X is replacing. It is difficult to extract predicate-argument structure for X, so in our automatic annotation algorithm we discount X when obtaining results.

When we achieved 85% coverage, we analysed the RHS of all the rules to see what percentage of categories received an annotation. We then analysed the resultant f-structures to see what needed to be improved. Comparing these f-structures to the previous set of f-structures generated, we checked that the fragmentation decreased, ensuring that the new annotations that we included improved the coverage results for our algorithm. The coverage shows how many nodes receive an annotation from the automatic annotation algorithm; we assess the quality of these annotations separately. Evaluation of coverage and quality is described in Chapter 5.

(145) shows the parse tree for the first of our 105 testset sentences, "*The investment community, for one, has been anticipating a speedy resolution*", from WSJ 2308, tree 24. The testset contains a 'gold standard' file containing a random sample of 105 sentences from section 23 of the WSJ part of the Penn-II treebank, which have been manually annotated by a linguist, and an automatically annotated file, which contains the same 105 sentences as annotated by our automatic annotation algorithm. It is used for qualitative evaluation, cf. Section 5.3. The f-structure derived automatically via our method is shown in Figure 4.6. The f-structure shows the results of the annotations applied as described in Section 4.3.1 and resolved by the constraint solver. Percolation of subjects into the XCOMP is included by use of '↑SUBJ=↓ SUBJ' annotations on the VP daughters in the VP rules.

(145)
```
(S (NP-SBJ (DT The)
           (NN investment)
           (NN community))
  (, ,)
  (PP (IN for)
      (NP (CD one)))
  (, ,)
  (VP (VBZ has)
      (VP (VBN been)
          (VP (VBG anticipating)
              (NP (DT a)
                  (JJ speedy)
                  (NN resolution))))))
```

4.4 Coordination

Coordinate structures present particular problems in our automatic annotation algorithm, due to the often flat treebank analyses provided. Coordinate structures have always posed intricate problems in linguistic analysis, as features such as number, person and tense may be distributed over the entire structure or may be limited to just one conjunct. Therefore, integrating the coordinate structures with the other annotation principles would complicate principles and make them harder to maintain and extend. For this reason, we decided to deal with the coordinate structures in a separate module in our algorithm. The module is divided into two main sections—the first for 'like' constituent coordination

```
subj : spec : det : pred : the
       adjunct : 1 : num : sing
                     pers : 3
                     pred : investment
       num : sing
       pers : 3
       pred : community
adj : 2 : obj : pred : one
              pred : for
xcomp : subj : spec : det : pred : the
               adjunct : 1 : num : sing
                             pers : 3
                             pred : investment
               num : sing
               pers : 3
               pred : community
        xcomp : subj : spec : det : pred : the
                       adjunct : 1 : num : sing
                                     pers : 3
                                     pred : investment
                       num : sing
                       pers : 3
                       pred : community
                obj : spec : det : pred : a
                      adjunct : 3 : pred : speedy
                      pred : resolution
                      num : sing
                      pers : 3
                participiple : pres
                pred : anticipate
        pred : be
        tense : past
pred : have
tense : pres
```

Figure 4.6: F-structure automatically generated for the first of our 105 Penn-II test sentences, *The investment community, for one, has been anticipating a speedy resolution*

(Sections 4.4.1 and 4.4.2) and the second for 'unlike' constituent coordination (Section 4.4.4).

4.4.1 Initial Coordination Principles

In the coordinating conjunction annotation principles, the annotation procedure is dependent, as in the other annotation principles, on locating the head daughter. Initially, our algorithm checks to see if the first non-punctuation element on the RHS is a CC or a CONJP. If this is the case, the CC or CONJP is annotated as an adjunct and the rest of the tree is processed as normal, referring to the annotation matrices described in Section 4.3.1. The annotations for such a case are given in (146), from WSJ 0319, tree 13:

(146)

NP →	CC	DT	JJ	NN
	$\downarrow \in \uparrow$ADJ	$\uparrow$SPEC=$\downarrow$	$\downarrow \in \uparrow$ADJ	$\uparrow = \downarrow$
	both	the	second	quarter

However, if the rule contains a CC in any other position on the RHS, our coordination component is applied. The initial algorithm for the CC component of [Cahill et al., 2002a] is very simple. Having established that the structure is indeed a coordinate structure (i.e. the rule contains a CC in any position other than the leftmost position on the RHS), it first annotates the rightmost conjunction (CC or CONJP) as the head (this was done following analysis of rules containing coordination, in which we found that the rightmost conjunction is most likely to be the head). The algorithm then checks if the categories to the immediate left and right of the CC or CONJP (excluding punctuation) are the same, and if so, annotates all categories of that type as '$\downarrow \in \uparrow$CONJ', as shown in (147), from WSJ 1041, tree 35:

(147)

VP →	VB ,	VB	CC	VB
	$\downarrow \in \uparrow$CONJ	$\downarrow \in \uparrow$CONJ	$\uparrow = \downarrow$	$\downarrow \in \uparrow$CONJ
	speak ,	walk	and	think

The f-structure associated with the rule in (147) for the string "**speak , walk and think** " is given in (148):

(148)
$$\begin{bmatrix} \text{CONJ} & \{ \begin{bmatrix} \text{PRED} & \text{speak} \end{bmatrix} \\ & \begin{bmatrix} \text{PRED} & \text{walk} \end{bmatrix} \\ & \begin{bmatrix} \text{PRED} & \text{think} \end{bmatrix} \} \\ \text{PRED} & \text{and} \end{bmatrix}$$

This works well for categories which are the same. If they are different, or if an adjective or adverb occurs between the conjunction and the conjuncts, then the nodes to the immediate left and right of the conjunction will be annotated as conjuncts. The nodes which should be annotated as they conjuncts are then left unannotated. Shared complements are also left unannotated, as in (149), from WSJ 0108, tree 15:

(149)

VP →	VB	CC	VB	NP
	↓∈↑CONJ	↑=↓	↓∈↑CONJ	
	hear	or	read	every viewpoint

The NP object, *every viewpoint*, is a shared complement of *hear* and *read*. It should be annotated with '↑OBJ=↓', but with this basic algorithm, it remains unannotated[7]. Of course if the categories are typologically different, they should be labelled UCPs (cf. Section 4.4.4), but unfortunately the correct labels are not always assigned.

4.4.2 Improved Method

Simple coordination (i.e. where both categories on either side of the conjunction are the same) can be handled quite easily by an automatic annotation process. However, when working with real data from the Treebank, it is not always the case that the categories on either side of the conjunction are the same. Section 4.4.4 deals with coordination of categories that are expected to differ, but unfortunately, the UCP label is not always used to mark unlike constituent coordination for typologically different constituents. We improved our strategy to extend the coverage of previously unannotated nodes (different categories in a CC rule, shared complements, or adjectives or adverbs occurring between the conjunction and the conjuncts), and we have now implemented the following rules in the coordination component in two main stages. As NP rules can be

[7]In the improved method, described in Section 4.4.2, the NP shared complement is fed back into the L/R context principles and receive an object annotation, cf. Section 4.4.2.

quite complex, due to the often flat Treebank analysis, the algorithm initially looks just at NP rules containing coordinate structures.

- The first stage in the improved coordination component of our automatic annotation algorithm checks noun phrases containing a CC (which is not the leftmost element) on the RHS.

    1. It searches first for noun sequences to the right of the head. If found, the rightmost nominal is annotated as the head of the phrase, '↑=↓', the CC is annotated with 'X=↓, X∈↑ADJ', where X is the f-number assigned to the CC node, and the nominals to the left and right of the CC are annotated with '↓∈X:CONJ'. An NP rule containing complex coordination is given in Figure 4.7, from WSJ 1855, tree 01.

NP →	NNS	CC	NNS	NN	NNS
	↓∈X:CONJ	X=↓, X ∈↑ADJ	↓∈X:CONJ	↓∈↑ADJ	↑=↓
	futures	and	options	trading	firms

Figure 4.7: NP containing noun sequence on the RHS, for the string *futures and options trading firms*

The NNS, *firms*, is first annotated as the head of the rule. The CC is annotated with 'X=↓, X∈↑ADJ'. X is instantiated as a number, and becomes a node identifier. CC becomes a member of the adjunct set. The other words tagged as NNS, *futures* and *options*, are annotated with '↓∈X:CONJ', i.e. the path 'X:CONJ' allows both to become members of the conjunct set. The NN, *trading* is annotated as an adjunct, as indicated in the NP annotation matrix—an NN to the left of the head is annotated with '↓∈↑ADJ'. The f-structure associated with Figure 4.7 is given in Figure 4.8.

The issue here is not so much unlike constituent coordination but rather the overtly flat treebank rules, which is the general approach taken by the Treebank annotators. Complex NPs are discussed more in Section 3.7. If there is no nominal to the left or right of the CC, constituents to the immediate left and right of the CC are annotated with '↓∈X:CONJ'. For all other categories in the rule, excluding punctuation, the annotation is found by looking up the NP matrix tables, as used in the L/R annotation principles (described in detail in Section 4.3).

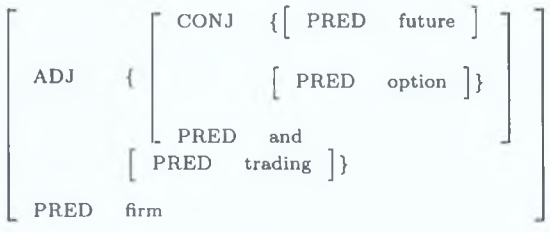


Figure 4.8: Partial f-structure for NP Coordination, for the string *futures and options trading firms*

2. If there is no noun sequence (i.e. more than a single nominal) to the right of the CC in the NP rule containing a coordinate structure, the algorithm annotates the CC as the head. It then looks to the left of the CC for nominals punctuated by commas, which are all annotated with '$\downarrow \in \uparrow$CONJ'. This is shown in Figure 4.13, p.105.

   The algorithm looks for nouns to the immediate left and right of the CC, which, if found, are annotated with '$\downarrow \in \uparrow$CONJ', as in (150), from WSJ 0472, tree 07:

   (150)

NP →	CD	NNS	NN	CC	NN
	$\downarrow \in \uparrow$ADJ	$\downarrow \in \uparrow$ADJ	$\downarrow \in \uparrow$CONJ	$\uparrow = \downarrow$	$\downarrow \in \uparrow$CONJ
	55,000	Machinists	rank	and	file

   For all other categories in the NP rule, excluding punctuation, the annotation is found by looking up the NP matrix tables.

- The second stage in the algorithm is applied if the mother category (on the LHS) is not an NP. We have constructed a list of similarity sets (e.g. for noun coordination, the categories NN and NNS are not exactly the same, but are both nouns, and therefore similar). The annotation algorithm first looks at the mother category and retrieves a list of all similar categories. The list of similarity sets used in the CC component is given in Table 4.8.

  For example, the list of similarity sets for NP contains NN, NNS, NNP, NNPS, NP, NAC and NX. The same list is used for NX and WHNP. These lists were compiled manually from our head rules, which specify the most likely head for each category.

  1. This second stage uses the list of similarity sets, and if any of the categories in the similarity set appear somewhere in the rule on the

Mother Category	Similar Categories
ADJP	JJ, JJR, JJS, VBN, VBG, VP
ADVP	RB, RBR, RBS
INTJ	INTJ
NAC	NAC
NP	NN, NNS, NNP, NNPS, NP, NAC, NX
NX	NN, NNS, NNP, NNPS, NP, NAC, NX
PP	IN, TO
PRN	NP
PRT	RP
QP	QP
RRC	RRC
S	S, SBAR, SBARQ, SQ, SINV, VP
SBAR	S, SBAR, SBARQ, SQ, SINV, VP
SBARQ	S, SBAR, SBARQ, SQ, SINV, VP
SINV	S, SBAR, SBARQ, SQ, SINV, VP
SQ	SQ, SBARQ
VP	MD, TO, VB, VBD, VBG, VBN, VBP, VBZ, VP
WHADJP	JJ, JJR, JJS, VBN, VBG, VP
WHADVP	RB, RBR, RBS
WHNP	NN, NNS, NNP, NNPS, NP, NAC, NX

Table 4.8: List of Similarity Sets used in the CC component

RHS, as in (151), from WSJ 2428, tree 41, they are annotated as '↓∈↑CONJ':

(151)

S →	S	CC	SINV
	↓∈↑CONJ	↑=↓	↓∈↑CONJ
	they have billions to invest	and	invest they will

The similarity set for S contains S, SINV, SBAR, SBARQ, SQ, and VP. The annotations for any other categories are found by looking up the matrix tables (for the mother node). The conjoined categories here are typologically similar, so they should not occur under the UCP label.

If there is a noun sequence (in a non-NP rule containing coordination), the similarity set is used to check for a set of similar conjuncts, as in Figure 4.9, from WSJ 0443, tree 05.

WHNP →	WP$	NNS	CC	NN	NNS
	↑POSS=↓	↓∈↑CONJ	↑=↓	↓∈↑CONJ	↓∈↑CONJ
	whose	pitchers	and	home-run	hitters

Figure 4.9: Noun Sequence (in a non-NP rule containing Coordination for the string *whose pitchers and home-run hitters*

Unfortunately, in this case the annotations assigned by our automatic f-structure annotation algorithm are incorrect, *home-run* modifies *hitters*, and should not be annotated as a conjunct. This is a problem with our generalisations, and needs to be investigated in further detail. As with the L/R context principles, we generalise to categories with functional tags, i.e. functional tag information is ignored at this stage in the algorithm, as in (152), from WSJ 0044, tree 133:

(152) Mrs. Yeargin says she pleaded guilty **because she realized it would no longer be possible to win reinstatement , and because she was afraid of further charges**

This sentence contains the rule SBAR-PRP → SBAR , CC SBAR-PRP. The text dominated by the mother SBAR-PRP is highlighted in bold. As functional information is ignored, it is a straightforward CC rule.

2. If the list of similar categories does not apply to provide us with a set of conjuncts, we check to see if there is just one non-punctuation element to the left and right of the CC, and if so, they are annotated as '$\downarrow \in \uparrow$CONJ', as in (153), from WSJ 0554, tree 16:

(153)

ADJP →	JJ	CC	DT
	$\downarrow \in \uparrow$CONJ	$\uparrow = \downarrow$	$\downarrow \in \uparrow$CONJ
	little	or	no

In this case the conjoined categories are typologically different so they should occur under the UCP label. As there is only one category to either side of the conjunction, the correct annotations are assigned without much difficulty. Our algorithm handles such instances of mistagging quite well, as illustrated by the results given in Section 5.3.3, Table 5.8, p.139, and Table 5.9, p.139.

3. The annotation method so far checks only for non-punctuation elements, as in most cases it ignores punctuation. The exception to this is a final principle which checks for cases where the rule on the RHS contains no CC or CONJP, but instead contains a colon (':'), with the same category occurring to the left and right of the colon. This can be shown by revisiting (46), p.33, *His answer reveals his vulnerability – it also draws the line that Soviet society must cross to enter the normal dialogue of Western culture.* which contains the rule S → S : S (where : is the POS tag assigned to the hyphen '–').

Our approach is to treat the punctuation as head (for rules of this type), and annotate the two sentences as members of the conjunct set; any other nodes are annotated according to the S annotation matrix. The two conjoined sentences are members of the conjunct set, each conjunct containing the complete f-structure for each sentence; the '–' is annotated as the main pred, as shown in the partial f-structure in Figure 4.10.[8]

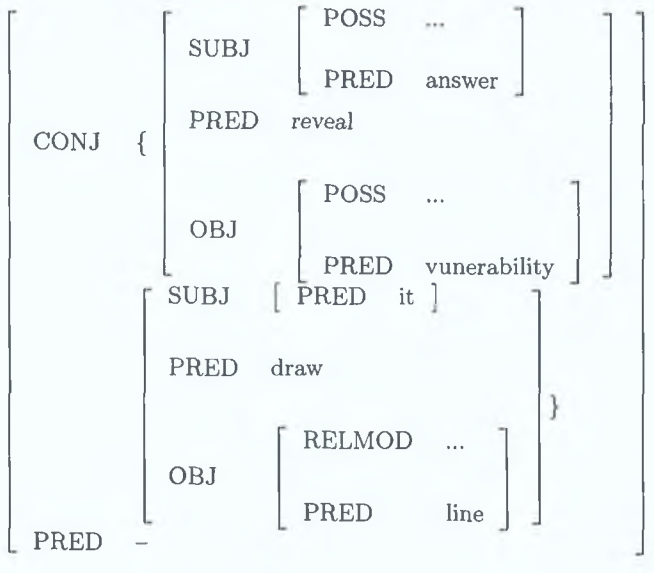


Figure 4.10: Partial f-structure where punctuation is the head for the string *His answer reveals his vulnerability - - it also draws the line that Soviet society must cross to enter the normal dialogue of Western culture*

4. When a conjunct has been annotated to both sides of the head, any other categories which do not have an annotation are annotated as adjuncts.

4.4.3 Our work in relation to LFG

Figure 2.5, p.32, shows the LFG rule for conjuncts separated by commas. An example of this from the Penn-II Treebank is given in Figure 4.11, from WSJ 0003, tree 15, which receives the correct annotations, as shown, by our algorithm.

[8]We are now revisiting Figure 2.6, p.34.

NP →	NP	, NP	CC	NP
	↓∈↑CONJ	↓∈↑CONJ	↑=↓	↓∈↑CONJ
	malignant mesothelioma	, lung cancer	and	asbestosis

Figure 4.11: Example coordination rule from the Penn-II Treebank for the string *malignant mesothelioma, lung cancer and asbestosis*

All the NPs are annotated as members of the conjunct set, as shown in the f-structure in Figure 4.12.

$$\begin{bmatrix} \text{CONJ} & \{ \begin{bmatrix} \text{ADJ} & \text{malignant} \\ \text{PRED} & \text{mesothelioma} \end{bmatrix} \\ & \begin{bmatrix} \text{ADJ} & \text{lung} \\ \text{PRED} & \text{cancer} \end{bmatrix} \\ & [\ \textit{PRED} \quad \text{asbestosis}\ ] \} \\ \text{PRED} & \text{and} \end{bmatrix}$$

Figure 4.12: Partial f-structure where punctuation is the head for the string *malignant mesothelioma, lung cancer and asbestosis*

The rules in (43) and Figure 2.5 are based on simple coordination; they assume that the categories on either side of the conjunction are the same. Unfortunately, however, when working with real data, this is not always the case, as can be seen in Figure 4.13, , from WSJ 0946, tree 32, which should be annotated as shown.

NP →	NN ,	NN ,	NN	CC	NNS
	↓∈↑CONJ	↓∈↑CONJ	↓∈↑CONJ	↑=↓	↓∈↑CONJ
	sex ,	class ,	race	and	politics

Figure 4.13: NP Coordination for the string *sex, class, race and politics*

This is an example taken from the Penn-II Treebank; the tag NN is used for a singular or mass noun, NNS is used for a plural noun, and CC is the tag used for the coordinating conjunction, cf. Section 3.2.3. Despite this mistagging by the Penn-II taggers, leading to the fact that the tags are not identical, our automatic f-structure annotation algorithm is designed to deal with such data.

We have constructed a list of 'similar' categories which is used by the algorithm, e.g. for noun coordination, the categories NN and NNS are not exactly the same, but are both nominals, and therefore similar. Further detail on the compilation of this list, which is based on a subset of the head rules used in our algorithm, is given in Section 4.4.1.

No special rules for two-part conjunctions have yet been implemented in the algorithm, as discussed in (44), p.32, but are planned for future work, along similar lines to the research outlined here. At the moment, a rule containing the structure *either... or* receives the annotations as shown in the sentence in (154), from WSJ 2321, tree 20:

(154)

ADJP →	CC	JJ	CC	JJ
	↓∈↑ADJ	↓∈↑CONJ	↑=↓	↓∈↑CONJ
	either	hard	or	soft

Three-part conjunctions also occur, as in Figure 4.14, from WSJ 1632, tree 06, which receive the following annotations from our automatic f-structure annotation algorithm.

NP →	CC	NP	CC	NP	CC	NP
	↓∈↑ADJ	↓∈↑ADJ	↓∈↑ADJ	↓∈↑CONJ	↑=↓	↓∈↑CONJ
	either	debt reduction	or	debt-service reduction	or	new loans (the Brady Plan)

Figure 4.14: Three-part conjunctions for the string *either debt reduction or debt-service reduction or new loans (the Brady Plan)*

In this case *either* should be annotated as PRECONJ, and *or* in both cases should be annotated as the conjunctions. The rightmost CC is currently annotated as the head of phrase by our automatic f-structure annotation algorithm. The NPs to the immediate left and right of the rightmost CC are annotated as the conjuncts, with all other categories on the RHS annotated as adjuncts. The scoping of such complex constructions is problematic for our annotation algorithm. In Figure 4.14, the scoping is NP → CC (NP CC NP) CC NP, but this is not always the case for examples containing *either ... or ... or ...*

4.4.4 UCP Rules

As mentioned previously, the coordination of unlike phrases is dominated by the UCP (Unlike Constituent Phrase) label, and this is dealt with in a separate

section within our coordination component.

The Penn-II Treebank bracketing guidelines state that a conjunction joining two or more elements which are typologically different is dominated by UCP ([Bies et al., 1995], p.120). The label UCP is meant to be used in all cases where the constituents to the left and right of the CC are typologically unlike, as shown in (155), from WSJ 0110, tree 02:

(155)

UCP →	NNP	CC	JJ
	↓∈↑CONJ	↑=↓	↓∈↑CONJ
	U.S.	and	foreign

This is a straightforward case of coordination for our automatic f-structure annotation algorithm. However, this is not always the case, as can be seen in (153), where the UCP label is not used, even though the two conjuncts are typologically different categories. Despite this inconsistency, this example is handled adequately in the CC section of the coordination component, so only phrases dominated by UCP are dealt with in this section. Of course, occurrences involving complex coordination do occur in which our annotations are not completely correct—evaluation results for the coordination component are given in Section 5.3.3.

Examples also occur where the UCP label is used but does not actually contain a coordinate structure, as shown in the rule in (156) from WSJ 2045, tree 26:

(156)

UCP →	PP-LOC	NP-TMP :
	↑=↓	↓∈↑ADJ
	In other commodity markets	yesterday :

This mistagging makes it difficult to distinguish what should be the head of the phrase. The automatic annotation algorithm searches from left-to-right, and picks the leftmost element as head, PP-LOC here. -LOC elements elsewhere are generally annotated as adjuncts but it is necessary to have a head in each phrase to ensure a pred at each level in the f-structure.

The UCP component has been updated since the work described by [Cahill et al., 2002c]. Following further analysis and evaluation, we now apply the following rules (in the sequence given) for the UCP section in the coordination component of our annotation algorithm. If a node has already received an annotation, that annotation holds and is not overwritten by a subsequent rule,

unless it is the case that there is no conjunct to one side of the conjunction, as in (161).

1. We begin by checking for just one non-punctuation element to the left of the CC, and if found it is annotated as a conjunct. The algorithm then checks for just one non-punctuation element to the right of the CC, and if found it is annotated as a conjunct, as in (155).

   As there is only one category to the both the left and right of the conjunction, they are annotated as conjuncts.

2. If the leftmost non-punctuation daughter is a conjunct and the same or 'similar' category[9] occurs somewhere on the RHS with no annotation, it is included in the conjunct set. The same rule is applied for the rightmost non-punctuation daughter, if it is a singleton conjunct, as shown in (157), from WSJ 0186, tree 14:

   (157)

UCP →	JJ ,	NN	CC	JJ
	↓∈↑CONJ	↓∈↑CONJ	↑=↓	↓∈↑CONJ
	federal ,	state	and	local

   In the above example, the JJ, being the only category to the right of the CC, is annotated as CONJ (by the first step in this component of the algorithm). The JJ and NN to the left of the conjunction are seen as being of 'similar' categories, so both are annotated as CONJ.

3. We proceed by checking for noun sequences to the right of the head, and, where found, we annotate the rightmost noun as one of the conjuncts. The other nouns in the sequence are annotated as adjuncts, as shown in (158), from WSJ 1936, tree 24:

   (158)

UCP →	JJ	CC	NN	NN
	↓∈↑CONJ	↑=↓	↓∈↑ADJ	↓∈↑CONJ
	critical	and	box	office

   As there is only one category to the left of the conjunction, the JJ, it has already been annotated as CONJ (by the first step in this component of the algorithm).

[9]Note the 'similar' categories vary slightly from before—a separate set is included here which lists all nominals and adjectives as being similar categories. The similarity sets described in Section 4.4.2 are not used here. The 'similar' category set for UCP was constructed following further analysis of the UCP rules, and due to mistagging in the Treebank, was deemed to be necessary.

Note that our annotation matrices are not used here to find further annotations—annotations for each category must be specified within this component of our algorithm, e.g. a determiner to the left of a CC is annotated as a specifier ('↑SPEC=↓'). This is done as the mother node is UCP, so it would be unclear as to which matrix the annotations should be retrieved from.

4. At this point the algorithm goes through all the daughters and checks for an RB and/or an ADVP to the left *and* the right (or if there is a PRN to the left *and* the right), as in the rule in Figure 4.15, from WSJ 1331, tree 09.

UCP →	ADVP ,	CC	ADVP	PP
	↓∈↑CONJ	↑=↓	↓∈↑CONJ	↓∈↑ADJ
	up 33% from August's	but	still below	of September 1988
	$227.1 million ,		the $328.2 million	

Figure 4.15: ADVP as conjuncts in a UCP rule for the string *up 33% from August's $227.1 million , but still below the $328.2 million of September 1988*

This is actually an instance of 'like' coordination, which should not be dominated by a UCP; the mother category here should be ADVP. If it is not the case that there is an RB and/or an ADVP to the left *and* the right, or if there is a PRN to the left *and* the right, all PRN, RB and ADVP are annotated as adjuncts, as shown in (159), from WSJ 0992, tree 14, and (160), from WSJ 1829, tree 22:

(159)

UCP→	NP ,	CC	RB	ADJP
	↓∈↑CONJ	↑=↓	↓∈↑ADJ	↓∈↑CONJ
	a bold stance ,	and	**thus**	suspicious

(160)

UCP→	ADJP	CC	PRN	PP
	↓∈↑CONJ	↑=↓	↓∈↑ADJ	↓∈↑CONJ
	newer	and	**, thus ,**	in step with the latest safety codes

The text tagged by RB and PRN is tagged in bold. Both contain the word *thus*, but PRN is the tag used in (160), as the word *thus* is enclosed by commas. However, in both cases, *thus* seems to be an adjunct of the final conjunct. Therefore our algorithm attaches it at the wrong level in these

instances, annotating it to modify the entire coordinate structure. The scoping of modifiers within coordinate structures is an avenue for further work.

5. If there is no conjunct to the left or the right of the head at this stage, the nodes to the immediate left and right of the conjunction are annotated as part of the conjunct set, as shown in (161), from WSJ 0554, tree 12:

(161)

UCP→	RB	ADVP ,	CC	NP
	↓∈↑ADJ	↓∈↑CONJ	↑=↓	↓∈↑CONJ
	not	once a month ,	but	two or three times a week

Both RB and ADVP to the left of the conjunction would initially have been annotated as adjuncts (by step four), but as there is no conjunct to the left of the head, the node to the immediate left is annotated as CONJ. The RB, *not*, attaches at the wrong level in this example—it should modify only *once a month*, not the entire coordinated phrase, but our algorithm gets it wrong in this case.

6. If there is no conjunct to the left or the right of the head at this stage, and there are still nodes with no annotation, the nodes nearest to the conjunction are annotated as part of the conjunct set. Finally, once conjuncts have been found on both sides of the conjunction, anything that does not have an annotation by now is annotated as an adjunct, as shown in Figure 4.16, from WSJ 0071, tree 06.

UCP→	VP ,	UCP ,	CC	PP
	↓∈↑ADJ	↓∈↑CONJ	↑=↓	↓∈↑CONJ
	limited , in production	**of exceptional quality , (or so perceived, at any rate)**	and	with exceedingly high prices

Figure 4.16: Final annotation in UCP rules for the string *One of the fastest growing segments of the wine market is the category of superpremiums – wines limited in production , of exceptional quality (or so perceived , at any rate) , and with exceedingly high prices*

The VP on the LHS of this rule would not have previously been annotated, as the matrix tables used in the L/R context principles and the

coordination component are not used here Therefore, it is necessary to specify that other categories are annotated as a final step to ensure that all nodes receive an annotation, to prevent fragmented f-structures, cf Section 5 2 2

This example contains an embedded UCP rule (shown by the text highlighted in bold) UCP → PP PRN The partial tree is shown in Figure 4 17

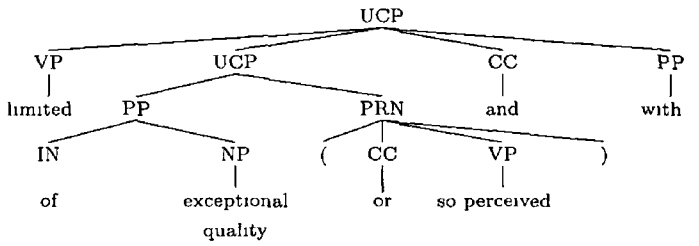


Figure 4 17 Partial tree containing embedded UCP rule for the string *One of the fastest growing segments of the wine market is the category of superpremiums – wines limited in production , of exceptional quality (or so perceived , at any rate) , and with exceedingly high prices*

This is another example (cf (156)) where the UCP label is used, but does not contain a tag for coordination This case is handled well by our algorithm, as here, PRN contains the conjunction *or* Our algorithm states that if PRN is the mother category, the algorithm then looks to its mother category (UCP), and annotates the daughters of PRN according to the annotation matrices for that mother category, cf (4 3) Therefore *or* will be annotated as the head of PRN

As can be seen from the rules detailed above, this section of the coordination component is similar to the section handling rules containing a CC, but with some significant differences which were deemed necessary following specific analysis of the UCP rules, notably that the annotation matrices are not used as it is unclear as to which matrix the annotations should be retrieved from Further analysis of the UCP rules is noted for further work In order to do this, it will be necessary to evaluate the UCP rules in isolation, by extracting rules in which the mother category is an UCP This will allow us to pinpoint exactly where in this component of the algorithm needs improvement

4.5 Traces

As described in Section 3.8, the Treebank annotation scheme provides a set of coindexed null elements, which mark the position for non-local phenomena such as wh-movement, passive, and the subjects of infinitival constructions. Before we incorporated the trace and index information in our algorithm, we could only produce 'proto' f-structures, which interpreted linguistic material locally, i.e. where it is found in the tree, but not always where it should be interpreted semantically in the tree, e.g. in a simple example sentence such as 'John promised Mary to leave', there was a missing subject (required by the verb *leave*) in the f-structure. We now use the trace and index information provided by the Treebank to capture these non-local dependencies, and encode them as corresponding reentrancies in a richer f-structure representation.

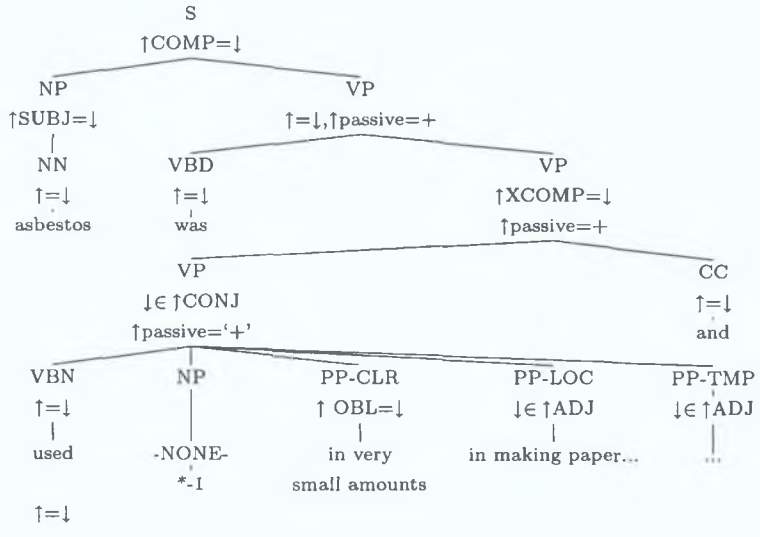
4.5.1 Passive

Passivisation is one class of constructions that can be classified as function-changing, and can be viewed as linking grammatical functions to semantic arguments. We make use of the following traces in the Treebank to annotate sentences as passive:

- If there is an empty NP in a VP to the right of a verb (VBD, etc.) with a * trace in object position, then the mother VP is annotated as passive ('passive=+' is included in the f-structure), as in the rule VP → VBN NP, where NP → (-NONE- *-1). The NP node receives no annotation. It is the only null constituent that is not treated as a full node (i.e. it receives no annotation); all other empty nodes receive an annotation even though they contain no lexical material, i.e. they are annotated according to the head rules and annotation matrices as though they contained lexical material. The partially annotated tree and the f-structure associated with the sentence in (162), from WSJ 0003, tree 11, are shown in Figure 4.18.

  (162) The Lorillard spokeswoman said asbestos was **used in " very modest amounts " in making paper for the filters in the early 1950s** and replaced with a different type of filter in 1956

  The text dominated by the VP is highlighted in bold. The passive annotation is included in the f-structure inside the sentential COMP.



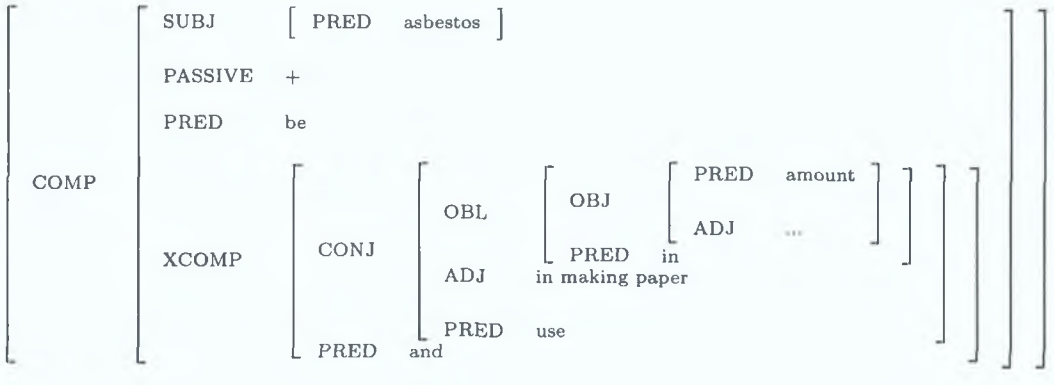


Figure 4.18: Partial tree and f-structure showing trace in object position for the string *The Lorillard spokeswoman said asbestos was used in " very modest amounts " in making paper for the filters in the early 1950s and replaced with a different type of filter in 1956*

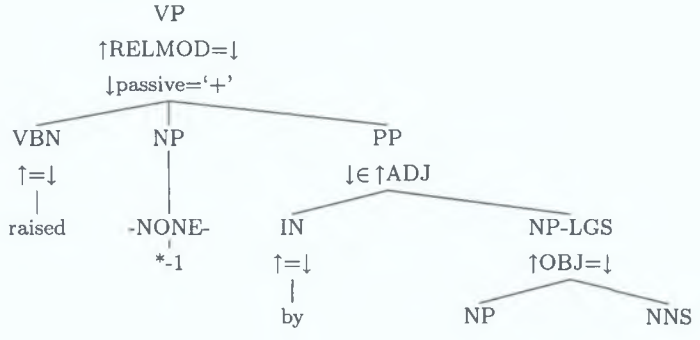


Figure 4.19: Partial tree containing -LGS tag for the string *If it does n't yield on these matters , and eventually begin talking directly to the ANC , the expectations and promise raised by yesterday 's releases will turn to disillusionment and unrest*

- In case the first condition did not catch all the passives, the algorithm also checks for an -LGS (logical subject) tag and makes the leftmost VP passive, provided that there is a VBN tag somewhere in the VP. The partially annotated tree associated with the sentence in (163), from WSJ 2454, tree 23, is shown in Figure 4.19. The text dominated by the VP is highlighted in bold.

  (163) If it does n't yield on these matters , and eventually begin talking directly to the ANC , the expectations and promise **raised by yesterday 's releases** will turn to disillusionment and unrest

4.5.2 Topic and Topicrel

A topicalised construction is one in which a displaced constituent bears a syntactic function usually associated with some other position in the sentence (this is discussed in detail in Section 2.7.1). Such long distance dependencies are encoded in the Treebank by means of index and trace information. The trace *T* is used to mark topicalisation (cf. Section 3.8.1). *T* bears a referential index that corresponds to the identity index of some other constituent in the sentence (e.g. a topicalised NP). The functional tag -TPC marks elements that appear before the subject in declarative sentences (cf. Section 3.2.3), and this is used

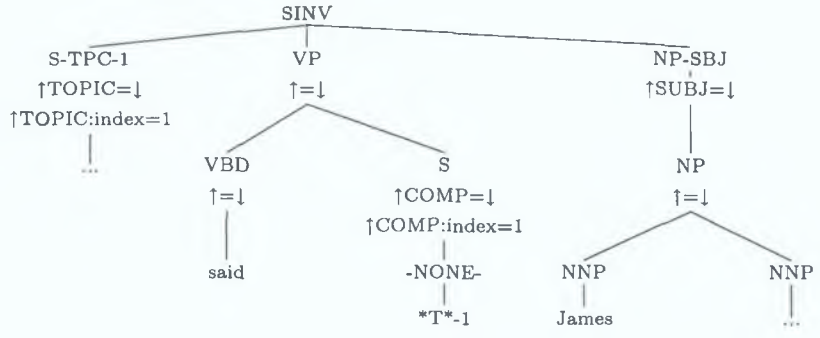


Figure 4.20: COMP linked to TOPIC for the string *We have no useful information on whether users are at risk, said James A. Talcott of Boston 's Dana-Farber Cancer Institute*

by our automatic f-structure annotation algorithm to annotate these nodes as a topic.

Nodes which have the functional tag -TPC receive the annotation '↑TOPIC=↓', even if the node has already been annotated, the previous annotation is overwritten. For example, S to left of head in a SINV is usually annotated as '↑COMP=↓' (cf. Annotation Matrices in Appendix B and C), but if it has the -TPC tag, the COMP annotation is overwritten with '↑TOPIC=↓'. The algorithm first applies the L/R context principles to the trees, annotating as many nodes as possible, and it is following this that the more specific rules are applied to refine the annotations. The partially annotated tree associated with the sentence in (164), from WSJ 0003, tree 09, is shown in Figure 4.20. The text dominated by S-TPC is highlighted in bold.

(164) **We have no useful information on whether users are at risk** ," said James A. Talcott of Boston 's Dana-Farber Cancer Institute

Annotating nodes with TOPIC information is done before the Catch-all and Clean-up component (cf. Section 4.6), in which most of the functional tags (described in Section 3.2.3) are used. The algorithm recurses through the tree, and looks for nodes that have the functional tag -TPC, or which have already received a TOPIC annotation, e.g. WHNP, which is annotated as a topic within a WHNP (cf. Annotation Matrices in Appendix B and C), as in (165), from WSJ 0111, tree 02:

(165) A successor was n't named , which fueled speculation that Mr. Bernstein may have clashed with S.I. Newhouse Jr. , **whose family company , Advance Publications Inc.** , owns Random House .

The annotations assigned by our algorithm are shown in the in (166):

(166)

WHNP→	WHNP ,	NP ,
	↑TOPIC=↓	↑=↓
	whose family company ,	Advance Publications Inc. ,

The algorithm then stores the index number associated with the node that receives the topic annotation (e.g. '↑TOPIC:index=1' if the trace is *-1, '↑TOPIC:index=2' if the trace is *-2, etc.), and the number of the node which contains that annotation (all the nodes are assigned f-numbers, cf. Section 2.3.3). When the empty node containing that trace is found, the topicalised node can be linked to where it belongs semantically by means of functional equations.[10]

'↑TOPICREL=↓' is used to annotate wh-elements in an SBAR in order to indicate that they are topics of a relative clause. As stated in Section 3.8.1.2, relative clauses are adjoined to the head noun phrase, and they bear an index that matches the reference index on the *T* in the position of the gap. The relative pronoun is given the appropriate wh-label, put inside the SBAR level and coindexed with a *T* in the position of the gap. In Figure 4.21, the NP node contains trace information, i.e. *NP → (-NONE- *T*-2)*, inside the PP-LOC node, and the NP node receives the annotation: '↑OBJ=↓,↑OBJ:index=2'. The OBJ is linked to the topic. The partially annotated tree associated with the sentence in (167), from WSJ 0102, tree 24, is shown in Figure 4.21.

(167) A driver has to find something **to hang the carrier on** , so the company supplies a window hook

This sentence contains the rule PP-LOC → IN NP highlighted in bold. In the f-structure, TOPICREL contains an index=2, which is indexed to the NP object within the PP, which is an adjunct. TOPIC in LFG is always associated with a grammatical function, but in the Treebank tagging, fronted adjuncts receive the -TPC label in cases where they are associated with a *T* in a clause

[10]There is a provision in our algorithm to ignore topics that are associated with a *T* in a clause inside the fronted construction, in order to prevent the algorithm going into a continuous loop, cf. Section 3.8.1.3.

contained inside the fronted construction (cf. Section 3.8.1.3, example (119), p.67).

4.5.3 Focus

The question word in an English wh-question appearing in initial position in the sentence is annotated with FOCUS. There are no functional tags for focus (corresponding to the -TPC tag for topic), so the algorithm recurses through the tree and looks for a focus annotation already on a node (which will be annotated using the matrix tables). The algorithm then stores the index number associated with it and the number of the node which contains that annotation.

When we find the empty node with the same index number, we find where the semantic origins of the node annotated with FOCUS. The empty node is annotated as it would have been if it contained lexical information, e.g. an SBAR occurring to the right of the head in a VP would be annotated as '↑COMP=↓', but if there is no annotation in the matrix tables it would be annotated as '↓∈↑ADJ' in the Catch-all and Clean-up component, cf. Section 4.6. The empty node is then linked to the node with the focus annotation (as is done for topic).

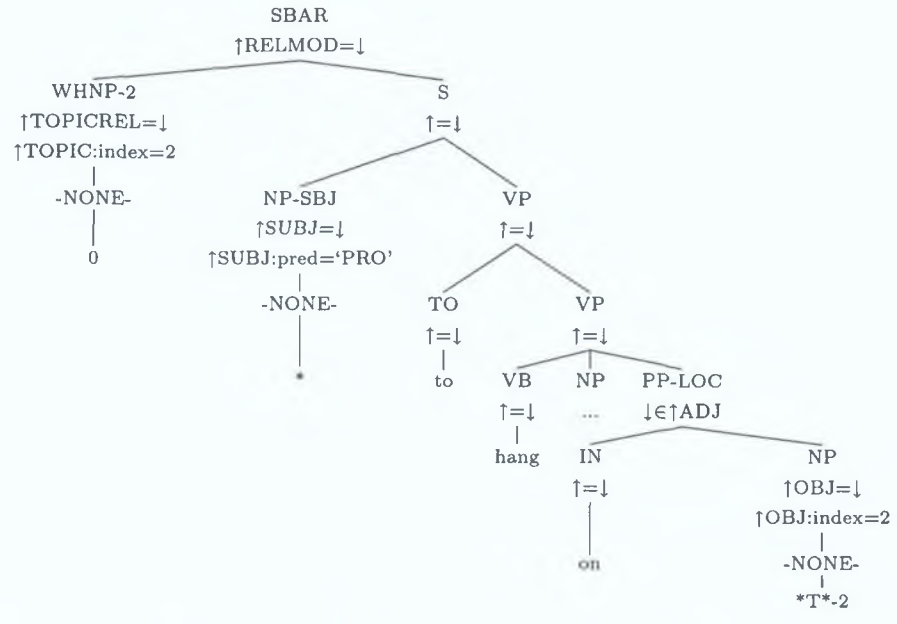
Wh-elements, e.g. WHADJP, WHADVP, WHNP, and WHPP, receive the annotation '↑FOCUS=↓' in an SBARQ (cf. Annotation Matrices in Appendix B), with an index annotation on the node (e.g. '↑FOCUS:index=1'). The gap is treated as if it were an ordinary (non-empty) constituent and an index annotation is added to link that node to the focus annotation. The partially annotated tree associated with the sentence in (168), from WSJ 0041, tree 62, is shown in Figure 4.22.

(168) **Who 's really lying** ? " asks a female voice

The '↑FOCUS=↑SUBJ' functional equation links the node containing the FOCUS annotation (in this case the WHNP-1 node), with the empty node that is linked to it by the *T*-1 trace.

4.6 Catch-all and Clean-up

'Catch-all and Clean-up' is the final component of our automatic annotation algorithm, and deals with any categories on the RHS of the rules that are left unannotated, or may cause clashes. In the 'Catch-all' step, we utilise some of



$$\left[\begin{array}{ll} \text{RELMOD} & \left[\begin{array}{ll} \text{TOPICREL} & \left[\text{INDEX} \quad 2\right] \\ \text{SUBJ} & \left[\text{PRED} \quad \text{PRO}\right] \\ \text{TO} & + \\ \text{INF} & + \\ \text{OBJ} & \left[\text{PRED} \quad \text{carrier}\right] \\ \text{ADJ} & \left[\begin{array}{ll} \text{OBJ} & \left[\text{INDEX} \quad 2\right] \\ \text{PRED} & \text{on} \end{array}\right] \\ \text{PRED} & \text{hang} \end{array}\right] \end{array}\right]$$

Figure 4.21: Topic linked to OBJ inside PP for the string *A driver has to find something to hang the carrier on, so the company supplies a window hook*

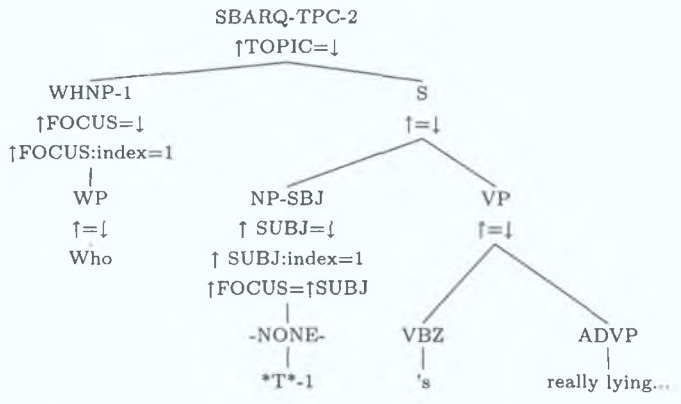


Figure 4.22: Focus linked to SUBJ for the string *Who 's really lying? " asks a female voice*

the functional tags provided in the Treebank annotations, which are described in Section 3.2.3, to annotate nodes which may not have received an annotation, or may have received an incorrect annotation. We initially designed a basic algorithm which did not utilise the functional tag information provided by the Treebank. Hence the reason it is in the final component of the algorithm that we utilise this tag information in the 'Catch-all' step.

In the case of partially annotated trees containing nodes with no annotations, we may get a number of unconnected f-structure fragments (cf. Section 5.2.2 for more discussion).

- A category with the functional tag -SBJ receives the annotation '↑SUBJ=↓'. If the node is empty, it will also be annotated with '↑PRED=PRO', where NP → (-NONE- *), or if it contains trace information, it will be annotated with '↑SUBJ:index=1', '↑SUBJ:index=2', etc., where NP → (-NONE- *T*-1), NP → (-NONE- *T*-2), etc.

- Categories with the functional tags -BNF and -DTV receive the annotation '↑OBL=↓'. -BNF is used only when the verb can undergo dative shift, and -DTV marks the dative object in the unshifted form of the double object construction (as described in Section 3.2.3). These tags can be used in our algorithm as it is always the case that the dative object is an oblique argument. This will overwrite any previous annotation that the node may have received from the annotation matrices.

- A category with the functional tag -PRD receives the annotation '↑=↓',

if no head has previously been assigned within that phrase (in which case the category with -PRD receives the annotation '↓∈↑ADJ').

- A preposition with the functional tag -CLR receives the annotation '↑OBL=↓' (cf. Section 4.3.1), unless PP-CLR is preceded by a comma, in which case it is annotated as '↓∈↑ADJ', as in (169), from WSJ 1037, tree 40:

  (169) Japanese consumers are increasingly eager to spend their money , **especially on high-priced goods such as 29-inch television sets and luxury cars**

  This sentence contains the rule VP → VB NP , PP-CLR. The text dominated by PP-CLR is highlighted in bold. It would be incorrect to annotate this phrase (and other examples found where PP-CLR is preceded by a comma) as an oblique, as the comma breaks up the verb phrase and provides a good indication that the prepositional phrase is not a required argument.

- All other categories with functional tags, such as -TMP (temporal), -LOC (locative) and -MNR (manner) receive the annotation '↓∈↑ADJ', as does PRN, unless previously annotated.

As a 'Clean-up' step, the algorithm checks for two (or more) subcategorisable grammatical functions which have received the same annotation, e.g. two NP nodes which are annotated as objects occurring under a mother VP node (in the case of a ditransitive verb). If more than one object or oblique appears at the same level, it will cause a clash, as the constraint solver would not be able to resolve the equations, resulting in no f-structure being generated. This indicates that errors were made in assigning the annotations which then leads to feature clashes, e.g. two objects assigned in a string, as in (170) from WSJ 0049, tree 33:

(170) The reason : the refusal of Congress to give federal judges a raise

Assigning the equation '↑OBJ=↓' to both *federal judges* and *a raise* would mean that the constraint resolution algorithm could not unify the paths *obj: pred: judge* and *obj: pred: raise*. '↑OBJ=↓' would be assigned to both following a look-up of the annotation matrices (described in Section 4.3.1, and listed in full in Appendix B and C), which state that an NP to the right of the head

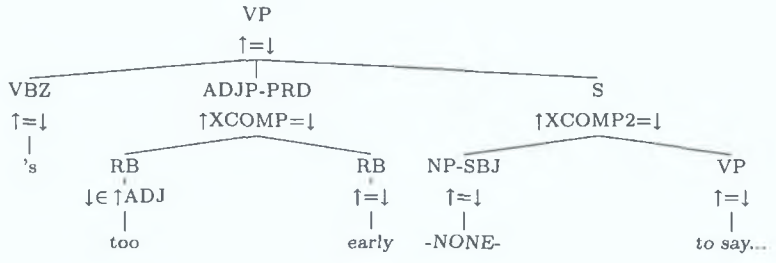


Figure 4.23: Partial tree containing XCOMP2 for the string *However , after two meetings with the Soviets , a State Department spokesman said that it 's " too early to say " whether that will happen*

in a VP is an object. However, this is handled adequately by the 'Catch-all and Clean-up' component of the annotation algorithm, in which the first NP remains annotated as '↑OBJ=↓', and the annotation for the second NP, *raise*, is overwritten as '↑OBJ2=↓', which prevents a unification clash.

The same is done for XCOMP, i.e. the first category (usually ADJP-PRD, S or VP) remains annotated as '↑XCOMP=↓', and the annotation for the second category is overwritten as '↑XCOMP2=↓', as in (171), from WSJ 0035, tree 3:

(171) However , after two meetings with the Soviets , a State Department spokesman said that it 's " too early to say " whether that will happen

There are no verbs in English which require more than one XCOMP, however, this clause is included in our algorithm to capture cases such as (171), in which it is necessary to capture the fact that the S should be annotated as an XCOMP. This needs to be analysed in further detail to assess which annotations should be kept and which overwritten. The partial tree for this string is given in Figure 4.23. It contains the annotation '↑XCOMP2=↓', having overwritten the previous '↑XCOMP=↓' annotation in order to allow the constraint solver to resolve the equations.

An unannotated nominal in an NP is annotated as an adjunct. If such a nominal is found and it is preceded by a comma, it is annotated as '↓∈↑APP', overwriting the '↓∈↑ADJ' annotation. This is not applied to the rules containing coordination, as nominals preceded by a comma in this instance would often be annotated as a conjunct, cf. Section 4.4 for more detail, but it catches occurrences of noun phrases annotated with functional tags, such as NP-LOC, which would otherwise be annotated as an adjunct, and should instead be annotated

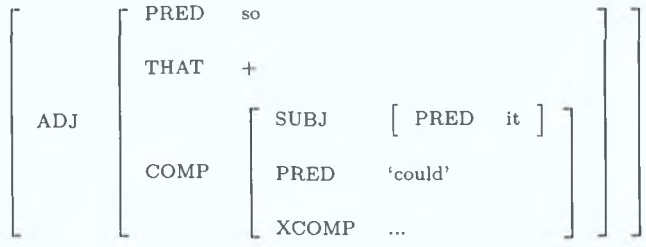


Figure 4.24: Parital f-structure where both INs are annotated as co-heads for the string *Northeast said it would refile its request and still hopes for an expedited review by the FERC so that it could complete the purchase by next summer if its bid is the one approved by the bankruptcy court*

as an apposition.

Obliques are treated as a separate case when searching for two of any sub-categorisable grammatical functions—they are both annotated as '↑OBL=↓' in the initial stages of the algorithm, but in this clean-up stage if there are two or more obliques at the same level, the second (rightmost) is overwritten as an adjunct, '↓∈↑ADJ'.

Two final cases conclude the 'Catch-all and Clean-up' component:

- In an SBAR rule, IN is the most likely category to be annotated as head, with an S to the right of the head annotated as '↑COMP=↓'. However, in a case such as (172), from WSJ 0013, tree 17, which contains the rule SBAR → IN IN S (with the text dominated by SBAR highlighted in bold), two INs are found and they are annotated as co-heads:

  (172) Northeast said it would refile its request and still hopes for an expedited review by the FERC **so that it could complete the purchase by next summer if its bid is the one approved by the bankruptcy court**

  IN IN covers the text *so that*. Annotating both in this case as co-heads will not cause a problem for our algorithm as *that* contains the lexical macro 'that=+' and does not contain a pred, so there will not be a problem with clashes where two different preds occur at the same level.

  The partial f-structure containing the text *so that* is shown in Figure 4.24, which shows that there is no clash with two different preds at the same level.

However, if both INs contained lexical entries with a pred, it would cause a problem with the constraint solver, as in (173), from WSJ 0112, tree 37, which contains the rule SBAR → IN IN S (with the text dominated by SBAR highlighted in bold):

(173) By 1987 , then-Speaker Jim Wright was discussing arms control in Moscow with Mikhail Gorbachev and then attempting to direct the president , through an appropriations rider , to treat the Soviets **as though the Senate had ratified SALT II**

IN IN covers the text *as though*. The two INs are found and they are annotated as co-heads. In this case, both contain lexical entries with a different pred, which would cause a problem when the constraint solver tries to resolve the equations. The Catch-all and Clean-up component checks to ensure this does not happen by overwriting the leftmost IN with the annotation '↓∈↑ADJ'. It is possible to amend the lexical macro for *though* to 'though = +', but that may not be the best possible solution to the problem. More analysis is needed, and noted for further work.

- A node which is annotated as '↑COMP=↓' is overwritten as '↑XCOMP=↓', if an empty node containing an index number (e.g. *-1, *-2) is found within that COMP. It does not happen for an empty node with no index number (*), a *?* node, or a null element (0). Our algorithm checks for a trace within the COMP annotated node (e.g. S or SINV), and if one is found with an index number, the annotation on that node is changed to XCOMP. An example of this is sentence (174), from WSJ 0054, tree 2:

(174) The company said it made the purchase in order **to locally produce hydraulically operated shovels**

This sentence contains the rule SBAR-TMP → IN NN S, with the text dominated by S highlighted in bold. According to the SBAR annotation matrix (cf. Appendix B), an S node occurring to the right of the head in an SBAR is annotated as '↑COMP=↓'. S is usually a COMP in an SBAR rule, i.e. in a rule such as SBAR → IN S, where IN is the tag used for *that*, and S is then usually a finite sentence. However, in this case, it should be an XCOMP, as indicated by the trace in the tree, which is shown in Figure 4.25.

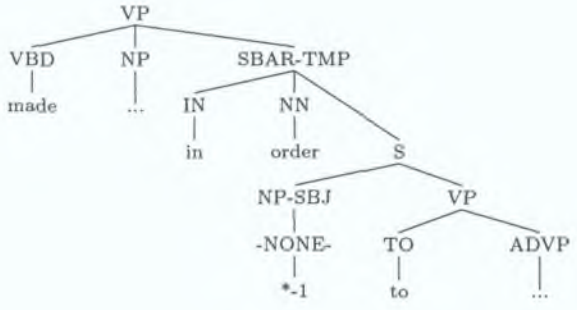


Figure 4.25: COMP annotation changed to XCOMP annotation for the string *The company said it made the purchase in order to locally produce hydraulically operated shovels*

4.7 Summary

In this chapter we have detailed the architecture and presented the linguistic content of our automatic f-structure annotation procedure which scales to the Penn-II Treebank. Each of the four main components, Left/Right (L/R) context, Coordination, Traces, and 'Catch-all and Clean-up' principles, used in the algorithm are described in detail. The annotation procedure begins by locating the head daughter and is driven by categorial, positional and functional tag information in the Penn-II trees. The annotation matrices we have constructed state generalisations in the form of a tripartition of local trees (of depth one, i.e. CFG rules). This allows a compact encoding of linguistic generalisations, with unfortunately some lack of detail and some mistakes. However, this gives us a methodology that can be easily scaled up to all of the Penn-II Treebank. Most errors are corrected in the Catch-all and Clean-up component of the algorithm. This allows the L/R context principles to be simple, clean, perspicuous, and maintainable. Coordinating constructions are treated in a separate component, as they can present particular problems in our automatic annotation algorithm, due to the often flat treebank analyses provided. The examples given in Section 4.4 give us some insight into the complex nature of coordinate structures, the complexity of which is exemplified by the often flat analysis provided by the Treebank taggers. LFG theory needs to be extended in cases such as three-part conjunctions (cf. Figure 4.14). However, examples such as (44), p.32, give a good basis for the necessary linguistic analysis of such structures.

We have advanced from 'proto' f-structures, which only interpret linguistic material locally, to detailed representations which capture long distance depen-

dencies. Using the traces and index information provided by the Treebank allows us to produce a more detailed representation in the f-structure of the sentence structure and meaning. Although further analysis is still needed before a more complete representation is produced, the algorithm scales up to a very large corpus of text, while maintaining robustness. Evaluation of this methodology is given in the next chapter.

Chapter 5

Evaluation

5.1 Introduction

The automatic f-structure annotation methodology described in Chapter 4 generates detailed f-structure representations for the more than one million words and 49,000 sentences of the Penn-II Treebank. In this chapter, we first evaluate our work by assessing the effectiveness of the annotation techniques we employ. This is done in two ways—in terms of quantity (the coverage and fragmentation of the annotations applied) and quality (the correctness of these annotations). In terms of quantity, we measure the coverage of the algorithm with respect to rule types and tokens, and calculate the degree of fragmentation, i.e. the number of unconnected f-structures produced by each sentence. The software used to calculate the coverage was implemented by [Cahill, forthcoming].

It is necessary for the annotation algorithm to be robust and encode linguistic generalisations while maintaining quality. In order to evaluate the quality of the f-structures produced, we compare the f-structure annotations automatically generated by the annotation algorithm against a manually constructed 'gold standard' set produced by this author. We first use the EVALB[1] test on the annotated trees to compare the automatically generated annotations on the trees against the gold standard annotated trees, and, as a further measure of comparison, we calculate precision and recall (P&R) on flat sets of term-based descriptions of the f-structures generated. This is done for all annotations, and then, as a further measure, coordination is evaluated in isolation, using the same qualitative measures, in order to provide a more in-depth analysis of the

[1] Available from `http://www.cs.nyu.edu/cs/projects/proteus/evalb/`

often complex coordinate structures. Integrating the coordinate structures with the other annotation principles would complicate principles and make them harder to maintain and extend, cf. Section 4.4. This chapter concludes with an exposition of some work related to our own, and a summary of the results produced by our evaluation techniques.

5.2 Quantitative Evaluation

One indication of the success of our method can be shown by evaluating the coverage it generates. The quantitative evaluation shown here measures first the percentage of nodes which receive an annotation with respect to rule types and tokens,[2] and then calculates the degree of fragmentation, i.e. the number of unconnected f-structure fragments produced by each sentence.

Some of the f-structures generated may be partial or unconnected, i.e. a sentence may be associated with two or more unconnected f-structure fragments, if some nodes have not received an annotation, cf. Table 5.1 and Table 5.2. Feature structure clashes (e.g. two categories which contain different preds being annotated as heads in the same phrase), cannot be resolved by the constraint solver and result in no f-structure being generated, cf. Section 5.2.2. Our aim is to associate each sentence with one complete (and correct) f-structure.

In previous work([Cahill et al., 2002c]), we gave results for the 'proto' f-structures generated, encoding basic predicate-argument-modifier relations. The results given here are for more detailed, 'proper', f-structures, encoding long distance dependencies using the trace and index information provided by the Penn-II Treebank (henceforth referred to as the Treebank). The f-structures evaluated here also contain root forms for pred values (e.g. the PRED value of the lexical entry for *invests* is *invest*), which was not done previously.

5.2.1 Coverage

The automatic f-structure annotation algorithm traverses trees top down and applies annotations at appropriate nodes. The string *The investment community, for one, has been anticipating a speedy resolution*, from WSJ 2308, tree 24, is the first of our 105 'gold standard' testset sentences.[3] The 'gold standard' set consists of 105 trees selected at random from the WSJ section 23 of

[2] An example of a rule *type* is NP → DT NN; *tokens* corresponding to NP → DT NN are *the man*, *a woman*, etc.

[3] Available from `http://www.computing.dcu.ie/~away/Treebank/testsent.html`

```
(S
   (NP-SBJ[up-subj=down]
      (DT[up-spec:det=down] The[up-pred='the'])
      (NN[down-elem=up:adj] investment[up-pred='investment',up-num=sing,up-pers=3])
      (NN[up=down] community[up-pred='community',up-num=sing,up-pers=3]))
   (, ,)
   (PP[down-elem=up:adj]
      (IN[up=down] for[up-pred='for'])
      (NP[up-obj=down]
         (CD[up=down] one[up-pred='one'])))
   (, ,)
   (VP[up=down]
      (VBZ[up=down] has[up-pred='have',up-tense=pres,up-pers=3,up-num=sing])
      (VP[up-xcomp=down,up-subj=down:subj]
         (VBN[up=down] been[up-pred='be',up-tense=past])
         (VP[up-xcomp=down,up-subj=down:subj]
            (VBG[up=down] anticipating[up-pred='anticipate',up-participle=pres])
            (NP[up-obj=down]
               (DT[up-spec:det=down] a[up-pred='a'])
               (JJ[down-elem=up:adj] speedy[up-pred='speedy'])
               (NN[up=down] resolution[up-pred='resolution',up-num=sing,up-pers=3])))))
   (. .))
```

Figure 5.1: Automatically annotated parse tree for the first of our 105 Penn-II test sentences, *The investment community, for one, has been anticipating a speedy resolution*

the Treebank. The average sentence length in this set is 23.98 words, with the shortest string 2 words, and the longest 45 words. These sentences were manually annotated and refined by this author to produce a complete and correct set of annotations.

Figure 5.1 shows the tree structure for the sentence with the automatic annotations generated by our algorithm. The resulting f-structure derived automatically from these annotations by our constraint solver is shown in Figure 4.6, p.97. The f-structure shows the results of the annotations applied (cf. Annotation Matrices, Appendix B), and resolved by the constraint solver. Percolation of subjects into the XCOMP is included by use of '↑SUBJ=↓SUBJ' annotations on the VP daughters in the VP rules.

In order to evaluate the coverage of our methodology, CFG rule types are extracted with the appropriate annotations. The CFG rules in (175) would be extracted from the annotated tree-structure in Figure 5.1:

(175)

S	$\longrightarrow$	NP-SBJ	PP	VP
		↑SUBJ=↓	↓∈↑ADJ	↑=↓
NP-SBJ	$\longrightarrow$	DT	NN	NN
		↑SPEC=↓	↓∈↑ADJ	↑=↓
PP	$\longrightarrow$	IN	NP	
		↑=↓	↑OBJ=↓	
NP	$\longrightarrow$	CD		
		↑=↓		
VP	$\longrightarrow$	VBZ	VP	
		↑=↓	↑XCOMP=↓	
			↑SUBJ=↓SUBJ	
VP	$\longrightarrow$	VBN	VP	
		↑=↓	↑XCOMP=↓	
			↑SUBJ=↓SUBJ	
VP	$\longrightarrow$	VBG	NP	
		↑=↓	↑OBJ=↓	
NP	$\longrightarrow$	DT	JJ	NN
		↑SPEC=↓	↓∈↑ADJ	↑=↓

This procedure is repeated for all 19,000 rule types in the Treebank. To evaluate the effectiveness of our automatic procedure, for a given CFG category, we measure the number of times the daughters of that category receive an annotation, e.g. for ADJP rules, we measure the number of times the daughters of ADJP receive an annotation. The results for the major rule types are given in Table 5.1. The software used to calculate the coverage was implemented by [Cahill, forthcoming]. The numbers shown in Table 5.1 illustrate the percentage of annotated daughters on the RHS of each of the main categories for all rule types in the Treebank. There are 30735 daughters appearing in the NP rules, 9 of which do not receive an annotation (0.03%). The results shown here for NP include NPs with functional tags, such as NP-TMP, NP-LOC, etc.

The coverage results for all the main categories, excluding PRN, are above 98%. The daughters (i.e. the categories on the RHS) of PRN are annotated

LHS Category	No. RHS Constituents	No. RHS annotated	Percentage annotated
ADJP	1641	1639	99.87
ADVP	605	603	99.66
NP	30735	30726	99.97
PP	1073	1071	99.81
PRN	1373	1283	93.44
QP	1555	1538	98.9
S	14817	14815	99.98
SBAR	409	409	100.00
SBARQ	256	256	100.00
SINV	1644	1643	99.93
SQ	650	648	99.69
UCP	649	647	99.66
VP	40824	40822	99.99
WHNP	367	365	99.45

Table 5.1: Percentage of Automatically Annotated Daughter Nodes in Rule Type RHSs. For example, for ADJP rules, there are 1641 daughters, 1639 of which receive an annotation.

according to the mother category of the PRN,[4] hence the reason that PRN is lower than the other results, as it is more difficult to analyse the results and improve on them when it is necessary to look one level higher, in order to analyse the rules. In comparison with the results given in [Cahill et al., 2002c], the results have improved by at least 5%, if not more in some cases (such as QP, which was 89.26%, and now stands at 98.9%). The rest of the main categories are all above 99%.

We also measure the percentage of annotated RHS constituents, i.e. the number of times a particular category (e.g. NP) occurs as a daughter in a rule, the results for which are shown in Table 5.2. For example, NP occurs as a daughter in a rule 8497 times, and is annotated 8470 times, i.e. 99.69% of the time. The lowest results shown are for NP and PP, which, as RHS daughters, are left unannotated the most, 27 and 22 times respectively, and even they are above 99%.

These results do not, however, give an indication of how many of the anno-

[4]If PRN is the mother category (i.e. occurs on the left-hand-side (LHS) of the rule), the algorithm then looks to the mother of PRN, and annotates the daughters of PRN according to the annotation matrices for that mother category. The PRN node is annotated as an adjunct, '$\downarrow \in \uparrow$ADJ', cf. Section 4.3.1, Figure 4.3, p.94.

RHS Category	No. RHS occurrences	No. RHS annotated	Percentage annotated
ADJP	1380	1377	99.79
ADVP	2421	2421	100.0
NP	8497	8470	99.69
PP	4294	4272	99.49
PRN	1028	1028	100.0
QP	334	334	100.0
S	2642	2639	99.89
SBAR	1287	1286	99.93
SBARQ	46	46	100.00
SINV	77	77	100.0
SQ	120	120	100.0
UCP	125	125	100.0
VP	3853	3852	99.98
WHNP	106	106	100.0

Table 5.2: Percentage of RHS Occurrences which Receive an Annotation. For example, the category ADJP occurs 1380 times on the RHS of rules, 1377 of which receive an annotation.

tations are correct, hence the need for a separate evaluation of the quality of the annotations, as described in Section 5.3.

5.2.2 Fragmentation

Once the nodes in the trees have been annotated by our automatic f-structure annotation algorithm, we collect these annotated nodes and feed them into the constraint solver. For each tree we aim to generate a single f-structure. However, in the case of partially annotated trees containing nodes with no annotations, as described in Section 5.2.1, we may get a number of unconnected f-structure fragments. No f-structure being generated indicates that errors were made in assigning the annotations which then leads to feature clashes, e.g. two objects assigned in a string, which can be shown by revisiting (170), p.120: *The reason : the refusal of Congress to give federal judges a raise.* Assigning the equation '↑OBJ=↓' to both *federal judges* and *a raise* would mean that the constraint resolution algorithm could not unify the paths *obj: pred: judge* and *obj: pred: raise.* '↑OBJ=↓' would be assigned to both following a look-up of the annotation matrices (described in Section 4.3.1, and listed in full in Appendix B and C), which state that an NP to the right of the head in a VP is an object. However, as

mentioned previously in Section 4.6, this is handled adequately by the 'Catch-all and Clean-up' component of the annotation algorithm, by assigning '↑OBJ2=↓' to *raise* which prevents a unification clash.

No. f-structure (fragments)	No. sentences	Percentage
0	226	0.4667
1	48141	99.41
2	58	0.1197

Table 5.3: Table for Fragmentation Results, illustrating sentences associated with no f-structure, one (complete) f-structure, or two unconnected f-structure fragments

Table 5.3 shows the current results achieved by our automatic annotation algorithm. There are 48257 f-structure fragments altogether. In the WSJ section of the Treebank there are 49,167 trees, but we only show results for 48,424 trees as **FRAG**(ment) and **X**(unknown constituents) are not included. They are the only 'constituents' that we have not covered to date. FRAG marks chunks of text that appear to be clauses, cf. (84), p.54, but lacks too many essential elements for the exact structure to be easily determined. X is the label used for unknown constituents. It would be extremely difficult to extract predicate-argument structure for FRAG or X, so they are not covered in our algorithm to date (cf. Section 4.3.1).

[Cahill et al., 2002b] reported that 78.836% of the trees in the Treebank received one f-structure, and this is only for "proto" f-structures. No use was made of the traces and index information in the Treebank when generating the f-structures reported in that research. 99.41% of the trees in the Treebank now receive one 'proper' f-structure. The figures reported by [Cahill et al., 2002b] also contained trees which received up to 11 f-structure fragments, but now the maximum number is 2. The 58 sentences which result in 2 fragments include annotations on empty nodes containing trace information which are not linked to a topic. An example of this is shown in Figure 5.2, from WSJ 1657, tree 24.

The problem here is that the mother S node (containing NP-SBJ and NP-PRD) is empty, and it is not annotated, leaving the annotation on the NP-SBJ disconnected. The empty NP-SBJ node is annotated with index information but is not linked to the NP-SBJ-1 node, *the Tela Accords* (as the algorithm only includes index traces for topicalised nodes). The two partial f-structures

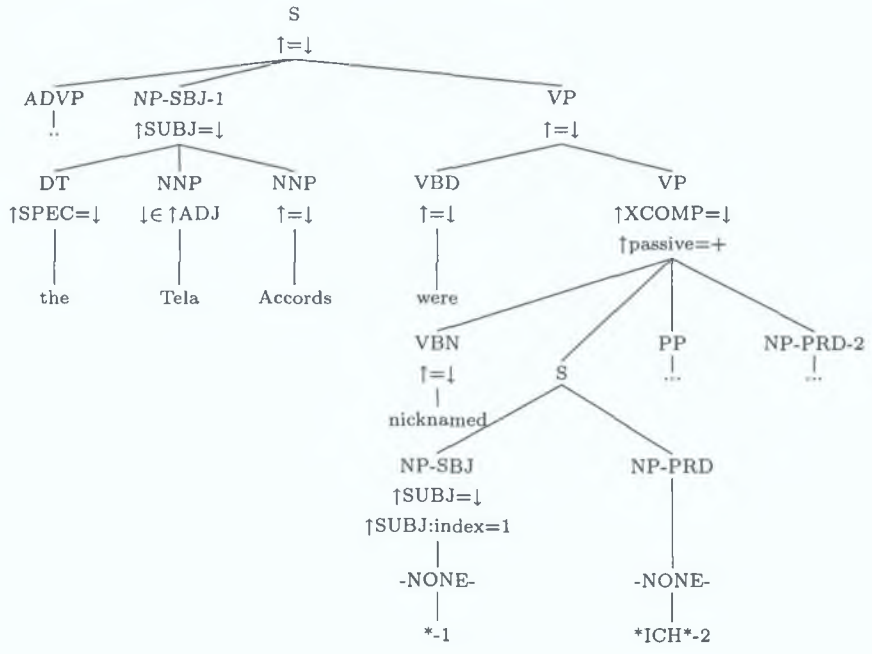


Figure 5.2: Partial tree for the string *Fittingly , the Tela Accords were nicknamed by Hondurans " the Dodd plan*

associated with this tree are given in Figure 5.3.

The second f-structure consists of the unlinked empty NP-SBJ node annotated with index information: "subj:index:1". Rather than amend the algorithm to include links to subject nodes without further analysis,[5] we continue to link up topic information and will try and resolve the fragmentation in further work.

226 sentences do not get an f-structure, and this is because of clashes such as those mentioned in (170). Instances of two objects at the same level are dealt with by the 'Catch-all and Clean-up' section of the automatic f-structure annotation algorithm (cf. Section 4.6). However, similar clashes have not yet been resolved. Further analysis of the trees involved is necessary in order to determine why, in some cases, sets of unresolvable annotations are generated.

Calculating fragmentation allows us to identify which sentences may contain attribute-value clashes, uninstantiated variables, etc. We can then re-examine the annotations assigned and improve annotations (as stated previously, they are linguistic generalisations and therefore may not capture specific cases which

[5]This would mean encoding specific cases, but the aim of the algorithm is to encode linguistic generalisations where possible.

```
subj : spec : det : pred : the
       adj : 2 : num : sing
                     pers : 3
                     pred : tela
       num : sing
       pers : 3
       pred : accords
adj : 1 : pred : fittingly
xcomp : subj : spec : det : pred : the
               adj : 2 : num : sing
                             pers : 3
                             pred : tela
               num : sing
               pers : 3
               pred : accords
        obj : spec : det : pred : the
                      adj : 4 : pred : dodd
                      pred : plan
                      num : sing
                      pers : 3
               tense : past
               pred : nicknamed
        adj : 3 : obj : num : pl
                            pers : 3
                            pred : honduran
                      pred : by
        passive : +
pred : be
tense : past



subj : index : 1
```

Figure 5.3: Two partial f-structures for the string *Fittingly , the Tela Accords were nicknamed by Hondurans " the Dodd plan*

occur only a few times).

Again, this does not assess the quality of the f-structures produced—it gives no indication of the correct grammatical function assignment (e.g. the distinction between obliques and arguments), for which qualitative evaluation is necessary.

5.3 Qualitative Evaluation

In order to assess the quality of the annotation algorithm, we constructed a 'gold standard' set of annotations and compared them against our automatic annotations. This also allows us to identify problems and therefore improve our automatic annotation algorithm still further.

We first use the EVALB test on the annotated trees to compare the automatically annotated trees against the gold standard annotated trees and, as a further measure of comparison, we calculate precision and recall (P&R) on flat sets of term-based descriptions of the f-structures generated.

The precision, recall, and F-score are measured using the formulae given:

$$\text{Precision} = \frac{\text{number of correct constituents in proposed parse}}{\text{number of constituents in proposed parse}}$$

$$\text{Recall} = \frac{\text{number of correct constituents in proposed parse}}{\text{number of constituents in treebank parse}}$$

$$\text{F-score} = \frac{2 \text{ x Precision x Recall}}{\text{Precision} + \text{Recall}}$$

Precision indicates how many of the annotations that we generate automatically are actually correct, and recall indicates how many of the correct annotations in the gold standard we actually generate automatically.

5.3.1 EVALB

EVALB[6] is a bracket scoring program designed by Sekine and Collins which reports precision, recall, non-crossing and tagging accuracy for tree data struc-

[6] Available from `http://www.cs.nyu.edu/cs/projects/proteus/evalb/`

tures. EVALB is an extremely severe evaluation metric, i.e. the set of equations produced automatically at a node must be identical to the set of manual annotations for the node in question. If the members of two sets of equations are the same, but do not appear in the same order, then EVALB would state that the two sets (and hence, the two nodes annotated with these sets, N{1,2,3} and N{3,2,1}) are not identical. For EVALB, therefore, the sets {2,1,3} and {1,3,2} are different (where 1, 2 and 3 represent f-structure annotations). Similarly, partial but correct annotations (e.g. {1,3} against {1,2,3}) are scored as mistakes by EVALB.

However, EVALB is freely available, and therefore is a convenient way to evaluate precision and recall as no special software needs to be designed. The results from running EVALB on our automatically annotated trees are presented in Table 5.4.

No. sentences	105
Bracketing Recall	93.83%
Bracketing Precision	93.83%
F-score	93.83%
Complete match	35.24

Table 5.4: Evaluation of Automatically Annotated Trees using EVALB on our Gold Standard Testset

Using the EVALB evaluation metric on all 105 test strings, we obtain 93.83% Precision and Recall (P&R), with 35 out of the 105 strings (33.56%) being completely correct. Precision and Recall are the same, as the configurational structure and the number of nodes in the test and gold standard trees are identical. The F-score, which is essentially the average of the precision and recall, is therefore also 93.58%.

This is an improvement of almost 4% from the results reported by [Cahill et al., 2002c], which were evaluated only on 'proto' f-structures. The f-structures that are now being generated encode trace and index information provided by the Treebank to capture long distance dependencies in terms of *reentrancies in the f-structures.*

5.3.2 Precision and Recall on F-structure Descriptions

As EVALB discounts some correct or partial automatic annotations, it gives us a lower bound with relation to the quality of our automatic f-structure annotation

algorithm. We also, therefore, calculate P&R directly on descriptions of f-structures, using a methodology similar to [Riezler et al., 2002], cf. Section 5.4. We translate the f-structure into a flat set of terms corresponding to f-structure descriptions. The computation from the f-structures to these terms was implemented by [Cahill, forthcoming]. Assume the simple f-structure in (176):

(176)
$$\begin{bmatrix} & & \begin{bmatrix} \text{PRED} & \text{'John'} \\ \text{NUM} & \text{SG} \\ \text{PERS} & \text{3rd} \end{bmatrix} \\ \text{SUBJ} & 2: & \\ \text{PRED} & \text{'love}\langle(\uparrow\text{SUBJ})(\uparrow\text{OBJ})\rangle\text{'} & \\ & & \begin{bmatrix} \text{PRED} & \text{'Mary'} \\ \text{NUM} & \text{SG} \\ \text{PERS} & \text{3rd} \end{bmatrix} \\ \text{OBJ} & 3: & \end{bmatrix}_1$$

We automatically 'translate' (176) into a flat set of terms corresponding to f-structure descriptions, as in Table 5.5.

`subj(1,2)`	pred(2,john)	num(2,sg)	pers(2,3rd)
`pred(1,love)`			
`obj(1,3)`	pred(3,mary)	num(3,sg)	pers(3,3rd)

Table 5.5: Flat Set of Terms corresponding to F-structure Descriptions

Each f-structure is represented as a set of terms of the form: 'relation(argument,argument)'. As the order of the terms within the set is not important (unlike EVALB, which treats the order of arguments in terms as relevant), the calculation should yield higher results (or at least no worse) than those given in Section 5.3.1. We also calculate preds-only f-structures. The 'preds-only' part of an f-structure is that part where every path from the root terminates in a PRED:lemma (PRED:semantic form) pair, i.e. we strip out lexical attributes, such as person and number, temporal attributes, etc. We then match these sets of triples from the gold standard to those obtained from the automatically generated f-structure. The P&R software used was written by [Crouch et al., 2002]. Table 5.6 shows the amount of annotations that we get correct for each grammatical function, when our automatic method is measured against the gold standard.

Table 5.6 breaks down the results so that it can be clearly seen which annotations are incorrect. For example, the 'conj' annotation occurs 151 times in the

Dependency	Precision	Recall	F-score
adjunct	897/1013 = 89	897/944 = 95	92
app	0/0	0/20	0
comp	67/71 = 94	67/80 = 84	89
conj	144/151 = 95	144/151 = 95	95
focus	1/1 = 100	1/1 = 100	100
obj	407/437 = 93	407/429 = 95	94
obl	20/21 = 95	20/55 = 36	53
part	7/9 = 78	7/11 = 64	70
poss	61/65 = 94	61/65 = 94	94
relmod	50/53 = 94	50/56 = 89	92
spec	265/269 = 99	265/269 = 99	99
subj	282/320 = 88	282/316 = 89	89
topic	12/12 = 100	12/13 = 92	96
topicrel	21/23 = 91	21/22 = 95	93
xcomp	135/174 = 78	135/163 = 83	80

Table 5.6: Precision and Recall on Descriptions of F-structures for each Grammatical Function

gold standard. Our automatic method correctly annotates nodes as 'conj' 144 times, resulting in an F-score of 95. There are 20 occurrences of the apposition annotation in the gold standard, but our algorithm does not get any of them right—at present it does not annotate anything as 'app' ($\downarrow\in, \uparrow$ADJ) in the gold standard set. 'obl' also yields a poor result, receiving only 20 out of 55 expected annotations. This needs to be addressed in further work.

A summary of the overall results for P&R calculated directly on descriptions of f-structures is shown in Table 5.7.

	All annotations	Preds-only annotations
Precision	93.53%	90.46%
Recall	94.69%	91.26%
F-score	94.11%	90.86%

Table 5.7: Overall Precision and Recall on Descriptions of F-structures

These results show an increase in recall on those given in [Cahill et al., 2002c], but a decrease in precision. The results here, however, are on f-structures encoding the index and trace information provided by the Treebank. 48.71% of the 105 strings are completely correct for all annotations, and 23.7% of the 105 strings are completely correct for the

preds-only annotations. The F-score is higher for *all* annotations *than* for preds-only, as the former takes into account matching lexical information. The F-score for all annotations shown here in Table 5.7 is, as expected, higher than the F-score resulting from the EVALB evaluation metric in Table 5.4.

5.3.3 Evaluation of Coordination

As coordination is handled in a separate component in the algorithm, we decided to evaluate the quality of coordinate structures in isolation in order to get an accurate assessment of how effective the component is[7]. In order to evaluate this correctly, we manually constructed a new gold standard for 50 randomly selected sentences from the WSJ section 23 of the Treebank, each of which contain at least one coordinating conjunction, CC. We then ran the quality experiments as described in Section 5.3.2. The results for EVALB are presented in Table 5.8.

No. sentences	50
Bracketing Recall	90.28%
Bracketing Precision	90.28%
F-score	90.28%
Complete match	28

Table 5.8: Evaluation of Automatically Annotated Trees containing CC using EVALB

Using EVALB, for all 50 test strings, we obtain 90.28% Precision and Recall (P&R), with 28 out of the 50 strings (56%) being completely correct.

As mentioned previously, EVALB discounts some correct automatic annotations, so again we calculate P&R directly on flat sets of term-based descriptions of f-structures for coordination, results of which are shown in Table 5.9.

	Preds-only annotations
Precision	88.82%
Recall	92.26%
F-score	90.5%

Table 5.9: Precision and Recall on Descriptions of F-structures containing CC

The F-score on the flat set of terms corresponding to f-structure descriptions again yields a higher F-score than the EVALB evaluation metric. Evaluation of

[7]The results given in Section 5.3.2 are for all annotations, including coordinate structures

rules which all contain at least one CC are very useful to pinpoint how we have improved the coordination component by any amendments to the algorithm (cf. Section 4.4). However, as the condition states that the sentences must contain at least one coordinating conjunction, CC, it is possible that a sentence may contain only one CC, and this may be the leftmost on the RHS, and the sentence is therefore not considered to contain a coordinate structure (cf. Section 4.4.1). It is necessary to extract the coordinate structures themselves to give more precise results of the amount of times we annotate the coordinate structure itself correctly, rather than entire sentences which contain coordinate structures.

5.4 Related Work

The resources produced by our automatic f-structure annotation algorithm can be used to automatically construct wide-coverage, probabilistic LFG parsing resources. As described in [Cahill et al., 2003], we automatically derive wide-coverage, rich unification grammars from the Penn-II Treebank resource via our automatic Lexical-Functional Grammar (LFG) f-structure annotation algorithm. Two parsing architectures are outlined: the *pipeline* and the *integrated* model. In the pipeline model, a Probabilistic Context-Free Grammar (PCFG) is extracted from the Treebank and used to parse new text. The tree with the highest probability is passed to the automatic annotation algorithm, where it is annotated with f-structure equations. These equations are collected and sent to a constraint solver to generate an f-structure, as shown in Figure 5.4.

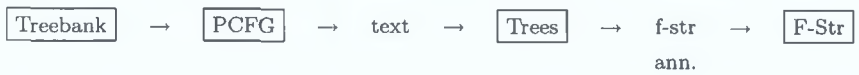


Figure 5.4: Pipeline Model

The quality of the output from the parser depends on the quality of the algorithm. In the integrated model, the trees are first annotated using the annotation algorithm, and then an annotated PCFG (A-PCFG) is extracted from the annnotated Treebank. Text is parsed with the annotated rules, and again the tree with the highest probability for a string is selected. The f-structure equations are collected and sent to a constraint solver to generate an f-structure, as shown in Figure 5.5.

As the trees are first annotated in the A-PCFG, there will be more rules

Figure 5.5: Integrated Model

extracted, and they will be different to those in the PCFG, and will be associated with different probabilities. One difference is that the A-PCFG distinguishes between subject and object NPs, therefore generating better results. As in the pipeline model, the quality of the output from the parser depends on the quality of the annotation algorithm. More detail on both models and the parsing experiments can be found in [Cahill et al., 2003].

Simple PCFG- and history-based parsing technology is used to enable the construction of systems for on-line parsing. Our best results to date use the integrated model and achieve over 63.04% F-score against the 2,400 section 23 treebank trees of length <40 as annotated by the automatic f-structure annotation algorithms. The f-structures derived from the parsers described in [Cahill et al., 2003] resolve long distance dependencies, which are very important for determining meaning representations or logical forms. Our results given in the previous sections are discussed in relation to some work on related parsing methods and results achieved:

- [Carroll et al., 1999] try to develop a language and application-independent corpus annotation scheme for evaluating syntactic parsers. They provide a gold standard with relations similar to information contained in the LFG representations to evaluate dependency relations. The gold standard corpus consists of 500 sentences (10K words), from the Susanne corpus given annotations according to their dependency relations.[8] They compute precision, recall, and F-score for each type of relation (e.g. subj, obj, comp, etc.) against the 10K-word test corpus.

- [Liakata and Pulman, 2002] employ a method for retrieving predicate-argument structures that encompass the complexity of tree structures, but employ few template rules. This system operates on a flattened, morphologically enriched version of the Treebank, allowing access to all levels of the tree simultaneously. They focus on building structures that reflect basic predicate-argument relations, rather than assigning the appropriate

[8]The corpus is available online at `http://www.cogs.susx.ac.uk/lab/nlp/carroll/greval.html`

case or thematic roles to phrases. To do this, they first transform the trees to take the form of embedded lists, and then separate the POS tags from the grammatical function and semantic role tags in order to clearly distinguish between the node's POS label, and its features and values. Use is made of the trace information provided by the Treebank to locate linguistic material where it should be interpreted semantically rather than where it occurs syntactically. Trace information is also used to map the Treebank trees into logical forms. A pre-processing trace resolution stage is applied to handle empty elements in infinitives, wh-movement and passives, while creating a new flattened version of the Treebank. (In our automated process, we do not transform the Treebank, we leave it intact.) Ellipsis and logical subject detection in passives are handled in a later stage. They manually constructed a 'gold standard', as we have also done (cf. Section 5.3.2), taking 100 sentences at random from the Treebank, counting the number of distinct predicates, the number of arguments expected for each predicate, and the level of embedding of the predicates. The automatic method is then evaluated against this set, resulting in a recall score of 86.32% for predicates, and 93.16% for arguments.

- [Riezler et al., 2002] develop a stochastic parsing system consisting of a grammar, a constraint-based parser and a stochastic disambiguation model. The LFG grammar for English developed in the ParGram project [Butt et al., 1999] is used. The WSJ section of the Treebank is used for estimation and testing purposes, and the grammar was modified to parse part of speech tags and labelled bracketing [Crouch et al., 2002]. They create a stripped-down version of the Treebank that uses only the POS tags and labelled brackets relevant for determining grammatical relations. LFG lexical entries are given to the WSJ labelled brackets which constrain the c- and f-structure of the parse. The stochastic disambiguation model is tested on [Carroll et al., 1999]'s gold standard from the Brown Corpus, and achieves a 79% F-score. The evaluation measure counts the number of predicate-argument relations in the f-structure of the parse selected by the stochastic model that match those in the gold standard annotation. It also maps predicate-argument relations in LFG f-structures to the dependency relations proposed by [Carroll et al., 1999]. This measures the matches of dependency relations to [Carroll et al., 1999]'s gold standard corpus. Under this measure, against a gold standard set of 700 sentences randomly

extracted from section 23 of the WSJ Treebank and annotated according to [Riezler et al., 2002]'s LFG scheme, they reach a 76% F-score.

We now give a comparison of our evaluation against other standards. [Carroll et al., 1999] constructed a gold standard corpus of 500 sentences from the Susanne corpus. This corpus is annotated according to dependency relations so it will be necessary to translate our f-structures into the same format as these dependency relations, and then it will be possible to more accurately compare results. [Liakata and Pulman, 2002] have also constructed a gold standard from the Penn-II Treebank, counting the number of distinct predicates, the number of arguments expected for each predicate, and the level of embedding of the predicates. [Riezler et al., 2002] have constructed a gold standard set of 700 sentences randomly extracted from section 23 of the WSJ Treebank and annotated according to [Riezler et al., 2002]'s LFG scheme. The PARC 700 Dependency Bank consists of 700 sentences which were randomly extracted from section 23 of the UPenn Wall Street Journal treebank, parsed with our LFG grammar, and given gold standard annotations of grammatical dependency relations by manual correction and extension [Crouch et al., 2003]. A possibility for further work is to translate our f-structures into the same format as the dependency relations described in [Carroll et al., 1999], or the f-structures of [Riezler et al., 2002]. Evaluating our work against these standards will reveal a more accurate comparison of our work with others in this area.

5.5 Summary

This chapter assesses the effectiveness of our automatic f-structure annotation algorithm in two ways—in terms of quantity (the amount of coverage and fragmentation of the annotations applied) and quality (the correctness of these annotations). There has been an improvement since the results shown in [Cahill et al., 2002c], which were generated for 'proto' f-structures, encoding only basic predicate-argument-modifier information, whereas the results shown here make use of the trace and index information in the Treebank to provide richer, 'proper', f-structure representations.

We measure the coverage of the algorithm with respect to rule types and tokens, and measure fragmentation, i.e. the number of complete f-structures produced by each sentence. At the moment 99.41% of the trees that we cover in the Treebank (excluding FRAG and X) now receive one f-structure. Some

of the f-structures generated may be partial or unconnected, i.e. a sentence may be associated with two or more unconnected f-structure fragments, if some nodes have not received an annotation, cf. Table 5.1 and Table 5.2. The maximum number of f-structure fragments is 2, with just 58 sentences resulting in 2 fragments. Feature structure clashes, e.g. two catgories which contain different preds being annotated as heads in the same phrase, cannot be resolved by the constraint solver and result in no f-structure being generated, cf. Section 5.3.2. No f-structure is currently generated for 226 sentences. The aim is to associate each sentence with one complete (and correct) f-structure. Work is ongoing to resolve this fragmentation.

In order to evaluate the quality, we compare the f-structure annotations generated by our algorithm against a manually constructed 'gold standard' set produced by a linguist. We first use the EVALB test on the annotated trees to compare the automatically annotated trees against the gold standard annotated trees and, as a further measure of comparison, we calculate precision and recall (P&R) on the flat set descriptions of the f-structures generated. The F-score for EVALB is 93.83%, with P&R on flat sets of term-based descriptions of the f-structures generated giving an F-score of 94.11% for all annotations. The P&R that we calculate yields higher results as the order of the terms within the set is not important (unlike EVALB, where the order of the equations matters).

This evaluation is done for all annotations, and then, as a further measure, coordination is evaluated in isolation, using EVALB and the P&R measures used for the entire annotation algorithm. The F-score for EVALB is 90.28%, with P&R on flat sets of term-based descriptions of the f-structures generated giving an F-score of 90.5%.

The various evaluation measures described in this chapter give a good indication of the quality of the annotation algorithm, which in turn affects the quality of the resources produced, and the quality of a parser or any other applications which use these resources. We continue in the next chapter by describing some applications of the resources produced, and further work to be implemented on the automatic annotation algorithm.

Chapter 6

Conclusion

This chapter concludes the thesis by summarising our work to date, giving a review of each of the main chapters, analysing the progress made, and outlining avenues for further work to improve the resources generated

In this thesis, we described the design and evaluation of the linguistic basis of an automatic f-structure annotation algorithm for the Wall Street Journal (WSJ) section of the Penn-II Treebank, which consists of more than 1,000,000 words, tagged for part-of-speech information, in about 50,000 sentences and trees

The Penn-II Treebank is automatically annotated with Lexical-Functional Grammar (LFG) f-structure equations, from which we can automatically extract a large-coverage unification grammar, thereby generating a new linguistic resource from the Treebank Such large-coverage unification grammars are extremely time-consuming, expensive and difficult to obtain

LFG was designed to be implementable in computational systems from the beginning, and is therefore particularly well suited to the construction of automatic f-structure annotation principles as we have done in our automatic f-structure annotation algorithm Chapter 2 introduces LFG as a theory of linguistic description, presenting the background to the theory and detailing some of the main principles, such as the Lexical Integrity Principle and the Principle of Economy of Expression It describes the grammatical representations used (c- and f-structure), including some detail on X′ theory and grammatical functions where appropriate It also provides a detailed step-by-step example of the construction of an f-structure from a simple sentence of English The way in which LFG deals with linguistic phenomena such as passivisation and long

distance dependencies, examining topicalisation, wh-questions, relative clauses, and functional uncertainty, is also discussed

Chapter 3 outlines the tagging and bracketing principles in the Penn-II Treebank, as detailed in [Santorini, 1991], [Marcus et al , 1994], and [Bies et al , 1995], paying particular attention to coordinate structures, the modification of noun phrases, and the handling of null elements Treebank grammars typically involve large sets of lexical tags and non-lexical categories, as syntactic information tends to be encoded in monadic category symbols They often feature flat rules in trees that do not express linguistic generalisations, making the task of automatic annotation more complex, as annotation principles have to identify subsequences on the RHS of the corresponding CFG rule for annotation Passivisation and long distance dependencies are encoded in the Penn-II Treebank by means of trace and index information on null elements, which differs from theories of LFG which limit the use of empty elements in c-structure However, use can be made of this in our automatic f-structure annotation algorithm, so we extend the theory of LFG to incorporate the linguistic assumptions of the Penn-II Treebank, describing some of the difficulties encountered when applying the theory to real data

Chapter 4 sketches the architecture and presents the linguistic content of our automatic f-structure annotation procedure The algorithm is implemented in Java [Cahill, forthcoming], and takes the form of a recursive procedure, which traverses the Penn-II treebank top-down and annotates the nodes with f-structure information Each of the four main components, Left/Right (L/R) context, Coordination, Traces, and 'Catch-all and Clean-up' principles, which are used in the algorithm are described in detail The annotation procedure begins by locating the head daughter and is driven by categorial, positional and functional tag information in the Penn-II trees The annotation matrices (cf Appendix B and C) we have constructed state generalisations in the form of a tripartition of local trees of depth one, i e CFG rules This allows a compact encoding of linguistic generalisations, with may include some lack of detail and errors However, this gives us a methodology that can be easily scaled up to the WSJ section of the Penn-II Treebank Most errors are corrected in the Catch-all and Clean-up component of the algorithm, which allows the L/R context principles to be simple, clean, perspicuous, and maintainable Coordinate structures present particular problems in our automatic annotation algorithm, due to the often flat treebank analyses provided Integrating the coordinate structures with the other annotation principles would complicate principles and

make them harder to maintain and extend For this reason, we decided to deal with the coordinate structures in a separate module in our algorithm

We have advanced from 'proto' f-structures, which only interpret linguistic material locally, to detailed representations which capture long distance dependencies Using the trace and index information provided by the Treebank allows us to produce a more detailed representation in the f-structure of the sentence structure and its meaning Although further analysis is still needed before a more complete representation is produced, the algorithm scales up to a very large corpus of text, while maintaining robustness

Chapter 5 assesses the effectiveness of our automatic f-structure annotation algorithm in two ways—in terms of quantity (the coverage and fragmentation of the annotations applied) and quality (the correctness of these annotations) [Cahill et al , 2002c] showed results which were generated for 'proto' f-structures, encoding only basic predicate-argument-modifier information, whereas the results shown here make use of the trace and index information in the Treebank to provide richer, 'proper', f-structure representations, and also show an improvement since the results shown in [Cahill et al , 2002c]

We measure the coverage of the algorithm with respect to rule types and tokens, and calculate the degree of fragmentation, i e the number of complete f-structures produced by each sentence, or tree At present 99 41% of the trees in the Treebank receive one 'proper' f-structure 0 12% of the f-structures generated are partial or unconnected, i e 58 sentences are associated with two unconnected f-structure fragments, because some nodes have not received an annotation The figures reported by [Cahill et al , 2002b] contained trees which received up to 11 f-structure fragments, but now the maximum number is 2

Feature structure clashes, e g two categories which contain different preds being annotated as heads in the same phrase, cannot be resolved by the constraint solver and result in no f-structure being generated (cf (170), p 120) At present, 0 47% (226 sentences) do not generate an f-structure Instances of two objects at the same level are dealt with by the 'Catch-all and Clean-up' section of the automatic f-structure annotation algorithm (cf Section 4 6) However, other clashes have not yet been resolved Further analysis of the trees involved is necessary in order to determine why, in some cases, sets of unresolvable annotations are generated, e g why two different preds occur at the same level The aim is to associate each sentence with one complete (and correct) f-structure Calculating fragmentation allows us to identify which sentences may contain attribute-value clashes or uninstantiated variables We can then re-examine the

annotations assigned and improve the annotations (as they are linguistic generalisations, they may not capture specific cases which occur only a few times)

In order to evaluate the quality of the f-structures generated, we compare the f-structure annotations automatically generated by our algorithm against a manually constructed 'gold standard' set of 105 annotated trees (cf Section 5 3) We first use the EVALB test on the annotated trees to compare the automatically annotated trees against the gold standard annotated trees and, as a further measure of comparison, we calculate precision and recall (P&R) on the flat sets of term-based descriptions of the f-structures generated The F-score for EVALB is 93 83%, with P&R on flat sets of term-based descriptions of the f-structures generated giving an F-score of 94 11% for all annotations The P&R that we calculate yields higher results as the order of the terms within the set is not important (unlike EVALB, where the order of the equations matters) At present, we discount trees containing **FRAG**(ment) and **X**(unknown constituents) when evaluating our automatically generated f-structures It would be extremely difficult to extract predicate-argument structure for FRAG or X (cf Section 5 2 2), but it is a possibility for further work

Comparing these annotations against our automatically annotated trees allows us to identify problem areas An example of this is apposition There are 20 occurrences of the apposition annotation in the gold standard, but our algorithm does not get any of them right—at present it does not annotate anything as '$\downarrow \in \uparrow$APP' in the gold standard 'obl' also yields a poor result, receiving only 20 out of 55 expected annotations This needs to be addressed in further work

This evaluation is performed for all annotations, and as a further measure, coordination is evaluated in isolation, using EVALB and the P&R measures used for the entire annotation algorithm The F-score for EVALB is 90 28%, with P&R on flat sets of term-based descriptions of the f-structures generated giving an F-score of 90 5% This evaluation allows a more in-depth analysis of coordinate structures, which can prove difficult for an automatic annotation process

The work contained in this thesis has shown that the automatic f-structure annotation algorithm we have constructed is robust, scales up to a very large corpus, and yields promising results However, further analysis is still needed before a more complete representation is produced

Further analysis is needed on CC rules Evaluation of rules which contain at least one CC are very useful to pinpoint how we have improved the coordination component by any amendments to the algorithm (cf Section 4 4) However, as

the condition states that the sentences must contain at least one coordinating conjunction, CC, it is possible that a sentence may contain only one CC, and if this is the first non-punctuation element on the RHS, the sentence is therefore not considered to contain a coordinate structure (cf Section 4 4 1) One possible way of analysing the CC rules in more detail is to extract the coordinate structures themselves to give more precise results of the amount of times we annotate the coordinate structure itself correctly, rather than entire sentences which contain the coordinating conjunction, CC When we reach the point in the algorithm where we have identified that it is a coordinate structure, we can then extract the rule containing the coordinate structure Manually constructing a gold standard of such structures and comparing them will allow us to see exactly how accurate our CC component is, and implement any necessary changes

At the moment, the algorithm attaches certain modifiers within coordinate structures at the wrong level, annotating it to modify the entire coordinate structure, when they should just modify a conjunct (cf Section 4 4 2) Noun phrases, in coordinate structures and otherwise, also cause problems for the automatic f-structure annotation algorithm when trying to encode generalisations, as, in the Penn-II Treebank, consecutive unrelated adjuncts are non-recursively attached to the NP they modify, resulting in a flat right-hand-side (cf Section 3 7) Noun phrases need to be analysed in greater detail in order to address issues such as these We have not yet analysed adjectival modifiers as HEADMOD, however, this is an avenue planned for further work Named entity recognition could help disambiguate complex coordinate phrases, we have not yet integrated a named entity recogniser with our automatic f-structure annotation algorithm, but it is an avenue for further work Section 3 9 2 proposes annotations for two-part conjunctions, such as *either or*, in which the first half of the conjunction is annotated as a 'PRECONJ', which is constrained to occur with a particular 'CONJ-FORM', provided by its paired conjunction ([Dalrymple, 2002], p 367) The features PRECONJ and CONJ are classified as non-distributive features (cf (44), p 32) No special rules for two-part conjunctions have yet been implemented in our automatic f-structure annotation algorithm, but are planned for future work Examples including multiple conjunctions also need to be analysed

Further analysis on the UCP rules (cf Section 4 4 4) is noted for further work In order to do this, it will be necessary to evaluate the UCP rules in isolation, by extracting rules where the mother category is UCP, manually annotating and refining these annotations to produce a complete and correct set

of annotations, and comparing these against our automatically generated annotations This will allow us to pinpoint exactly where this component of the algorithm needs improvement

TOPIC in LFG is always associated with a grammatical function, but in the Treebank tagging, fronted adjuncts receive the -TPC label in cases where they are associated with a *T* in a clause contained inside the fronted construction (cf Section 3 8 1 3, example (119), p 67) As our algorithm is implemented using generalisations, these adjuncts are annotated as TOPIC This is done as a general rule in our algorithm states that any node with the -TPC tag receives the TOPIC annotation This needs to be investigated in more detail to decide the most appropriate annotations for such cases Where two INs are found, they are annotated as co-heads As mentioned in Section 4 6, if both contain lexical entries with a different pred, it would cause a problem when the constraint solver tries to resolve the equations The Catch-all and Clean-up component checks to ensure this does not happen by overwriting the leftmost IN with the annotation '$\downarrow \in \uparrow$ADJ' It is possible to amend the lexical macros, but that may not be the best possible solution to the problem More analysis is needed, and noted for further work

The various evaluation measures described in this chapter give a good indication of the quality of the annotation algorithm, which in turn affects the quality of the resources produced, and the quality of a parser or any other applications which use these resources

However, it is also necessary to evaluate our work against other standards than our own manually constructed set [Carroll et al , 1999] constructed a gold standard corpus of 500 sentences from the Susanne corpus This corpus is annotated according to dependency relations so it will be necessary to translate our f-structures into the same format as these dependency relations, and then it will be possible to more accurately compare results [Liakata and Pulman, 2002] have also constructed a gold standard from the Penn-II Treebank, counting the number of distinct predicates, the number of arguments expected for each predicate, and the level of embedding of the predicates [Riezler et al , 2002] have constructed a gold standard set of 700 sentences randomly extracted from section 23 of the WSJ Treebank and annotated according to [Riezler et al , 2002]'s LFG scheme (cf Section 5 4) The PARC 700 Dependency Bank consists of 700 sentences which were randomly extracted from section 23 of the UPenn Wall Street Journal treebank, parsed with our LFG grammar, and given gold standard annotations of grammatical dependency relations by manual correction

and extension [Crouch et al , 2003] Evaluating our work against these standards will reveal a more accurate comparison of our work with others in this area

The further work mentioned here is needed to improve the existing algorithm We have not fully exploited the information encoded in the Penn-II Treebank, so another avenue for further work is to make more use of the functional tags (cf Section 3 2 3), and the pseudo-attachment notation (cf Section 3 8 4) which has not yet been incorporated in our algorithm

We also need to implement grammaticahty checks (cf Section 2 4 and [van Genabith et al , 1999]), i e completeness, coherence and unification Completeness and coherence checks ensure that the f-structure is grammatically correct, i e that the f-structure contains all the necessary grammatical functions, but no unnecessary functions The uniqueness condition ensures that attributes have at most one value, thereby disallowing multiple differing values (such as num=pl and num=sg, cf (37), p 28)

Before we made use of the trace information, we could only produce 'proto' f-structures, which interpreted linguistic material locally, but not semantically, e g in a simple example sentence such as "John promised Mary to leave", there was a missing subject (required by the verb *leave*) in the f-structure We now use the trace and index information provided by the Treebank to capture these reentrancies, and encode them in the f-structure In addition to the functional uncertainty equations already encoded in our algorithm (specifying paths in the f-structure between 'moved' elements and where they originally belonged, cf Section 4 5), subcategorisation information is necessary for the treatment of long distance dependencies in LFG Subcategorisation requirements are treated in LFG lexically in terms of semantic forms (subcategorisation lists) and completeness and coherence conditions on f-structure representations Developing these subcategorisation frames is another avenue for further work

These c- and f-structure pairs generated by our automatic f-structure annotation algorithm can be used for LFG-DOP and LFG-DOT (cf Section 1 3) We can automatically extract large-scale, wide-coverage LFG grammars from the Penn-II Treebank, which is a very useful linguistic resource The automatic f-structure annotation algorithm we have constructed yields very promising results With further linguistic analysis, a more complete representation of the trees in the Penn-II Treebank can be generated, enhancing the quality of the resources produced

Despite these issues for further work, this thesis has shown the design and

evaluation of an automatic f-structure annotation algorithm which scales to the Penn-II Treebank

Appendix A

Lexical Macros

POS tag	Lexical Info
CC	pred=lemma
CD	pred=lemma
DT	pred=lemma
EX	form=lemma
FW	pred=lemma
IN	pred=lemma
JJ	pred=lemma
JJR	pred=lemma, adegree=comparative
JJS	pred=lemma, adegree=superlative
LS	
MD	pred=lemma, modal='+'
NN	pred=lemma, num=sg, pers=3
NNS	pred=lemma, num=pl, pers=3
NNP	pred=lemma, num=sg, pers=3
NNPS	pred=lemma, num=pl, pers=3
PDT	pred=lemma
POS	
PRP	pred=lemma
PRP$	pred=pro, wh=' ', case=gen
RB	pred=lemma
RBR	pred=lemma, adegree=comparative
RBS	pred=lemma, adegree=superlative
RP	pred=lemma
SYM	pred=lemma
TO	to='+', inf='+'
UH	pred=lemma
VB	pred=lemma
VBD	pred=lemma, tense=past
VBG	pred=lemma, participle=pres
VBN	pred=lemma, tense=past
VBP	pred=lemma, tense=pres
VBZ	pred=lemma, tense=pres
WDT	pred=lemma
WP	pred=pro, wh='+'
WP$	pred=pro, wh='+'
WRB	pred=lemma
#	pred=lemma
$	pred=lemma
	pred=lemma

Table A 1 Lexical Macros

Appendix B

Annotation Matrices ADJP–UCP

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
#	$\downarrow \in \uparrow$ ADJ	#	$\downarrow \in \uparrow$ ADJ
$	$\downarrow \in \uparrow$ ADJ	ADJP	$\downarrow \in \uparrow$ ADJ
ADJP	$\downarrow \in \uparrow$ ADJ	ADVP	$\downarrow \in \uparrow$ ADJ
ADVP	$\downarrow \in \uparrow$ ADJ	CD	$\downarrow \in \uparrow$ ADJ
CD	$\downarrow \in \uparrow$ ADJ	FW	$\downarrow \in \uparrow$ ADJ
DT	$\uparrow$ SPEC DET= $\downarrow$	IN	$\downarrow \in \uparrow$ ADJ
FW	$\downarrow \in \uparrow$ ADJ	JJ	$\downarrow \in \uparrow$ ADJ
IN	$\downarrow \in \uparrow$ ADJ	JJR	$\downarrow \in \uparrow$ ADJ
JJ	$\downarrow \in \uparrow$ ADJ	JJS	$\downarrow \in \uparrow$ ADJ
JJR	$\downarrow \in \uparrow$ ADJ	NN	$\downarrow \in \uparrow$ ADJ
JJS	$\downarrow \in \uparrow$ ADJ	NNP	$\downarrow \in \uparrow$ ADJ
NN	$\downarrow \in \uparrow$ ADJ	NNS	$\downarrow \in \uparrow$ ADJ
NNP	$\downarrow \in \uparrow$ ADJ	NP	$\downarrow \in \uparrow$ ADJ
NNS	$\downarrow \in \uparrow$ ADJ	NP-TMP	$\downarrow \in \uparrow$ ADJ
NP	$\downarrow \in \uparrow$ ADJ	PDT	$\uparrow$ SPEC DET= $\downarrow$
PDT	$\uparrow$ SPEC DET= $\downarrow$	PP	$\downarrow \in \uparrow$ ADJ
PP	$\downarrow \in \uparrow$ ADJ	PRN	$\downarrow \in \uparrow$ ADJ
PRN	$\downarrow \in \uparrow$ ADJ	PRT	$\uparrow$ PART= $\downarrow$
PRT	$\uparrow$ PART= $\downarrow$	QP	$\downarrow \in \uparrow$ ADJ
RB	$\downarrow \in \uparrow$ ADJ	RB	$\downarrow \in \uparrow$ ADJ
RBR	$\downarrow \in \uparrow$ ADJ	RBR	$\downarrow \in \uparrow$ ADJ
RBS	$\downarrow \in \uparrow$ ADJ	RBS	$\downarrow \in \uparrow$ ADJ
RP	$\downarrow \in \uparrow$ ADJ	RP	$\downarrow \in \uparrow$ ADJ
S	$\uparrow$ XCOMP= $\downarrow$, $\uparrow$ SUBJ= $\downarrow$SUBJ	S	$\uparrow$ XCOMP= $\downarrow$
TO	$\uparrow$ OBL= $\downarrow$	S-NOM	$\uparrow$ COMP= $\downarrow$
VB	$\downarrow \in \uparrow$ ADJ	SBAR	$\uparrow$ COMP= $\downarrow$
VBD	$\downarrow \in \uparrow$ ADJ	TO	$\uparrow$ OBL= $\downarrow$
VBG	$\downarrow \in \uparrow$ ADJ	VB	$\downarrow \in \uparrow$ ADJ
VBN	$\downarrow \in \uparrow$ ADJ	VBD	$\downarrow \in \uparrow$ ADJ
VBP	$\downarrow \in \uparrow$ ADJ	VBG	$\downarrow \in \uparrow$ ADJ
VP	$\downarrow \in \uparrow$ ADJ	VBN	$\downarrow \in \uparrow$ ADJ
WRB	$\downarrow \in \uparrow$ ADJ	VBP	$\downarrow \in \uparrow$ ADJ
		VP	$\downarrow \in \uparrow$ ADJ
		WRB	$\downarrow \in \uparrow$ ADJ

Table B 1 Mother category ADJP

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADJP	↓∈↑ADJ	ADJP	↑OBJ=↓
ADVP	↓∈↑ADJ	ADVP	↓∈↑ADJ
CC	↓∈↑ADJ	CC	↓∈↑ADJ
DT	↑SPEC DET=↓	DT	↑OBL=↓
EX	↓∈↑ADJ	EX	↓∈↑ADJ
FW	↓∈↑ADJ	FW	↓∈↑ADJ
IN	↑OBL=↓	IN	↑OBL=↓
JJ	↓∈↑ADJ	JJ	↑OBL=↓
JJR	↓∈↑ADJ	JJR	↓∈↑ADJ
JJS	↓∈↑ADJ	JJS	↓∈↑ADJ
NN	↓∈↑ADJ	NN	↓∈↑ADJ
NNP	↓∈↑ADJ	NNP	↓∈↑ADJ
NNS	↓∈↑ADJ	NNS	↓∈↑ADJ
NP	↑OBJ=↓	NP	↓∈↑ADJ
PDT	↑SPEC DET=↓	PDT	↑SPEC DET=↓
PP	↓∈↑ADJ	PP	↓∈↑ADJ
PRN	↓∈↑ADJ	PRN	↓∈↑ADJ
QP	↓∈↑ADJ	QP	↓∈↑ADJ
RB	↓∈↑ADJ	RB	↓∈↑ADJ
RBR	↓∈↑ADJ	RBR	↓∈↑ADJ
RBS	↓∈↑ADJ	RBS	↓∈↑ADJ
RP	↓∈↑ADJ	RP	↓∈↑ADJ
S	↓∈↑ADJ	S	↑XCOMP=↓, ↑SUBJ=↓SUBJ
TO	↑OBL=↓	SBAR	↓∈↑ADJ
VB	↓∈↑ADJ	TO	↑OBL=↓
VBD	↓∈↑ADJ	VB	↓∈↑ADJ
VBG	↓∈↑ADJ	VBD	↓∈↑ADJ
VBN	↓∈↑ADJ	VBG	↓∈↑ADJ
VBZ	↓∈↑ADJ	VBN	↓∈↑ADJ
VP	↓∈↑ADJ	VBZ	↓∈↑ADJ
		VP	↓∈↑ADJ

Table B 2 Mother category ADVP

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
JJ	↓∈↑ADJ	IN	↑OBL=↓
IN	↓∈↑ADJ	JJ	↓∈↑ADJ
NN	↓∈↑ADJ	NN	↓∈↑ADJ
RB	↓∈↑ADJ	RB	↓∈↑ADJ
TO	↓∈↑ADJ	TO	↓∈↑ADJ
VB	↓∈↑ADJ	VB	↓∈↑ADJ

Table B 3 Mother category CONJP

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
DT	$\downarrow \in \uparrow$ ADJ	DT	$\downarrow \in \uparrow$ ADJ
INTJ	$\downarrow \in \uparrow$ ADJ	INTJ	$\downarrow \in \uparrow$ ADJ
JJ	$\downarrow \in \uparrow$ ADJ	JJ	$\downarrow \in \uparrow$ ADJ
NN	$\downarrow \in \uparrow$ ADJ	NN	$\downarrow \in \uparrow$ ADJ
NNP	$\downarrow \in \uparrow$ ADJ	NNP	$\downarrow \in \uparrow$ ADJ
NP	$\downarrow \in \uparrow$ ADJ	NP	$\downarrow \in \uparrow$ ADJ
PP	$\downarrow \in \uparrow$ ADJ	PP	$\downarrow \in \uparrow$ ADJ
PRP$	$\downarrow \in \uparrow$ ADJ	PRP$	$\downarrow \in \uparrow$ ADJ
RB	$\downarrow \in \uparrow$ ADJ	RB	$\downarrow \in \uparrow$ ADJ
UH	$\downarrow \in \uparrow$ ADJ	UH	$\downarrow \in \uparrow$ ADJ
VB	$\downarrow \in \uparrow$ ADJ	VB	$\downarrow \in \uparrow$ ADJ

Table B 4 Mother category INTJ

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADVP	$\downarrow \in \uparrow$ ADJ	ADVP	$\downarrow \in \uparrow$ ADJ
CD	$\downarrow \in \uparrow$ ADJ	CD	$\downarrow \in \uparrow$ ADJ
DT	$\uparrow$SPEC DET=$\downarrow$	DT	$\uparrow$SPEC DET=$\downarrow$
JJ	$\downarrow \in \uparrow$ ADJ	JJ	$\downarrow \in \uparrow$ ADJ
NNP	$\downarrow \in \uparrow$ ADJ	NNP	$\downarrow \in \uparrow$ ADJ
NNPS	$\downarrow \in \uparrow$ ADJ	NNPS	$\downarrow \in \uparrow$ ADJ
NP	$\downarrow \in \uparrow$ ADJ	NP	$\downarrow \in \uparrow$ ADJ
PRN	$\downarrow \in \uparrow$ ADJ	PP	$\downarrow \in \uparrow$ ADJ
PRP	$\downarrow \in \uparrow$ ADJ	PRN	$\downarrow \in \uparrow$ ADJ
SBAR	$\uparrow$XCOMP=$\downarrow$, $\uparrow$SUBJ=$\downarrow$SUBJ	PRP	$\downarrow \in \uparrow$ ADJ
TO	$\downarrow \in \uparrow$ ADJ	SBAR	$\uparrow$XCOMP=$\downarrow$, $\uparrow$SUBJ=$\downarrow$SUBJ
VBN	$\downarrow \in \uparrow$ ADJ	TO	$\downarrow \in \uparrow$ ADJ
		VBN	$\downarrow \in \uparrow$ ADJ

Table B 5 Mother category NAC

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
#	↓ ∈ ↑ ADJ	#	↓ ∈ ↑ ADJ
ADJP	↓ ∈ ↑ ADJ	ADJP	↓ ∈ ↑ ADJ
ADVP	↓ ∈ ↑ ADJ	ADVP	↓ ∈ ↑ ADJ
CD	↓ ∈ ↑ ADJ	CD	↓ ∈ ↑ ADJ
DT	↑SPEC DET=↓	DT	↑SPEC DET=↓
FW	↓ ∈ ↑ ADJ	FW	↓ ∈ ↑ ADJ
IN	↓ ∈ ↑ ADJ	IN	↓ ∈ ↑ ADJ
INTJ	↓ ∈ ↑ ADJ	INTJ	↓ ∈ ↑ ADJ
JJ	↓ ∈ ↑ ADJ	JJ	↓ ∈ ↑ ADJ
JJR	↓ ∈ ↑ ADJ	JJR	↓ ∈ ↑ ADJ
JJS	↓ ∈ ↑ ADJ	JJS	↓ ∈ ↑ ADJ
LST	↓ ∈ ↑ ADJ	LST	↓ ∈ ↑ ADJ
MD	↓ ∈ ↑ ADJ	MD	↓ ∈ ↑ ADJ
NAC	↓ ∈ ↑ ADJ	NAC	↓ ∈ ↑ ADJ
NN	↓ ∈ ↑ ADJ	NN	↓ ∈ ↑ ADJ
NNP	↓ ∈ ↑ ADJ	NNP	↓ ∈ ↑ ADJ
NNPS	↓ ∈ ↑ ADJ	NNPS	↓ ∈ ↑ ADJ
NNS	↓ ∈ ↑ ADJ	NNS	↓ ∈ ↑ ADJ
NP	↓ ∈ ↑ ADJ	NP	↓ ∈ ↑ APP
NX	↓ ∈ ↑ ADJ	NX	↓ ∈ ↑ ADJ
PDT	↑SPEC DET=↓	PDT	↑SPEC DET=↓
PP	↓ ∈ ↑ ADJ	PP	↓ ∈ ↑ ADJ
PRN	↓ ∈ ↑ ADJ	PP-TMP	↓ ∈ ↑ ADJ
PRP	↓ ∈ ↑ ADJ	PRN	↓ ∈ ↑ ADJ
PRP$	↑POS=↓	PRP	↓ ∈ ↑ ADJ
PRT	↓ ∈ ↑ ADJ	PRP$	↓ ∈ ↑ ADJ
QP	↑SPEC QUANT=↓	PRT	↓ ∈ ↑ ADJ
RB	↓ ∈ ↑ ADJ	QP	↑SPEC QUANT=↓
RBR	↓ ∈ ↑ ADJ	RB	↓ ∈ ↑ ADJ
RBS	↓ ∈ ↑ ADJ	RBR	↓ ∈ ↑ ADJ
RP	↓ ∈ ↑ ADJ	RBS	↓ ∈ ↑ ADJ
RRC	↑RELMOD=↓	RP	↓ ∈ ↑ ADJ
S	↓ ∈ ↑ ADJ	RRC	↑RELMOD=↓
SBAR	↑COMP=↓	S	↑XCOMP=↓, ↑SUBJ=↓SUBJ
SBARQ	↑COMP=↓	SBAR	↑RELMOD=↓
SYM	↓ ∈ ↑ ADJ	SBARQ	↑COMP=↓
TO	↓ ∈ ↑ ADJ	SINV	↑COMP=↓
UCP	↓ ∈ ↑ ADJ	SQ	↑RELMOD=↓
UH	↓ ∈ ↑ ADJ	SYM	↓ ∈ ↑ ADJ

Table B 6 Mother category NP

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
VB	↓ ∈ ↑ ADJ	TO	↓ ∈ ↑ ADJ
VBD	↓ ∈ ↑ ADJ	UCP	↓ ∈ ↑ ADJ
VBG	↓ ∈ ↑ ADJ	UH	↓ ∈ ↑ ADJ
VBN	↓ ∈ ↑ ADJ	VB	↓ ∈ ↑ ADJ
VBP	↓ ∈ ↑ ADJ	VBD	↓ ∈ ↑ ADJ
VBZ	↓ ∈ ↑ ADJ	VBG	↓ ∈ ↑ ADJ
VP	↓ ∈ ↑ ADJ	VBN	↓ ∈ ↑ ADJ
WDT	↑SPEC DET=↓	VBP	↓ ∈ ↑ ADJ
WHNP	↓ ∈ ↑ ADJ	VBZ	↓ ∈ ↑ ADJ
WHPP	↑RELMOD=↓	VP	↑RELMOD=↓
WP	↑SPEC DET=↓	WDT	↑SPEC DET=↓
WP$	↓ ∈ ↑ ADJ	WHNP	↓ ∈ ↑ ADJ
WRB	↓ ∈ ↑ ADJ	WHPP	↑RELMOD=↓
		WP	↑SPEC DET=↓
		WP$	↓ ∈ ↑ ADJ
		WRB	↓ ∈ ↑ ADJ

Table B 7 Mother category NP (continued)

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADJP	↓ ∈ ↑ ADJ	CD	↓ ∈ ↑ ADJ
CD	↓ ∈ ↑ ADJ	DT	↑SPEC DET=↓
DT	↑SPEC DET=↓	FW	↓ ∈ ↑ ADJ
FW	↓ ∈ ↑ ADJ	IN	↓ ∈ ↑ ADJ
IN	↓ ∈ ↑ ADJ	JJ	↓ ∈ ↑ ADJ
JJ	↓ ∈ ↑ ADJ	JJR	↓ ∈ ↑ ADJ
JJR	↓ ∈ ↑ ADJ	JJS	↓ ∈ ↑ ADJ
JJS	↓ ∈ ↑ ADJ	NAC	↓ ∈ ↑ ADJ
NAC	↓ ∈ ↑ ADJ	NN	↓ ∈ ↑ ADJ
NN	↓ ∈ ↑ ADJ	NNPS	↓ ∈ ↑ ADJ
NNP	↓ ∈ ↑ ADJ	NNS	↓ ∈ ↑ ADJ
NNPS	↓ ∈ ↑ ADJ	NP	↓ ∈ ↑ ADJ
NNS	↓ ∈ ↑ ADJ	NX	↓ ∈ ↑ ADJ
NP	↓ ∈ ↑ ADJ	PP	↓ ∈ ↑ ADJ
NX	↓ ∈ ↑ ADJ	PRN	↓ ∈ ↑ ADJ
PP	↓ ∈ ↑ ADJ	PRP$	↓ ∈ ↑ ADJ
PRN	↓ ∈ ↑ ADJ	QP	↓ ∈ ↑ ADJ
PRP$	↓ ∈ ↑ ADJ	RB	↓ ∈ ↑ ADJ
QP	↓ ∈ ↑ ADJ	RRC	↑RELMOD=↓
RB	↓ ∈ ↑ ADJ	SBAR	↑COMP=↓
RRC	↑RELMOD=↓	SBARQ	↑COMP=↓
SBAR	↑COMP=↓	UCP	↓ ∈ ↑ ADJ
SBARQ	↑COMP=↓	VBG	↓ ∈ ↑ ADJ
UCP	↓ ∈ ↑ ADJ	VBN	↓ ∈ ↑ ADJ
VBG	↓ ∈ ↑ ADJ	WHPP	↑RELMOD=↓
VBN	↓ ∈ ↑ ADJ	WP	↑SPEC DET=↓
WHPP	↑RELMOD=↓		
WP	↑SPEC DET=↓		

Table B 8 Mother category NX

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADVP	$\downarrow \in \uparrow$ ADJ	INTJ	$\downarrow \in \uparrow$ ADJ
INTJ	$\downarrow \in \uparrow$ ADJ	PRN	$\downarrow \in \uparrow$ ADJ
		SINV	$\downarrow \in \uparrow$ ADJ

Table B 9 Mother category PRN

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
#	$\downarrow \in \uparrow$ ADJ	#	$\downarrow \in \uparrow$ ADJ
ADJP	$\downarrow \in \uparrow$ ADJ	$	$\downarrow \in \uparrow$ ADJ
ADVP	$\downarrow \in \uparrow$ ADJ	ADJP	$\downarrow \in \uparrow$ ADJ
CD	$\downarrow \in \uparrow$ ADJ	ADVP	$\downarrow \in \uparrow$ ADJ
DT	$\downarrow \in \uparrow$ ADJ	CD	$\downarrow \in \uparrow$ ADJ
IN	$\downarrow \in \uparrow$ ADJ	IN	$\uparrow$OBJ=$\downarrow$
JJ	$\downarrow \in \uparrow$ ADJ	JJ	$\downarrow \in \uparrow$ ADJ
JJR	$\downarrow \in \uparrow$ ADJ	JJR	$\downarrow \in \uparrow$ ADJ
JJS	$\downarrow \in \uparrow$ ADJ	JJS	$\downarrow \in \uparrow$ ADJ
NN	$\downarrow \in \uparrow$ ADJ	NN	$\downarrow \in \uparrow$ ADJ
NNS	$\downarrow \in \uparrow$ ADJ	NNP	$\downarrow \in \uparrow$ ADJ
NP	$\downarrow \in \uparrow$ ADJ	NNS	$\downarrow \in \uparrow$ ADJ
PDT	$\uparrow$SPEC DET=$\downarrow$	NP	$\downarrow \in \uparrow$ ADJ
PRN	$\downarrow \in \uparrow$ ADJ	PRN	$\downarrow \in \uparrow$ ADJ
RB	$\downarrow \in \uparrow$ ADJ	RBR	$\downarrow \in \uparrow$ ADJ
RBR	$\downarrow \in \uparrow$ ADJ	RBS	$\downarrow \in \uparrow$ ADJ
RBS	$\downarrow \in \uparrow$ ADJ	RP	$\downarrow \in \uparrow$ ADJ
RP	$\downarrow \in \uparrow$ ADJ	SYM	$\downarrow \in \uparrow$ ADJ
SYM	$\downarrow \in \uparrow$ ADJ	TO	$\uparrow$OBL=$\downarrow$
TO	$\uparrow$OBL=$\downarrow$	VB	$\downarrow \in \uparrow$ ADJ
VBN	$\downarrow \in \uparrow$ ADJ	VBN	$\downarrow \in \uparrow$ ADJ

Table B 10 Mother category QP

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADJP	$\downarrow \in \uparrow$ ADJ	ADVP	$\downarrow \in \uparrow$ ADJ
ADVP	$\downarrow \in \uparrow$ ADJ	PP	$\downarrow \in \uparrow$ ADJ
ADVP-TMP	$\downarrow \in \uparrow$ ADJ	PP-LOC	$\downarrow \in \uparrow$ ADJ
NP-TMP	$\downarrow \in \uparrow$ ADJ	PP-TMP	$\downarrow \in \uparrow$ ADJ
PP	$\downarrow \in \uparrow$ ADJ		
PP-LOC	$\downarrow \in \uparrow$ ADJ		
PP-TMP	$\downarrow \in \uparrow$ ADJ		

Table B 11 Mother category RRC

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADJP	↓∈↑ADJ	ADJP	↓∈↑ADJ
ADVP	↓∈↑ADJ	ADVP	↓∈↑ADJ
ADVP-TMP	↓∈↑ADJ	DT	↓∈↑ADJ
DT	↓∈↑ADJ	FW	↓∈↑ADJ
FW	↓∈↑ADJ	IN	↓∈↑ADJ
IN	↓∈↑ADJ	INTJ	↓∈↑ADJ
INTJ	↓∈↑ADJ	LST	↓∈↑ADJ
LST	↓∈↑ADJ	MD	↓∈↑ADJ
MD	↓∈↑ADJ	NNP	↓∈↑ADJ
NP	↑SUBJ=↓	NP	↓∈↑ADJ
NP-NOM	↑SUBJ=↓	PP	↓∈↑ADJ
NP-TMP	↓∈↑ADJ	PRN	↓∈↑ADJ
PP	↓∈↑ADJ	RB	↓∈↑ADJ
PP-TMP	↓∈↑ADJ	RBR	↓∈↑ADJ
PRN	↓∈↑ADJ	S	↑XCOMP=↓
RB	↓∈↑ADJ	SBAR	↑XCOMP=↓
RBR	↓∈↑ADJ	SBARQ	↓∈↑ADJ
S	↓∈↑ADJ	SINV	↑XCOMP=↓
ADV	↓∈↑ADJ	SQ	↑COMP=↓
SBAR	↑XCOMP=↓	UCP	↓∈↑ADJ
SBAR-ADV	↓∈↑ADJ	VBD	↓∈↑ADJ
SBAR-TMP	↓∈↑ADJ	VBP	↓∈↑ADJ
SINV	↑XCOMP=↓	VBZ	↓∈↑ADJ
SQ	↑COMP=↓	VP	↑XCOMP=↓, ↑SUBJ=↓SUBJ
S-TPC	↑COMP=↓	WHNP	↑SUBJ=↓
UCP	↓∈↑ADJ		
VBD	↓∈↑ADJ		
VBP	↓∈↑ADJ		
VBZ	↓∈↑ADJ		
VP	↓∈↑ADJ		
WHNP	↑SUBJ=↓		

Table B 12 Mother category S

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADJP	↓ ∈ ↑ ADJ	ADJP	↓ ∈ ↑ ADJ
ADVP	↓ ∈ ↑ ADJ	ADVP	↓ ∈ ↑ ADJ
DT	↓ ∈ ↑ ADJ	IN	↓ ∈ ↑ ADJ
IN	↑ = ↓	LST	↓ ∈ ↑ ADJ
LST	↓ ∈ ↑ ADJ	MD	↓ ∈ ↑ ADJ
MD	↓ ∈ ↑ ADJ	NN	↑OBJ=↓
NN	↓ ∈ ↑ ADJ	NNP	↓ ∈ ↑ ADJ
NNP	↓ ∈ ↑ ADJ	NP	↑OBJ=↓
NP	↓ ∈ ↑ ADJ	PP	↓ ∈ ↑ ADJ
PP	↓ ∈ ↑ ADJ	PRN	↓ ∈ ↑ ADJ
PRN	↓ ∈ ↑ ADJ	QP	↓ ∈ ↑ ADJ
QP	↓ ∈ ↑ ADJ	RB	↓ ∈ ↑ ADJ
RB	↓ ∈ ↑ ADJ	S	↑COMP=↓
S	↓ ∈ ↑ ADJ	SBAR	↓ ∈ ↑ ADJ
SBAR	↓ ∈ ↑ ADJ	SINV	↑COMP=↓
SINV	↑COMP=↓	SQ	↓ ∈ ↑ ADJ
SQ	↓ ∈ ↑ ADJ	TO	↓ ∈ ↑ ADJ
TO	↓ ∈ ↑ ADJ	UCP	↓ ∈ ↑ ADJ
UCP	↓ ∈ ↑ ADJ	VB	↓ ∈ ↑ ADJ
VB	↓ ∈ ↑ ADJ	VBD	↓ ∈ ↑ ADJ
VBD	↓ ∈ ↑ ADJ	VBN	↓ ∈ ↑ ADJ
VBN	↓ ∈ ↑ ADJ	VP	↓ ∈ ↑ ADJ
VP	↓ ∈ ↑ ADJ	WDT	↓ ∈ ↑ ADJ
WDT	↑TOPICREL=↓	WHADVP	↓ ∈ ↑ ADJ
WHADJP	↑TOPICREL=↓	WP	↓ ∈ ↑ ADJ
WHADVP	↑TOPICREL=↓	WP$	↓ ∈ ↑ ADJ
WHNP	↑TOPICREL=↓	WRB	↓ ∈ ↑ ADJ
WHPP	↑TOPICREL=↓		
WP	↑TOPICREL=↓		
WP$	↑TOPICREL=↓		
WRB	↑TOPICREL=↓		

Table B 13 Mother category SBAR

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADVP	$\downarrow \in \uparrow$ ADJ	ADVP	$\downarrow \in \uparrow$ ADJ
INTJ	$\downarrow \in \uparrow$ ADJ	INTJ	$\downarrow \in \uparrow$ ADJ
MD	$\downarrow \in \uparrow$ ADJ	MD	$\downarrow \in \uparrow$ ADJ
PP	$\downarrow \in \uparrow$ ADJ	PP	$\downarrow \in \uparrow$ ADJ
PRN	$\downarrow \in \uparrow$ ADJ	PRN	$\downarrow \in \uparrow$ ADJ
RB	$\downarrow \in \uparrow$ ADJ	S	$\downarrow \in \uparrow$ ADJ
S	$\downarrow \in \uparrow$ ADJ	SBAR	$\downarrow \in \uparrow$ ADJ
SBAR	$\downarrow \in \uparrow$ ADJ	SQ	$\downarrow \in \uparrow$ ADJ
SBAR-ADV	$\downarrow \in \uparrow$ ADJ	VBZ	$\downarrow \in \uparrow$ ADJ
SQ	$\downarrow \in \uparrow$ ADJ	WHADJP	$\downarrow \in \uparrow$ ADJ
VBZ	$\downarrow \in \uparrow$ ADJ	WHADVP	$\downarrow \in \uparrow$ ADJ
WHADJP	$\uparrow$FOCUS=$\downarrow$	WHNP	$\downarrow \in \uparrow$ ADJ
WHADVP	$\uparrow$FOCUS=$\downarrow$	WHPP	$\downarrow \in \uparrow$ ADJ
WHNP	$\uparrow$FOCUS=$\downarrow$		
WHPP	$\uparrow$FOCUS=$\downarrow$		

Table B 14 Mother category SBARQ

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADJP-PRD	$\downarrow \in \uparrow$ ADJ	INTJ	$\downarrow \in \uparrow$ ADJ
ADVP	$\downarrow \in \uparrow$ ADJ	NP SBJ	$\uparrow$SUBJ=$\downarrow$
ADVP-LOC	$\downarrow \in \uparrow$ ADJ	NP	$\uparrow$SUBJ=$\downarrow$
INTJ	$\downarrow \in \uparrow$ ADJ	PP	$\downarrow \in \uparrow$ ADJ
NP-SBJ	$\uparrow$SUBJ=$\downarrow$	PRN	$\downarrow \in \uparrow$ ADJ
PP	$\downarrow \in \uparrow$ ADJ	RB	$\downarrow \in \uparrow$ ADJ
PRN	$\downarrow \in \uparrow$ ADJ	S	$\uparrow$COMP=$\downarrow$
RB	$\downarrow \in \uparrow$ ADJ	S-ADV	$\downarrow \in \uparrow$ ADJ
S	$\uparrow$COMP=$\downarrow$	SBAR	$\downarrow \in \uparrow$ ADJ
SBAR	$\downarrow \in \uparrow$ ADJ	SBARQ	$\downarrow \in \uparrow$ ADJ
SBARQ	$\downarrow \in \uparrow$ ADJ	SINV	$\downarrow \in \uparrow$ ADJ
SINV	$\downarrow \in \uparrow$ ADJ	VBD	$\downarrow \in \uparrow$ ADJ
S-TPC	$\uparrow$COMP=$\downarrow$	VP	$\uparrow$XCOMP=$\downarrow$, $\uparrow$SUBJ=$\downarrow$SUBJ
VBD	$\downarrow \in \uparrow$ ADJ		
VP	$\downarrow \in \uparrow$ ADJ		

Table B 15 Mother category SINV

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADVP	↓ ∈ ↑ ADJ	ADJP-PRD	↑XCOMP=↓, ↑SUBJ=↓SUBJ
INTJ	↓ ∈ ↑ ADJ	ADVP	↓ ∈ ↑ ADJ
NNP	↓ ∈ ↑ ADJ	ADVP-MNR	↓ ∈ ↑ ADJ
NNS	↓ ∈ ↑ ADJ	ADVP-TMP	↓ ∈ ↑ ADJ
NP-SBJ	↑SUBJ=↓	CC	↓ ∈ ↑ ADJ
PP	↓ ∈ ↑ ADJ	INTJ	↓ ∈ ↑ ADJ
PRN	↓ ∈ ↑ ADJ	NNP	↓ ∈ ↑ ADJ
RB	↓ ∈ ↑ ADJ	NNS	↓ ∈ ↑ ADJ
S	↑XCOMP=↓, ↑SUBJ=↓SUBJ	NP	↑OBJ=↓
SBAR-ADV	↓ ∈ ↑ ADJ	NP-PRD	↑XCOMP=↓, ↑SUBJ=↓SUBJ
SBARQ	↓ ∈ ↑ ADJ	NP-SBJ	↑SUBJ=↓
SQ	↓ ∈ ↑ ADJ	PP	↓ ∈ ↑ ADJ
WHNP	↓ ∈ ↑ ADJ	PP-LOC	↓ ∈ ↑ ADJ
		PP-PRD	↑OBJ=↓
		PRN	↓ ∈ ↑ ADJ
		RB	↓ ∈ ↑ ADJ
		S	↑XCOMP=↓, ↑SUBJ=↓SUBJ
		SBAR	↑COMP=↓
		SBAR ADV	↓ ∈ ↑ ADJ
		SBAR-PRP	↑XCOMP=↓, ↑SUBJ=↓SUBJ
		SBARQ	↓ ∈ ↑ ADJ
		SQ	↓ ∈ ↑ ADJ
		VP	↑XCOMP=↓, ↑SUBJ=↓SUBJ
		WHNP	↓ ∈ ↑ ADJ

Table B 16 Mother category SQ

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADJP	↓ ∈ ↑ ADJ	ADJP	↓ ∈ ↑ ADJ
ADVP	↓ ∈ ↑ ADJ	ADVP	↓ ∈ ↑ ADJ
CONJP	↓ ∈ ↑ ADJ	CD	↓ ∈ ↑ ADJ
DT	↓ ∈ ↑ ADJ	NP	↓ ∈ ↑ ADJ
IN	↓ ∈ ↑ ADJ	PP	↓ ∈ ↑ ADJ
JJ	↓ ∈ ↑ ADJ	PRN	↓ ∈ ↑ ADJ
NP	↓ ∈ ↑ ADJ	RB	↓ ∈ ↑ ADJ
PP	↓ ∈ ↑ ADJ	SBAR	↓ ∈ ↑ ADJ
RB	↓ ∈ ↑ ADJ	SQ	↓ ∈ ↑ ADJ
SBAR	↓ ∈ ↑ ADJ		

Table B 17 Mother category UCP

Appendix C

Annotation Matrices VP–WHPP

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADJP	↓ ∈ ↑ ADJ	ADJP	↑XCOMP=↓, ↑SUBJ=↓SUBJ
ADVP	↓ ∈ ↑ ADJ	ADJP-PRD	↑XCOMP=↓, ↑SUBJ=↓SUBJ
DT	↑SPEC DET=↓	ADVP	↓ ∈ ↑ ADJ
IN	↓ ∈ ↑ ADJ	ADVP-PRD	↑XCOMP=↓, ↑SUBJ=↓SUBJ
INTJ	↓ ∈ ↑ ADJ	ADVP-TMP	↓ ∈ ↑ ADJ
JJ	↓ ∈ ↑ ADJ	DT	↓ ∈ ↑ ADJ
LST	↓ ∈ ↑ ADJ	IN	↓ ∈ ↑ ADJ
NN	↑OBJ=↓	INTJ	↓ ∈ ↑ ADJ
NNP	↑OBJ=↓	JJ	↑XCOMP=↓, ↑SUBJ=↓SUBJ
NNS	↓ ∈ ↑ ADJ	LST	↓ ∈ ↑ ADJ
NP	↑OBJ=↓	NNS	↓ ∈ ↑ ADJ
NP-ADV	↓ ∈ ↑ ADJ	NP	↑OBJ=↓
PDT	↓ ∈ ↑ ADJ	NP-EXT	↑OBJ=↓
PP	↓ ∈ ↑ ADJ	NP PRD	↑OBJ=↓
PRN	↓ ∈ ↑ ADJ	NP ADV	↑OBJ=↓
PRT	↓ ∈ ↑ ADJ	NP-CLR	↑OBJ=↓
PRT—ADVP	↓ ∈ ↑ ADJ	NP-TMP	↓ ∈ ↑ ADJ
QP	↓ ∈ ↑ ADJ	PDT	↓ ∈ ↑ ADJ
RB	↓ ∈ ↑ ADJ	PP	↓ ∈ ↑ ADJ
RBR	↓ ∈ ↑ ADJ	PP-DIR	↑OBL=↓
RBS	↓ ∈ ↑ ADJ	PP-DTV	↓ ∈ ↑ ADJ
S	↓ ∈ ↑ ADJ	PP-LOC	↓ ∈ ↑ ADJ
SBAR	↓ ∈ ↑ ADJ	PP-MNR	↓ ∈ ↑ ADJ
SBARQ	↑COMP=↓	PP-PRD	↑XCOMP=↓, ↑SUBJ=↓SUBJ
SINV	↑XCOMP=↓, ↑SUBJ=↓SUBJ	PP-PRP	↓ ∈ ↑ ADJ
SQ	↑COMP=↓	PP-TMP	↓ ∈ ↑ ADJ
SYM	↓ ∈ ↑ ADJ	PRN	↓ ∈ ↑ ADJ
TO	↑=↓	PRT	↑PART=↓
VBD	↓ ∈ ↑ ADJ	PRT—ADVP	↓ ∈ ↑ ADJ
VBG	↓ ∈ ↑ ADJ	QP	↓ ∈ ↑ ADJ
VBP	↓ ∈ ↑ ADJ	RB	↓ ∈ ↑ ADJ
VBZ	↓ ∈ ↑ ADJ	RBR	↓ ∈ ↑ ADJ
VP	↓ ∈ ↑ ADJ	RBS	↓ ∈ ↑ ADJ

Table C 1 Mother category VP

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
		S	↑XCOMP=↓
		S ADV	↓ ∈ ↑ ADJ
		SBAR	↑COMP=↓
		SBAR-ADV	↓ ∈ ↑ ADJ
		SBAR-PRD	↑COMP=↓
		SBAR-NOM	↑COMP=↓
		SBAR-NOM-PRD	↑COMP=↓
		SBARQ	↑COMP=↓
		SBAR-TMP	↓ ∈ ↑ ADJ
		S-CLR	↑XCOMP=↓
		SINV	↑XCOMP=↓, ↑SUBJ=↓SUBJ
		S-PRD	↑XCOMP=↓
		S-PRP	↑XCOMP=↓
		SQ	↑COMP=↓
		SYM	↓ ∈ ↑ ADJ
		TO	↓ ∈ ↑ ADJ
		UCP	↓ ∈ ↑ ADJ
		UCP	↑OBJ=↓
		VB	↑XCOMP=↓, ↑SUBJ=↓SUBJ
		VBD	↓ ∈ ↑ ADJ
		VBG	↓ ∈ ↑ ADJ
		VBN	↑XCOMP=↓, ↑SUBJ=↓SUBJ
		VBP	↓ ∈ ↑ ADJ
		VBZ	↓ ∈ ↑ ADJ
		VP	↑XCOMP=↓, ↑SUBJ=↓SUBJ

Table C 2 Mother category VP (continued)

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADJP	↓ ∈ ↑ ADJ	ADJP	↓ ∈ ↑ ADJ
RB	↓ ∈ ↑ ADJ	RB	↓ ∈ ↑ ADJ
WHADVP	↓ ∈ ↑ ADJ		
WRB	↓ ∈ ↑ ADJ		

Table C 3 Mother category WHADJP

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADVP	↓ ∈ ↑ ADJ	ADJP	↓ ∈ ↑ ADJ
WRB	↓ ∈ ↑ ADJ	ADVP	↓ ∈ ↑ ADJ
		JJ	↓ ∈ ↑ ADJ
		NN	↓ ∈ ↑ ADJ
		NP	↓ ∈ ↑ ADJ
		NP-ADV	↓ ∈ ↑ ADJ
		RB	↓ ∈ ↑ ADJ
		WRB	↓ ∈ ↑ ADJ

Table C 4 Mother category WHADVP

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
ADJP	↓ ∈ ↑ ADJ	ADJP	↓ ∈ ↑ ADJ
ADVP	↓ ∈ ↑ ADJ	ADVP	↓ ∈ ↑ ADJ
CD	↓ ∈ ↑ ADJ	CD	↓ ∈ ↑ ADJ
DT	↓ ∈ ↑ ADJ	DT	↑SPEC DET=↓
JJ	↓ ∈ ↑ ADJ	JJ	↓ ∈ ↑ ADJ
JJR	↓ ∈ ↑ ADJ	JJR	↓ ∈ ↑ ADJ
JJS	↓ ∈ ↑ ADJ	JJS	↓ ∈ ↑ ADJ
NN	↓ ∈ ↑ ADJ	NN	↑SUBJ=↓
NNPS	↓ ∈ ↑ ADJ	NNP	↑SUBJ=↓
NNS	↓ ∈ ↑ ADJ	NNPS	↓ ∈ ↑ ADJ
NP	↓ ∈ ↑ ADJ	NNS	↓ ∈ ↑ ADJ
NX	↓ ∈ ↑ ADJ	NP	↑SUBJ=↓
PRN	↓ ∈ ↑ ADJ	NX	↓ ∈ ↑ ADJ
QP	↓ ∈ ↑ ADJ	PP	↓ ∈ ↑ ADJ
RB	↓ ∈ ↑ ADJ	PRN	↓ ∈ ↑ ADJ
RBS	↓ ∈ ↑ ADJ	QP	↓ ∈ ↑ ADJ
S	↓ ∈ ↑ ADJ	RB	↓ ∈ ↑ ADJ
VBG	↓ ∈ ↑ ADJ	RBS	↓ ∈ ↑ ADJ
VBN	↓ ∈ ↑ ADJ	S	↓ ∈ ↑ ADJ
VBZ	↓ ∈ ↑ ADJ	VBG	↓ ∈ ↑ ADJ
VP	↓ ∈ ↑ ADJ	VBN	↓ ∈ ↑ ADJ
WDT	↓ ∈ ↑ ADJ	VBZ	↓ ∈ ↑ ADJ
WHADVP	↑SPEC=↓	VP	↓ ∈ ↑ ADJ
WHNP	↑TOPIC=↓	WDT	↓ ∈ ↑ ADJ
WP$	↑POS=↓	WP$	↓ ∈ ↑ ADJ
WRB	↓ ∈ ↑ ADJ	WRB	↓ ∈ ↑ ADJ

Table C 5 Mother category WHNP

Category occurring to the left of the head	Annotation	Category occurring to the right of the head	Annotation
		NP	↑OBJ=↓
		SBAR	↑RELMOD=↓
		WHADVP	↑OBJ=↓
		WHNP	↑OBJ=↓

Table C 6 Mother category WHPP

Bibliography

[Bies et al , 1995] Bies, A , M Ferguson, K Katz and R MacIntyre (1995) *Bracketing Guidelines for Treebank II Style Penn Treebank Project* Penn Treebank Project, University of Pennsylvania, PA

[Bod, 1998] Bod, R (1998) *Beyond Grammar An Experience-Based Theory of Language*, CSLI Publications, Stanford, CA

[Bod and Kaplan, 1998] Bod, R and R Kaplan (1998) A Probabilistic Corpus-Driven Model for Lexical-Functional Analysis, in *17th International Conference on Computational Linguistics, COLING '98*, Montreal, Quebec, Canada, pp 145–151

[Bresnan, 1978] Bresnan, J (1978) A realistic transformational grammar In M Halle, J Bresnan, and G A Miller, (eds), *Linguistic Theory and Psychological Reality*, MIT Press, Cambridge, MA, pp 1–59

[Bresnan, 2001] Bresnan, J (2001) *Lexical-Functional Syntax* Blackwell Publishers, Oxford, UK

[Butt et al , 1999] Butt, M , T H King, M Niño and F Segond (1999) *A Grammar Writer's Cookbook*, CSLI publications, Stanford, CA

[Cahill and van Genabith, 2002] Cahill, A and J van Genabith (2002) TTS - A Treebank Tool, in *Proceedings of Third International Conference on Language Resources and Evaluation (LREC)*, Las Palmas, Canary Islands, Spain, pp 1712–1717

[Cahill et al , 2002a] Cahill, A , M McCarthy, J van Genabith and A Way (2002) Automatic Annotation of the Penn Treebank with LFG F-Structure Information, in *Proceedings of the LREC Workshop*

on Linguistic Knowledge Acquisition and Representation Bootstrapping Annotated Data, Las Palmas, Canary Islands, Spam, pp 8–15

[Cahill et al , 2002b] Cahill, A , M McCarthy, J van Genabith and A Way (2002) Parsing with PCFGs and Automatic F-Structure Annotation, in M Butt and T H King (eds), *Proceedings of the 7th International Lexical-Functional Grammar Conference*, CSLI Publications, Stanford, CA, pp 76-95, ISSN 1098–6782, CA, `http //cslipublications stanford edu/`

[Cahill et al , 2002c] Cahill, A , M McCarthy, J van Genabith and A Way (2002) Evaluating Automatic F-Structure Annotation for the Penn-II Treebank, in E Hinrichs and K Simov (eds), *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT '02) Workshop*, Sozopol, Bulgaria, pp 42–60, `http //www bultreebank org/Proceedings html`

[Cahill et al , 2003] Cahill, A , M McCarthy, R O'Donavan, J van Genabith and A Way (2003) Lexicalisation of Long-Distance Dependencies in a Treebank-Based, Statistical LFG Grammar, to appear in *Proceedings of the 8th International Lexical Functional Grammar Conference*, Saratoga Springs, NY

[Cahill, forthcoming] Cahill, A (forthcoming) *Probabilistic Parsing with Treebank Grammars and Automatic F-Structure Annotation*, Ph D Thesis, School of Computer Applications, Dublin City University, Dublin 9, Ireland

[Carroll et al , 1999] Carroll, J , G Minnen and T Briscoe (1999) Corpus annotation for parser evaluation, in *Proceedings of the EACL Workshop on Linguistically Interpreted Corpora (LINC-99)*, Bergen, Norway, pp 35–41

[Chomsky, 1970] Chomsky, N (1970) Remarks on nominalization In *Readings in English Transformational Grammar*, Roderick Jacobs and Peter Rosenbaum (eds), Waltham, MA Blaisdell, pp 184-221

[Church, 1988] Church, K W (1988) A stochastic parts program and noun phrase parser for unrestricted text, in *Proceedings of the Sec-*

ond Conference on Applied Natural Language Processing, Austin, Texas, pp 136–143

[Collins, 1996] Collins, M (1996) A New Statistical Parser Bases on Bigram Lexical Dependencies, in *34th Annual Meeting of the Association of Computational Linguistics*, Santa Cruz, CA, pp 184–192

[Crouch et al , 2002] Crouch, R , R Kaplan, T H King and S Riezler (2002) A Comparison of Evaluation Metrics for a Broad-Coverage Stochastic Parser, in *Proceedings of Third International Conference on Language Resources and Evaluation (LREC)*, Las Palmas, Canary Islands, Spain, pp 67–74

[Crouch et al , 2003] Crouch, R , M Dalrymple, R Kaplan, T H King and S Riezler (2003) The PARC 700 Dependency Bank, in *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora*, held at the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03), Budapest

[Dalrymple, 2002] Dalrymple, M (2002) *Lexical Functional Grammar*, Syntax and Semantics, Volume 34, Academic Press, San Diego, CA / London, U K

[Falk, 2001] Falk, Y (2001) *Lexical-Functional Grammar An Introduction to Parallel Constraint-Based Syntax* CSLI Publications, Stanford, CA

[Francis and Kucera, 1982] Francis, W and H Kucera (1982) *Frequency analysis of English usage Lexicon and grammar* Houghton Mifflin, Boston, MA

[Frank, 2000] Frank, A (2000) Automatic F-structure Annotation of Treebank Trees, in M Butt and T H King (eds), *Proceedings of the 5th International Lexical-Functional Grammar Conference*, The University of California at Berkeley, CSLI Publications, Stanford, CA, pp 139–160, ISSN 1098–6782, `http //www-csli stanford edu/publications/`

[Frank et al , 2001] Frank, A , J van Genabith, and A Way (2001) Treebank vs X-BAR based automatic F-structure annotation, in M Butt

and T H King (eds), *Proceedings of the 6th International Lexical-Functional Grammar Conference*, The University of Hong Kong, Hong Kong, CSLI Publications, Stanford, CA, pp 127–146, ISSN 1098-6782, http //cslipublications stanford edu/

[Frank et al, 2003] A Frank, L Sadler, J van Genabith and A Way (2003) From Treebank Resources to LFG F-Structures, in Anne Abeille (ed), *Treebanks Building and Using Syntactically Annotated Corpora*, Kluwer Academic Publishers, Dordrecht/Boston/London, The Netherlands, to appear (2003)

[Forst, 2003] Forst, M (2003) Treebank Conversion - Establishing a test-suite for a broad-coverage LFG from the TIGER treebank, in the *4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, to be held at EACL-03, the 10th Conference of the European Chapter of the Association for Computational Linguistics

[Hindle, 1989] Hindle, D (1989) Acquiring disambiguation rules from text, in *27th Annual Meeting of the Association for Computational Linguistics*, Vancouver, British Columbia, Canada, pp 118–125

[Jackendoff, 1990] Jackendoff, Ray S (1990) *Semantic Structures*, MIT Press, Cambridge, MA

[Johnson, 2002] Johnson, M (2002) A simple pattern-matching algorithm for recovering empty nodes and their antecedents, in *40th Annual Meeting of the Association of Computational Linguistics*, University of Pennsylvania, Philadelphia, PA, pp 136–143

[Kaplan and Bresnan, 1982] Kaplan, R and J Bresnan (1982) Lexical Functional Grammar a Formal System for Grammatical Representation, in J Bresnan (ed), *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge, MA, pp 173–281

[Kaplan and Zaenan, 1989] Kaplan, R and Zaenan, A (1989) Long-distance dependencies, constituent structure, and functional uncertainty, in M Baltin and A Kroch (eds), *Alternative Conceptions of Phrase Structure*, Chicago University Press, Chicago, IL, pp 17–42

[Lapointe, 1980] Lapointe, S (1980) *A Theory of Grammatical Agreement* Doctoral dissertation, University of Massachusetts at Amherst

[Liakata and Pulman, 2002] Liakata, M and S Pulman (2002) From trees to predicate-argument structures, in *Proceedings of 19th International Conference on Computational Linguistics, COLING '02*, Taipei, Taiwan, pp 563–569

[Magerman, 1994] Magerman, D (1994) *Natural Language Parsing as Statistical Pattern Recognition*, PhD Thesis, Stanford University, CA

[Marcus et al , 1993] Marcus, M , M A Marcinkiewicz, and B Santorini (1993) Building a large annotated corpus of English the Penn Treebank, in *Computational Linguistics*, **19**(2) 313-330 Reprinted in *Using large corpora*, Susan Armstrong (ed), 1994, Cambridge, MIT Press, MA, pp 273-290

[Marcus et al , 1994] Marcus, M , G Kim, M A Marcinkiewicz, R MacIntyre, M Ferguson, K Katz and B Schasberger (1994) The Penn Treebank A Revised Corpus Design for Extracting Predicate Argument Structure, in *Human Language Technology, ARPA Workshop*, Princeton, NJ, Morgan Kaufmann

[Pollard and Sag, 1994] Pollard, C and I Sag (1994) *Head-Driven Phrase Structure Grammar*, University of Chicago Press, Chicago, IL, and CSLI Publications, Stanford, CA

[Riezler et al , 2002] Riezler, S , R Kaplan, T King, M Johnson, R Crouch, and J Maxwell III (2002) Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques, in *40th Annual Meeting of the Association of Computational Linguistics*, University of Pennsylvania, Philadelphia, PA, pp 271–278

[Sadler et al , 2000] Sadler, L , A Way, J van Genabith (2000) Automatic F-Structure Annotation from the AP Treebank, in M Butt and T H King (eds), *Proceedings of the 5th International Lexical-Functional Grammar Conference*, CSLI Publications, Stanford, CA, pp 226–243, ISSN 1098-6782, `http //cslipublications stanford edu/`

[Santorini, 1991] Santorini, B (1991) *Part-of-Speech Tagging Guidelines for the Penn Treebank Project* Technical report MS-CIS-90-47, Department of Computer and Information Science, University of Pennsylvania

[van Genabith and Crouch, 1996] van Genabith, J and D Crouch (1996) Direct and Underspecified Interpretations of LFG f-structures, in *Proceedings of 16th International Conference on Computational Linguistics, COLING '96*, Copenhagen, Denmark, pp 262–267

[van Genabith et al , 1999] van Genabith, J , L Sadler and A Way (1999) Data-Driven Compilation of LFG Semantic Forms, in *Proceedings of the EACL Workshop on Linguistically Interpreted Corpora (LINC-99)*, Bergen, Norway, pp 69–76

[Way, 1999] Way, A (1999) A Hybrid Architecture for Robust MT using LFG-DOP, in *Journal of Experimental and Theoretical Artificial Intelligence*, Taylor and Francis, London, pp 441–471

[Way, 2003] Way, A (2003) Translating with Examples The LFG-DOT Models of Translation, in M Carl and A Way (eds), *Recent Advances in Example-Based Machine Translation*, Kluwer Academic Publishers, Dordrecht, The Netherlands (in press), pp 443–472