Treebank-Based Deep Grammar Acquisition for French Probabilistic Parsing Resources

Natalie Schluter

A dissertation submitted in fulfilment of the requirements for the award of

Doctor of Philosophy (Ph.D.)

to

Dublin City University

Faculty of Engineering and Computing, School of Computing

Supervisor: Josef van Genabith

January 2011

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy 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 :____

(Natalie Schluter)

Student ID : 55131328

Date : January 10, 2011

To my husband, Kristian, and our daughter, Magdalena.

Contents

Abstract

Motivated by the expense in time and other resources to produce hand-crafted grammars, there has been increased interest in wide-coverage grammars automatically obtained from treebanks. In particular, recent years have seen a move towards acquiring deep (LFG, HPSG and CCG) resources that can represent information absent from simple CFG-type structured treebanks and which are considered to produce more language-neutral linguistic representations, such as syntactic dependency trees. As is often the case in early pioneering work in natural language processing, English has been the focus of attention in the first efforts towards acquiring treebank-based deep-grammar resources, followed by treatments of, for example, German, Japanese, Chinese and Spanish. However, to date no comparable large-scale automatically acquired deep-grammar resources have been obtained for French. The goal of the research presented in this thesis is to develop, implement, and evaluate treebank-based deep-grammar acquisition techniques for French.

Along the way towards achieving this goal, this thesis presents the derivation of a new treebank for French from the Paris 7 Treebank—the Modified French Treebank—a cleaner, more coherent treebank with several transformed structures and new linguistic analyses. Statistical parsers trained on this data outperform those trained on the original Paris 7 Treebank, which has five times the amount of data.

The Modified French Treebank is the data source used for the development of treebank-based automatic deep-grammar acquisition for LFG parsing resources for French, based on an f-structure annotation algorithm for this treebank. LFG CFG-based parsing architectures are then extended and tested, achieving a competitive best f-score of 86.73% for all features. The CFG-based parsing architectures are then complemented with an alternative dependency-based statistical parsing approach, obviating the CFG-based parsing step, and instead directly parsing strings into f-structures.

Acknowledgements

Coming to Dublin to do my postgraduate studies has definitely been one of the best decisions of my life. I must thank first and foremost my supervisor Josef van Genabith for offering this chance to a Canadian student in Montreal (me) that he had never met. Without his tireless patience, motivation, and positivity, this thesis would definitely never have gotten written.

I was one of the last to join the GramLab team and owe many thanks to Grzegorz Chrupała, Yuqing Guo, and Ines Rehbein, who filled me in on their own trial and error experiences with software and new ideas. In particular, I would like to thank Grzegorz and Yuqing for lending me their own implementations of the triples evaluation software, which made my life much easier!

On a personal note, I would like to thank my husband Kristian Krohn Djurhuus, who has patiently listened to my complaints about treebanks for the last five years and who constantly pushed me to *just finish that research anyways*. And, of course, I owe much to my mother-in-law, Susanne Krohn Djurhuus, who lived several weeks with us, taking care of my baby daughter, while I finished writing my thesis.

Finally, many thanks are due to Science Foundation Ireland who supported my research with grant 04/IN/I527.

List of Tables

List of Figures

List of Acronyms

CFG	context-free grammar
CCG	Combinatory Categorial Grammar
DCULFG	Dublin City University's Treebank-Based Unification Grammar Acquisition project for the Automatic Annotation of the Penn-II Treebank with Feature-Structure Information
FU	functional uncertainty equation
GramLab	Multilingual Treebank-Based Deep LFG Grammars for Parsing, Generation and Machine Translation
HPSG	Head-Phrase Structure Grammar
LDD	Long Distance Dependency
LFG	Lexical Functional Grammar
MFT	Modified French Treebank
MFTDB	MFT Dependency Bank
P7T	Paris 7 Treebank
SVM	Support Vector Machine
TAG	Tree Adjoining Grammar

Chapter 1

Introduction

Motivated by the expense in time and other resources to produce hand-crafted grammars, there has been increased interest in wide-coverage grammars automatically obtained from treebanks. In particular, recent years have seen a move towards acquiring deep resources that can represent information absent from simple CFG-type structured treebanks and which are considered to produce more language-neutral linguistic representations, such as syntactic dependency trees. As is often the case in early pioneering work in natural language processing, English has been the focus of attention in the first efforts towards acquiring treebank-based deep-grammar resources, followed by treatments of, for example, German, Japanese, Chinese and Spanish. However, to date, no comparable large-scale automatically acquired deep-grammar resources have been obtained for French. The goal of the research presented in this thesis is to develop, implement and evaluate treebank-based deep-grammar acquisition techniques French.

1.1 Context of the Research

Phrase-structure grammars (CFGs) are the main syntactic representation formalism for many mainstream linguistic theories. A central concern for natural language processing of raw text over the past decades has been to find ways to automatically assign syntactic structure to text. Hand-crafted grammars built to these ends, however, have turned out to be limited in several important respects. The main problem has been that manual grammar development for a wide-coverage and robust system is extraordinarily time-consuming and, as a result, very expensive. Moreover, phrase-structure grammar representations tend to be rather language dependent, and this has lead to a considerable body of research on whether more abstract and language-neutral representations can be found for describing common linguistic relations across languages.

Motivated by the expense in time and other resources to produce hand-crafted grammars, there has been increased interest in automatically obtained wide-coverage grammars that can represent information absent from simple CFG-type structured treebanks to provide more language-neutral linguistic representations. Dublin City University's *Treebank-Based Unification Grammar Acquisition* project for the *Automatic Annotation of the Penn-II Treebank with Feature-Structure Information* (DCULFG, 2001-2004) was an early project, which built technology for robust, large-scale, data-driven acquisition and parsing within the framework of a linguistic theory comprising a deeper, more abstract, syntactic formalism—Lexical Functional Grammar (LFG). The project is a natural development and extension of the basic, automatic treebank PCFG acquisition paradigm (Charniak, 1997).

The DCULFG project, as with most early efforts in NLP, was developed on and for English. In essence, syntax is language dependent, even though the syntactic formalism may be thought to be language independent. Automatic tools for the syntactic analysis of raw text will essentially have a language dependent component also, though one might use language independent techniques to "learn" these language dependent structures, assuming suitable training resources exist (eg. treebanks).

At the start of the DCULFG project, a database containing more abstract syntactic representations was not available to support training a deep probabilistic parser. Therefore, a major part of the DCULFG project was to augment the existing CFG-based syntactic information provided by the Penn-II treebank as well as the CFG output of parsers trained on this treebank with information describing the deeper representation and relations among syntactic units: grammatical "function" equations (essentially describing bilexical labeled dependencies augmented with grammatical features, such as aspect, number, etc., and non-local dependencies). Though the actual extension of the information represented in the original treebank was carried out in an automatic fashion, via an *f-structure annotation algorithm*, the implementation of the algorithm as well as the construction of the corresponding annotation program itself is strongly language and treebank data-structure dependent and therefore based on and constructed for Penn-II style tree representations of English phrases only. However, it was thought that the technology could be migrated to other languages and treebanks with some additional effort to construct language tailored implementations of the established annotation algorithm. Toy projects for Spanish, German, and Chinese aimed to show early and limited proof-of-concept results (O'Donovan, Cahill, van Genabith and Way, 2005; Cahill, 2004; Burke et al., 2004). Following this, the GramLab project (2004-2008) was begun to effectively test this idea in depth. The research presented in this thesis is part of GramLab.

The aims of the GramLab project are twofold. The first aim was to build treebank-based multilingual deep-grammar parsing systems based on and adapting the technology established by DCULFG, for Chinese, Japanese, Arabic, Spanish, French, and German. At the time, such resources did not exist. Secondly, the project aimed to evolve the DCULFG technology, and explore ways in which the original components can be improved or added to for overall improvement of the system. My research is a contribution towards the two aims of the GramLab project in a very specific manner. The primary objective of my research is the automatic acquisition of wide-coverage, robust LFG resources for French. The secondary objective of my research concerns the evolution of the original DCULFG model by exploring other parsing paradigms—in particular, dependency parsing into f-structures, obviating the CFG-based parsing step in more traditional LFG parsing architectures.

Concerning the first aim of the GramLab project, early proof-of-concept trials in the migration of this technology to other languages (German, Spanish and Chinese) attempted to show how the model developed for English can be applied to to other languages (O'Donovan, Cahill, van Genabith and Way, 2005; Burke et al., 2004; Cahill, 2004). However, with the possible exception of Spanish, these proof-of-concept trials did not produce results as impressive as for English in any of the languages and treebank resources investigated. Also, the proof-of-concept research was further put into question by the, at the time, insufficient evaluation schemes. Applying the original DCULFG approach to the respective languages and treebank resources has generally proven more complicated than was originally thought. Though the DCULFG model has provided guidelines for the development of acquisition and parsing system within other linguistic contexts, several important factors impact on the effectiveness of the corresponding systems.

Work on other languages such as Spanish, Chinese, German, and Arabic as well as my work on French, in the GramLab project has shown that the DCULFG annotation algorithm as well as component algorithms such as the long-distance dependency resolution approach make assumptions about both the language and the data structures (treebank trees) in question.

For my work on French, unlike for the other GramLab languages, I did not have a consistent and reliable treebank resource with syntactic structures compatible with LFG annotations. The Paris 7 French Treebank (Abeillé and Clément (2003), for example) was lacking in several important respects. The starting point of my research, therefore, was to first derive a usable treebank resource for French grammar acquisition and parsing—the *Modified French Treebank* (MFT) (Chapter 2). I also show that for French, a smaller but high-quality resource supports better statistical modeling than a larger, less consistent resource.

Equipped with this new treebank, I developed the LFG parsing resources for French (Chapter 3). A new f-structure annotation algorithm was developed for French that takes into account some of the criticisms of the English model,[1] as well as important linguistic and structural differences between French and English and the different treebank tree data-structures of the Penn-II treebank and the MFT. Also, following my work on the treebank,

[1]For example, the representation of tense and aspect.

the MFT is now equipped with a complete and reliable treebank functional tag annotation which can be exploited directly in an annotation algorithm, and which makes another part of the original LFG annotation algorithm inefficient for certain constituents (Chapter 4).

The second goal of the GramLab project concerns various additions to the DCULFG model, such as implementing machine learning algorithms in various components or as pre/post-processing (Chrupała and van Genabith, 2006b), exploring the use of multi-word unit recognition (Cafferkey et al., 2007), implementing automatic morphological annotation for disambiguation (Rehbein and van Genabith, 2006), obtaining better training instances and a wider array of sublanguages in training data (Chrupała and van Genabith, 2006b; Guo, Wang and van Genabith, 2007; O'Donovan, Burke, Cahill, van Genabith and Way, 2005), as well as exploring the integration of hand-crafted and data-driven technologies. My research on directly parsing with dependency structures, rather than going through CFG-based technology as is traditionally done in LFG, for French contributes further towards accomplishing this goal.

Parsing within the LFG framework has always first considered c-structures. However, the question remains as to the utility of this integrated c-structure parsing step; in particular, in a context where efficient and accurate data-driven parsers[2] exist which directly parse strings into dependency structures which can be obtained from f-structures. My research aims to answer this question for French (Chapter 5).

Treebank Based Deep-Grammar Induction within other Linguistic Frameworks. The last decade has also seen active research in treebank-based deep grammar acquisition within the deep-grammar frameworks of Tree Adjoining Grammar (TAG) (Chen et al., 2006; Xia, 1999), Combinatory Categorial Grammar (CCG) (Hockenmaier, 2006; Hockenmaier and Steedman, 2007) and Head-Phrase Structure Grammar (HPSG) (Miyao and Tsujii, 2005; Nakanishi et al., 2004), which happen to be constraint-based also. This research has been concentrated on English and the Penn Treebank, though there has been some work on

[2]For example, MST parser (McDonald et al., 2005) and MALT parser (Nivre et al., 2006).

other languages—for example, German (Hockenmaier, 2006) and Turkish Cakici (2005) for CCG, Korean (Park, 2006) for TAG, and Japanese (Yoshida, 2005) for Japanese. No work has been carried out to date on treebank-based deep-grammar acquisition for French in any other linguistic framework.

GramLab remains the first systematic investigation into treebank-based deep-grammar acquisition within a single linguistic framework—LFG.

1.2 Preliminaries and Review of Related Research

1.2.1 Lexical-Functional Grammar

Lexical-Functional Grammar (LFG) is a constraint based theory of language, whose basic architecture distinguishes two levels of syntactic representation : *c-structure* (constituent structure) and *f-structure* (functional structure) —c-structures corresponding to traditional constituent tree representation, and f-structures to a traditional dependency representation in the form of an attribute value matrix.[3]

Consider, for example, the following sentence.

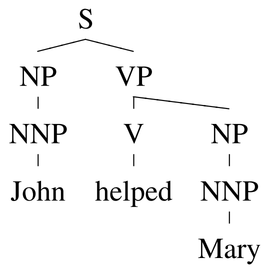
(1) John helped Mary

Sentence (1) has the c-structure shown to at the top in Figure 1.1, which corresponds to the f-structure shown in the middle in the same figure.

Like any attribute-value matrix, f-structures are the minimal solution to a set of functional equations such as $(f\ a) = v$, where f is an f-structure, a is some attribute, and v is the value taken by that attribute, possibly another f-structure.

These two levels of representation (f-structure and c-structure), for a given phrase, are explicitly related by a structural mapping, called the *f-description*, often denoted by ϕ, which maps c-structure nodes to f-structure nodes.

[3]A detailed introduction to LFG may be found in (Dalrymple, 2001).



$$\begin{bmatrix} \texttt{pred} & \text{‘help’} \\ \texttt{subj} & \begin{bmatrix} \texttt{pred} & \text{‘John’} \end{bmatrix} \\ \texttt{obj} & \begin{bmatrix} \texttt{pred} & \text{‘Mary’} \end{bmatrix} \end{bmatrix}$$

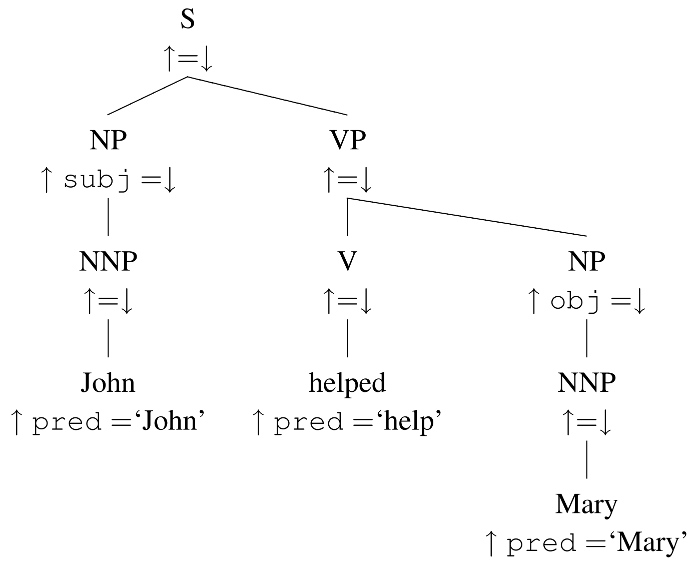


Figure 1.1: C-structure (top), basic f-structure (middle), and basic annotated c-structure (bottom) for Example (1).

In the LFG framework, this mapping may be given by functional annotations inserted into the c-structure tree, as in Figure 1.1 on the bottom.

The metavariables $\uparrow$ and $\downarrow$ refer to the f-structure of the mother node and that of the node itself, respectively. So that if node n is annotated $\uparrow=\downarrow$, then n's f-structure is mapped to the same f-structure as n's mother's f-structure. Also, if n has the annotation $\uparrow\texttt{obj}=\downarrow$, this means that the f-structure associated with n is mapped to the value of the mother's f-structure `obj` attribute. LFG also has equations for members of sets, such as $\downarrow \in \uparrow \texttt{adjunct}$, which states that the node's f-structure is mapped to an element of the mother's ADJ attribute.

The f-structure and the annotated tree in Figure 1.1 are derived from the following annotated rules, productions whose left-hand sides and right-hand sides can be viewed as (simple) regular expressions.

$$S \longrightarrow \begin{array}{cc} \text{NP} & \text{VP} \\ \uparrow \texttt{subj} = \downarrow & \uparrow = \downarrow \end{array}$$

$$NP \longrightarrow \begin{array}{c} \text{NNP} \\ \uparrow = \downarrow \end{array}$$

$$VP \longrightarrow \begin{array}{cc} \text{V} & \text{NP} \\ \uparrow = \downarrow & \uparrow \texttt{obj} = \downarrow \end{array}$$

1.2.2 Overview of the DCULFG Project: the Original Methodology

The technology for treebank-based acquisition of multilingual LFG probabilistic parsing resources is based on the English model developed in DCULFG and adapted to the language and treebank data structures in question. There are three main stages in initiating this process, the basic input for which is a CFG-type treebank. These stages include the construction and application of an f-structure annotation algorithm combined with satisfiability verification, subcategorisation frame extraction, and long-distance dependency extraction (Section 1.2.2.1). Given the resources produced in these initial stages, two probabilistic parsing architectures were developed (Section 1.2.2.2).

1.2.2.1 Initial Stages

Augmenting the Penn-II Treebank with Deep Dependency Representation Annotation. The treebank is automatically annotated with f-equations by the implementation of an annotation algorithm constructed specifically for English and the Penn-II treebank.

The annotation algorithm for the Penn-II treebank is composed of four separate modules (McCarthy, 2003; Burke, 2006). The first module consists of left-right context annotation principles: the head of a phrase is detected (adapting Magerman's (1994) scheme) and annotated, then the sister tags of the head are annotated depending on whether they are in the right context of the head or in the left context. Because coordination in the Penn-II treebank is highly ambiguous, a separate module was needed to provide appropriate annotations; this constitutes the second module. The third module carries out the annotation of trace elements, covering such linguistic phenomena as passivisation, topicalisation, wh-questions and relative clauses. The fourth module performs a catch-all and clean-up, which attempts to correct errors and overgeneralisations caused by the three previous modules.

Following the automatic annotation of Penn-II trees, f-equations are collected and sent to a constraint-solver to produce f-structures. The annotation algorithm is evaluated in terms of coverage and quality. It achieves 99.83% coverage, an f-score of 96.93% for all grammatical features and 94.28% for preds-only against the DCU150 and an f-score of 87.33% for all grammatical features and 84.45% for preds-only against the PARC 700.[4]

Subcategorisation Frame Extraction from Deep Representations of the Penn-II/III Treebank. Access to adequate lexical resources is crucial in the functioning of any wide-coverage computational system carrying out a syntactic analysis of text. As with grammar writing, manual construction of such resources is time-consuming, expensive and rarely ever complete. O'Donovan, Burke, Cahill, van Genabith and Way (2005) give an approach to automating subcategorisation frame acquisition, given the availability of the augmented Penn-II/III treebanks obtained from the method above.

O'Donovan, Burke, Cahill, van Genabith and Way (2005)'s system for automatised lexical resource acquisition takes f-structures from the automatically augmented treebanks as input. The central algorithm recursively traverses these f-structures, recording for each local `pred` value, the governable argument attributes. Other information recorded includes

[4]See, for example, (Burke, 2006) or (McCarthy, 2003) for more details.

the syntactic categories of the predicate and its subcategorised arguments, prepositions required by obliques, verbal passivity, particles accompanying verbs, as well as counts (for the assignment of conditional probabilities).

Following the appropriate mapping of lexical information, an evaluation is carried out against COMLEX (MacLeod et al., 1994), obtaining a best f-score of 72%.

Long-Distance Dependency Extraction. Linguistic phenomena such as topicalisation and wh-movement are characterised by the dislocation between surface realisation and semantic interpretation of linguistic material; these phenomena are referred to as long-distance dependencies. The Penn-II treebank contains empty nodes and co-indexation to represent long-distance dependencies at the c-structure. The f-structure annotation algorithm for English annotates these, and long-distance dependencies appear as reentrancies at the f-structure level also. All long-distance dependencies are extracted from f-structures and associated with relative frequencies conditioned on the local (communicative) function (eg. `topic`, `focus`) of the reentrant f-structure in question (Cahill et al., 2004).

1.2.2.2 Parsing into Deep-Syntactic Structures Using the Augmented Penn-II Treebank.

With the augmentation of the Penn-II treebank with f-structure annotations, a resource was made available to which one could apply already available machine learning methods. The DCULFG project developed two parsing architectures for PCFG-based approximations of LFG grammars: the *pipeline* and the *integrated* architectures (Figure 1.2).

For the pipeline architecture, the parser is trained on the original treebank trees. Parser output trees are then annotated by the annotation algorithm. In the integrated architecture, the original treebank is augmented with f-equations and the parser is trained on the augmented version of the treebank.

In both architectures, long-distance dependency resolution is carried out at the f-structure level. The pipeline architecture achieves an f-score of 84.76%, and the integrated architec-

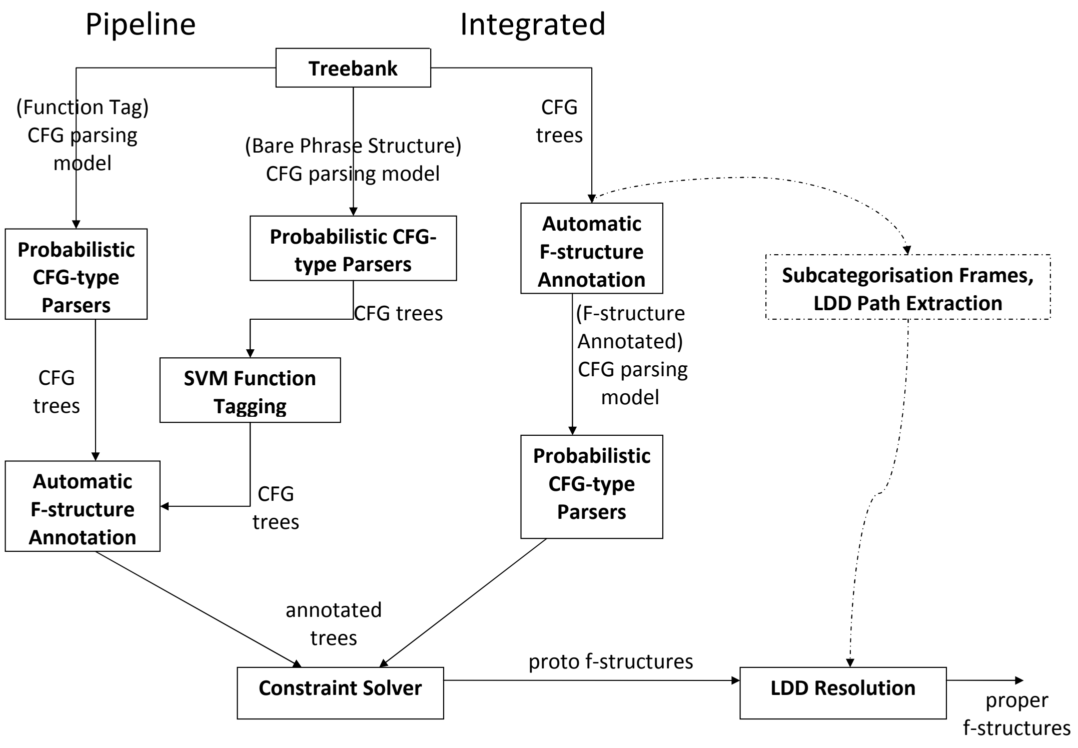

Pipeline
Integrated
Treebank
(Function Tag) CFG parsing model
(Bare Phrase Structure) CFG parsing model
CFG trees
Probabilistic CFG-type Parsers
Probabilistic CFG-type Parsers
Automatic F-structure Annotation
Subcategorisation Frames, LDD Path Extraction
CFG trees
(F-structure Annotated) CFG parsing model
CFG trees
SVM Function Tagging
Automatic F-structure Annotation
CFG trees
Probabilistic CFG-type Parsers
annotated trees
proto f-structures
Constraint Solver
LDD Resolution
proper f-structures

Figure 1.2: Overview of treebank-based LFG parsing architectures.

ture achieves 87.09%, against the DCU 105. Against the PARC700, the pipeline achieves an f-score of 80.33%, whereas the integrated architecture achieves 78.74%. [5]

Long-Distance Dependency Resolution in Deep Syntax. In LFG, long-distance dependencies are resolved at the f-structure level, accounted for by functional uncertainty equations which capture a regular set of optional non-local dependencies along a specific path of dependencies. DCULFG's system extracts a finite approximation of paths from f-structures obtained from the augmented Penn-II treebank, along with their counts relative to the communicative attributes `topic`, `topic-rel` or `focus` for deriving conditional probabil-

[5]See (Cahill, 2004) for details. A new parsing best score of 82.73% has recently been reported by Cahill et al. (2008).

ities. It then uses these finite approximations and subcategorisation information obtained as described in Section 1.2.2.1 to resolve these dependencies in f-structures obtained from parsed (augmented) text.[6]

1.2.3 GramLab: Towards Multilingual LFG Resources

In GramLab, LFG based parsing systems are under development for German, Chinese, Spanish and Arabic. As a contribution to GramLab, this thesis presents the research on acquiring French LFG parsing resources.

LFG Resources for German. Following the proof-of-concept work carried out by Cahill (2004) on German, Ines Rehbein has worked towards overcoming the limitations of the initial annotation algorithm and evaluation scheme for the TiGer treebank.

German is less configurational than English, but morphologically richer. This provides a first example of the problems to be overcome in the migration of language technology developed on English to other languages. Rehbein first attempted to account for the information exploited by the DCULFG version of the annotation algorithm in terms of right and left contexts by observing first that such information, though not expressed always in terms of word order, may be available in terms of morphology in German. This led to her work on the automatic morphological annotation of the TiGer treebank (Rehbein and van Genabith, 2006).

To render the resulting f-structures more suitable to be evaluated against the publicly available TiGer Dependency Bank, Rehbein fundamentally revised and extended the f-structure annotation algorithm, incorporating a substantially larger set of features, achieving an f-score of 77.5% for all grammatical features. She developed an independent and more compatible German dependency bank of 250 sentences for the purposes of evaluation, showing that her new annotation algorithm actually obtains an f-score of 93.5% when the triples mapping is fair. Rehbein's first parsing experiments under the integrated archi-

[6]See (Cahill et al., 2004) for details.

tecture result in an f-score of 69.9%.

LFG Resources for Chinese. Proof-of-concept work for Chinese was carried out by Burke et al. (2004). However, closer study of Chinese linguistic phenomena important to the construction of an adequate annotation algorithm, specifically regarding "empty productions" in the Chinese Penn treebank, revealed that much was not considered in the original development of the system for Chinese.

Guo, van Genabith and Wang (2007)'s work involved revising and improving the f-structure annotation algorithm to increase robustness and accuracy, and to more genuinely reflect the linguistic structure of Chinese. Also, this new system now generates true f-structures with long-distance dependencies effectively resolved, elaborating a new non-local dependency resolution algorithm specifically designed for Chinese. The approach is now scaled to the entire Penn Chinese treebank. Finally, Yuqing Guo collaborated with Xerox PARC in the construction of a new 200 sentence gold standard for Chinese, for evaluation purposes.

Specific fundamental modifications to the DCULFG technology, based on the linguistic structures of Chinese, involved, for example, non-local dependency resolution (which includes long-distance dependency resolution), and the basic f-structure annotation algorithm structure which creates intermediate (dependency tree-like) f-structure templates to be annotated rather than carrying out the annotation on treebank c-structures.

The modified annotation algorithm obtains an f-score of 96% on gold trees and 80.01% on parser output.

LFG Resources for Spanish. Chrupała and van Genabith (2006b)'s work extends that of O'Donovan, Cahill, van Genabith and Way (2005), by improving the annotation algorithm for Spanish to take into account more linguistic phenomena, by extending the annotations to account for functional optionality (necessary for the account of some linguistic phenomena in Spanish), and by exploring machine learning techniques to account for the reliance of the annotation algorithm for Spanish on existent function labels in the Spanish Cast3LB treebank (see Figure 1.2, centre). This work improves on already decent scores for Spanish LFG parsing by an impressive 3% (with a 75.67% f-score in the integrated model for all grammatical functions).

LFG Resources for Arabic. Tounsi et al. (2009a) use the Penn Arabic Treebank (Bies and Maamouri, 2003) to induce an arabic LFG grammar. Though the annotation of this treebank is in strong correspondence with that of the Penn-II Treebank (M. Marcus and Schasberger, 1994), linguistic differences bear some effect on the design and implement an f-structure annotation algorithm for Arabic using this treebank, especially at the morphological level of analysis. For this reason, the left-right context principles module of the f-structure annotation algorithm for English is much less important. On the other hand new modules are introduced to handle syntactic complexity. Initial parsing experiments in the pipeline parsing architecture yields an f-score of 77.78% (Tounsi et al., 2009b).

1.2.4 Related and More Recent Work on Parsing French

Essentially motivated by Rehbein and van Genabith (2007)'s observation that unlabeled dependency evaluation is a more annotation-neutral metric, the series of papers (Candito et al., 2009; Candito, Crabbé and Denis, 2010; Candito, Nivre, Denis and Anguiano, 2010) explore the question of the most successful parsing architecture in terms of this type of evaluation for French. Following a preprocessing step, two strategies are compared for deriving dependencies from a new reduced version of the Paris 7 Treebank, which greatly mirror a coarser version of the pipeline and integrated architectures presented here. The dependencies sought are based on the constituent head-finding rules, as proposed by Lin (1995); dependency labels are taken from the Paris 7 Treebank if available or obtained by heuristics. One strategy, called integrated parsing (*analyse intégrée*) uses a PCFG and statistical dependency parsers to recover Paris 7 Treebank function labels and dependencies. The second strategy, called sequential parsing (*analyse séquentielle*) uses a PCFG parser and recovers function labels with a classifier in a post-processing step. The essential difference with this (later) work and my own work presented in this thesis is in the deepness of the syntactic description: I worked towards deep-grammar parsing whereas this more recent work aimed at a sort of surface-dependency parsing.

1.3 Thesis Outline

This thesis is organised as follows.

- Chapter 2 presents the data source which is used in the remainder of the thesis: the Modified French Treebank.

  The Modified French Treebank is a new French Treebank, derived from the Paris 7 Treebank, which is cleaner, more coherent, has several transformed structures, and introduces new linguistic analyses. In this chapter, I investigate one important effect of a clean treebank on corpus-based linguistics, providing a strong argument in favour of quality versus quantity in statistical parsing: a probabilistic parser trained on clean and transformed data performs better than its counterpart trained on the original French treebank, which consists of five times the data. Moreover, I show how data which has a high error rate and is not "parser-friendly" can lead to the potentially erroneous conclusions about the impact of lexicalisation on probabilistic parsing of French.

- Chapter 3 outlines the design and implementation of the f-structure annotation algorithm for French and the Modified French Treebank. Building on the ideas of Burke (2006); McCarthy (2003); Sadler et al. (2000), the f-structure annotation algorithm for French reflects criticism about certain syntactic representations of the original model for English as well as differences in terms of language and data source from the other language models. In addition, it implements a simple coordination distribution algorithm, based on the LFG analyses for coordination as presented by, for example, Dalrymple (2001), which is a novel and significantly beneficial addition to the parsing architecture laid out by Cahill et al. (2005, 2008).

This chapter also discusses the final steps (namely, hand verification and completion) in the construction of the MFT Dependency Bank (MFTDB), an f-structure gold standard to be used for evaluation of all deep parsing work reported in this thesis.

- In Chapter 4, I present the application of treebank-based LFG acquisition and parsing to French. I show that with modest changes to the established parsing architectures, encouraging results can be obtained for French, with an overall best dependency structure f-score of 86.73% for all features. I also extend the existing parsing architectures—introducing (1) a simplified architecture and a (2) machine learning approximation of an established parsing architecture (in the spirit of Chrupała and van Genabith (2006b))—and evaluate this as well.

- Chapter 5 presents work on directly parsing into f-structures using statistical dependency parsing technology. It gives a *mise-en-scène* between theoretical dependency syntax and dependency parser practical requirements, an *entrée en scène* for f-structures in the literature for dependency parsing, an approach to representing f-structures in LFG as pseudo-projective dependencies, a first attempt to reconcile parsing LFG and dependency parsing, and, finally, the first treebank-based statistical dependency parsing results for French.

- In chapter 6, I provide some concluding remarks and outline directions for future work.

Chapter 2

Preparing, Restructuring, and Augmenting a French Treebank: Construction of the Modified French Treebank

This chapter presents the data source adopted and adapted for my research: the Paris 7 Treebank adapted into the Modified French Treebank. It also presents preliminary CFG parsing scores for parsers trained on the adapted treebank.[1]

2.1 Introduction

The construction of the Paris 7 Treebank (P7T) resulted in the first treebank available for French (Abeillé et al., 2004; Abeillé and Barrier, 2004). Its use in research, however, has proven challenging. Arun and Keller (2005), for example, observe a number of points in which the treebank should be improved or even completely structurally reorganised before any serious study can be carried out using it.

[1]This work was previously published as (Schluter and van Genabith, 2007).

My goal has been to create a French treebank with consistent and coherent annotations and with a comparatively low error rate, that supports efficient statistical parsing paradigms while compromising as little as possible on linguistically relevant structural information. I aimed to achieve this, while carrying out only the minimum number of changes to the P7T necessary to meet this goal.

The necessary correction and modification of the P7T has led to the creation of the *Modified P7T*, which I will simply call Modified French Treebank (MFT). My research focusses on the functionally annotated subset of 9357 sentences from the P7T, and the MFT now consists of the the first[2] half of these sentences.

Following an overview of the P7T (Section 2.2), I introduce the MFT via the various structural changes (Section 2.3), formatting and error mining (Section 2.4) applied to the P7T source material. Using statistical analysis techniques, I show that the MFT and P7T have become very different treebanks (Section 2.5). As a means of showing the importance of such changes in treebank-based linguistic analysis, I provide results for statistical parsing in Section 2.6, and draw some important conclusions. Finally, in Section 2.7, I touch upon some more recent result in parsing French.

2.2 The Paris 7 Treebank

Work on the P7T was carried out by a research team at the Université Paris 7, under the direction of Anne Abeillé. The treebank consists of *Le Monde* newspaper article excerpts published between 1989 and 1993, written by various authors, and covering an array of topics. The full P7T contains 20,648 sentences annotated for phrase structure, (and additionally, about half with grammatical function tags) comprising 580,945 words. Table 2.1 gives the phrase tags of the P7T. In particular, there is no VP, except in the cases of some participial phrases (VPpart) and infinitival phrases (VPinf).[3]

Table 2.2 gives the syntactic function labels used in the functionally annotated sections

[2]In alphabetical order of the filenames.

[3]The phrase VN is considered to be more of a convention, grouping together all parts of composed verbs into one unit with their clitic pronouns, as well as any modifier phrases occurring between these.

label	**syntactic role**
AP	adjectival phrase
VPinf	infinitival phrase
AdP	adverbial phrase
Srel	relative clause
COORD	coordinated phrase
Ssub	subordinated clause
NP	noun phrase
Sint	internal, inflected sentence
PP	prepositional phrase
VN	verb kernel
VPpart	participial phrase
SENT	independent sentence

Table 2.1: Phrase Tags of the Paris 7 Treebank.

of the P7T. Only some clitics and those phrases which are sisters of a VN constituent carry functional annotations. This assumes that any phrase which is a sister element of VN functionally depends directly on the verb kernel; I show that this is not always the case and present a new functional annotation scheme in Section 2.3.5.

label	**functional role**
SUJ	subject
DE-OBJ	de (*of/from*)-object
OBJ	object
A-OBJ	à (*to*)-object
P-OBJ	prepositional-object
MOD	modifier
ATS	subject attribute
ATO	object attribute

Table 2.2: Syntactic Function Labels of the Paris 7 Treebank.

My research focusses on the first half of the functionally annotated sentences of the treebank; there are, in total, 20 files that contain the 9357 functionally annotated sentences, and I am working with the first ten of these files. These files originally contain 4741 sentences, comprising 134,445 words.

2.3 Structural Changes

The MFT differs significantly from the P7T, in terms of its phrase structure as shown by the statistical tests in Section 2.5. Major structural changes to the original P7T trees include increased rule stratification, introduction of analyses for untreated structures, information propagation, coordination raising, the addition of missing functional tags, and the introduction of functional path tags.

2.3.1 Rule Stratification

While maintaining a relatively flat syntactic analysis, the MFT has the property that there is one distinct head (and sometimes also one co-head) for each constituent. For example, NP, AP, and AdP constituents that have modifiers will have separate constituents for those modifiers. Figure 2.1 provides an example of increased stratification for AdP in Example (1); note that some underspecification has been maintained between the modifying adverbial phrases of the head adverb *bien* ('well').[4]

(1)	encore	pas	très	bien
	still	not	very	well

'still not very well'[5]

2.3.2 Introduction of Analyses for Untreated Structures

Compared to the P7T, the MFT offers increased coverage of linguistic phenomena. 'It'-cleft constructions provide an important example of structures that remained untreated in the P7T annotation guidelines, and therefore received a variety of treatments throughout the P7T. Figures 2.2 and 2.3 (for Examples (2) and (3)) illustrate the new analysis, inspired mainly by separate transitive and intransitive clefting analyses outlined in (van der Beek, 2003). In particular, in Figure 2.2, the P7T representation (above) shows that (possibly

[4]Dates, time periods and phrases involving adverbs of quantity provide further frequent examples of phrases which always lacked internal structure, and into which I introduced structure.

[5]Sentence 88, file flmf7ag1ep.cat.xml.

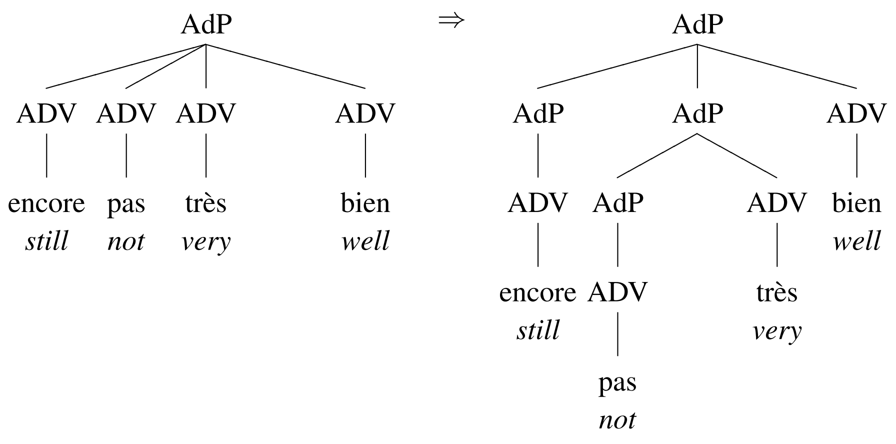


Figure 2.1: P7T representation (left) and MFT representation (right) of Example (1).

due to a lack of analysis being provided in (Abeillé, 2003) for it-cleft constructions) the Srel phrase is analysed incorrectly as a modifier the subject's attribute, whereas in the MFT representation (below) the whole sentential structure is recognised as a transitive it-cleft construction, the attachment of the Srel phrase is corrected and given a path function tag SUJ.MOD. In Figure 2.3, the P7T representation (above) erroneously calls the Ssub phrase a simple modifier, whereas in the MFT representation (below) the sentential structure is recognised as an intransitive it-cleft construction and the Ssub phrase is accordingly given the SUJ function tag.

(2) C'est [...] l'URSS [...] qui se trouve prise [...]
It is [...] the USSR [...] who herself finds taken [...]
'It is the USSR that finds itself trapped'[6]

(3) C'est á ce prix que l'Ukraine peut convaincre sa population de la nécessité
It is at this price that the Ukraine can convince its population of the need
'It is this cost that the Ukraine can convince its population of the need.'[7]

[6]Sentence 8151, file cflmf3_08000_08499ep.xd.cat.xml of the MFT.
[7]Sentence 416, file flmf7ag1ep.cat.xml.

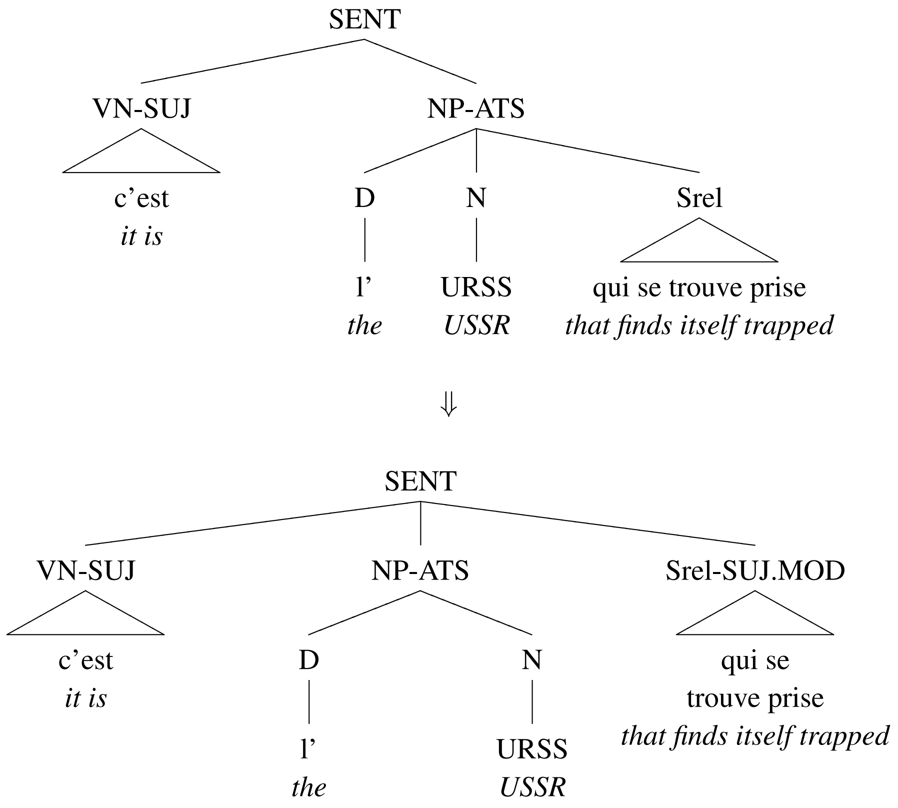


Figure 2.2: P7T representation (above) and MFT representation (below) of Example (2) (transitive clefting).

2.3.3 Information Propagation

Some constituent categories in the P7T derive terminal strings with grammatical patterns not reflected in the intervening levels of syntactic representation. VPinf, VPpart, and Srel are the three categories which were found to have this property. For instance, VPinf requires a VN daughter that has a V daughter which is an infinitive, and Srel requires a PP or NP daughter whose head is or has an argument that has a relative pronoun daughter. The phrase structure trees in the P7T do not capture these requirements. Cases such as these amount to information loss across levels of representation, thereby introducing CFG ambiguity. A parser must guess the daughters of these VN, NP, and PP constituents in order to produce a correct syntactic analysis. This potentially leads to poor statistical parsing. The required information can be automatically propagated, augmenting the MFT with extended

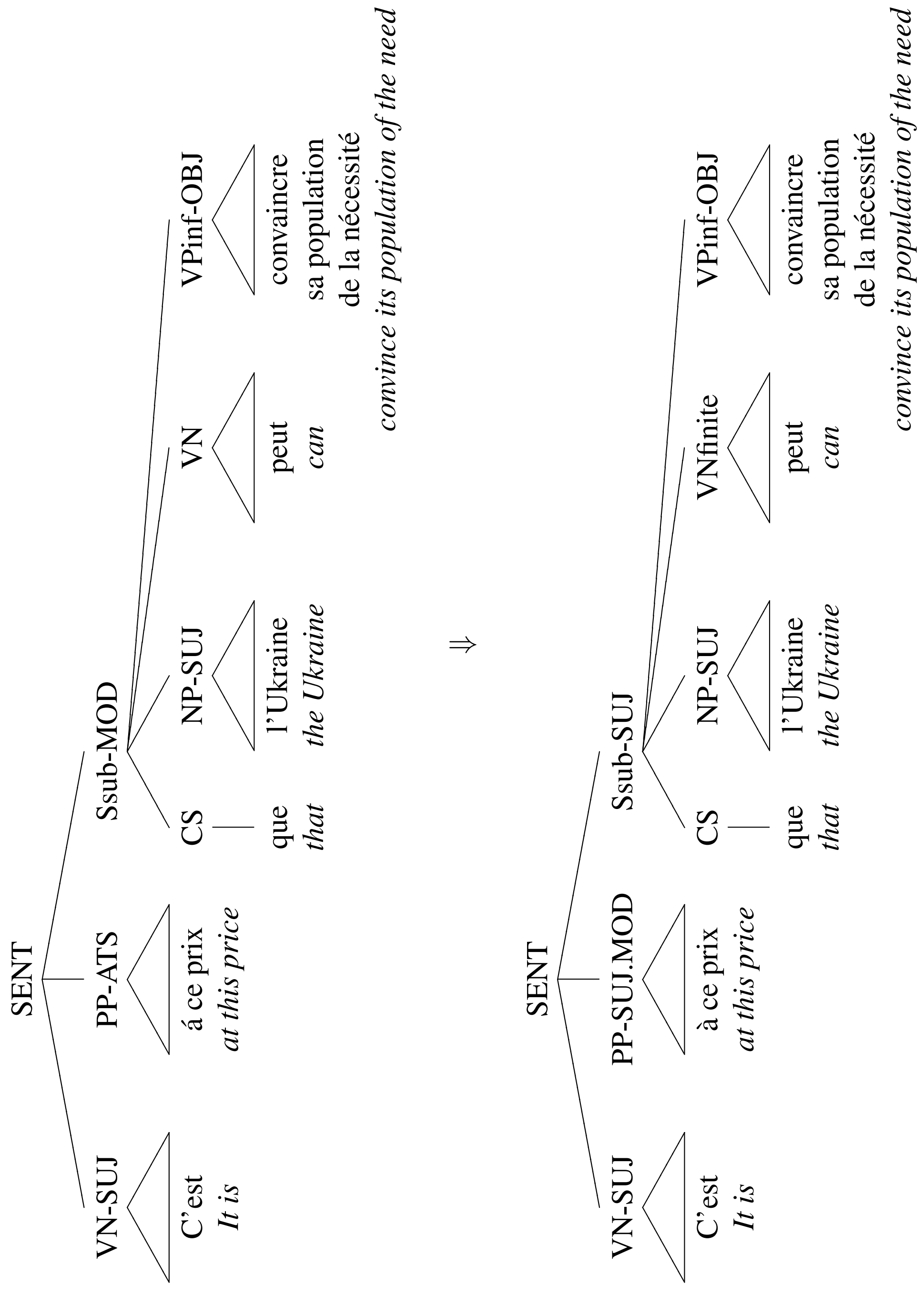


Figure 2.3: P7T representation (left) and MFT representation (right) of Example (3) (intransitive clefting).

constituent labels.[8]

- In the MFT, the part-of-speech V is separated into three different categories: Vfinite, Vinf, and Vpart, according to whether the verb is tensed, infinite, or a participial. The XML representation of the constituent VN for the MFT now has an attribute "type", which records the first verb's grammatical category: finite, inf, or part. The VPinf constituent will now only have a VN constituent with type "inf", and similarly for VPpart.

- Relative pronouns in the P7T are already indicated in the "subcat" attribute. I propagate this information as "type" attributes through the dominating nodes, until the node Srel is reached, thus introducing the constituent categories PPrel and NPrel.

Example (4), whose tree structure is shown in Figure (4), illustrates both these changes.

(4)	[...]	qui	risquait	de	brouiller	l'image	[...]
	[...]	who	was risking	of	shake-up	the image	[...]

'who risked messing up the image'[9]

2.3.4 Raised Coordination

Coordination in the P7T is represented as a sort of adjunction of a COORD phrase as a sister or daughter of the element it is to be coordinated with. This is interpreted in two different ways in the treebank, illustrated in Figure 2.5, making coordinated structures in the P7T highly ambiguous and inconsistent. Either of the two analyses shown are attested

[8]Note that this is similar to the strategy suggested by Johnson (1998), but with two important differences. First, the information propagation is done here in a bottom-up fashion and, therefore, retains the central linguistic motivation behind phrase structure trees, that of constituents making up and determining the type of a phrase. On the other hand, Johnson (1998) suggests a sort of information propagation in a top-down fashion—a sort of after the fact description of a phrase's context within a given tree. Second, I am not carrying out transformations to be undone after some parsing process; I am carrying out a permanent re-annotation of treebank trees. In this latter sense my work is different from that of, for example, Klein and Manning (2003).

[9]Sentence 8009, file flmf3_08000_08499ep.xd.cat.xml.

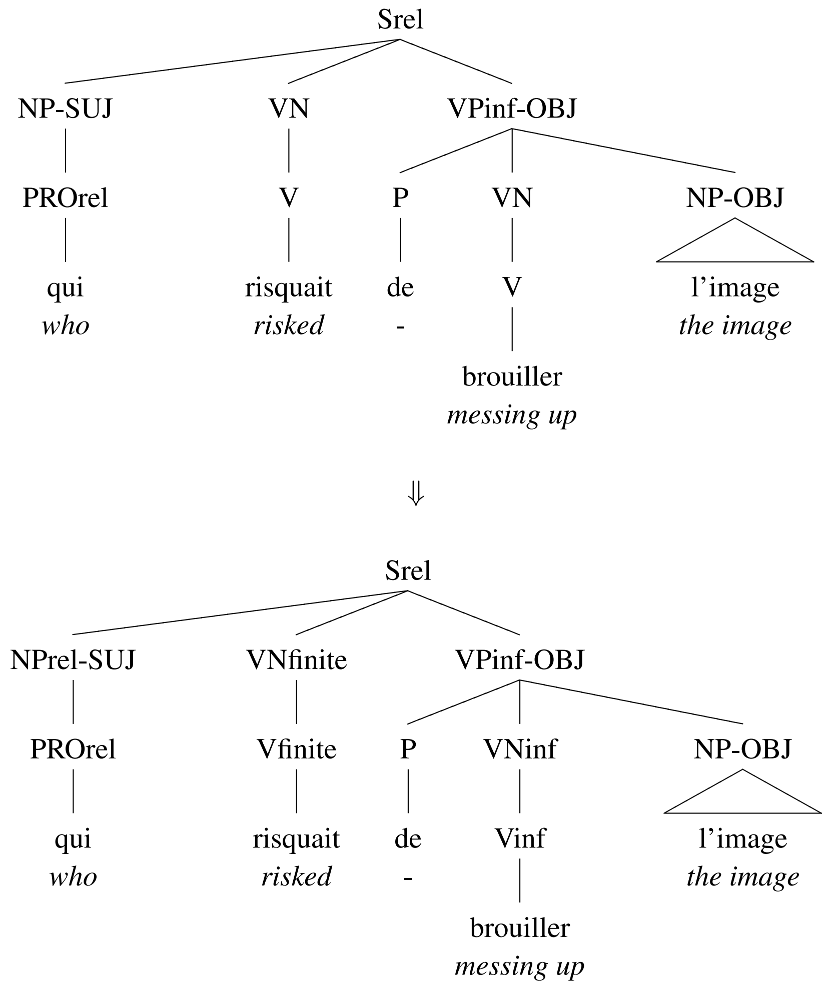


Figure 2.4: P7T (above) and MFT representation (below) of Example (4).

in the P7T, as well as a third, sometimes, for PP coordination.[10,11]

The coordination analysis adopted for the MFT is similar to that of the Penn Treebank (Bies et al., 1995), except for one important fact: I do not get rid of the COORD constituent. Coordination has been modified to be structured as a single phrase consisting of coordinate daughters. This process of restructuring was carried out in a semi-automatic fashion. All

[10]The annotation guidelines of the P7T suggest that there is a difference in distribution; however, upon working with the P7T, one realises that, in fact, this is not the case. It seems that the flatness of analyses in the trees of the P7T combined with their analysis of coordination has resulted in confused structures. Thus, for any type of constituent coordination, both of the structures in Figure 2.5 are attested in the P7T.

[11]I have also found another regularly used form of coordination for PP coordination, where a PP is coordinated with the mother node of its mother node. However, I believe that this is perhaps a consistent error, and not an analysis.

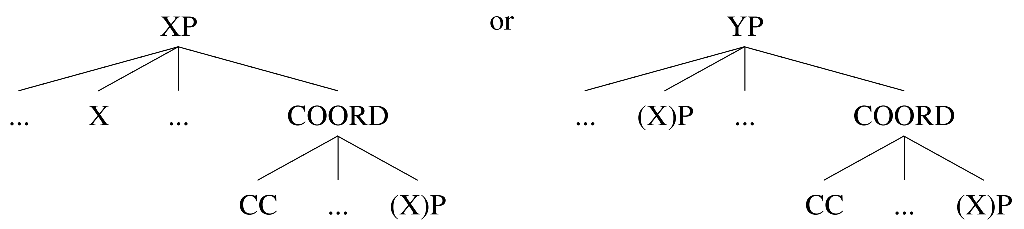


Figure 2.5: Coordination with Mother or with Sister Node in the P7T. The (X)P are coordinated.

sentences had to be hand corrected after automatic transformation, due to the ambiguity in the structures of the P7T. Generally, the goal of the transformation was to arrive at a structure such as the one in Figure 2.6, from those in Figure 2.5 (as well as from any other erroneous coordinated structures encountered).

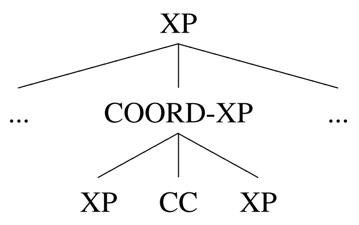


Figure 2.6: MFT coordination with arguments.

For like-constituent coordination, COORD XML elements now have a “type” attribute, whose value is the type of coordinated constituent (i.e., NP, AP, etc.). In Figure 2.6, the COORD phrase is of type XP. In addition, it is enclosed in an XP phrase along with any of its shared arguments or modifiers.

Nonconstituent coordination and unlike constituent coordination required slightly different, but similarly structured, analyses. Unlike constituent coordination was labeled with the type UC, and nonconstituent coordination with the type NC, or VP in the case of an NC that really corresponds to a VP.[12]

COORD-UC phrases may take a functional label if they are sister to a VN, whereas COORD-NC phrases do not. In NC coordination, parallel elements are enclosed in a special

[12]Recall that VP is not a constituent in the P7T, and is not introduced into the MFT, except where NC would correspond to a VP.

NC phrase, if they are not argumentally complete verbal phrases (for example, in argument cluster coordinations). The functional roles of each of their constituents is given on the constituents themselves within the NC or Sint constituent.[13] Figure 2.7 illustrates a type of NC coordination for the following example.

(5) la personalité morale de la Cinq disparaît, et avec elle l'autorisation
the personality moral of the Five disappears, and with her the authorisation
d'émettre
of broadcast
'the moral personality of the Five is disappearing, and with it the permission to broadcast' [14]

2.3.5 Functional Path Tags

Approximately half of the P7T was automatically functionally annotated and hand corrected (Abeillé and Barrier, 2004).[15] In the original subsection of the P7T (before being modified and hand corrected by the present author) the functional tag counts are as given in Table 2.3.

functional tag	**count**
SUJ	8036
OBJ	5949
MOD	6023
A-OBJ	833
DE-OBJ	1354
P-OBJ	913
ATS	560
ATO	104

Table 2.3: Original Functional Tag Counts for the Relevant P7T Subset.

The functional annotation scheme adopted for the P7T assumed that all sisters of the VN phrase are functionally dependent on that phrase. However, this is not always the

[13]In reality, like VN, NC is not really a phrase; rather, it is a convention permitting the expression of parallel structures. I explicitly use the tag "NC" to make this clear.
[14]Sentence 154, file flmfaa1ep.cat.xml.
[15]cf. Section 2.2.

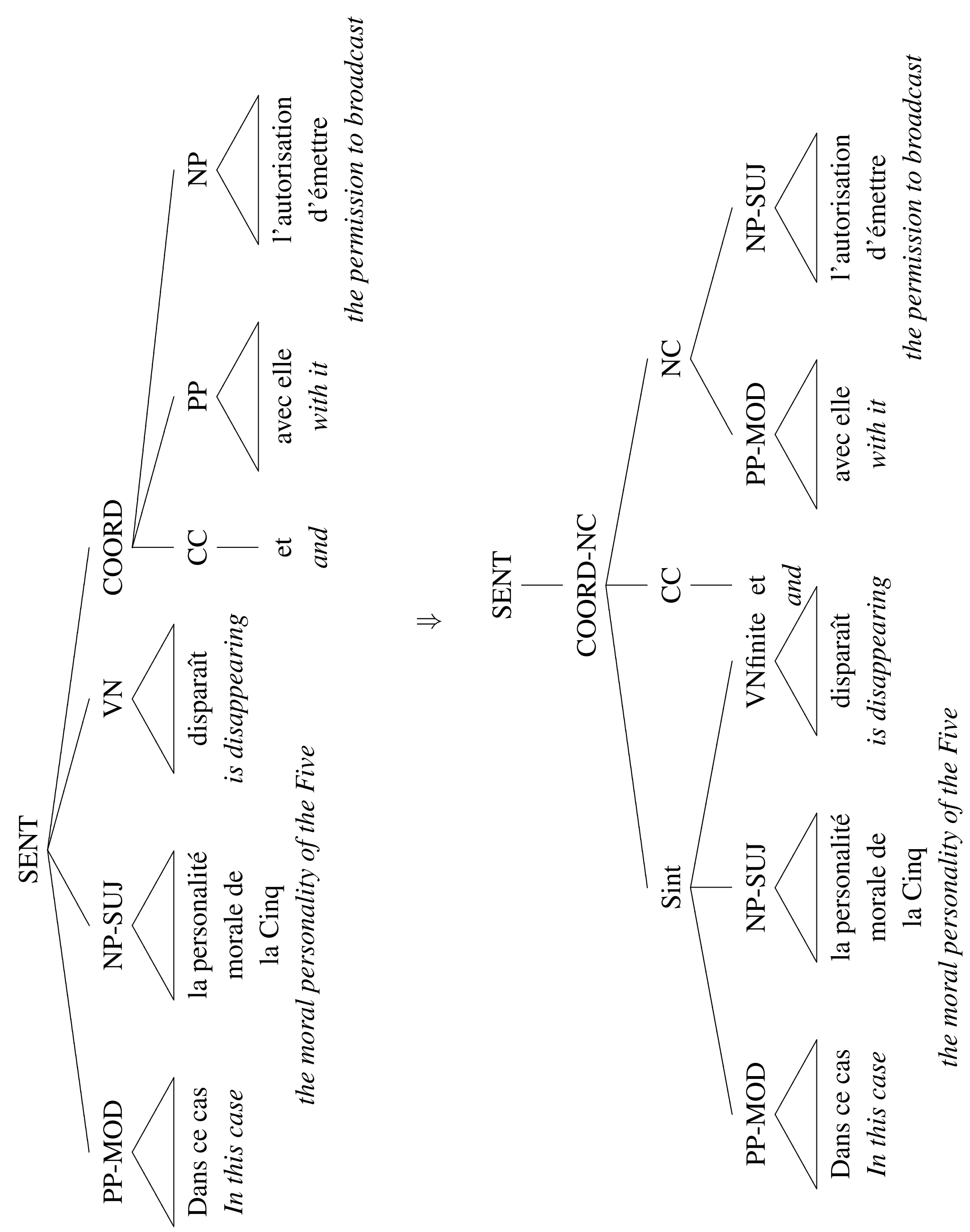


Figure 2.7: P7T (left) and MFT (right) representation of example (5).

case; it-cleft constructions provide a first example (cf. Section 2.3.2). Other cases involve, for example, pronouns for DE prepositional phrases (pronouns such as *dont* or *en*) and daughters of NC. Inspired by the functional paths in the LFG framework,[16] I assign new

[16]See, for example, (Dalrymple, 2001).

path functions, as illustrated in Figure 2.2, where the Srel constituent takes the functional path tag SUJ.MOD, representing the fact that Srel has the function MOD, and is dependent on the constituent whose function is SUJ.

Note, in addition, that much of the functional annotation was missing in the functionally annotated subset of the P7T: only 23,772 functional tags were found in the relevant subsection of the P7T. In contrast, the MFT contains 30,399 functional tags. Table 2.4 presents the MFT counts of the new functional path tags.

functional tag	**count**
SUJ	7969
OBJ	6667
MOD	10615
A-OBJ	1432
DE-OBJ	956
ATS	1470
SUJ.MOD	158
P-OBJ	1022
ATO	126
A-OBJ.OBJ	1
ATS.MOD	14
DE-OBJ.OBJ	1
OBJ.MOD	38
OBJ.DE-OBJ	1
OBJ.OBJ	3
SUJ.A-OBJ	1
DE-OBJ.OBJ.MOD	2
OBJ.A-OBJ	2
SUJ.DE-OBJ	1
A-OBJ.OBJ.MOD	1

Table 2.4: MFT Counts of Functional Path Tags.

2.4 Formatting and Error Mining

In order to be usable by software, and before any restructuring of the P7T could take place, I carried out an extensive clean-up of the original P7T formatting. This involved, for example, reinserting missing part-of-speech tags, and repairing the XML formatting.[17]

Following the reformatting and restructuring of the treebank, a phase of general error mining and correction was undertaken to reduce any noise that I had introduced into the new MFT version of the treebank, and to try to catch any important errors that I had as yet left untreated or that I had missed. Error mining has been shown to improve the results of even very robust techniques for comparatively large corpora (Dickinson and Meurers, 2005, 2003a,b).

This phase has been carried out semi-automatically, in three steps. The first step simply involved automatically extracting a CFG grammar from the treebank, and verifying manually that the productions were consistent with P7T and MFT annotation guidelines, correcting any deviations. The next two steps consisted of applying error-mining software created under the Decca project (Dickinson and Meurers, 2005). This involved applying software for the detection of part-of-speech variations and constituent-string relation variations, examining non-fringe results, and manually correcting any detected erroneous annotations.[18]

2.5 Comparative Statistics

The comparative counts of tokens and types of CFG rules for the relevant subset of sentences, given a certain left-hand side, is presented in Table 2.5.[19] Observe that in all instances (except for AdP[20]), the number of tokens has increased from P7T to MFT, whereas, except for Sint, the number of types has decreased. In addition, I have used a 2-sample χ^2 test for equality to show that all type-token proportions have significantly decreased, as shown in the last column of Table 2.5 (ranging from the largest P-value 2.546E-02 to the

[17]For example, in the whole of the functionally annotated section of the P7T, I found 5 empty SENT constituents, 3 cases of word-forms floating outside of their XML elements, 15 misformatted lemmas, 24 missing parts-of-speech for words not belonging to a multi-word expression, 16,222 missing parts-of-speech for words belonging to a multi-word expression, 18 misused attributes, etc.

[18]For example, Decca POS software detects 28 7-gram variations of which 15 are non-fringe. The non-fringe variations were examined for errors. The same softwares detects only 11 7-gram variations in the MFT, of which 5 are non-fringe.

[19]COORD and VN, and any new constituents added to the MFT are not mentioned for reasons of incomparability. Also note that these rule counts abstract over any punctuation or functional tagging.

[20]Observe that the AdP phrase in the original P7T was comparatively rarely employed.

smallest 2.2E-16). The differences reflect the consistency and comparative simplicity of the MFT with respect to the P7T.

left side	**P7T types/tokens**	**MFT types/tokens**	**p-value**
SENT	1476/4741	1114/4739	2.2E-16
AP	93/5506	64/8440	5.428E-07
AdP	37/290	44/5755	2.2E-16
NP (or NPrel)	1086/34747	690/38036	2.2E-16
PP (or PPrel)	129/19071	60/19930	1.416E-07
VPinf	300/2940	221/3047	6.229E-05
VPpart	249/2009	160/2115	2.838E-07
Srel	302/1567	233/1590	6.475E-04
Ssub	361/1426	284/1513	2.235E-05
Sint	191/597	273/1024	2.546E-02

Table 2.5: Productions of the P7T versus the MFT.

2.6 Parsing Results and Regression Analysis

Arun and Keller (2005) explore the question of the role of lexicalisation in parsing French and report parsing results on the P7T. Post-publication, Arun discovered (personal communication) that the results reported in these publications were erroneously obtained; Arun and Keller (2005) mistakenly discarded over half of the treebank trees, believing that the contracted words were XML errors. Their new results for sentences of length ≤ 40 words were given in their presentation at ACL, and are reported in Table 2.6.[21],[22]

Arun and Keller (2005) present results for BitPar (Schmid, 2004) (a simple PCFG parser), as well as for several modifications made to Bikel's parser (Bikel, 2002) (a lex-

[21]The ACL slides presenting these new results may be obtained at http://homepages.inf.ed.ac.uk/s0343799/acl2005slides.pdf .

[22]In Table 2.6, *perfect tagging mode* means that the parser does not carry out any POS tagging; it is run with all POS tags supplied. Bikel's parser does not automatically run in perfect tagging mode when all POS tags are supplied. It still carries out its own POS tagging, unless the word-forms have not been seen in the training set, in which case it uses the supplied POS tag; so in this case, it is run with *unknown POS tags supplied.*

parser and mode	LR	LP	f-score
BitPar (own POS tagging)	64.49	64.36	64.42
BitPar (perfect tagging mode)	67.78	67.07	67.42
Bikel (own POS tagging)	79.94	79.36	79.65
Bikel (unknown POS tags supplied)	80.79	80.23	80.50

Table 2.6: Arun and Keller's P7T parsing results ($\leq$ 40 words).

icalised statistical parser).[23] What they term as "Collins Model 2" is essentially Bikel's parser without any of the added modifications; results from this model applied to the best of Arun and Keller (2005)'s transformations of the P7T (contracted compounds and raised coordination[24]) will serve as a baseline for comparison with the results presented here.

Upon finding that Bikel's parser outperforms BitPar when trained on the P7T by over 15%, Arun and Keller (2005) concluded that French, like English but unlike German, parses best in a lexicalised statistical parsing framework, leading to the conjecture that word order, and not flatness of annotation, is crucial for lexicalisation. By contrast, parsing results with the MFT lead to a less extreme conclusion, and provide further evidence that a coherent and well-structured treebank leads to better parsing results.

Experiments were repeated on the MFT using both BitPar and Bikel's parser. The MFT was randomly subdivided into a training set (3800 sentences), development set (509 sentences) and a test set (430 sentences).[25] My training set roughly corresponds (in quantity) to only 20.5% of the training data used by Arun and Keller (2005) in their most recent experiments (18,548 sentences), yet my results show improvements on results using the P7T.

[23]BitPar is a PCFG parser that considers most likely parses based on phrase structural (CFG) information only, with associated rule frequencies. Bikel's parser is a implementation of Collin's parsing model (Collins, 1997; Bikel, 2004). Following this model, the parser first carries out a heavy preprocessing step, and then proceeds to carry out training of a lexicalised PCFG based on, in addition, the associated frequencies resulting from a number of smoothing and back-off techniques to compensate for fine-grained data due to lexicalisation.

[24]As in Arun and Keller (2005)'s work, I contracted compounds for all experiments. Their method of raising coordination is completely different from the way coordination is treated in the MFT. See (Arun and Keller, 2005) for details.

[25]This data partition will remain the same for all experiments that I report on in this thesis.

The results for the MFT are shown in Table 2.7.

parser and mode	**LR**	**LP**	**f-score**
BitPar (own POS tagging)	70.66	70.62	70.64
BitPar (perfect tagging mode)	78.07	77.36	77.71
Bikel (own POS tagging)	79.76	80.13	79.95
Bikel (unknown POS tags supplied)	83.09	83.31	83.20
Bikel (perfect tagging mode)	84.62	84.69	84.66

Table 2.7: MFT parsing results (≤ 40 words).

BitPar trained on the MFT outperforms across the board its scores when trained on more than five times the amount of data from the P7T. On sentences of length less than 40 words, BitPar trained on the MFT scores 6.22% (absolute) better, and in perfect tagging mode, BitPar scores 10.29% (absolute) better than when trained on the substantially larger training set from the P7T.

Smaller increases are also achieved for Bikel's parser, when trained on the small training set of the MFT. When Bikel's parser carries out its own POS tagging, it scores 0.3% (absolute) better, and when unknown POS tags are supplied, it performs 2.51% (absolute) better than its counterpart trained on the large training set of the P7T.

Table 2.7 also shows how scores using Bikel's parser increase further, when run in perfect tagging mode.[26] Arun and Keller do not report results for running Bikel in perfect tagging mode.

The variances in the increases of f-scores seem to be the direct results of the parsing mechanisms adopted by each of the parsers. BitPar is less flexible to inconsistent and error-ridden data, than Bikel's parser, which assumes independence relations among sister nodes (with respect to the phrase head), compensating for this with only a distance measurement.

[26]Bikel's parser can be tricked into perfect tagging mode, by appending the part-of-speech to the end of each word-form.

parser and mode	**transform**	R^2	α	**p-value** (α)	β	**p-value** (β)
BitPar (own POS tagging)	$y = \alpha \cdot ln(x) + \beta$	0.9978	6.7206	1.53E-10	14.8734	8.11E-07
BitPar (perfect tagging)	$y = \frac{1}{\alpha \cdot ln(x) + \beta}$	0.9616	-0.0003	3.283E-06	0.0156	1.1472E-11
Bikel (own POS tagging)	$y = \frac{\alpha}{ln(x)} + \beta$	0.9943	-298.6927	4.03E-09	115.4334	2.42E-12
Bikel (unknown POS supplied)	$y = \frac{ln(x)}{\alpha + \beta \cdot ln(x)}$	0.977	0.01551	5.42E-07	0.0102	8.32E-12

Table 2.8: Linear regression on learning curve data from Figure 2.8.

The learning curves in Figure 2.8 present the changes in parser performance trained on increasingly larger subsets of the MFT training set. For this experiment, I also train on the development set to obtain further information about possible increases in parser performance and its possible correlation to training set size.

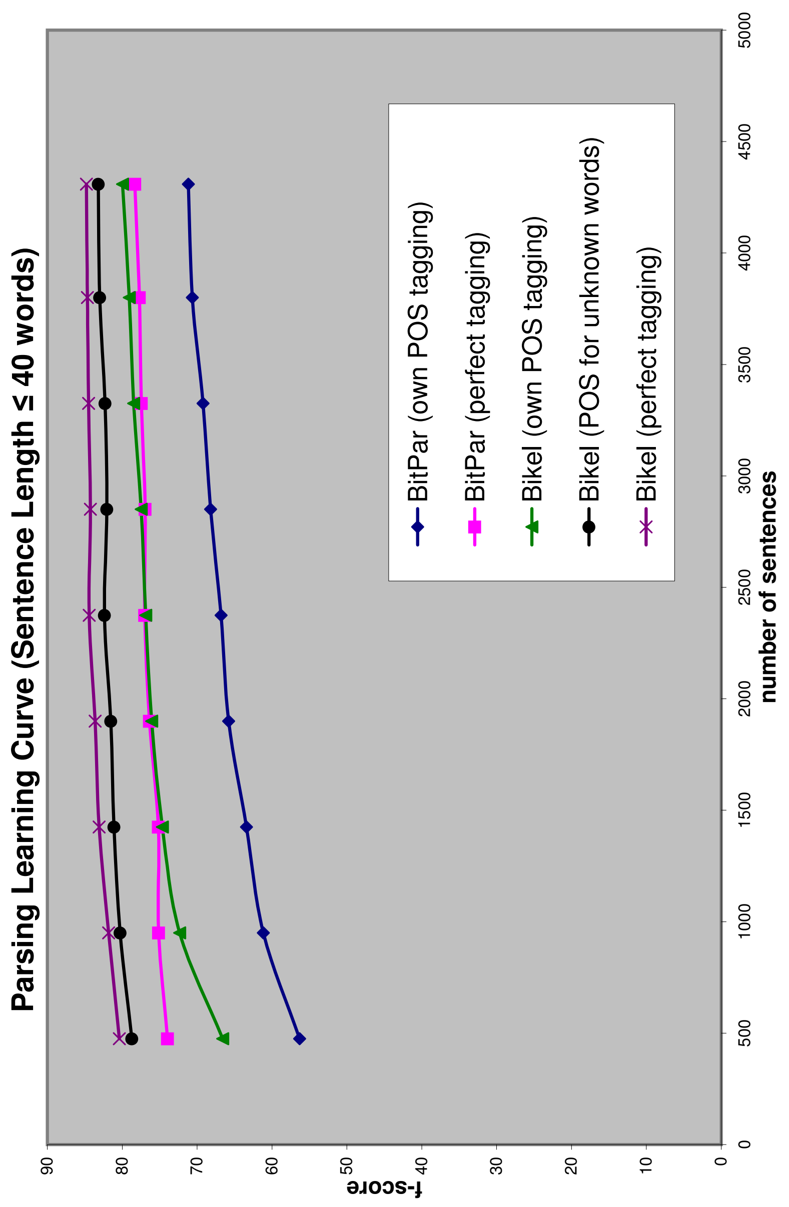


Figure 2.8: Learning curve for the MFT.

Due to the small number of observations, any nonlinear growth curve fitting method would be parsimonious; I therefore applied linear regression analysis. Using four different combinations of power transformations, I found these learning curves to be approximately linear with a very strong positive relationship between transformed number and f-score. Table 2.8 shows the transforms, R^2, parameters, and parameter p-values (using the standard t-test). F-score extrapolation for a training set of size 18,548 (the size of the training set for experiments by Arun and Keller on the P7T) are given in Table 2.9. These predictions show an increase in f-score across the board.[27]

parser and mode	**P7T f-score**	**MFT predicted f-score**
BitPar (own POS tagging)	64.42	75.72
BitPar (perfect tagging)	67.42	81.08
Bikel (own POS tagging)	79.65	82.44
Bikel (unknown POS tags supplied)	80.50	83.99

Table 2.9: F-score and f-score prediction comparison for training set of size 18,548.

The largest increase between P7T parsing scores and predicted MFT parsing scores with a larger training set is for BitPar, whose predicted score is 11.3% higher when doing its own POS tagging, and 13.66% higher in perfect tagging mode. In fact, the performance gap between BitPar and Bikel's parser seems to be steadily closing as MFT training data sizes increase. These results suggest that lexicalisation for statistical parsing of French is perhaps not as crucial as was concluded by Arun and Keller (2005).

2.6.1 What is *not* concluded here?

Some authors (for example, Rehbein and van Genabith (2007); Kűbler (2005)) argue that parsing results for treebanks with different annotation schemes are not comparable. There

[27]Significance tests are not applicable.

is also concern when comparing parsing results of treebanks having different test sets (even though the MFT is a subset of the P7T). However, these concerns remain with respect to my own argumentation. I do *not* conclude that our parsing results are necessarily *better* than Arun and Keller (2005)'s—only that their conclusion about the critical role of lexicalisation for the statistical parsing of French may be erroneous.

2.7 More Recent Work on Parsing French

Since this research was carried out and first published as (Schluter and van Genabith, 2007), the P7T has undergone some changes, the most important of which seems to be a discarding slightly less than half of the treebank trees; for example, Candito and Crabbé (2009) report the P7T to contain only 12531 sentences. However, there has been no account of the syntactic or other structural changes carried out in the static treebank, if any have taken place.[28]

Recently, parsing experiments experiments with this new and reduced P7T treebank have been carried out, by Candito and Crabbé (2009), using the Berkeley Parser (Petrov and Klein, 2007) and word clustering with the (Brown et al., 1992) hard clustering algorithm, both demonstrating the usability of the new version of the P7T and giving encouraging results for this new form of the treebank, using this parsing method.

Seddah et al. (2009) consider the differences between the MFT and the new version of the P7T in their comprehensive study on the influence of tag set on a number of parsing models for French. In this work they show further signs of success in parsing French with the Berkeley Parser, trained on the new version of the P7T enhanced by tag set transformations inspired by those carried out for the MFT (in Section 2.3.3).

[28]Here we are referring to changes within the *static* treebank and not to any preprocessing for specific parsing experiments.

2.8 Concluding Remarks

In this chapter, I have presented the Modified French Treebank, a new French Treebank, derived from the P7T, which is cleaner, more coherent, has several transformed structures, and introduces new linguistic analyses. The positive effect of transformations on and cleaning up treebanks is well documented (for example, by Dickinson and Meurers (2005)). I investigated one important effect of a clean treebank on corpus-based linguistics. The MFT provides a strong example of how quantity does not always make up for quality in statistical parsing. A probabilistic parser trained on clean and transformed data performs better than its counterpart trained on the original French treebank, which consists of five times the data. Moreover, I have shown how data which has a high error rate and that is not "parser-friendly" can lead to the potentially erroneous conclusions about the impact of lexicalisation on probabilistic parsing of French.

Apart from the research on automatically detecting inconsistencies in treebank annotations by Dickinson and Meurers (2003a,b, 2005), there has been very little research on restructuring and correcting treebank resources: an exception is the work of Hockenmaier and Steedman (2007) who describe the substantial clean-up and re-analysis of of Penn Treebank structures as a prerequisite to their automatically deriving the CCGbank.

The MFT will serve as our new data source for the LFG acquisition and parsing research reported in the remainder of this thesis.

Chapter 3

An F-Structure Annotation Algorithm and Dependency Gold Standard for French

In this chapter, I discuss the design and implementation of an f-structure annotation algorithm for the MFT, based on and substantially adapting and extending earlier work on English by Burke (2006), McCarthy (2003), and Sadler et al. (2000) and complemented by syntactic analyses for French outlined especially by Frank and Berman (1996) (as well as Butt et al. (1999) and Dalrymple (2001) for the general perspective), within the framework of LFG.[1] This is combined with the implementation of a simple coordination distribution algorithm, based on the LFG analyses for coordination as presented by, for example, Dalrymple (2001), which is a novel and significantly beneficial addition to the treebank-based LFG parsing architecture laid out by Cahill et al. (2005, 2008). I also discuss the final steps (namely, hand verification and completion) in the construction of the MFT Dependency Bank (MFTDB), a 430 sentence f-structure gold standard to be used for evaluation of all deep parsing work reported in this thesis.

[1]Note that the research reported in this thesis makes no arguments for one linguistic analysis or another. It simply models itself after work by the DCULFG compensating for lacunae in syntactic analysis for French by consultation of Frank and Berman (1996).

3.1 Introduction

Automatically acquired deep-grammar parsing resources require a treebank with deep grammatical annotations. If no such 'deep' treebank exists, it may be possible to automatically derive such a resource from existing treebank resources; in an LFG-based approach, this is the role of the f-structure annotation algorithm. An f-structure annotation algorithm simply derives an f-structure bank from a simple CFG style treebank. It takes as input a c-structure and outputs an f-structure annotated c-structure from which an f-structure bank can be derived using a constraint solver. The work presented in this chapter provides such an f-structure annotation algorithm for French.

Moreover, (statistical) dependency parsing and automatic dependency derivation from treebanks require evaluation against established gold standard resources, providing a benchmark for resource quality and allowing direct comparisons to be made among outputs from differing grammar development paradigms and statistical methods. Gold standards for languages such as English, (for example, the PARC700 Dependency Bank (King et al., 2003) and the DCU 105 (Cahill et al., 2002b), German (for example, TiGer dependency bank (Brants et al., 2002)), and Arabic (for example, the DCU 250 (Al-Raheb et al., 2006)) have been developed and used for evaluation of dependency parsing and automatic grammar and lexicon extraction architectures. However, to my knowledge, the construction of the MFT Dependency Bank (MFTDB), reported in this chapter, produced the first dependency gold standard resource available for French.[2]

I briefly outline the original annotation algorithm developed for English (Section 3.2). The new annotation algorithm for French and a selection of important novel linguistic analyses[3] it implements is presented in Section 3.3. I then discuss the construction of the dependency gold standard (Section 3.4) and present an evaluation of the f-structure annotation algorithm against this gold standard (Section 3.5).

[2]Bick (2004), for example, is forced to measure the precision and recall of his rule-based French dependency parser by hand, and necessarily takes only a small chunk (1790 words) of the Europarl corpus to do so.

[3]The linguistic analyses are "novel" for this LFG parsing approach, not for linguistic analysis.

3.2 An F-Structure Annotation Algorithm for English

The Annotation Algorithm for English Penn-II treebank-style CFG representations (McCarthy, 2003; Burke, 2006) is composed of five separate modules, as shown in Figure 3.1. The first module consists of left-right context annotations: the head of a phrase is detected and annotated (Head-Lexicalisation Module), then the sister nodes of the head are annotated depending on whether they occur in the right context of the head or in the left context (Left-Right Context Annotation Principles Module). As the representation of coordination in the Penn-II treebank is highly ambiguous, a separate module is provided to keep the Left-Right Context Annotation Principles Module simple and perspicuous; this makes up the second module (Coordination Annotation Principles Module). The third module covers the annotation of trace elements, including the treatment for such linguistic phenomena as passivisation, topicalisation, wh-questions and relative clauses (Traces Module). The fourth module performs a catch-all and clean-up, attempting to correct errors and overgeneralisations caused by the three previous modules (Catch-All and Clean-Up Module).

3.3 The Annotation Algorithm for French

There are several differences in the architecture of the annotation algorithm for French and the one for English. In this section I motivate and outline these differences.

In contrast to English and the Penn-II treebank representations, French and the MFT are rich in morphological information, as inherited from the P7T. In addition to this, because of the extension, completion and verification of the MFT function tag annotation, our algorithm relies less on f-equation decision heuristics than on simple translation of functional information already present in MFT trees. This supports the construction of simple *lexical macros* (Section 3.3.1), as well as the *LFG Conversion Module* (Section 3.3.2). For this reason, and in particular for phrases with a verb kernel, rather than relying on annotation approximations by means of a left-right context annotation module, the LFG Conversion Module is used to simply translate existent function tags into LFG functional equations.

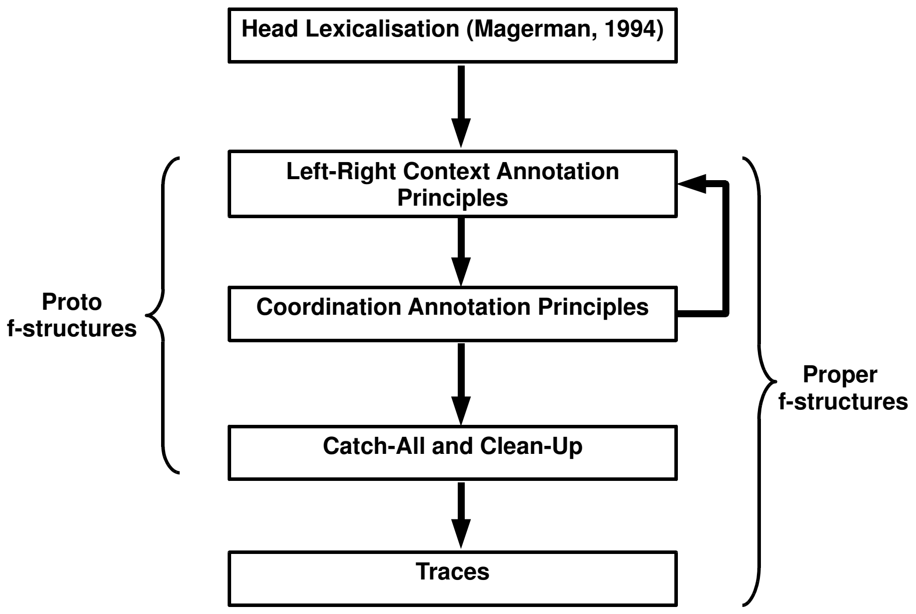

Head Lexicalisation (Magerman, 1994)
Left-Right Context Annotation Principles
Coordination Annotation Principles
Catch-All and Clean-Up
Traces
Proto f-structures
Proper f-structures

Figure 3.1: Annotation algorithm for English.

Ambiguities implicit in the original P7T treebank have been reduced considerably in the MFT. In particular, there is no ambiguity between coordinated structures and modification of these coordinated structures in the representation of coordination in the MFT. Therefore, there is no need for a special coordination module.

In addition, to reflect the current linguistic analyses of verbs adopted in LFG (for example, Frank and Berman, 1996; Butt et al., 1999; Dalrymple, 2001), I provide a monoclausal treatment of compound verbs, resulting in the introduction of a *Verb Combinatorics Module* (Section 3.3.3).

Finally, the MFT inherits from the P7T the absence of traces and empty productions. In the Penn-II treebank representations, most traces and empty productions resolve long-distance dependencies. These are accounted for by the path function tags of the MFT; it is straightforward that communicative re-entrancies (such as `topic` or `focus`) are coannotated *in situ* in terms of corresponding f-structure reentrancies in the LFG conversion module.[4] Other traces and empty productions may indicate passivisation, which is accounted for in the annotation algorithm for French in the Verb Combinatorics Module.

For other constituents such as NP or PP, I maintain the *left-right context* annotation principles in the corresponding annotation module (Section 3.3.4).

A *catch-all and clean-up module* is virtually non-existent, annotating only the sentence type of the outermost f-structure (declarative, interrogative, etc.) (Section 3.3.5).

To boost the performance of the f-structure annotation algorithm, I introduce a post-processing step of argument *distribution over coordination*, which significantly improves the performance of the annotation algorithm (Section 3.3.6).

The goal of the f-structure annotation algorithm is to comprehensively annotate all phrasal and lexical nodes of all the trees of the MFT, augmenting its tag and categorical information with LFG f-structure annotations, in order to derive the dependency representation given by the attribute value matrix representing an f-structure. Given a tree, the annotation algorithm iterates over its list of n non-terminal nodes, T, which are in some order that maintains the dominance relation (root node downwards). For each non-terminal node $T(i)(1 \leq i \leq n)$, $T(i)$s daughters are annotated as follows. If $T(i)$ is a verb phrase, send $T(i)$ to the Verb Combinatorics Module. Otherwise, if $T(i)$'s head is a verb phrase, send $T(i)$ to the LFG Conversion Module. Otherwise, send $T(i)$ to the Left-Right Context Annotation Module. For the annotation of any leaf node, directly use the lexical macros. Once the list has been exhausted, it is sent to the Catch-All and Clean-Up Module.[5] The functional equations are then extracted from the trees, and sent to post-processing for distribution over coordination. Figure 3.2 shows the basic structure of the flow of control in the annotation algorithm.

[4]Other approaches to long distance dependency resolution are tested and reported in Chapter 4.

[5]There is no ambiguity in annotations for coordination in the MFT, unlike the Penn-II treebank. For this reason, there is no need for a special Coordination Module of the annotation algorithm as was adopted for the English version (Cahill et al., 2004).

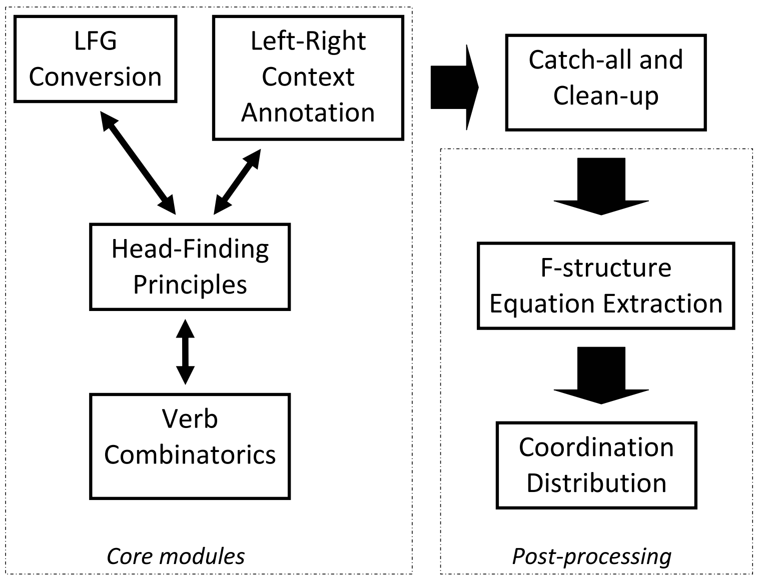


Figure 3.2: Three Modules of the Annotation Algorithm for French.

A Note on Clitics. The MFT inherited a relatively flat phrase-structure annotation from the P7T, including a lexicalised treatment for clitic pronouns. As such, clitic pronouns are not the head of any constituent, but are enclosed in the verb kernel (VN) together with the verb lemma to which they are attached. Also, the function label for clitics is on the VN in which they are enclosed. F-structure equations may not be labeled in this manner in LFG; the predicate node cannot also be the value of the subject. As a preprocessing step, I have enclosed clitics in their own constituents (CLP[6]) and propagated the function labels down to this level from VN. Figure 3.3 shows a transformed main VN subtree derived from the corresponding MFT subtree.

This is a preprocessing step and not an actual change in the MFT because clitics, linguistically speaking, should not have their own constituent.

[6]In reality, I have added two different constituents: CLseP for all reflexive clitics, and CLP for the rest. For ease in description, I will only refer to CLP here, even though there are actually both CLP and CLseP.

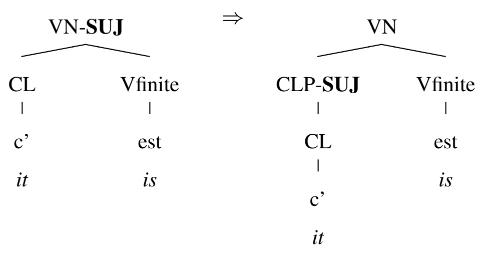


Figure 3.3: Clitic transformation as a preprocessing step for the MFT. The subtree on the left is replaced by the subtree on the right.

3.3.1 Lexical Macros

The annotation of terminal nodes is greatly facilitated by the richness in morphological and lemma information provided by the MFT, which is directly inherited from the P7T. Morphological information is directly translated into lexical features, and lemma information provides predicate values for f-structure annotation. Table 3.1 provides simplified examples of some noun annotation macros.

MFT XML encodings	**LFG function equations**
mph ="(G)N"	↑-`gend`= G ↑-`num`= N
$lemma$ ="l"	↑-`pred`= l
$subcat$ ="s"	↑-`n-type`= s

Table 3.1: Noun lexical macros.

3.3.2 MFT LFG Conversion Module

MFT functional tags are directly translated into LFG functional equations for the constituent under consideration. Table 3.2 gives the simplified macros adopted for the f-structure annotation algorithm.

Note, in particular, the translation of the function path tags, which permit the creation of (LDD-resolved) proper f-structures, rather than only proto-f-structures, unlike the algorithm designed for the Penn-II treebank (Cahill et al., 2004).

functional tag	function equation
SUJ	$\uparrow$subj=$\downarrow$
OBJ	$\uparrow$obj=$\downarrow$
MOD	$\downarrow\in\uparrow$adjunct
A-OBJ	$\uparrow$a_obj=$\downarrow$
DE-OBJ	$\uparrow$de_obj=$\downarrow$
ATS	$\uparrow$xcomp=$\downarrow$,$\uparrow$subj=$\downarrow$subj
SUJ.MOD	$\downarrow\in\uparrow$subj:adjunct
P-OBJ	$\uparrow$obl=$\downarrow$
ATO	$\uparrow$xcomp=$\downarrow$,$\uparrow$obj=$\downarrow$subj
A-OBJ.OBJ	$\uparrow$xcomp:obj=$\downarrow$
ATS.MOD	$\downarrow\in\uparrow$xcomp:adjunct
DE-OBJ.OBJ	$\uparrow$xcomp:obj=$\downarrow$
OBJ.MOD	$\downarrow\in\uparrow$obj:adjunct
OBJ.DE-OBJ	$\uparrow$xcomp:de_obj=$\downarrow$
OBJ.OBJ	$\uparrow$xcomp:obj=$\downarrow$
SUJ.A-OBJ	$\uparrow$subj:a_obj=$\downarrow$
DE-OBJ.OBJ.MOD	$\downarrow\in\uparrow$xcomp:obj:adjunct
OBJ.A-OBJ	$\uparrow$xcomp:a_obj=$\downarrow$
SUJ.DE-OBJ	$\uparrow$subj:de_obj=$\downarrow$
A-OBJ.OBJ.MOD	$\uparrow$a_obj:obj:rel_mod=$\downarrow$

Table 3.2: MFT function tag macros.

3.3.3 Verb Combinatorics Module

To render the f-structures more similar to those output by the current XLE systems (Butt et al., 1999), in my f-structure analyses for the French annotation algorithm, I have made the move away from the multi-clausal treatment of compound tenses (as implemented for the English annotation algorithm of Cahill et al. (2008)), elaborating a verb combinatorics module for the mono-clausal analysis.

The verb combinatorics module involves the exhaustive provision of annotations for the (finite) number of composed tenses with and without coordinated verbal parts.[7] The

[7]Frank (2000) proposed and tested a method for automatically inducing LFG f-structures from treebank tree representations based on CFG subtree f-structure annotations rather than CFG rule f-structure annotations. This is similar to the approach taken for the Verb Combinatorics Module here, except in one important respect. The Verb Combinatorics Module annotations are exhaustive: it is possible to enumerate all verb tenses in French, and with them the verb word-forms forming these tenses. This is what we have done here. In this sense, the Verb Combinatorics Module is *more* than data induced, unlike the method proposed by Frank (2000), and therefore does not suffer the same consequences of loss of robustness of the f-structure annotation algorithm.

analysis may be carried out over several levels of phrase-structure representation, depending on the presence of coordinated tenses. A verb in a compound tense is said to be made up of a verb complex, which may include some tensed component, an auxiliary (which may coincide with the tensed component), one or more past participles, and/or an infinitive. The features associated with a verb complex at the f-structure level of representation are the following:

- **`factive=+/-`**: French factive constructions, *faire* + infinitive, are essentially similar to causative constructions in English. The MFT (like the P7T) treats factive phrases as mono-clausal. This seems to be the more accepted linguistic analysis (Yates, 2002). The following sentences, adapted from Yates (2002), illustrate the factive construction.

  (1) Pierre fait courir Paul.
  Pierre is making [to] run Paul.
  '*Pierre is making Paul run.*'

  (2) Pierre lui fait écrire une lettre.
  Pierre to him is making [to] write a letter.
  '*Pierre is making him write a letter. / Pierre is having a letter written to him.*'

  (3) Pierre fait téléphoner Marie à Paul.
  Pierre is making [to] phone Marie to Paul.
  '*Pierre is making Marie phone Paul.*'

  Though the mono-clausal analysis at the f-structural level is not universally accepted, it also seems to be the most accepted one (see, for example, Alsina, 1992; Zaenen and Dalrymple, 1996). Therefore, I adopt this analysis also. For the sentence in Example (1), the outermost predicate is *courir* ('[to] run'), which has the attribute, `factive=+`. Note that the semi-auxiliary verb *faire* ('[to] make'), with any auxiliary associated with it, holds all the morphological information of the verb complex. Basic f-structures for Examples (1) through (3) are shown in Figures 3.4 through 3.6.

- **`passive=+/-`**: The passive voice is normally indicated, as in English, by the compound form of the verb complex, and the introduction of the verb *être* ('[to] be'), along with any

$$\begin{bmatrix} \texttt{pred} & \text{‘courir’} \\ \texttt{subj} & \begin{bmatrix} \texttt{pred} & \text{‘Pierre’} \end{bmatrix} \\ \texttt{obj} & \begin{bmatrix} \texttt{pred} & \text{‘Paul’} \end{bmatrix} \\ \texttt{factive} & + \end{bmatrix}$$

Figure 3.4: Basic f-structure for Example (1).

$$\begin{bmatrix} \texttt{pred} & \text{‘écrire’} \\ \texttt{subj} & \begin{bmatrix} \texttt{pred} & \text{‘Pierre’} \end{bmatrix} \\ \texttt{obj} & \begin{bmatrix} \texttt{pred} & \text{‘lettre’} \\ \texttt{spec} & \begin{bmatrix} \texttt{det} & \text{‘un’} \end{bmatrix} \end{bmatrix} \\ \texttt{aobj} & \begin{bmatrix} \texttt{pred} & \text{‘pro’} \end{bmatrix} \\ \texttt{factive} & + \end{bmatrix}$$

Figure 3.5: Basic f-structure for Example (2). (No disambiguation has taken place at the f-structure level.)

$$\begin{bmatrix} \texttt{pred} & \text{‘téléphoner’} \\ \texttt{subj} & \begin{bmatrix} \texttt{pred} & \text{‘Pierre’} \end{bmatrix} \\ \texttt{obj} & \begin{bmatrix} \texttt{pred} & \text{‘Marie’} \end{bmatrix} \\ \texttt{aobj} & \begin{bmatrix} \texttt{pred} & \text{‘Paul’} \end{bmatrix} \\ \texttt{factive} & + \end{bmatrix}$$

Figure 3.6: Basic f-structure for Example (3).

associated auxiliary, for the morphological expression of tense, and mood.[8]

- **`aux_select=v`**: The auxiliary in a compound verb complex is indicated through the `aux_select` feature, where v is the auxiliary in question (either *avoir* or *être*).
- **`tense=t`**, **`mood=m`**: Tense and mood are indicated according to the traditional tenses of French, found, for example, in any complete conjugation reference for French.

[8]Other manners of expression for the passive voice are not annotated as such in this version of the annotation software, since they are not only rare in the data, but also ambiguous in form with pronominal verbs and therefore difficult to catch. Also, pronominality is indicated on the relevant clitic pronoun and, therefore, does not come into play in the verb combinatorics module.

- **`perf=+/-`**, **`superperf=+/-`**, **`inf=+/-`**, **`part=+/-`** : The aspectual attribute of perfectivity is recorded in a simplistic manner, in keeping with the mainstream LFG treatment of aspect as based on English verbal morphology. A tense that is absent in English, but present in French is then the *superperfect*, composed of the same elements as for the perfect tenses, but where the auxiliary is itself in a perfect tense.[9] The attribute `inf` indicates whether or not the verbal form is an infinitive and the attribute `part` indicates whether it is a participial.

In Sections 3.3.3.1 through 3.3.3.4, I give a description of the various possible annotations for the possible cases, defined in Table 3.3.[10]

Case	**Subcase**	**Figures**	**Description**
1			*VNfinite* is a sister of *VPpart-OBJ-cmp*
	i	3.7	• *VNfinite* has 1 *V* daughter (Vpart)
	ii	3.8	• *VNfinite* has 2 *V* daughters
	iii	3.9	• *VNfinite* has 3 *V* daughters
			VNfinite is a sister of *VPinf-OBJ-cmp* (composed factive)
	iv	3.10	• *VNfinite* has 2 *V* daughters
	v	3.11	• *VNfinite* has 3 *V* daughters
2		3.13	lone *VN*
	i		• *VN* has 1 daughter
	ii		• *VN* has 2 daughters
	iii		• *VNfinite* has 3 daughters
	iv		• *VNfinite* has 4 daughters
3			*VNfinite/VNpart/VNinf* with COORD-VNfinite/COORD-VNpart/COORD-VNinf daughter
	i	3.14	• *VN* has 1 *V* daughter
	ii	3.15	• *VNfinite* has 2 *V* daughters
	iii	3.16	• *VNfinite* has 3 *V* daughters
4	i	3.7, 3.8, 3.9, 3.10, 3.11	*VNpart/VNinf* daughter of *VPpart-cmp/VPinf-cmp*
	ii	3.14, 3.15, 3.16	*VNpart/VNinf* daughter of *COORD-VNpart/COORD-VNinf*, in *VN*

Table 3.3: Legend for cases of verbal complexes defined for treatment in the verb combinatorics module.

[9]For example, *il a eu aimé* (literally, 'he has had loved').

[10]When making an abstraction over the type of verbal element (*Vfinite*, *Vpart*, or *Vinf*), I simply refer to *V* in this chapter. Similarly for *VN* (*VNfinite*, *VNpart*, *VNinf*) and *VP* (*VPpart*, *VPinf*).

3.3.3.1 Case 1

In this section, I present Case 1, where there is no coordination in the *VNfinite* phrase.

Subcase 1i The presence of the sister phrase *VPpart-OBJ-cmp* (*VPinf-OBJ-cmp*) implies that the verb complex has a coordinated structure with past participials (infinitives). For coordination in verb complexes, I take the first conjunct as representative of grammatical information of all conjuncts. More complex coordination cannot be represented (and is probably awkward or agrammatical anyways). Therefore, I consider all auxiliary and semi-auxiliary material to be collected in the *VNfinite* phrase.

Subcase 1i, where VNfinite phrases have only a single *V* daughter, is illustrated in Figure 3.7. For this subcase, in the annotation algorithm, first the value for passivity and the auxiliary are detected (since there is only one *V* daughter of *VNfinite*, one knows that there is no possibility for factivity). Then a certain combination of morpholgical forms are translated by the macros given in Table 3.5, to obtain the tense and mood values. The decision tree algorithm for this subset of annotations is given in Algorithm 1,[11] where the verb variables v_1, v_2, v_3 reflect assignments in Figure 3.7, and the morphology variables mph_1, mph_2, mph_3 are the respective morphological encodings for verbs v_1, v_2, and v_3.[12] The function **translate**($[mph_1 \cdots mph_n]$) takes a combination (ordered set) of morphological encodings and returns the grammatical (atomic) feature/value set for the verb complex, according to the macros given in Tables 3.4 and 3.5. The function **isMovementVerb**(v) returns `true` if v can only be conjugated with auxiliary *être* and `false` otherwise; this is the case for special intransitive verbs.

Subcase 1ii. Subcase 1ii, where VNfinite phrases have two *V* daughters, is illustrated in Figure 3.8. Note that factivity is impossible for this subcase. Also, the auxiliary is given by the second *V*. This case is annotated using Algorithm 2.

[11] Note that, though this algorithm tests for factivity, for this case, this test will always result in `false`.

[12] $mph_i \in \{P, K, I, J, F, S, T, S, C, W, G\}$, the set of possible morphological encodings of a verb component. For example, K encodes the fact that the verb component is a past participle. See Table 3.4.

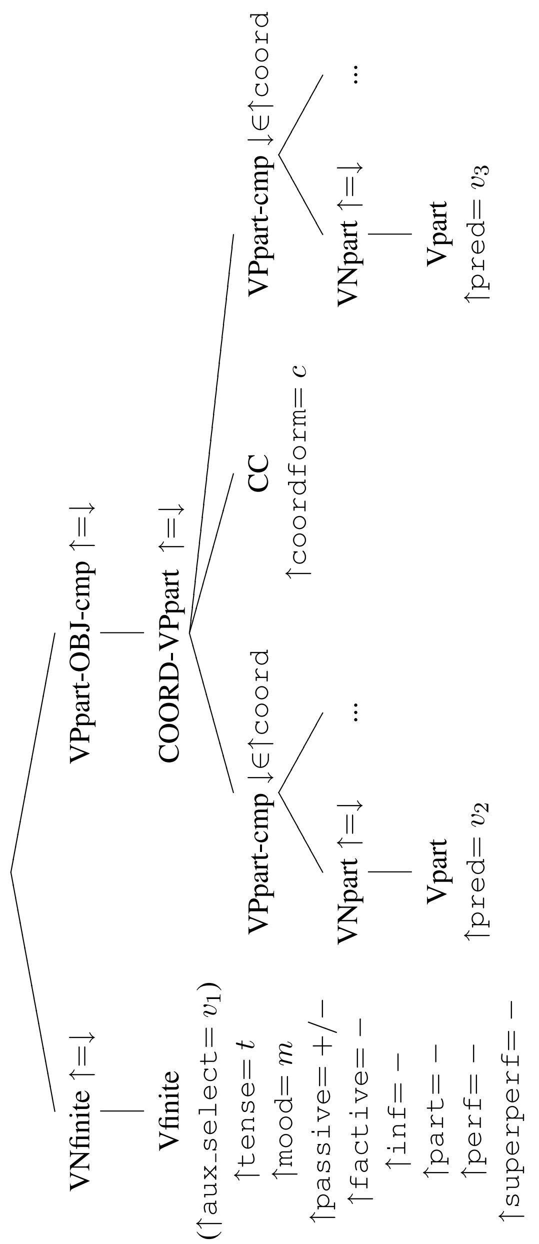


Figure 3.7: Template for Subcase 1i verb complex annotations.

Subcase 1iii. Subcase 1iii, where *VNfinite* has three *V* daughters, is the single case of the passive superperfect, annotated as in Figure 3.9 and Algorithm 3.

Algorithm 1 Decision-tree algorithm for two *V* verb combinatorics.

```
if $v_1$ = faire then
  $\uparrow$factive= $+$, $\uparrow$passive= $-$, translate([$mph_1$])
else
  if $v_1$ = avoir then
    $\uparrow$ factive = $-$, $\uparrow$ passive = $+$, $\uparrow$ aux_select =avoir,
    translate([$mph_1$ $mph_2$])
  else
    if isMovementVerb($v_1$) then
      $\uparrow$factive= $-$, $\uparrow$passive= $-$, $\uparrow$aux_select=être,
      translate([$mph_1$ $mph_2$])
    else
      $\uparrow$factive= $-$, $\uparrow$passive= $+$, $\uparrow$aux_select=être, translate([$mph_1$])
    end if
  end if
end if
```

Algorithm 2 Decision-tree algorithm for three *V* verb combinatorics.

```
if $v_2$ = faire then
  $\uparrow$factive= $+$, $\uparrow$passive= $-$, translate([$mph_1$ $mph_2$])
else
  if $v_2$ = avoir then
    $\uparrow$factive= $-$, $\uparrow$passive= $-$, $\uparrow$aux_select=avoir,
    translate([$mph_1$ $mph_2$ $mph_3$])
  else
    if isMovementVerb($v_2$) then
      $\uparrow$factive= $-$, $\uparrow$passive= $-$, $\uparrow$aux_select=être,
      translate([$mph_1$ $mph_2$ $mph_3$])
    else
      $\uparrow$factive= $-$, $\uparrow$passive= $+$, translate([$mph_1$ $mph_2$])
    end if
  end if
end if
```

Algorithm 3 Decision-tree algorithm for four *V* verb combinatorics.

```
if $v_3$ = faire then
  $\uparrow$factive= $+$, $\uparrow$passive= $-$, translate([$mph_1$ $mph_2$ $mph_3$])
else
  $\uparrow$factive= $-$, $\uparrow$passive= $+$, translate([$mph_1$ $mph_2$ $mph_3$])
end if
```

Subcases 1iv and 1v. These subcases describe only the factive compound tenses, where there is more than one *V* daughter of the *VNfinite* phrase (Figures 3.10 and 3.11).[13] Note

[13]Recall that factivity is represented monoclausally in the MFT.

TRADITIONAL TENSE NAME	**MORPHOLOGICAL ENCODING**	**ANNOTATION**
présent indicatif	P	`perf`=- `tense`=pres `inf`=- `part`=- `mood`=indicative
imparfait	I	`perf`=- `tense`=past `inf`=- `part`=- `mood`=indicative
passé simple	J	`perf`=- `tense`=passesimple `inf`=- `part`=- `mood`=indicative
futur simple	F	`perf`=- `tense`=fut `inf`=- `part`=- `mood`=indicative
présent subjonctif	S	`perf`=- `tense`=pres `inf`=- `part`=- `mood`=subjunctive
imparfait subjonctif	T	`perf`=- `tense`=pres `inf`=- `part`=- `mood`=subjunctive
présent impératif	Y	`perf`=- `tense`=pres `inf`=- `part`=- `mood`=imperative
présent conditionnel	C	`perf`=- `tense`=pres `inf`=- `part`=- `mood`=cond
infinitif présent	W	`perf`=- `part`=- `inf`=+ `tense`=pres
participe présent	G	`perf`=- `part`=+ `inf`=+ `tense`=pres

Table 3.4: Simple verb morphological macros for translating combinations of morphological encodings.

that the passive voice is not possible here.

Subcase 1iv is annotated by Algorithm 2 and subcase 1v, by Algorithm 3.

3.3.3.2 Case 2

Case 2 describes the verb complex that is fully given in the VN phrase and that does not involve any coordination, as illustrated in Figure 3.13. Note that for the case where COORD-VNfinite/COORD-VNpart/COORD-VNinf does not have any V sister (Figure 3.12), each VN conjunct branch is annotated individually according to Subcases 2i through 2iv.

TRADITIONAL TENSE NAME	MORPHOLOGICAL ENCODING	ANNOTATION	
passé composé	P K	perf=+ inf=- mood=indicative	tense=pres part=-
passé surcomposé indicatif	P K K	perf=+ inf=- superperf=+	mood=indicative tense=pres part=-
plus-que-parfait indicatif	I K	perf=+ inf=- mood=indicative	tense=past part=-
passé antérieur	J K	perf=+ inf=- mood=indicative	tense=passesimple part=-
futur antérieur	F K	perf=+ inf=- mood=indicative	tense=fut part=-
passé subjonctif	S K	perf=+ inf=- mood=subjunctive	tense=pres part=-
plus-que-parfait subjonctif	T K	perf=+ inf=- mood=subjunctive	tense=pres part=-
passé impératif	Y K	perf=+ inf=- mood=imperative	tense=pres part=-
passé conditionnel 1re forme	C K	perf=+ inf=- mood=cond	tense=pres part=-
passé conditionnel 2e forme	T K	perf=+ inf=- mood=cond	tense=past part=-
infinitif passé	W K	perf=+ inf=+	tense=past part=-
participe passé	G K	perf=+ inf=+	tense=past part=+

Table 3.5: Compound verb morphological macros for translating combinations of morphological encodings.

If there is only one V daughter of VNfinite (Subcase 2i), then annotation is carried out using Algorithm 4.

Subcases 2ii, 2iii, and 2iv use respectively Algorithms 1, 2 and 3, which is also straightforward.

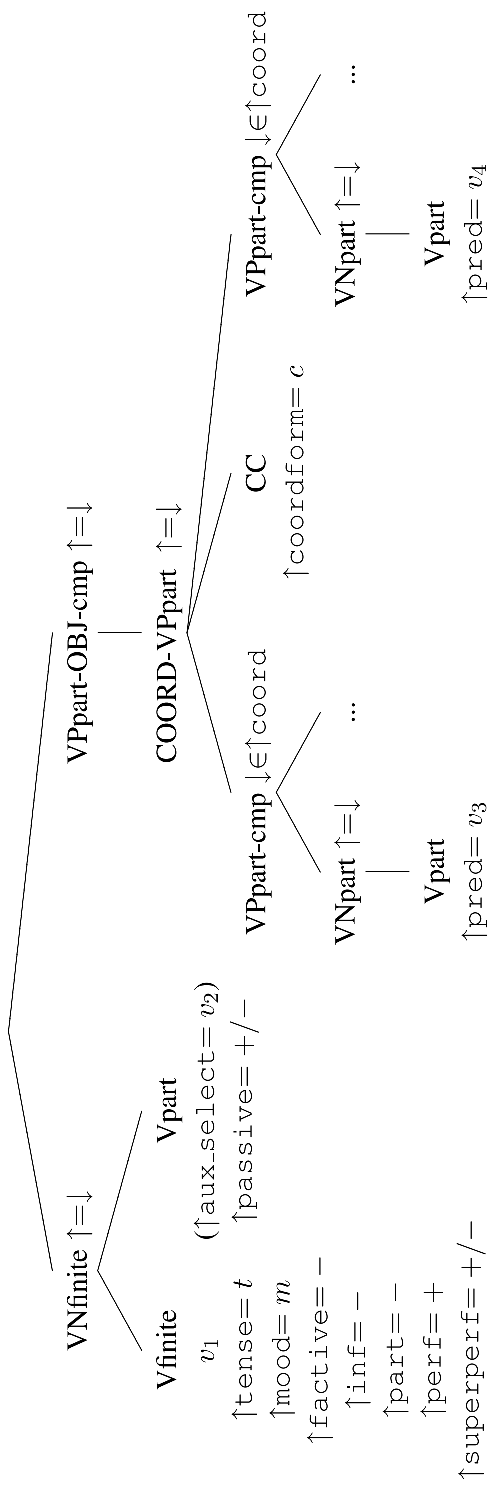


Figure 3.8: Template for Subcase 1ii verb complex annotations.

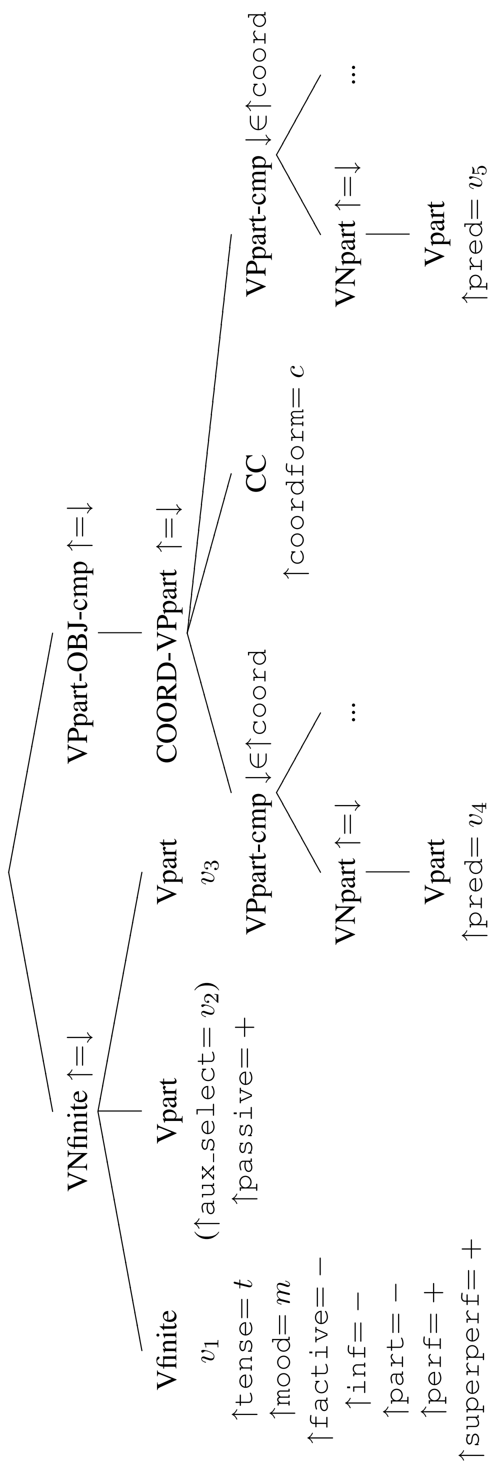


Figure 3.9: Template for Subcase 1iii verb complex annotations.

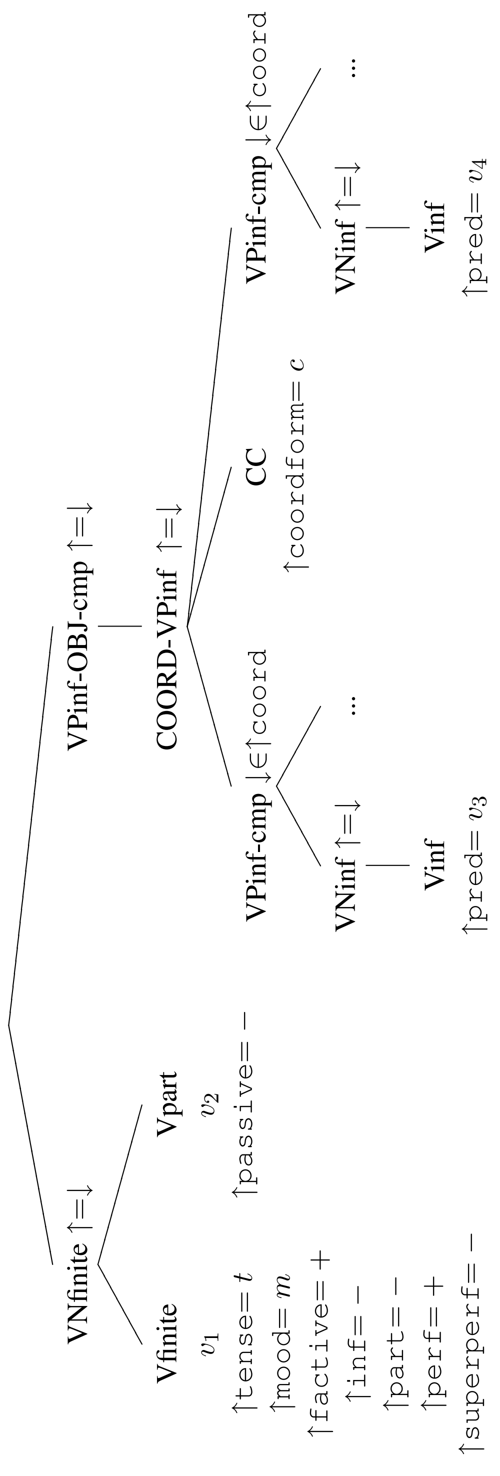


Figure 3.10: Template for Subcase 1iv verb complex annotations.

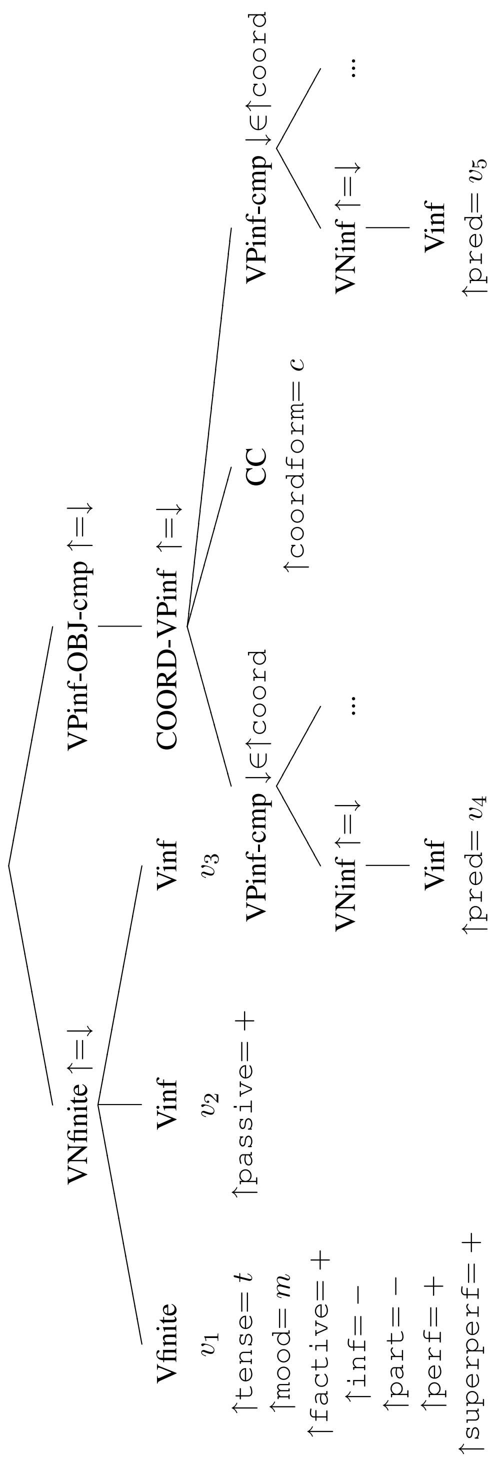


Figure 3.11: Template for Subcase 1v verb complex annotations.

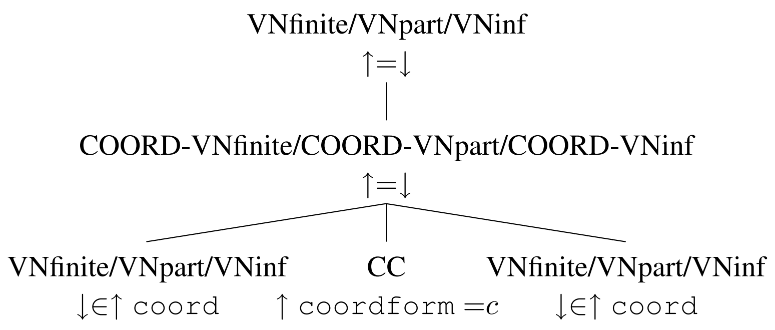


Figure 3.12: A possible context for Subcases 2i-2iv.

Algorithm 4 Decision-tree algorithm for single *V* verb combinatorics.

```
$\uparrow$factive= $-$, $\uparrow$passive= $-$, translate([$mph_1$])
```

3.3.3.3 Case 3

Case 3 covers those verbal configurations where VN has at least one V daughter as well as a *COORD-VNpart/COORD-VNinf* daughter; this is the case of coordination among verbal components (similar to Case 1). Cases 3i, 3ii and 3iii are annotated using Algorithms 1, 2, and 3, respectively.

3.3.3.4 Case 4

Case 3 describes annotation of those verb configurations that involve predicate coordination; it is the complement to Cases 1 and 2. The contexts of this case are

(4i) either *VNpart*(*VNinf*) is a daughter of *VPpart-cmp*(*VPinf-cmp*), or

(4ii) *VNpart* (*VNinf*) is a daughter of *COORD-VNpart* (*COORD-VNinf*), in a *VNfinite* phrase

The annotation of these cases are simple. Only predicate information is annotated as shown in Figures 3.7, 3.8, 3.9, 3.10, 3.10, 3.14, 3.15, and 3.16.

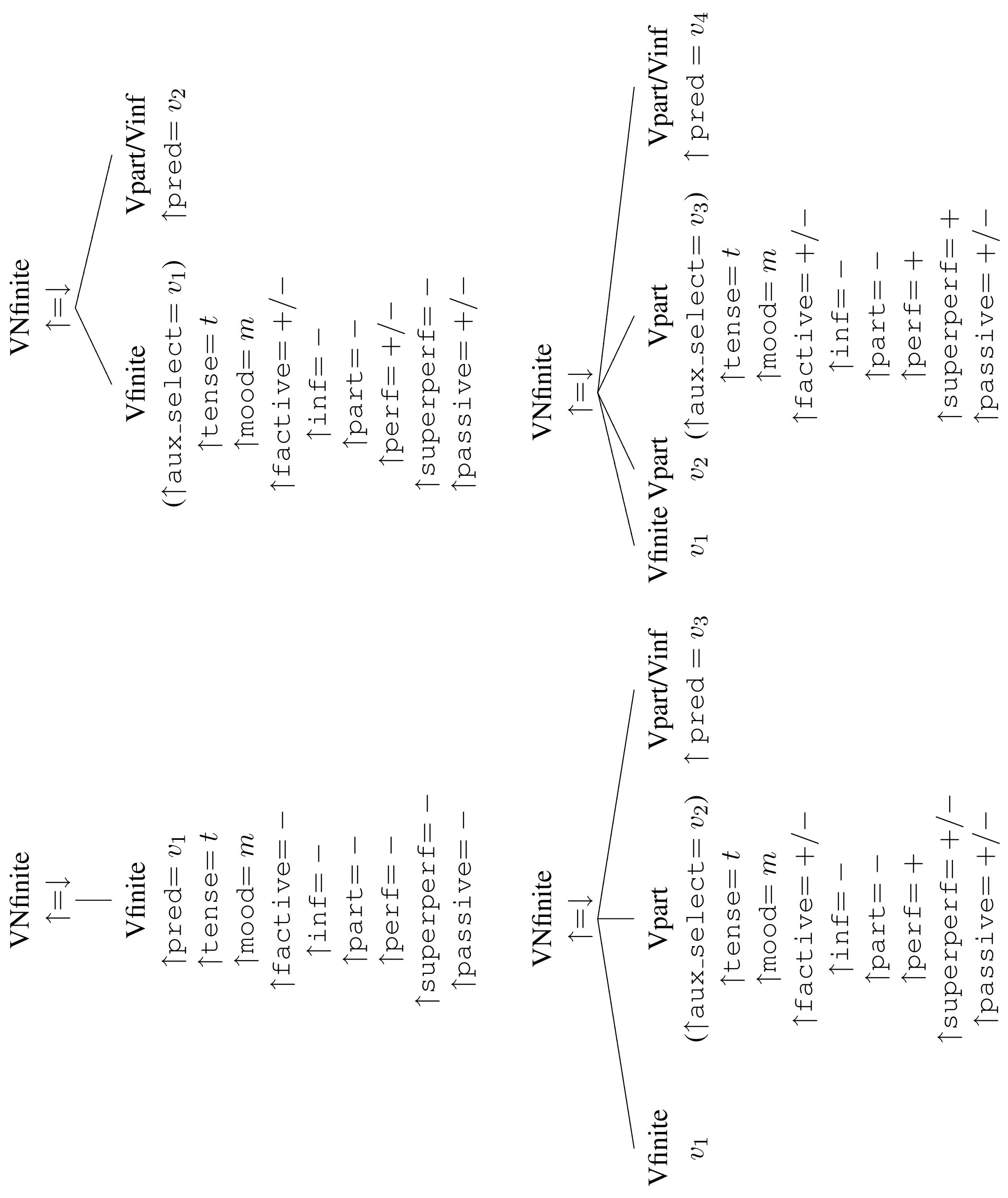


Figure 3.13: Template for Subcases 2i (top left), 2ii (top right), 2iii (bottom right), 2iv (bottom left) verb complex annotations.

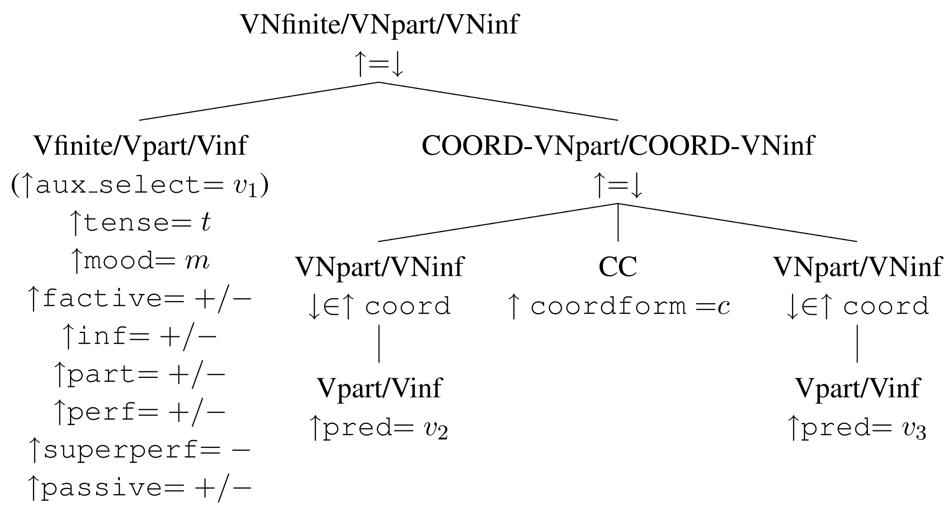


Figure 3.14: Template for Subcase 3i verb complex annotations (2 *V* combination).

3.3.4 Left-Right Context Annotation Module

Like the annotation algorithm for English (Cahill et al., 2004), the annotation algorithm for French also (but to a considerably lesser extent) makes use of a Left-Right Context Annotation Module. The module proceeds by first determining the head of a constituent with respect to head-finding rules provided by the author (see Appendix C), then by consulting annotation tables developed through the analysis of the most frequent rule types in the corpus, covering at least 85% of the respective token instances. The table encodes information on how to annotate CFG node types to the left or right of the constituent head. This is combined with the simple rule that any argument annotation may only be used once and therefore is assigned to the first applicable phrasal tag found. Such annotation principles support easy maintenance and development of this basic annotation module. Table 3.6 gives a simplified annotation table for PP rules.

left context	head	right context
*:↓∈↑adjunct	P:↑=↓	NP, Sint, AdP, AP: ↓=↑obj *:↓∈↑adjunct

Table 3.6: Simplified annotation table for PP rules.

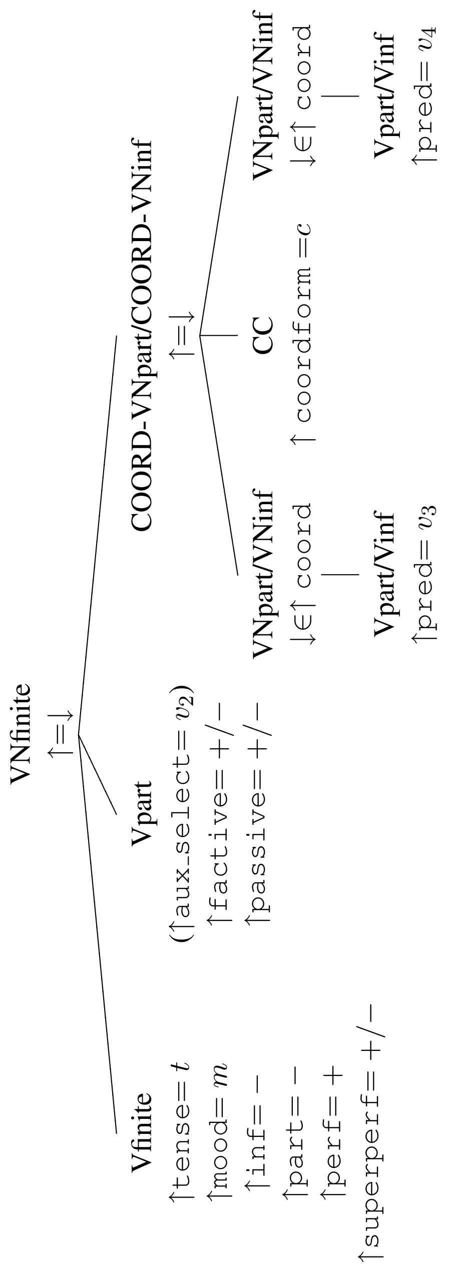


Figure 3.15: Template for Subcase 3ii verb complex annotations (3 *V* combination).

3.3.5 Catch-All and Clean-Up Module

The catch-all and clean-up module handles any special annotation that was not covered earlier, and repairs predictable errors resulting from overgeneralisation in earlier components of the annotation algorithm. For now, just the statement type (declarative, interrogative,

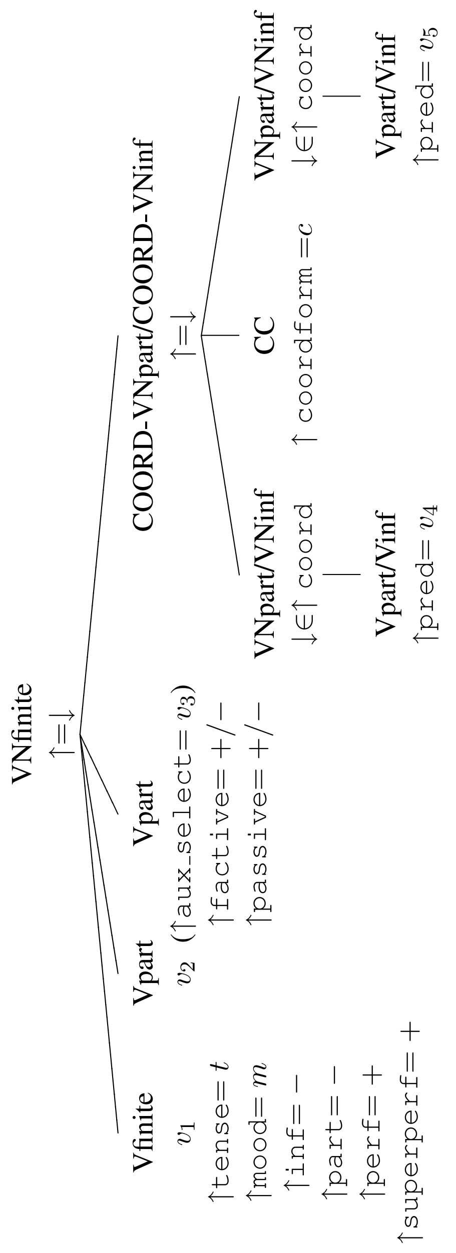


Figure 3.16: Template for Subcase 3iii verb complex annotations (4 *V* combination).

etc.) of the sentence is annotated in this module.

3.3.6 Coordination Distribution

Within the framework of LFG, coordinate phrases in coordination are generated by a production rule which places them as sisters among the right-hand side of the rule (Dalrymple, 2001). The left-hand side is the type of coordination in question. Moreover, coordination at the c-structure level is isomorphic to the corresponding representation at the f-structure level (without considering shared dependencies within control structures or long-distance dependencies–that is, in proto f-structures or distribution of shared elements into the coordinate phrases). The MFT treatment of coordination is essentially inspired by such analyses; the following rules illustrate the resulting configurations (where, for the treebank, the function equations would be annotated by the f-structure annotation algorithm).

$$\begin{array}{llcccc} \text{COORD-NP} & \longrightarrow & \text{NP} & \text{NP} & \text{CC} & \text{NP} \\ & & \downarrow\in\uparrow\texttt{coord} & \downarrow\in\uparrow\texttt{coord} & \uparrow\texttt{coordform}=p & \downarrow\in\uparrow\texttt{coord} \end{array}$$

$$\begin{array}{llccc} \text{COORD-NC} & \longrightarrow & \text{SINT} & \text{CC} & \text{NC} \\ & & \downarrow\in\uparrow\texttt{coord} & \uparrow\texttt{coordform}=p & \downarrow\in\uparrow\texttt{coord} \end{array}$$

For verb phrase or non-constituent coordination, coordinates may share arguments or a predicate (including the predicates lexical attributes). In such cases, the shared predicate or arguments are distributed among the coordinates at the f-structure level (Dalrymple, 2001). Example (4) provides an example of gapping. Figures 3.17 and 3.18 are the f-structure annotated tree and the f-structure for this example, respectively.[14] In this example, the predicate *être* ('to be') from the first conjunct has been distributed into the other in the f-structure.

(4)									
	Le	premier	est	collectionneur,	l'autre	passionné	de	vieux	papiers.
	The	first	is	collector,	the other	fascinated	of	old	papers.

'*The first is a collector, the other has a passion for historic documents*.'[15]

[14]See, for example, Hudson (1976); Maxwell and Manning (1996), for discussion of non-constituent coordination such as gapping, argument clustering, and right-node raising.

[15]Sentence 919, file cflmf7ak2ep.xd.cat.xml of the MFT.

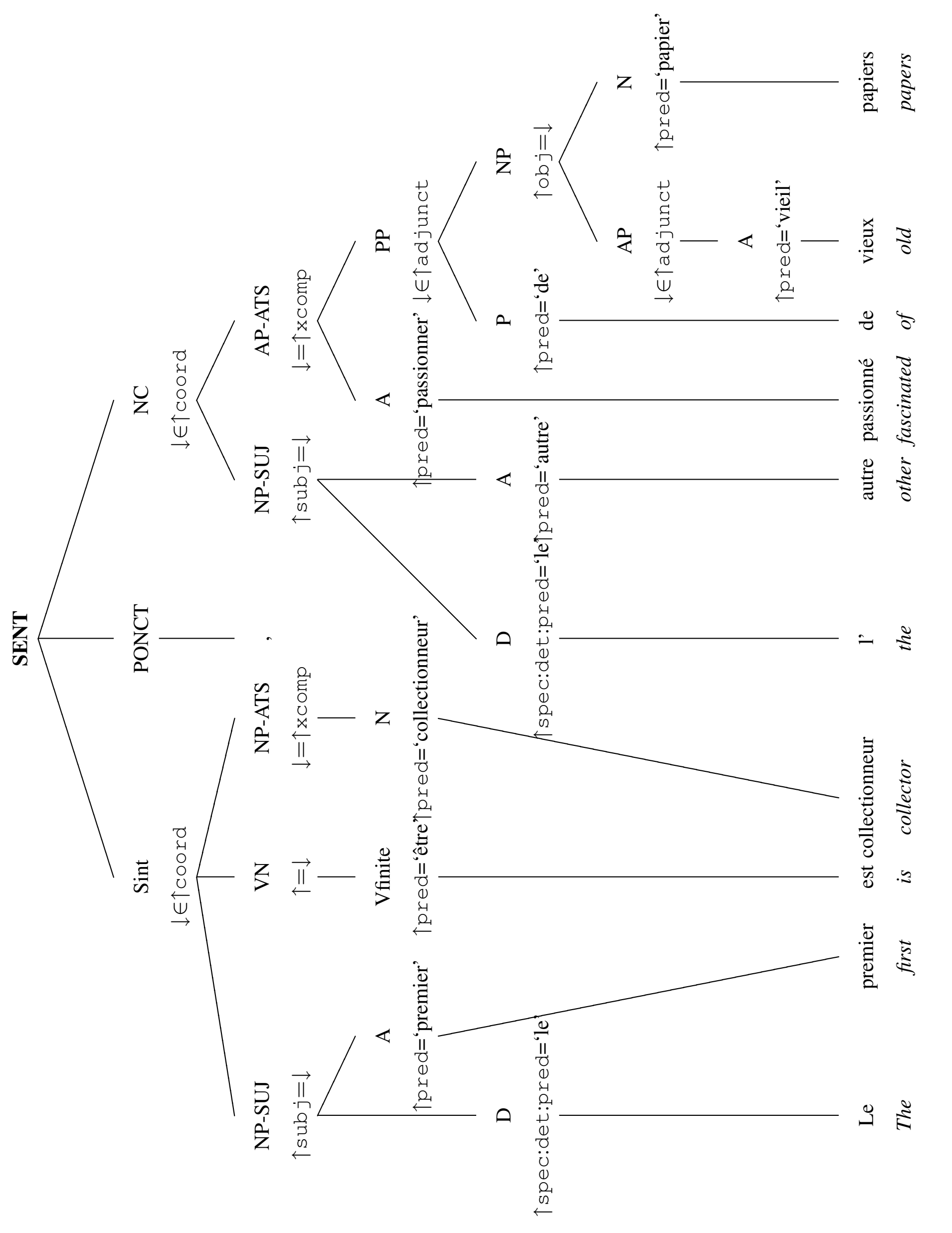


Figure 3.17: F-structure annotated tree for Example (4) on gapping non-constituent coordination.

Distribution over coordination must be carried out for verb phrase (shared subject), predicate, argument-cluster, gapping, and right-node raising types of coordination, and is accounted for as a post-processing step among equations extracted from f-structure annotated MFT trees. The decision tree heuristics adopted for this process are given in Figure 3.19. Distribution is essential for long-distance dependencies through coordinated struc-

$$\left[\begin{array}{ll} \texttt{coord} & \left\{\begin{array}{l} \left[\begin{array}{ll} \texttt{subj} & \boxed{1}\left[\begin{array}{ll}\texttt{pred} & \text{‘premier’}\\ \texttt{spec} & \left[\begin{array}{ll}\texttt{det} & \left[\begin{array}{ll}\texttt{pred} & \text{‘le’}\end{array}\right]\end{array}\right]\end{array}\right]\\ \texttt{pred} & \text{‘être’}\\ \texttt{xcomp} & \left[\begin{array}{ll}\texttt{pred} & \text{‘collectionneur’}\\ \texttt{subj} & \boxed{1}\end{array}\right]\end{array}\right]\\ \left[\begin{array}{ll} \texttt{subj} & \boxed{2}\left[\begin{array}{ll}\texttt{pred} & \text{‘autre’}\\ \texttt{spec} & \left[\begin{array}{ll}\texttt{det} & \left[\begin{array}{ll}\texttt{pred} & \text{‘le’}\end{array}\right]\end{array}\right]\end{array}\right]\\ \texttt{pred} & \text{‘être’}\\ \texttt{xcomp} & \left[\begin{array}{ll}\texttt{pred} & \text{‘passionner’}\\ \texttt{subj} & \boxed{2}\\ \texttt{adjunct} & \left\{\left[\begin{array}{ll}\texttt{pred} & \text{‘de’}\\ \texttt{obj} & \left[\begin{array}{ll}\texttt{pred} & \text{‘papier’}\\ \texttt{adjunct} & \left\{\left[\begin{array}{ll}\texttt{pred} & \text{‘vieil’}\end{array}\right]\right\}\end{array}\right]\end{array}\right]\right\}\end{array}\right]\end{array}\right]\end{array}\right\}\end{array}\right]$$

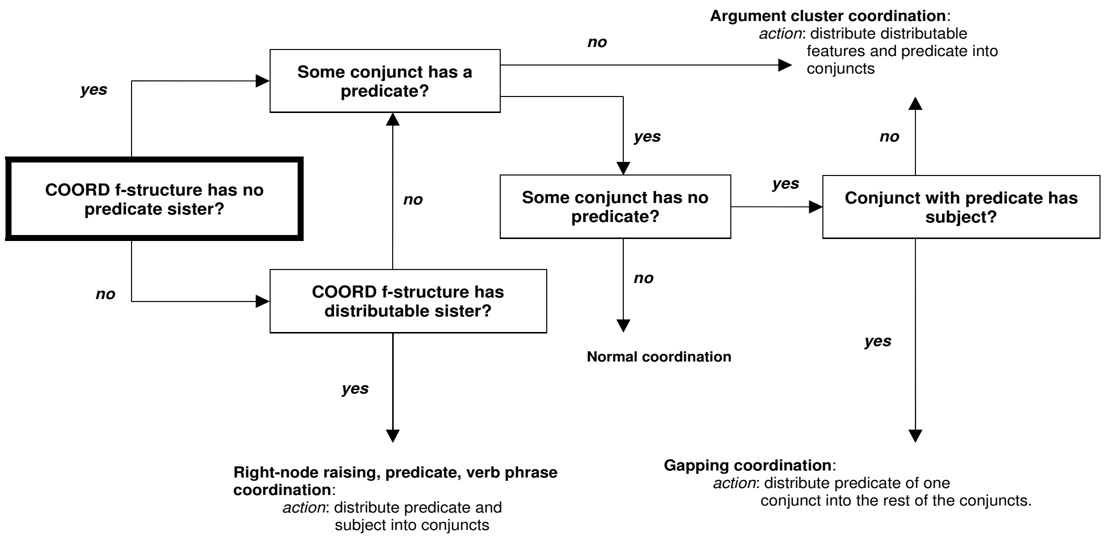
Figure 3.18: F-structure for Example (4) on gapping non-constituent coordination, with *être* distributed into both coordinate f-structure components.

tures, control verb phrases with coordinated structures as verbal arguments (Burke, 2006), as well as for the extraction of lexical resources (O’Donovan, Cahill, van Genabith and Way, 2005). Previous versions of automatically acquired LFG parsing resources do not carry out any distribution over conjuncts and pay the price in performance.[16] Moreover, the MFT contains 3355 cases of coordination; that is, in terms of proportions, well over two thirds of MFT sentences contain coordination. This makes coordination in the MFT, if not in French sentences in general, important enough of a phenomenon to consider coordination distribution as an essential part of the f-structure annotation algorithm.[17]

[16]Burke (2006), for example, derives a sort of hack which treats conjunction as a control structure, in attempting to boost recall (but to the detriment of precision).

[17]For the reader’s information, I include a table of counts of types of coordination in the whole MFT below.

In Section 3.5, I present the evaluation of the annotation algorithm with and without this post-processing step. It turns out that distributing over coordination leads to statistically significant improvements on the performance of the annotation algorithm.



Argument cluster coordination: action: distribute distributable features and predicate into conjuncts
no
Some conjunct has a predicate?
yes
yes
no
COORD f-structure has no predicate sister?
no
Some conjunct has no predicate?
yes
Conjunct with predicate has subject?
no
COORD f-structure has distributable sister?
no
Normal coordination
yes
yes
Right-node raising, predicate, verb phrase coordination: action: distribute predicate and subject into conjuncts
Gapping coordination: action: distribute predicate of one conjunct into the rest of the conjuncts.

Figure 3.19: Decision tree heuristics and actions for distribution over coordination.

coordination type	count
NP	1183
VP	186
VPinf	118
VPpart	66
AP	261
PP	641
Ssub	56
UC	138
Sint	225
unary	320
NC	87
VN_inf	13
VN_part	8
VN_finite	7
Srel	31
AdP	15

3.3.7 Other New Linguistic Annotations

Though linguistic analyses for the French annotation algorithm are very similar to those for English, there are some exceptions. One exception involves the direct translation of the new analyses for cleft constructions outlined in Section 2.3.2. Other exceptions involve including positional information for some AP modifiers and the treatment of clitic-doubling, which differs from that adopted for the Spanish annotation algorithm (Chrupała and van Genabith, 2006a).

3.3.7.1 Modifier Positional Information

French is commonly known to have adjectival modification of noun phrases which either follows or precedes the head noun (phrase). However, there are some adjectives which require the prenominal position, and some that require the postnominal position. The following examples, taken from (Frank and Berman, 1996) illustrate the point.

(5) le soin paternel
the care fatherly
'*fatherly care*'

(6) un petit garon
a little boy
'*a little boy*'

Positional information would be very important to obtain good results in generation. Therefore, following the suggestion of Frank and Berman (1996), I record this positional information through the *POS* attribute at the f-structure level, which may take either of the values *pre* or *post*.

3.3.7.2 Clitic-Doubling in French

In French, argument constituent daughters of the sentence kernel may be topicalised, "dislocating" them to the sentence initial position and marking their syntactic role by a (coindexed) pronoun. The effect is to put emphasis on the topicalised phrase, which, in English, would normally be signalled via stressed prosody (Mel'čuk, 2001). In essence, these pronouns co-occur with semantically full phrases. For this reason, this phenomenon is called *clitic-doubling*. The following sentences, taken from Frank and Berman (1996), illustrate clitic doubling in a tensed phrase, an infinitival phrase, and a noun phrase.

(7) Que l'industrie fasse des progrès, **cela** agace le Japon.
That the industry makes some progress, that annoys Japan.
'It annoys Japan that the indrustry is going better.'

(8) Réussir, **cela** ne suffit pas.
Succeed, that suffices not.
'It is not sufficient to [just] succeed.'

(9) Le Japon, **il** est en avance.
Japan, he is in advance.
'Japan is ahead.'

Clitic-doubling in French seems to differ from that of, for example, Spanish, where there are strong relations with the pro-drop phenomenon. Therefore, whereas in my analysis I simply annotate such dislocated phrases as topics, assigning the pronouns or clitics the non-communicative syntactic annotation (illustrated in Figure 3.20), Chrupała and van Genabith (2006a) adopt an analysis based on optional annotations, assigning both constituents the non-communicative syntactic annotation where the clitic annotation is disregarded if the other constituent is realised.

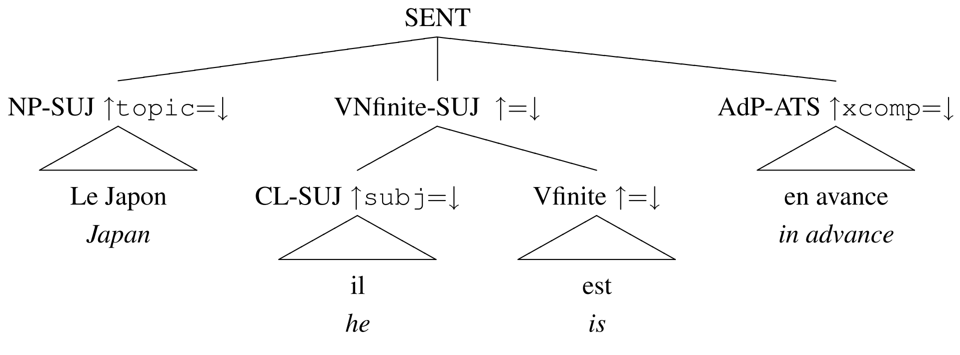


Figure 3.20: Basic Annotated C-structure for Example (9)

	with coordination distribution			**without coordination distribution**		
features	**precision**	**recall**	**f-score**	**precision**	**recall**	**f-score**
all	99.42	99.80	99.61	99.10	96.48	97.78
preds only	99.50	99.77	99.63	98.57	96.38	97.47

Table 3.7: Evaluation of the French annotation algorithm against the MFTDB.

3.4 The MFT Dependency Bank

In order to evaluate the quality and coverage of the f-structure annotation algorithm, a gold standard f-structure (i.e., dependency) bank is required. The MFT was divided into three sets of trees as follows: training set (3800 sentences), development set (509 sentences), and test set (430 sentences). The annotation algorithm was applied to the test set. Functional equations were extracted and sent to a constraint solver to verify consistency (and, therefore, producing the f-structures). All consistent f-structures from the test set were then hand verified and corrected, if necessary, twice by the author. As much of the annotation algorithm is basically LFG conversion software, the hand correction was straightforward. The resulting hand-corrected f-structures were then compiled into dependency triples following Crouch et al. (2002). The result of this work is the MFT Dependency Bank (MFTDB).

3.5 Evaluation

The f-structure annotation algorithm is measured for coverage (i.e., all output sets of equations are consistent and the corresponding f-structures are connected) and accuracy. The f-structure annotation algorithm described here achieves 98.40% coverage on the training set. Table 3.7 gives the scores for the algorithm, evaluated against the MFTDB, both with coordination distribution post-processing and without.

The f-structure annotation algorithm achieves a high f-score of 99.61, with an equally high preds-only (f-structures with only paths ending in a predicate value) f-score of 99.63. This is the highest f-score for an f-structure annotation algorithm to date (Cahill and van Genabith, 2006a; Chrupała and van Genabith, 2006a; Guo, van Genabith and Wang, 2007;

Rehbein and van Genabith, 2009; Oya and van Genabith, 2007; Tounsi et al., 2009a). Moreover, there is a drop in performance when the coordination distribution post-processing step is not carried out. In particular, recall drops by more than 3.3% for both preds-only and full f-structures. This is clearly a statistically significant difference in recall (p-value $< 2.2E$-16 for both), perhaps partially explaining the high score of this annotation algorithm compared to previous models as well as demonstrating the importance of coordination in the MFT.

3.6 Concluding Remarks

The objectives of this chapter have been twofold. First I described the f-structure annotation algorithm for French, used to automatically obtain a treebank with deep grammar annotations, along with a post-processing module over extracted equations for coordination distribution. I have also presented the MFTDB, a new gold standard within the LFG framework, for dependency based resource development. With the construction of these two resources, a first major hurdle in the automatic acquisition of LFG parsing and lexical resources for French is overcome. Moreover, the availability of a dependency gold standard for French now provides a much needed benchmark for comparison among different grammar engineering frameworks and methods.

Chapter 4

Parsing into F-structure Annotated C-structures

4.1 Introduction

This chapter presents the application of the treebank- and CFG-based LFG resource acquisition methodologies presented in Chapter 3 to parsing French text.[1] I present a number of changes to the established treebank-based LFG parsing architectures (Cahill et al., 2002a; Cahill, 2004; Cahill et al., 2008), and show that encouraging parsing results can be obtained for French, with an overall best dependency structure f-score of 86.73% for all features and 81.35% for preds-only.

I start the presentation with a discussion of the parsing architectures that I am building upon (Section 4.2)—namely (1) simplified, (2) with a machine learning component, and (3) with a long-distance dependency resolution component. Following this, the versions of the MFT on which the experiments are carried out are presented in Section 4.3, as well as the performance of the f-structure annotation algorithm for French on these treebank versions and the associated probabilistic parsing results. In Sections 4.4, 4.5, and 4.6, I present the f-structure dependency results obtained for the various parsing architectures, followed by

[1]This work was previously published as (Schluter and van Genabith, 2008).

some concluding remarks in Section 4.7.

4.2 CFG-Based LFG Parsing Architectures Tested for French

In Section 1.2.2.2, I presented the original pipeline and integrated treebank-based CFG parsing architectures (see especially Figure 1.2). I also briefly presented Chrupała and van Genabith (2006a)'s addition to this original pipeline parsing architecture by means of a separate machine learning component for recovering treebank function tags after parsing trees with phrase structure tags only (no function tags) (Section 1.2.3). In this chapter, I present results on French data for these parsing architectures. In addition, I evaluate a further extension of the pipeline parsing architecture, extending Chrupała and van Genabith (2006a)'s work: I incorporate a separate machine learning component for f-structure equations as annotated by the annotation algorithm (See Figure 4.1, in dark grey), rather than treebank function tags as in the earlier work. It will be of interest to compare this component with results on pure dependency parsing into LFG f-structures in Chapter 5.

4.2.1 The Simplified Parsing Architectures

Because of the MFT's completed function tagging where function tag paths encode non-local dependencies (Section 2.3.5), one can carry out LDD dependency resolution at the f-structure annotation or parsing stages, without recourse to a separate LDD resolution component as in the earlier work of Cahill et al. (2004). This is what I will call the *simplified parsing architecture*, as depicted in Figure 4.1, on the bottom, bypassing the separate LDD Resolution step (Section 4.4).

4.3 Generating Different Versions of the MFT: Clitic Constituents and MFT Function Labels

In general, the adaption of the original approach to treebank-based LFG grammar acquisition (for English data and Penn-II style representations) to other languages is based on

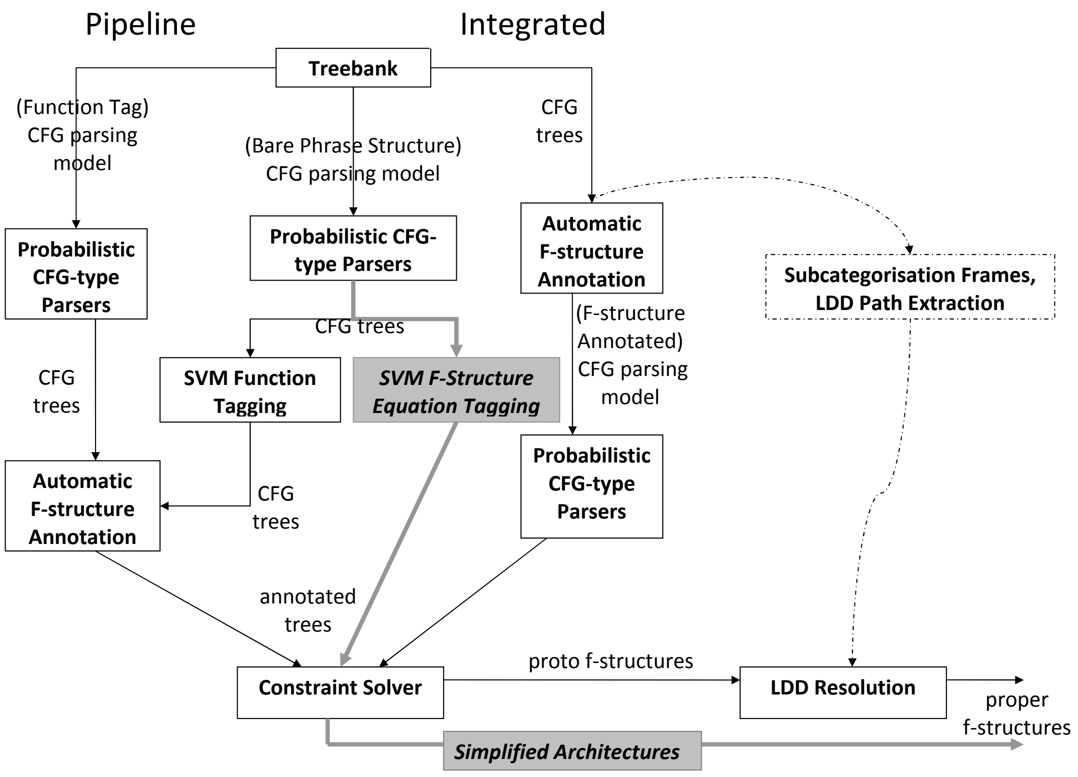

Pipeline
Integrated
Treebank
(Function Tag) CFG parsing model
(Bare Phrase Structure) CFG parsing model
CFG trees
Probabilistic CFG-type Parsers
Probabilistic CFG-type Parsers
Automatic F-structure Annotation
Subcategorisation Frames, LDD Path Extraction
CFG trees
(F-structure Annotated) CFG parsing model
CFG trees
SVM Function Tagging
SVM F-Structure Equation Tagging
Automatic F-structure Annotation
CFG trees
Probabilistic CFG-type Parsers
annotated trees
Constraint Solver
proto f-structures
LDD Resolution
proper f-structures
Simplified Architectures

Figure 4.1: Overview of CFG treebank-based LFG parsing architectures for French. Paths including dark grey segments involve parsing architectures introduced here.

both linguistic and data-structure specific considerations. In terms of data-structure specific considerations, several important differences between MFT data structures and Penn-II data structures (on which the original technology is based) require special attention. This concerns the absence of empty nodes and coindexation in the MFT, as well as the lexicalised treatment of some personal pronouns (clitic pronouns).

In this section, I introduce the different versions of the MFT (and grammars extended from these and in some cases further processed by the f-structure annotation algorithm or the SVM-based treebank function tag or f-structure annotation labeler) used in my CFG-based parsing experiments, summarised in Table 4.1:

- The versions MFT-norm, MFT-fct, and MFT-comm (numbers 1-3 in the table) will be explained in Sections 4.3.1 and 4.3.2.

- For the prefix A- and $X \in$ {MFT-fct, MFT-comm, SVM-MFT-fct, SVM-MFT-comm}, A-X denotes the output of the annotation algorithm with input X.
- For the prefix SVM- and $Y \in$ {MFT-fct, MFT-comm, A-MFT-fct, A-MFT-comm}, SVM-Y denotes the approximation of Y by the Support Vector Machine classifier.

reference number	**version**	**explanation**
1	MFT-norm	MFT without any functional information
2	MFT-fct	MFT with functional information (including LDD function paths)
3	MFT-comm	MFT without LDD function paths, but with derived communicative function tags
4	A-MFT-fct	output of f-structure annotation algorithm with input MFT-fct
5	A-MFT-comm	A-MFT-fct stripped of long-distance dependencies
6	SVM-MFT-fct	SVM approximation of MFT-fct: SVM labeling of function tags from MFT-fct onto MFT-norm
7	SVM-MFT-comm	SVM approximation of MFT-comm: SVM labeling of function tags from MFT-comm onto MFT-norm
8	SVM-A-MFT-fct	SVM approximation of A-MFT-fct: SVM labeling of f-structure annotation from A-MFT-fct onto MFT-norm
9	SVM-A-MFT-comm	SVM approximation of A-MFT-comm: SVM labeling of f-structure annotation from A-MFT-comm onto MFT-norm
10	A-SVM-MFT-fct	output of annotation algorithm with input SVM-MFT-fct
11	A-SVM-MFT-comm	output of annotation algorithm with input SVM-MFT-comm

Table 4.1: Generated versions of the MFT.

4.3.1 Deriving MFT-norm and MFT-fct

Similar to the preprocessing step before application of the f-structure annotation algorithm for French to the MFT, for my parsing experiments for both the pipeline and integrated architectures, I have enclosed clitics in their own constituents and propagated the function labels down to this level from VN. The CLP transformed version of the MFT without function labels will be denoted by MFT-norm, and with function labels, by MFT-fct.

4.3.2 Deriving MFT-comm

Unlike the Penn-II treebank, the MFT does not have any communicative function annotation such as TOPIC or FOCUS. Rather, whenever local c-structural constituents are not locally dependent on the constituent head, there is a function path annotation indicating where this dependency is resolved (see Section 2.3.5). Therefore, I generated a complementary version of the treebank to implement the existing treebank-based LFG architectures for other languages for comparison–this version of the treebank has no function path tags corresponding to communicative LFG functions, instead introducing communicative function tags.[2,3] This is carried out automatically, using the automatically (f-structure) annotated version of the treebank: since the f-structure annotation algorithm introduces communicative f-structure equations, I project this communicative annotation onto the MFT, replacing most path function tags (and some simple function tags) in the MFT's function tag set.[4] Figure 4.2 shows the representation in MFT-comm of Example (2) (Figure 2.2): SUJ.MOD is replaced by TOPICREL. The MFT-fct with communicative function annotation will be denoted by MFT-comm.

[2]These experiments are in addition to the simplified parsing architectures introduced in Section 4.2.1.

[3]Note that introduced communicative function tags will not only replace function path tags; they can also replace simple function tags.

[4]Not all path function tags can be replaced, as they are not all communicative. This is specific to French. See Section 4.7 for some discussion on this.

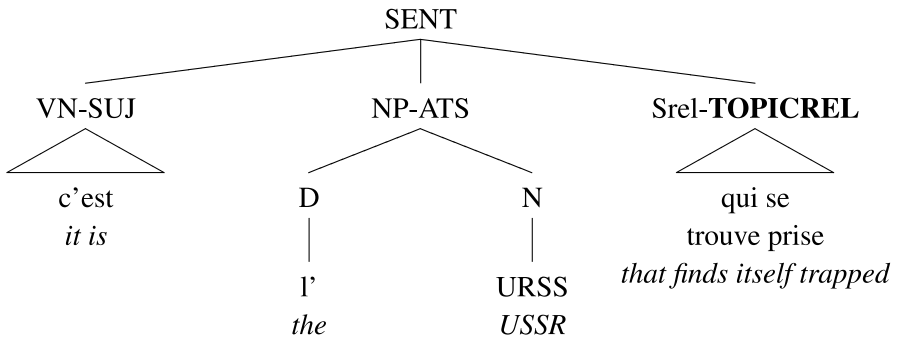


Figure 4.2: MFT-comm representation for Example (2) (Figure 2.2).

4.3.3 Performance of the F-Structure Annotation Algorithm on MFT-fct and MFT-comm

The f-structure annotation algorithm was adjusted to also take MFT-comm as input. Table 4.2 provides evaluation results using gold treebank trees. Tables 4.2 and 3.7 (which measures the performance of the annotation algorithm on MFT-fct) show that without accounting for LDDs (as in MFT-comm in Table 4.2), the f-structure annotation algorithm's performance is not optimal: scores decrease by 3-4% from Table 3.7 to Table 4.2.

coordination distribution	**features**	**precision**	**recall**	**f-score**
no	all	99.26	89.29	94.02
	preds only	98.58	90.33	94.28
yes	all	99.54	91.60	95.41
	preds only	99.35	92.85	95.99

Table 4.2: Performance of the f-structure annotation algorithm on MFT-comm.

4.3.4 CFG Parsing Results for MFT-norm, MFT-fct and MFT-comm

PARSEVAL parsing results for the MFT-norm, the MFT-fct, and the MFT-comm, are reported in Table 4.3.[5] Notice in particular that Bikel's parser outperforms BitPar on all

[5]Table 4.4 gives the parsing results for MFT-norm shown in Table 4.3, but collapsing CLP at evaluation. In both these tables, parsing scores for BitPar of MFT-norm are slightly higher than those of MFT (Compare especially Tables 4.4 and 2.7). This is in support of the findings of Kűbler (2005); Maier (2006); Kűbler and Prokic (2006), which give empirical proof that the omission of unary nodes is detrimental to parsing results.

treebank versions. However, the divergence between precision and recall is always much greater for Bikel's parser than for BitPar; the difference between precision and recall for BitPar is consistently close to 1%, whereas for Bikel, it fluctuates from 2.2-4.05%. Moreover, with respect to the treebank versions, the difference in f-score is smaller between BitPar and Bikel's parser performances on MFT-fct (2.98) and MFT-comm (2.59) compared to MFT-norm (7.39). The quality of these parsed c-structures will be put into question in light of the results on the derived f-structures in Section 4.4.

treebank version	**parser**	**precision**	**recall**	**f-score**
MFT-norm	BitPar	77.6	78.25	77.92
	Bikel	84.21	86.41	85.31
MFT-fct	BitPar	68.84	69.96	69.39
	Bikel	70.45	74.40	72.37
MFT-comm	BitPar	69.33	70.41	69.87
	Bikel	70.49	74.54	72.46

Table 4.3: PARSEVAL parsing results for three derived MFT versions.

treebank version	**parser**	**precision**	**recall**	**f-score**
MFT-norm	BitPar	77.67	78.40	78.03
(pruning CLP)	Bikel	83.73	86.00	84.85

Table 4.4: PARSEVAL parsing results for MFT-norm, collapsing CLP at evaluation.

For both parsers, as expected, MFT-norm produces the best parse results; MFT-fct produces the worst parse results, as it provides the sparsest information (due to the inflation of its functionally annotated phrasal category set). Since MFT-comm results are slightly better than those for MFT-fct, it is worth looking into the original approach to LDD resolution for French.

4.4 Simplified Parsing into F-Structures and Proto F-Structures

Deep grammar parsing experiments, for the MFT-fct, A-MFT-fct, MFT-comm, and A-MFT-comm by the two simplified pipeline and integrated architectures were carried out and the extracted f-structures evaluated. The results are presented in Tables 4.5 through 4.9.

Similar differences in CFG parsing results as observed in Table 4.3 are also observable in Table 4.5; that is, A-MFT-comm parses slightly better with both parsers, than A-MFT-fct. However, a separate LDD resolution component would need to work very well, to build on these differences to translate the highter CFG parsing scores for A-MFT-comm into corresponding higher dependency scores for the resulting f-structures; I look into this in Section 4.6.

treebank version	**parser**	**precision**	**recall**	**f-score**
A-MFT-fct	BitPar	71.18	72.54	71.85
	Bikel	73.71	75.90	74.79
A-MFT-comm	BitPar	71.47	72.83	72.15
	Bikel	74.25	76.38	75.30

Table 4.5: PARSEVAL results for simplified integrated parsing architecture.

coord dist	**features**	**parser**	**precision**	**recall**	**f-score**
no	all	BitPar	89.61	80.92	85.04
	all	Bikel	91.63	68.20	78.20
	preds only	BitPar	78.93	72.24	75.44
	preds only	Bikel	82.16	61.99	70.66
yes	all	BitPar	89.25	79.39	84.04
	all	Bikel	91.27	67.82	77.82
	preds only	BitPar	79.10	71.22	74.96
	preds only	Bikel	82.37	61.82	70.63

Table 4.6: MFT-fct pipeline simplified architecture f-structure triples evaluation.

Comparing Tables 4.6 with 4.8 and 4.7 with 4.9, three important observations can be made. Firstly, the pipeline architecture consistently outperforms the integrated architecture. Secondly, though Bikel's parser outperformed BitPar in terms of c-structure labeled bracketing scores (Tables 4.3, 4.4, 2.7), BitPar output seems to better preserve dependency relations between lemmata: the results show that f-structures extracted from BitPar parse trees are generally of better quality under either parsing architecture. Finally, I observe that coordination distribution post-processing almost never entails a boost in scores for parser

coord dist	**features**	**parser**	**precision**	**recall**	**f-score**
no	all	BitPar	91.12	75.51	82.58
	all	Bikel	91.10	66.04	76.92
	preds only	BitPar	80.21	68.39	73.83
	preds only	Bikel	82.49	60.53	69.83
yes	all	BitPar	90.67	74.03	81.51
	all	Bikel	91.84	65.53	76.48
	preds only	BitPar	80.27	67.40	73.27
	preds only	Bikel	82.68	60.25	69.71

Table 4.7: MFT-comm pipeline simplified architecture f-structure triples evaluation.

coord dist	**features**	**parser**	**precision**	**recall**	**f-score**
no	all	BitPar	89.89	64.57	75.15
	all	Bikel	89.12	47.63	62.08
	preds only	BitPar	78.71	58.35	67.02
	preds only	Bikel	77.39	42.51	54.87
yes	all	BitPar	89.39	64.59	74.99
	all	Bikel	88.55	47.21	61.59
	preds only	BitPar	78.81	58.62	67.23
	preds only	Bikel	77.42	42.25	54.67

Table 4.8: A-MFT-fct integrated simplified architecture f-structure triples evaluation.

coord dist	**features**	**parser**	**precision**	**recall**	**f-score**
no	all	BitPar	90.45	61.93	73.52
	all	Bikel	89.52	46.93	61.58
	preds only	BitPar	78.94	56.80	66.07
	preds only	Bikel	77.74	42.49	54.94
yes	all	BitPar	90.00	61.84	73.31
	all	Bikel	88.91	46.67	61.21
	preds only	BitPar	79.08	56.99	66.24
	preds only	Bikel	77.58	42.37	54.81

Table 4.9: A-MFT-comm integrated simplified architecture triples evaluation.

output extracted f-structures.[6] This is probably due to the difficulty in obtaining good parsing results for coordinate structures in the first place, which may be compounded by the "explosion" of phrasal category tag set in some of the MFT versions. Indeed there are 42

[6]The only case is under the integrated architecture (for both MFT-fct and MFT-comm) using the preds only metric.

different constituent tags paired with up to 27 different function tags in MFT-fct for the parser to recognise (Table 4.8).

Among the results presented for the *integrated* parsing architecture in this chapter, the highest LFG all-features f-score of 75.15 is achieved by BitPar (Table 4.8), under the simplified integrated parsing, without coordination distribution.

4.5 Using Generic Machine Learning Techniques for Annotation

In the previous section we saw that higher PARSEVAL labeled bracketing scores did not necessarily correspond to better f-structures. The question remained, however, as to whether an explosion of the treebank phrasal category tag set was at fault. In the pipeline architecture one must train parsers on the regular MFT phrasal tag set with function labels (either MFT-fct or MFT-comm) and in the integrated architecture, parsers are trained on f-structure annotated MFT-norm trees (i.e., A-MFT-fct or A-MFT-comm).

Moreover, general results could perhaps be boosted if one can use parse trees from MFT-norm that were found to have a c-structure f-score of over 10% higher than the annotated treebank version parse trees.

Chrupała and van Genabith (2006b) introduced an approach to side-stepping this tag set explosion, enlisting machine learning of function tag information as provided by the Spanish CAST3LB treebank; this is a sort of *approximation of the original pipeline parsing architecture*. Several algorithms were tested, with the Support Vector Machine (SVM) classifier performing best. I therefore applied this method to the French data. In addition, I investigated the application of this method to directly learning f-structure equations from f-structure annotated MFT trees (rather than MFT function tags), as an *approximation of the original integrated parsing architecture* (see Figure 4.1).

As in Chrupała and van Genabith (2006b), I use Chang and Lin (2001)'s implementation of SVM; I also use exactly the same feature information.[7] Before applying the methods on probabilistic parser output, I present an evaluation on gold trees.

4.5.1 Evaluation of the SVM Approximation Architectures on Gold Trees

Table 4.10 gives the accuracy of the SVM classifier on gold trees. Observe that there is no difference in performance of the classifier between SVM-MFT-fct and SVM-MFT-comm (where the SVM learns MFT function tag labels), and only a small difference in performance between SVM-A-MFT-fct and SVM-A-MFT-comm (+0.13%) (where the SVM learns f-structure annotations).

treebank version	**accuracy**
SVM-MFT-fct	96.73
SVM-MFT-comm	96.73
SVM-A-MFT-fct	96.68
SVM-A-MFT-comm	96.81

Table 4.10: Accuracy of the SVM classifier on gold trees.

Table 4.11 gives the PARSEVAL results on precision of the SVM classifier on gold trees (all instances are given to the classifier, so precision, recall, and therefore f-score are identical). Observe that once again there are very similar scores, with the classifier on SVM-A-MFT-comm performing slightly better than the others.

[7]The feature information used is the following:

1. Node features:
    (a) position relative to the constituent head
    (b) tag
    (c) definiteness
    (d) head lemma of the constituent
2. Local features:
    (a) parent constituent category
    (b) head lemma of the verb
3. Context features:
    (a) node features of the two previous and the following two nodes

(Chrupała and van Genabith, 2006b)

treebank version	precision
SVM-MFT-fct	96.61
SVM-MFT-comm	96.56
SVM-A-MFT-fct	96.52
SVM-A-MFT-comm	96.67

Table 4.11: PARSEVAL f-score precision for the SVM approximations on gold trees.

I ran the f-structure annotation algorithm on the gold trees from SVM-MFT-fct and SVM-MFT-comm, resulting in A-SVM-MFT-fct and A-SVM-MFT-comm. The f-structure equations were then extracted from A-SVM-MFT-fct and A-SVM-MFT-comm for the pipeline approximations, SVM-A-MFT-fct, and SVM-A-MFT-comm for the integrated approximations, and the triples evaluation of the resulting f-structures is given in Table 4.12. On just the gold trees, we see that already the pipeline approximation outperforms the integrated approximation by more than 10% in f-score. The question remains as to whether this difference still holds and also, whether it is equally pronounced for parser output.

treebank version	coord dist	features	precision	recall	f-score
A-SVM-MFT-fct	no	all	97.85	87.04	92.13
		preds only	96.17	86.28	90.96
	yes	all	97.88	88.89	93.17
		preds only	96.63	88.28	92.26
A-SVM-MFT-comm	no	all	97.76	83.43	90.03
		preds only	95.81	83.44	89.20
	yes	all	97.87	85.24	91.12
		preds only	96.36	85.43	90.57
SVM-A-MFT-fct	no	all	96.86	69.73	81.08
		preds only	95.13	69.00	79.99
	yes	all	97.06	71.62	82.42
		preds only	95.72	70.89	91.45
SVM-A-MFT-comm	no	all	97.54	66.22	78.89
		preds only	95.57	66.31	78.30
	yes	all	97.66	67.24	79.65
		preds only	95.99	67.44	79.22

Table 4.12: Triples evaluation for f-structures derived from SVM approximations on gold trees
.

4.5.2 Evaluation of the SVM Approximation Architectures on Parser Output

The PARSEVAL evaluation results for the SVM approximations of MFT-fct (SVM-MFT-fct) and A-MFT-fct (SVM-A-MFT-fct) on parser output are presented in Table 4.13. The results show that Bikel's parser outperforms BitPar consistently, by more than 5%. Triples evaluation, however, will show much closer scores for the two parsers.

treebank version	**parser**	**precision**	**recall**	**f-score**
SVM-MFT-fct	BitPar	71.50	72.10	71.80
	Bikel	79.00	81.08	80.03
SVM-MFT-comm	BitPar	71.40	72.10	71.80
	Bikel	78.91	81.00	79.94
SVM-A-MFT-fct	BitPar	74.55	75.17	74.86
	Bikel	79.73	81.83	80.77
SVM-A-MFT-comm	BitPar	74.54	75.15	74.84
	Bikel	79.82	81.92	80.86

Table 4.13: PARSEVAL results for the SVM approximations on parser output trees.

By comparison to the results for the simplified architecture, for triples evaluation of the f-structures (Tables 4.14 to 4.17), observe that while BitPar makes modest gains in f-score (around +2.5%) in the pipeline approximation, f-structure quality actually decreases in the approximation of the integrated architecture. On the other hand, Bikel's parser remarkably gains over 5-7% in f-score for both parsing architectures. Interestingly, this suggests that BitPar is less influenced by larger constituent tag sets than Bikel's parser.

Also, the pipeline approximation outperforms the integrated approximation. Two possible explanations for this discrepancy are immediately evident. Firstly, the MFT-fct has manual (and therefore, probably, better quality) function tag annotation, to be learned by the SVM method. Moreover, there are much fewer function tags to be learned compared to f-structure annotations (27 function tag types as opposed to 69 f-structure annotation types), which makes learning f-structure equations the harder of the two tasks.

Observe also that the coordination distribution post-processing is only beneficial to Bikel's parser output under both approximation architectures. This suggests that Bikel's parser is better able to recognise coordinate structures than BitPar on MFT-norm trees.

coord dist	features	parser	precision	recall	f-score
no	all	BitPar	91.38	81.21	85.99
	all	Bikel	94.42	79.62	86.39
	preds only	BitPar	82.89	74.37	78.41
	preds only	Bikel	87.99	74.92	80.93
yes	all	BitPar	90.86	79.77	84.95
	all	Bikel	94.01	80.30	86.61
	preds only	BitPar	83.00	73.27	77.84
	preds only	Bikel	87.98	75.65	81.35

Table 4.14: LFG triples evaluation of the SVM approximation of the pipeline architecture performance on MFT-fct.

coord dist	features	parser	precision	recall	f-score
no	all	BitPar	91.97	58.59	71.58
	all	Bikel	93.26	57.61	71.22
	preds only	BitPar	83.65	54.32	65.87
	preds only	Bikel	87.20	54.14	66.80
yes	all	BitPar	91.56	57.26	70.46
	all	Bikel	93.09	58.96	72.20
	preds only	BitPar	83.87	53.16	65.08
	preds only	Bikel	87.51	55.45	67.88

Table 4.15: LFG triples evaluation of the SVM approximation of the integrated architecture performance on MFT-fct.

coord dist	features	parser	precision	recall	f-score
no	all	BitPar	91.47	79.26	84.93
	all	Bikel	94.22	77.46	85.02
	preds only	BitPar	82.56	73.10	77.54
	preds only	Bikel	87.51	73.43	79.85
yes	all	BitPar	90.95	77.97	83.96
	all	Bikel	93.96	78.15	85.33
	preds only	BitPar	82.52	72.1	76.96
	preds only	Bikel	87.61	74.19	80.35

Table 4.16: Triples evaluation of the SVM approximation of the pipeline architecture performance on MFT-comm (SVM-MFT-comm).

Using the SVM classifier for approximations of the pipeline and integrated parsing architectures, I achieve the highest predicates only f-scores for results in this chapter:

coord dist	features	parser	precision	recall	f-score
no	all	BitPar	92.32	56.46	70.07
	all	Bikel	93.65	56.72	70.65
	preds only	BitPar	83.62	52.83	64.75
	preds only	Bikel	87.15	53.75	66.49
yes	all	BitPar	92.01	55.38	69.15
	all	Bikel	93.41	57.22	70.96
	preds only	BitPar	83.78	51.87	64.08
	preds only	Bikel	87.22	54.32	66.95

Table 4.17: Triples evaluation of the SVM approximation of the integrated architecture performance on MFT-comm (SVM-A-MFT-comm).

1. For the pipeline architecture (approximation), the highest predicates only f-score of 81.35 is achieved, using Bikel's parser and coordination distribution (Table 4.14).

2. For the integrated architecture (approximation), the highest predicates only f-score of 67.88 is achieved, using Bikel's parser and coordination distribution (Table 4.15).

4.6 Long-Distance Dependency Resolution

In this section, following a brief presentation of LDDs in LFG (Section 4.6.1), I discuss experiments when LDD resolution is a separate component—that is, with MFT-comm as the base treebank version.

4.6.1 Long-Distance Dependency in LFG

There are several types of grammatical features that may appear in f-structures. Of exceptional value are the communicative attributes `topic`, `focus`, and `topic-rel`, which represent the communicative organisation of a given phrase. I will call those f-structures that are the values of these communicative attributes, communicative f-structures. Long-distance dependencies in LFG are those non-local f-structural dependencies between non-communicative f-structures and their re-entrancies as communicative f-structures. Therefore, under this linguistic framework, long-distance dependencies are resolved at the f-structural level of representation and not in c-structures. In standard LFG, this is carried out by means of function uncertainty equations (FUs) (Kaplan and Zaenen, 1989).

FUs are regular expressions which denote the set of proposed possible paths in an f-structure between a source communicative f-structure and a target non-communicative f-structure. For example, the equation $\uparrow\texttt{topic}{=}\uparrow\texttt{comp}^{+}\texttt{comp}$ reflects the fact that a `topic` f-structure may be resolved with a `comp` f-structure along some non-null path of `comp` attributes. Among the paths, the only possible ones are those that maintain the principles of completeness and coherence in LFG with regards to f-structures. That is, the target's local predicate must subcategorise for the argument in question, and this argument must not already be filled.

4.6.2 Long-Distance Dependency Resolution for Treebank-Based LFG Parsing Resources and its Application to French

A technique for the automatic resolution of long-distance dependencies, based on learning finite approximations of FUs from f-structure annotated treebank resources in English was first outlined by Cahill et al. (2004). A finite approximation of an FU is a single path, rather than a set of possible paths. It is this technique for long-distance dependency resolution that I test for the French case, with slight adaptations due to the particularities of the MFT treebank encoding schemes.

In Section 3.3.2, I explained that the MFT has function path tags that are directly translated into f-structure equations. Moreover, it has no communicative function tags. These are added in the translation of MFT's function tags to f-structure equations. So, the f-structures generated in the previous sections, using MFT-fct or SVM-MFT-fct (for the pipeline architecture), or A-MFT-fct or SVM-A-MFT-fct (for the integrated architecture), as the treebank input to the parsers, are in fact full f-structures. No further long distance dependency resolution is needed. The question remained, however, as to how efficiently the function path tags or functional uncertainty approximations are being recovered during the parsing or prediction phase. I can test this, by rerunning my experiments on MFT-comm or A-MFT-comm, and by separating out LDD resolution.

The technique starts with the MFT-fct version of the MFT. This treebank includes path function tags that are exploited for simple annotation of long-distance re-entrancies at the f-structural level. These f-structures are complete f-structures; moreover, they are of high quality as the f-structure annotation algorithm achieves an f-score that is close to perfect on gold trees. Subcategorisation information and long-distance dependency paths for each predicate are extracted from these f-structures from the training section of the original gold treebank trees and stored with their associated relative frequencies from treebank f-structures.

As an illustration, an example of subcategorisation information extracted for the verb *assurer* 'to reassure' (along with relative frequencies), is given in Table 4.18. Observe that in this approach, `([subj],p)` and `([subj])` are considered to be different, where the `p` indicates passivity. Also, the possible LDD paths for `topic_rel` extracted from the MFT are given in Table 4.19 along with their relative frequencies.

relative frequency	**subcat info**
0.5227	`([subj,obj])`
0.1364	`([subj,comp])`
0.0909	`([subj],p)`
0.0455	`([subj])`
0.0455	`([subj,obj,de_obj])`
0.0455	`([subj,obl_agt],p)`
0.0227	`([obj])`
0.0227	`([subj,xcomp],p)`
0.0227	`([subj,obj,a_obj])`
0.0227	`([subj,obj,comp])`
0.0227	`([subj,de_obj,obl_agt],p)`

Table 4.18: Subcategorisation information for *assurer* 'to reassure' extracted from the MFT.

Armed with this information the LDD resolution algorithm works for MFT-comm or A-MFT-comm derived f-structures as follows.

Given an f-structure of type $GF \in \{\texttt{focus}, \texttt{topic}, \texttt{topic\_rel}\}$, the possible paths associated with GF are retrieved. The list of paths is pruned with regards to the principle

relative frequency	**path**
0.62378303	`subj`
0.13908206	`adjunct`
0.12169680	`obj`
0.05632823	`subj:adjunct`
0.01738526	`de_obj`
0.01112656	`a_obj`
0.01043115	`obj:adjunct`
0.00208623	`xcomp:obj:adjunct`
0.00208623	`obl`
0.00695410	`xcomp`
0.00347705	`xcomp:obj`
0.00208623	`xcomp:adjunct`
0.00208623	`xcomp:de_obj`
6.954E-4	`xcomp:a_obj`
6.954E-4	`xcomp:xcomp`

Table 4.19: LDD path information for `topic_rel` extracted from the MFT.

of coherence, to obtain a list of possible paths for GF. Each possible path, p, has a relative frequency, $P(p|GF)$. The end (or target) of the path p is in the f-structure of the predicate l. The principle of completeness requires that the predicate l subcategorise for the target of p. Therefore, the subcategorisation information for l is retrieved, each instance s associated with a relative frequency $P(s|l)$. The path with the highest ranking $P(p|GF) \times P(s|l)$ is the resolution of the long-distance dependency for GF.

The results of this technique are given in Table 4.22. All scores are reported with coordination distribution. For completion, we also report in Figures 4.20 and 4.21 the triples evaluation of the technique on gold MFT-comm trees and the SVM approximation of gold MFT-comm trees.

Observe that the scores for BitPar are all poorer than in the simplified parsing architecture (on MFT-fct and A-MFT-fct). However, Bikel's parser in the pipeline architecture sometimes achieves slight gains from separating the process of LDD resolution. In fact, the overall highest f-score of 86.73 measures on all features is achieved with the separation of LDD resolution, using Bikel's parser under the pipeline approximation.

features	**precision**	**recall**	**f-score**
all	99.2	95.75	97.44
preds only	99.01	95.77	97.37

Table 4.20: LFG triples evaluation of LDD resolution on MFT-comm gold trees

treebank version	**features**	**precision**	**recall**	**f-score**
A-SVM-MFT-comm	all	97.53	88.87	93.00
	preds only	96.02	87.90	91.78
SVM-A-MFT-comm	all	96.64	69.62	80.94
	preds only	95.18	68.88	79.92

Table 4.21: LFG triples evaluation of LDD resolution on SVM approximations on MFT-comm gold trees

parsing architecture	**features**	**parser**	**precision**	**recall**	**f-score**
pipeline	all	BitPar	90.44	77.00	83.18
	all	Bikel	91.67	66.81	77.29
	preds only	BitPar	80.15	69.21	74.28
	preds only	Bikel	82.46	60.71	69.93
integrated	all	BitPar	89.43	64.00	74.61
	all	Bikel	88.76	47.86	62.18
	preds only	BitPar	78.72	58.15	66.89
	preds only	Bikel	77.63	42.84	55.21
SVM pipeline approximation	all	BitPar	90.33	80.82	85.31
	all	Bikel	93.62	80.78	86.73
	preds only	BitPar	82.26	73.89	77.85
	preds only	Bikel	87.40	75.71	81.13
SVM integrated approximation	all	BitPar	91.27	57.43	70.50
	all	Bikel	92.37	58.95	71.97
	preds only	BitPar	83.31	53.10	64.86
	preds only	Bikel	86.59	55.17	67.39

Table 4.22: LFG tiples parsing results with separate LDD resolution.

4.7 Concluding Remarks

In this chapter, I have shown that the techniques applied to other languages for treebank-based grammar acquisition can be successfully adapted to the French case. I introduced a number of innovations with respect to function labeling (using machine learning techniques to predict LFG functional equations for tree nodes) and LDD resolution based on function tag path annotation. I have also acquired evidence that Bikel's parser is more sensitive to larger tag sets than BitPar, overcoming this obstacle by means of integrating a machine learning component for function tag and f-structure annotation. Tables 4.23 and 4.24 present summaries of the best f-scores achieved in the parsing experiments presented in this chapter.

features	**parser**	**parsing architecture**	**coord dist**	**score**
all	Bikel	SVM approximation with separate LDD resolution	yes	86.73
preds only	Bikel	SVM approximation of simplified	yes	81.35

Table 4.23: Summary of best results for (the approximation of) the pipeline architecture.

features	**parser**	**parsing architecture**	**coord dist**	**score**
all	BitPar	simplified	no	75.15
preds only	Bikel	SVM approximation of simplified	yes	67.88

Table 4.24: Summary of best results for (the approximation of) the integrated architecture.

An avenue for future work concerns further adaption of the LDD resolution for French. Indeed, not all path functions in the MFT-fct correspond to communicative local features. The phenomenon of *de*-phrase extraction from NPs provides the exception. Future research on treebank-based grammar acquisition for French should work towards a targeted account of this phenomenon.

Chapter 5

Direct Parsing into F-Structure Dependencies

5.1 Introduction

Research on automatically obtained wide-coverage deep grammars for natural language processing, given reliable and large CFG-like treebanks, within the Lexical Functional Grammar framework are typically based on an extended PCFG parsing architecture from which dependencies are extracted. However, recent developments in statistical dependency parsing Nivre et al. (2006); McDonald et al. (2005) suggest that such deep grammar approaches to statistical parsing could be streamlined in terms of a novel approach where strings are directly parsed into LFG f-structures, effectively obviating the CFG-based parsing step in traditional treebank-based (and hand-crafted) approaches to LFG parsing. In this chapter, I present my research on this novel approach to deep grammar parsing within the framework of LFG, for French, showing that best predicates only results (an f-score of 69.46) for the established integrated parsing architecture can in fact be obtained by the direct dependency parsing approach.[1]

This chapter presents a *mise-en-scène* between theoretical dependency syntax and de-

[1]This work was previously published as (Schluter and van Genabith, 2009).

pendency parser practical requirements, an *entrée en scène* for f-structures in the literature for dependency parsing, an approach to representing f-structures in LFG as pseudo-projective dependencies, a first attempt to reconcile parsing LFG and dependency parsing, and, finally, the first treebank-based statistical dependency parsing results for French.

I begin with the discussion of LFG f-structure dependencies, comparing previously mentioned theoretical frameworks for statistical dependency parsing in the literature and showing their pseudo-projectivity (Section 5.2). In Section 5.3 I describe the data conversion involved in this research. In Section 5.4 I overview the dependency parsing architecture adopted. Finally, in Section 5.5 I discuss the dependency-parsing extension of the established LFG parsing architectures and present and discuss the parsing results.

5.2 LFG F-structure Dependencies

In this section, I present an overview of the target frameworks for previous conversions of CFG-like data into dependency tree data (Section 5.2.1). I then consider projectivity in light of these conversions, and explain why projectivity need not be a problem in LFG dependency parsing as a result of the f-structure's property of pseudo-projectivity (Section 5.2.2).

5.2.1 A Comparison of Theoretical Frameworks

In the statistical dependency parsing literature, there are generally two sources of modern linguistic theoretical justification behind parsing models: the theoretical framework of Meaning-Text Theory (Mel'čuk, 1998), and the annotation guidelines of the Prague Treebank (Hajič et al., 1999). Moreover, software converting phrase-structure style treebanks into dependencies (for example, Nivre et al. (2007); Johansson and Nugues (2007)) for statistical dependency parsing usually quote these two annotation styles in the treatment of hard cases. Therefore, if our aim is to directly parse strings into LFG f-structures (obviating the CFG parsing step in standard LFG architectures), it is vital to consider what sorts of dependencies existing dependency parsers were intended to parse.

Meaning-Text Theory (MTT) represents the syntactic organisation of sentences strictly by dependencies. Under this framework, syntax is separated into surface and deep syntactic dependency-based tree representations. The deep-syntactic structure of a sentence has nodes that are semantically full lemmata (full lexemes), abstracting away from any auxiliary or structural lemmata at this level. Also, lemmata are subscripted by the grammatical information (grammemes) expressed by their associated word-form(s), rather than those imposed by government and agreement. Arcs are labeled by a selection of around ten language-independent relations. On the other hand, the surface-syntactic structure of a sentence contains all lemmata of the sentence and its arcs are labeled with the names of language-specific surface-syntactic relations, each of which represents a particular construction of the language (Mel'čuk, 2003, 1998). Furthermore, communicative functions such as `topic` or `focus` are not associated with a pure syntactic structure in the Meaning-Text Theory (Mel'čuk, 2001). On the other hand, re-entrancies in the deep-syntactic representation associated with coreference (belonging to the Deep-Syntactic Anaphoric Structure) are possible, as in Figure 5.2.

Figures 5.1 and 5.2 respectively show the surface and deep-syntactic structure for Example (1) (taken from (Mel'čuk, 2003)).

(1) For decades, cocoa farming has escaped such problems by moving to new areas in the tropics.

For Figure 5.2, the Roman numerals on arcs indicate the actant (or argument) number, ATTR indicates a modifier relationship, and in both figures, subscripted information provides essentially grammatical features of the lemmata in the sentence. The dotted line indicates a coreference link between the two occurrences of the lemma FARMING (subject raising). The structure in Figure 5.1 is just a projective dependency tree, whereas the structure in Figure 5.2 is a non-projective DAG.

Note that there is no existing treebank constructed within the framework of MTT. There-

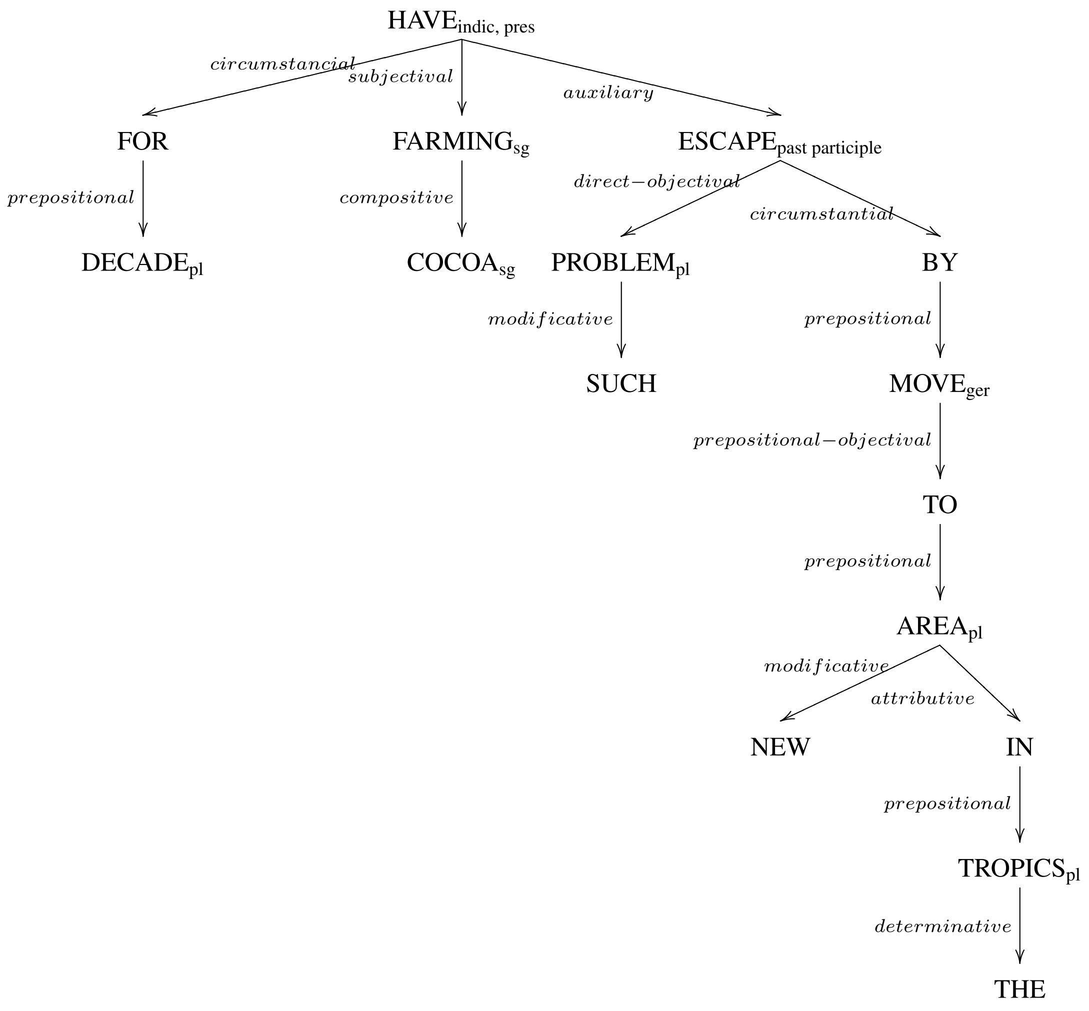


Figure 5.1: MTT surface-syntactic structure for Example (1).

fore, no statistical parsing has ever been carried out after training on MTT structures (only on perhaps a very small subset of structures transformed to resemble some aspects of MTT syntax).[2] On the other hand, the data and annotations in Prague Treebank have been used many times in statistical parsing experiments.

The Prague Treebank (PT) annotation guidelines Hajič et al. (1999) also distinguishes between two levels of dependency-based syntactic representation: analytical and tectogrammatical. These guidelines are written in the spirit of Functional Generative Description.[3]

[2]For example, Nilsson et al. (2006) explores coordination and verb group transformations resembling MTT syntactic representations in statistical dependency parsing.

[3]See, for example, (Hajičová and Sgall, 2003) for a discussion of dependency syntax according to Functional

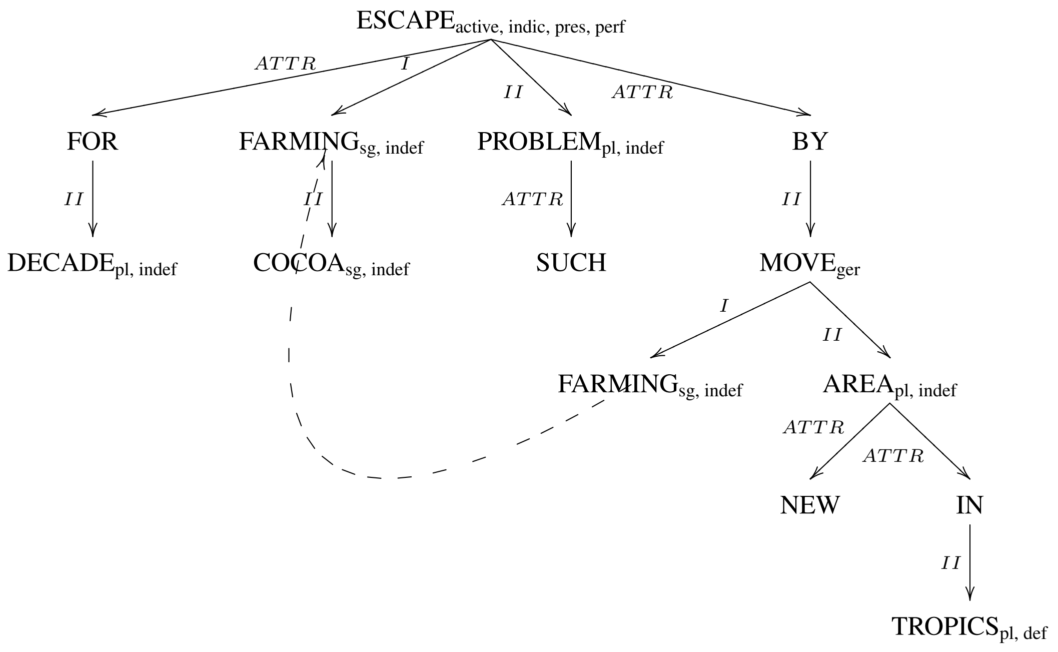


Figure 5.2: MTT deep-syntactic structure for Example (1).

These two levels of syntactic representation roughly correspond to those of the Meaning-Text Theory—the analytical level corresponding to the surface-syntax of the MTT and the tectogrammatical level corresponding to the deep-syntactic level of the MTT (Žabokrtský, 2005). In the PT, word-forms have attributes for their lemmata as well as for grammatical and lexical information expressed morphologically. The syntactic structure of the treebank is given for the analytic level of representation, though work is under way on complementing this with a tectogrammatical level of representation (Sgall et al., 2004). Also similarly to the MTT, communicative structure is not associated with pure syntax in Functional Generative Description, and therefore does not figure among annotations defined for the PT.

Figure 5.3 shows the syntactic structures specified by Hajič et al. (1999) for Example (2), taken directly from the annotation guidelines. Arcs are labeled with syntactic relations defined in the annotation guidelines.

(2)	Jeho	výklad	je,	že	mají	hrát.
	his	interpretation	is	that	they ought	to play

Generative Description.

His interpretation is that they ought to play.

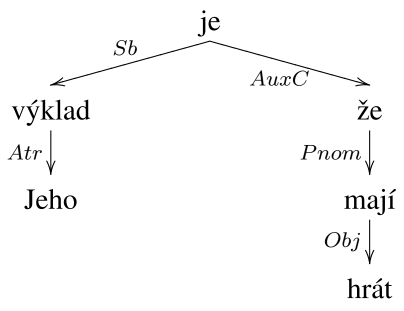


Figure 5.3: PT syntactic structure for Example (2).

LFG does not have a uniform dependency syntax, instead distinguishing between c-structure and f-structure. These two systems express different sorts of information, represented by means of phrase-structure trees, for the c-structure, and dependency DAGs, for f-structures. F-structure is an abstract functional syntactic representation of a sentence, thought to contain deeper or more language-independent information than the c-structure (Dalrymple, 2001).

There are several important ways in which f-structures differ from the tree-dependencies outlined in the literature on dependency syntax within the MTT framework or the annotation guidelines of the Prague Treebank. For instance, f-structures can include communicative information, such as `topic` and `focus`, that LFG theorists consider to be grammaticised or syntacticised components of *information structure*. This introduces the notion of long-distance dependencies. Moreover, subject and object-raising are represented with re-entrancies at the f-structure syntactic level of description in LFG, as in the deep-syntactic MTT representation. This creates DAGs rather than just dependency trees, since some grammatical functions share the same f-structure value; these shared f-structures are called *re-entrancies*. In fact, f-structure syntax corresponds, to a sort of mix of surface and deep dependency MTT syntax (respectively, a mix of analytic and tectogrammatical syntax in Functional Generative Description). Like a surface dependency syntax, some lemmata, like copular verbs, that are not semantically full, appear in f-structures. On the other hand, like deep dependency syntax, some lemmata that are not semantically full are excluded (for example, for the monoclausal treatment of compound tenses of verbs) and re-entrancies are represented.

Other differences between dependency structures may be found in the notions of grouping and sets. In particular, coordination receives different treatments that must be considered. According to the PT annotation guidelines, coordination is treated as sets (conjuncts are sister nodes, elements of a set of conjuncts). Also, every node of a dependency tree must be associated with a word-form, which makes the coordinating conjunction or punctuation the governor of the set. On the other hand, in the MTT, coordination has a cascaded representation, with the first conjunct as governor. To distinguish between modifiers or arguments of the first conjunct and those of the coordinated structure, MTT theorists resort to *grouping*: the first conjunct essentially forms a distinguished group with its modifiers and arguments, much like the notion of constituent (Mel'čuk, 2003). In this sense, the first conjunct grouping is really the governor of the coordination.[4] Also according to the MTT, every node of a dependency tree must be associated with a word-form. But in LFG this is not necessary, in particular, in the representation of coordination; coordinated elements, like in the PT, are treated as sets. In DAG form, it can be seen that these coordinated structures have a *null* governor; that is, they do not have a governor that corresponds to any word-form as the node has no label. Because today's statistical dependency parsers cannot handle null elements, some pre-processing will be needed to convert my LFG representation of coordination (Section 5.3.1).

Finally, f-structures may be specified in terms of annotated c-structures with the local meta-variables $\uparrow$ and $\downarrow$, and grammatical function regular paths. This restricts the structure of dependencies actually occurring in LFG f-structure syntax, as we will show in Section 5.2.2.

[4]Grouping may be indicated on labels (Nilsson et al., 2006).

5.2.2 The Breadth of Functional Equations in LFG

LFG's f-structures often have re-entrancies (or shared sub-f-structures)—two functional equations resolve to take the same (f-structure) value—making them DAGs, rather than simple dependency trees. In LFG, the term *functional uncertainty* describes the uncertainty in the resolution given a simple grammatical function, in the definition of the grammar. The set of options for resolution may be finite and given by a disjunction, in which case resolution is down a chain of f-structure nodes of bounded length, or (theoretically) infinite in which case they are given by a regular expression (including the Kleene star operator) and resolution is down a chain of f-structure nodes of unbounded length. We note, however, that in statistical parsing of f-structures, the functional uncertainty in the resolution of a grammatical function will never be infinite, since the data is finite.

5.2.2.1 Projectivity

Consider a labeled dependency tree (directed tree) $T = (V, E, L)$, where V is its set of vertices (or nodes), $E = \{(a, l, b) \mid a, b \in V, l \in L\}$ its set of directed edges, and L the set of labels for edges. If $e = (a, l, b) \in E$, we say that a *immediately dominates* b; in this case, we say that a is the governor of b, or that b is a dependent on a. We say that v_1 *dominates* v_n if there is a chain of arcs $e_1, e_2, \ldots, e_{n-1}$, such that $e_1 = (v_1, l_1, v_2), e_2 = (v_2, l_2, v_3), \ldots, e_{n-1} = (v_{n-1}, l_{n-1}, v_n)$. In this case, we also say that v_n is a descendent of v_1 or that v_1 is an ancestor of v_n.

An *ordered tree* is a tree having a total order, $(V, \leq)$, over its nodes, which for dependency trees is just the linear order of the symbols (or natural language words) in the generated string. An edge $e = (a, l, b)$ *covers* nodes $v_1, v_2, \ldots, v_n$ if $a \leq v_1, \ldots, v_n \leq b$, or $b \leq v_1, \ldots, v_n \leq a$.

An edge, $e = (v_1, l, v_2)$, of a tree is said to be projective if and only if for every vertex v covered by e, v is dominated by v_1. A tree T is projective if and only if all its edges are projective (Robinson, 1970). Gaifman (1965) explains that a projective dependency tree can be associated with a dependency tree whose constituents are the projections of the nodes of the dependency tree, showing that projectivity in dependency trees corresponds to constituent continuity in phrase-structure trees.

These definitions are easily extended to DAGs. However in the case of DAGs, there are sometimes two governors for a single node that must be considered. For f-structure DAGs, we must additionally consider the mixed surface/deep dependency structure: some lemmata do not appear in f-structures as predicates. For those f-structure DAGs for which there is a one-to-one correspondence between predicates and original word-forms, these extended definitions may easily be applied.

However, LFG's treatment of long-distance dependency resolution and of subject/object raising is non-projective. Consider the following example (Example (3)), adapted from Example (4) of Chapter 3 (and from the corresponding f-structure in Figure 3.18).

(3) l'autre est passionné de vieux papiers.
the other is fascinated of old papers.
'*the other has a passion for historic documents*.'[5]

Example (3) has the dependency structure (f-structure but with word-forms rather than predicates and other LFG features) in Figure 5.4, which is non-projective: referring to nodes by their word forms, the edge (*passionné*, `subj`, *l'autre*) covers the node *est*, but *passionné* does not dominate *est*.

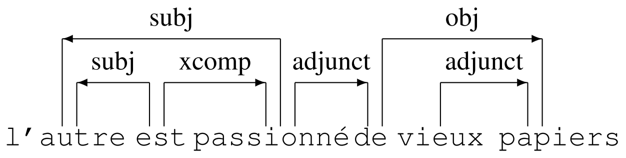


Figure 5.4: (Non-projective) dependency graph for Example (3).

For French, another interesting non-projective structure is found in *en* pronouns and NP

[5]Adapted from sentence 919, file cflmf7ak2ep.xd.cat.xml of the MFT.

extraction. In fact, we can replace the phrase *de vieux papiers* in Example (3) to obtain the following phrase (Example (4)).

(4)	l'autre	en	est	passionné.
	the other	of it	is	fascinated.
	'*the other has a passion for it.*'[6]			

Example (4) then has the dependency structure in Figure 5.5 which non-projective for a second reason: the edge (*passionné*, `adjunct`, *en*) covers *est*, but, again, *passionné* does not dominate *est*.

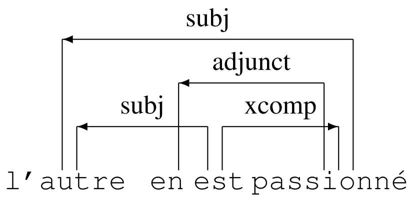


Figure 5.5: (Non-projective) dependency graph for Example (4).

Projectivity in dependency trees or syntax DAGs is obviously a result of the definition of the generating dependency grammar. This is true also of cases that are not like LFG re-entrancies. For example, Johansson and Nugues (2007) propose a conversion of the Penn Treebank into dependency trees that introduces more non-projective edges than the conversion proposed by Yamada and Matsumoto (2003) and Nivre (2006). In addition to long-distance dependencies, for example, their representation of gapping always introduces non-projective branches (Johansson and Nugues, 2007).

LFG is capable of locally representing non-projective dependencies in phrase structures, which should, by definition, be impossible. This is because the only types of non-projective dependencies theoretically represented in LFG are actually pseudo-projectivities.

[6]Adapted from sentence 919, file cflmf7ak2ep.xd.cat.xml of the MFT.

5.2.2.2 Non-Projectivity and Pseudo-Projectivity

Dependency trees also model non-projective structures that have no correspondence with any constituent trees—that is, they may be non-projective. This added "increase" in power for dependency grammars is shown to be useful for syntactic representations of certain languages (for example, the cross-serial dependencies of Dutch). However, as Kahane et al. (1998) explain, pseudo-projective dependency trees may be parsed as projective trees with the aid of a simple transformation.[7]

Consider three non-projective labeled dependency trees, $T_1 = (V, E_1, L_1), T_2 = (V, E_2, L_2)$, and $T_3 = (V, E_3, L_3)$. T_2 is called a *lift* (of T_1) if one of the following conditions hold, for some $e = (a, l, b), e' = (b, l', c) \in E_1$.[8]

(L1) $E_2 = (E_1 - \{e'\}) \cup \{(a, l : l', c)\}, L_2 \subseteq L_1 \cup \{l : l'\}$, or

(L2) T_3 is a lift of T_1 and T_2 is a lift of T_3.

A labeled ordered dependency tree T is said to be *pseudo-projective* if there is some lift T' of T that is projective.

Corresponding to (L1), the action of creating the tree T_2 from T_1 by removing the edge e' (adjacent to e) and adding the edge e'', will be referred to as *lifting*. Note that lifting results in path labels: in (L1), lifting for edge e' (adjacent to e) results in a new edge $e'' = (a, l : l', c)$, with path label $l : l'$. Building a projective tree by means of lifting results in arcs with path labels. Projecting the nodes would result in a sort of annotated c-structure. Turning to LFG and abstracting away from any contractions resulting from $\uparrow=\downarrow$ annotations, lifting is the opposite of the correspondence ϕ from c-structure to f-structure.[9] Re-entrancies may simply be considered as complex labels. Let us call the transformation opposite to lifting a *de-contraction* (used to undo the lifting transformation). Since

[7]Nivre and Nilsson (2005) carries out parsing experiments based on this notion, with various levels of precision in the transformation description.

[8]This definition is equivalent to the one given in (Kahane et al., 1998), where a lift was defined as in terms of governance for *unlabeled* dependency trees.

[9]Kahane et al. (1998) remark that the idea of building a projective tree by means of lifting can be compared to the functional uncertainty of LFG.

generating an f-structure from an annotated c-structure involves simple contractions or lifts of the form $\uparrow=\downarrow$ and de-contractions, all f-structures are at most pseudo-projective. That means, we do not have to worry about non-projective structures in the direct parsing of LFG dependencies in f-structures.

As illustration, figures 5.6 and 5.7 give projective representations of the non-projective dependency graphs in Figures 5.4 and 5.5, respectively. In particular, the dependency graphs of Figures 5.4 and 5.5 have undergone a lifting of the non-projective edge (*passionné*, `subj`, *l'autre*), adjacent to (*est*, `xcomp`, *passionné*), producing the projective edge (*passionné*, `xcomp:subj`, *l'autre*) in the dependency graphs of Figures 5.6 and 5.7. Also, in the dependency graph of Figure 5.5 has undergone a further lift of the non-projective edge (*passionné*, `adjunct`, *en*), adjacent to (*est*, `xcomp`, *passionné*), producing the projective edge (*passionné*, `xcomp:adjunct`, *en*) in the dependency graph of Figure 5.7.

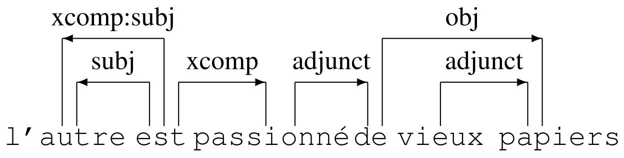


Figure 5.6: Projectivised dependency graph for Example (3).

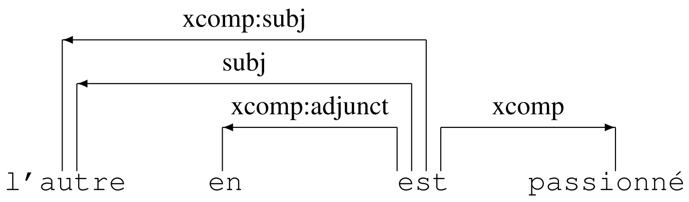


Figure 5.7: Projectivised dependency graph for Example (4).

5.3 Transforming Annotated C-Structures into Dependency Trees

To generate dependency trees, rather than using f-structures, we start with annotated c-structures. The motivation for this choice is straightforward: we need only carry out a certain number of contractions for the equations $\uparrow = \downarrow$ in order to get a projective dependency tree (rather than just a pseudo-projective dependency tree on which we must perform lifts). Moreover, the association of labels for handling re-entrancies is sitting in the annotated tree and does not need to be re-calculated. There are some problems that remain in the result.

Firstly, not every terminal will get have a predicate annotation. For example, in causative constructions like for the phrase *faire danser* ('to make dance'), the word-form *faire* would only be annotated with the feature $\uparrow$ `factive` $= +$, not as a predicate. These will simply be turned into f-structures rather than features, by changing annotations such as these to $\uparrow$ `factive:pred` $=$ '$faire'$. For predicates only evaluation against the MFTDB, which is an LFG gold standard, these f-structures will be thrown out.

Another problem is that coordination structures have no governor. These structures must be transformed. We choose to follow the annotation guidelines for the PT for this transformation, due to its similarity with LFG analyses. Some coordination structures of the treebank need alternative treatment. In particular, non-constituent coordination and unlike constituent coordination require analyses that are not covered in the those guidelines. We resort to extended dependency tag sets to treat these cases and retain projectivity.

5.3.1 Coordination Transformations

In general, coordination will be transformed in the spirit of the PT annotation guidelines. If there is a coordinating conjunction, then the last of these will be taken as the governor of the coordination, as in Figure 5.8. In the case where there is no coordinating conjunction but there is coordination punctuation (like a comma or semicolon), we will take the last of these as the governor. Otherwise we will take the first conjunct of the coordination as the governor and revert to grouping through extended labels.

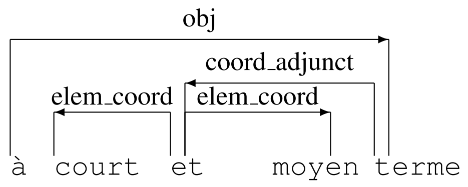


Figure 5.8: Dependency graph for *a court et moyen terme* (‘short and mid-term’).

For non-constituent coordination, the goal is twofold: (1) show that the different elements of each of the conjuncts belong together[10] and (2) show that they are missing something that is present in the first conjunct (done by the function tags). For this reason, the LFG analysis is ideal. LFG allows groupings of elements based on dependence on an item that is physically there or not (i.e., by means of predicate-less (sub-)f-structures). However, a surface dependency analysis cannot do this; constituent structure is not simply dependency structure that projects lexical units to terminals. To make up for this impossibility, extended labels are used, forcing a ”fake” lexical head.

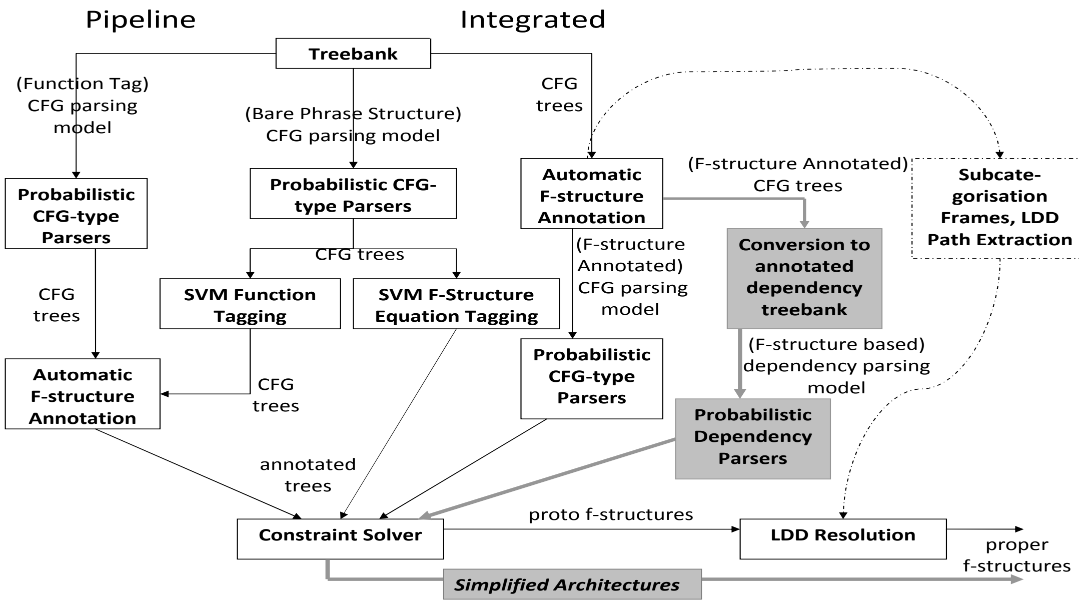
With the conversion of the f-structure annotated c-structures into dependency trees described here (especially with the use of extended labels for the transformation of various coordination analyses), we have more than doubled the size of the f-structure tag set, from 69 tags to 152 tags for the simplified architecture. This will obviously have a detrimental effect on parsing scores.

5.4 Dependency Parsing Architecture

The new direct f-structure dependency parsing architecture works as follows. The f-structure annotation algorithm is applied to MFT trees, creating f-structure annotated trees that are then transformed into the projective dependency representation described in Section 5.3, using the c-structure with the (only) f-structure equations. A dependency parser is then trained

[10]The dependency treatment of coordination outlined by Johansson and Nugues (2007) for the treatment of gapping also introduced ambiguity for the case where there are more than two conjuncts; in their approach, they have removed the relation that the components of gapping are part of the same element/constituent.

on this data, and the test set parsed. The dependency parser output is then transformed back to f-structure equations, which are evaluated against the f-structure gold standard (MFTDB) along predicate paths only.[11] Figure 5.9 shows the new probabilistic dependency-based integrated architecture that is presented here, in grey, to the right.



Pipeline
Integrated
Treebank
(Function Tag) CFG parsing model
(Bare Phrase Structure) CFG parsing model
CFG trees
Probabilistic CFG-type Parsers
Probabilistic CFG-type Parsers
Automatic F-structure Annotation
(F-structure Annotated) CFG trees
Subcate-gorisation Frames, LDD Path Extraction
CFG trees
SVM Function Tagging
SVM F-Structure Equation Tagging
(F-structure Annotated) CFG parsing model
Conversion to annotated dependency treebank
CFG trees
(F-structure based) dependency parsing model
Automatic F-structure Annotation
CFG trees
Probabilistic CFG-type Parsers
Probabilistic Dependency Parsers
annotated trees
Constraint Solver
proto f-structures
LDD Resolution
proper f-structures
Simplified Architectures

Figure 5.9: Overview of treebank-based LFG parsing architectures. The dependency-based architecture presented in this chapter is in bold grey to the right.

5.5 Evaluation

Two statistical dependency parsers were used for this research: MST parser (McDonald et al., 2005) and MALT parser (Nivre et al., 2006).[12] Experiments were done with the

[11]The construction of conversion software for dependency representation from simple f-structures (not f-structure annotated c-structures) is outside the scope of this thesis. In addition, excepting any hand-correction, the f-structures of the MFTDB are the product of an annotation algorithm that included, in particular, a verb combinatorics module. The monoclausal verbal analysis requires that f-structure outputs have their own sort of f-structure algorithm for the verb combinatorics module, which is outside the scope of this thesis. For these reasons, evaluation is only carried out for predicate paths of the MFTDB (preds-only), not in the format of the dependency conversion and not for all features.

[12]Using the default parameters for both. In particular, for MALT parser, the nivreeager parsing algorithm (with the libsvm learner (Chang and Lin, 2001) and no pre/post-processing) is used and the default feature

simplified architecture (in which long-distance dependencies are given as complex path equations in training), and in the established architecture (with a separate long-distance dependency resolution task). The results are given in Tables 5.1 and 5.2.

Parser	coord dist	precision	recall	f-score
MST	*no*	87.46	54.67	67.28
	yes	87.45	54.66	67.27
MALT	*no*	86.23	52.17	65.01
	yes	86.17	51.95	64.82

Table 5.1: Simplified Architecture Parsing Results.

Parser	LDDs resolved	coord dist	precision	recall	f-score
MST	*no*	*no*	86.90	57.07	68.89
		yes	86.89	57.06	68.88
	yes	*yes*	86.48	58.03	69.46
MALT	*no*	*no*	85.98	51.13	64.13
		yes	86.02	50.9	63.96
	yes	*yes*	86.08	51.62	64.54

Table 5.2: Parsing Results with Long Distance Dependency Resolution.

Best results are obtained by the MST parser when LDD recovery is separated and coordination distribution is carried out. The overall best score of 69.46 for the dependency parsing architecture is also the best predicates only score for the integrated architecture for all work presented in this thesis. We observe, however, the characteristic divergence in precision and recall for the integrated architectures, most probably due to the large dependency tag set. Best overall results, are still obtained via the original PCFG based LFG parsing approach, in the pipeline architecture.

The method described in this chapter has been a labeled dependency parsing approach. It has been shown, for example, in (Chen et al., 2007), that combining the use of an unlabeled dependency parser with a separate machine learning algorithm for dependency labels may improve results. However, the parsers used here are unsuitable for this task; MALT

model that came with that for the 1.0.4 release. For MST parser, the version number is 0.5.0; some important default settings are for projective parsing, the inclusion of punctuation in hamming loss calculation, and a feature scope of 1.

parser and MST parser are essentially labeled dependency parsers (with at least their default feature sets).[13]

5.6 Concluding Remarks

In this chapter, I have reviewed the differences between the f-structures and dependency representations parsed by MALT parser and MST parser, showing that f-structures may be parsed non-projectively. I have shown that best statistical parsing results for French in the integrated LFG parsing architecture are achievable by extending this architecture for statistical dependency parsing (a preds-only f-score of 69.46%). This is 1.58% higher than the best result obtained in the traditional PCFG and SVM approximation based integrated parsing architecture (67.88%) reported in Chapter 4. However, best overall results are still obtained via the original PCFG based pipeline LFG parsing approach (c.f. Chapter 4).

[13]An attempt was made to separate the labelling task and employ MALT parser and MST parser as unlabeled dependency parsers, by using only 'blank' labels and employing the SVM classifier for labeling. The SVM classifier performs well, with accuracy of 96.45% for MFTfct and 96.54% for MFTcomm, using similar features as given in Section 4.5. On the other hand, the dependency parsers achieved discouraging results, as is observed in the following table.

simplified architecture	parser	coord dist	ldds resolved	precision	recall	f-score
yes	MALT	no	n/a	84.45	1.82	3.56
		yes	n/a	84.45	1.82	3.56
	MST	no	n/a	0	0	0
		yes	n/a	0	0	0
no	MALT	no	no	83.69	1.79	3.50
		yes	no	83.69	1.79	3.50
		yes	yes	83.59	1.81	3.54
	MST	no	no	90.91	0.39	0.79
		yes	no	90.91	0.39	0.79
		yes	yes	88.41	0.40	0.80

Chapter 6

Concluding Remarks

In this thesis, I have presented research on the automatic treebank-based generation of French LFG parsing resources.

In order to carry out this research, a good quality French treebank was necessary. In particular, this thesis has presented the derivation of a new treebank for French from the Paris 7 Treebank—the Modified French Treebank—a cleaner, more coherent treebank with several transformed structures and new linguistic analyses. In doing so, I show that treebank design and quality has a large impact in statistical parsing.

The Modified French Treebank is the data source used for the development of treebank-based automatic deep-grammar acquisition for LFG parsing resources for French. I developed an f-structure annotation algorithm for this treebank (and language). Already established LFG CFG-based parsing architectures were then extended and tested, achieving a competitive best f-score 86.73% for all features and 81.35% pred-only against the MFTDB, in the pipeline architecture. The CFG-based parsing architectures were then complemented with dependency-based statistical parsing, for the direct parsing of French strings into f-structures, effectively providing another type of the integrated parsing method, obviating the c-structure (CFG) parsing step in the traditional treebank-based and hand-crafted LFG parsing architectures. The direct dependency-based parsing approach is a novel contribution, and though it achieved the highest predicates only f-score (69.46%) for the integrated parsing architecture, overall parser performance was significantly higher for the CFG-based pipeline parsing architecture.

The findings of the research presented here on the performance of the LFG parsing architectures for French are in general disagreement with that of the other languages: the integrated architectures generally perform worst for parsing French and the MFT than the pipeline architectures. One plausible explanation would be the exceptionally high score of the f-structure annotation algorithm for the MFT, with respect to that of f-structure annotation algorithms for other languages; indeed, quality parses seems to lead to satisfactory recall. A second plausible explanation relates to the size of the data; the integrated architecture introduces a comparatively large tag set, which, given such a small dataset as the MFT, yields too coarse-grained a (lexicalised) PCFG for, especially, Bikel's parser. Further research would be required to verify these hypotheses.

6.1 Future Work

6.1.1 Treebank-Based Deep-Grammar Induction of Parsing Resources for French

Since the research presented in Chapter 2 was carried out and first published as (Schluter and van Genabith, 2007), the P7T has undergone some changes, the most important of which seems to be the discarding slightly less than half of the treebank trees; for example, Candito and Crabbé (2009) report the P7T to contain only 12531 sentences. However, there has been no account of the syntactic or other structural changes carried out in the treebank, if any have taken place. Parsing experiments with this reduced treebank have been carried out, for example by Candito and Crabbé (2009), showing relative success by using the Berkeley Parser (Petrov and Klein, 2007). Unfortunately, no comparative results have been shown in this recent research using the MFT and the Berkeley Parser. This is one avenue of future research with respect to treebank-based deep-grammar induction of parsing resources for French: if better parses are achievable on the MFT using the Berkeley

Parser and since best triples scores to date for French have been obtained via the pipeline architecture, the Berkeley Parser should translate to even more competitive results within, at least, this architecture.

Future research in treebank-based deep-grammar induction of parsing resources for french could also study the new version of the P7T to determine whether any significant changes have been carried out in this treebank, converting it into a usable and reliable resource for deep-grammar induction.

6.1.2 Related Areas

A natural extension and completion for developing LFG deep-grammar resources for French would be developing LFG generation resources for French and evaluating (via automatic means) the extracted lexicon presented in this thesis.

LFG Generation Resources for French. A specific research aim for GramLab is the use of the treebank-based multilingual LFG resources for a statistical machine translation system that (1) parses f-structures of a source language from raw text, (2) "translates" these f-structures to f-structures of a target language automatically, and (3) generates surface realisations for the f-structures in the target languages.

As a contribution to the GramLab project Graham et al. (2009) designed and implemented a statistical deep grammar transfer decoder as the transfer component to a transfer-based machine translation system.[1]

Also arising from work within GramLab, there has been research on statistical generation from f-structures for Chinese and English. This research was based on grammars extracted from the f-structure annotated Penn-II and Penn Chinese Treebanks for parsing resources (Cahill and van Genabith, 2006b; Hogan et al., 2007; Guo, van Genabith and Wang, 2008; Guo, Wang and van Genabith, 2008). Similar work on generation with treebank-based LFG resources for French has yet to be carried out. Such work would also extend the

[1]See also (Graham and van Genabith, 2009) and (Graham and van Genabith, 2008).

above-mentioned transfer-based SMT system to fully cover French.

French Subcategorisation Lexicon Evaluation. Previous lexicons have been automatically extracted for French, in each case relying on text parsed by rule-based parsers. Chesley and Salmon-Alt (2006) extracted 104 verbs subcategorisation frames, which were evaluated manually using two native French speakers. Messiant et al. (2008) extracted subcategorisation frames for 3297 verbs, testing only 20 of those verbs against a manually derived gold standard dictionary. Both methods are fairly basic and almost identical. A corpus is parsed, a CFG is automatically extracted and augmented with lexical head information. The extracted CFG rules are then filtered in terms of some measure of frequency and these rules are taken as the subcategorisation frames.

The method for subcategorisation lexicon extraction presented in this thesis is superior to that of the methods employed by Chesley and Salmon-Alt (2006) and Messiant et al. (2008) in that more sophisticated means are used (e.g. fully reflecting long-distance dependencies in the data) and that we have a cleaner corpus from which to extract subcategorisation frames (rather than automatically parsed text). Future work would automate the evaluation of extracted lexicons for French through the use of existing electronic dictionary resources for French, in a manner similar to O'Donovan, Cahill, van Genabith and Way (2005)'s use of the Oxford Learner's Dictionary of Current English. The results of this new evaluation method for my extracted lexicon and that of any others that are available (notably that of Messiant et al. (2008)) should then be compared.[2]

[2]Chesley and Salmon-Alt (2006)'s extracted subcategorisation lexicon is not publicly available.

Appendix

Appendix A: List of Atomic-Valued Features

Feature	*Values*
stmt_type	decl, imp, int
aux_select	avoir, être
passive	+/-
refl	+/-
perf	+/-
superperf	+/-
part	+/-
inf	+/-
factive	+/-
mood	cond, imperative, indicative, subjunctive
tense	pres, fut, past, passesimple
degree	comparative, positive, superlative
pos	post, pre
ne	+/-
neg	+/-
neg_form	*wordform of negation*
case	acc, dat, nom
ntype	common, proper, card
pron_form	*lemma information*
pron_type	dem, expl, pers, card, refl, rel, int
pform	*lemma information*
comp_form	*lemma information*
coord_form	*lemma information*
precoord_form	*lemma information*
introducteur_form	*lemma information*
dtype	def, dem, ind, part
anaph_num	sg, pl
agr_num	sg, pl
pred	*lemma information*
gend	fem, masc
num	sg, pl
pers	1, 2, 3

Appendix A: List of Grammatical Functions

Grammatical Function	*Explanation*
`adjunct`	adjunct
`name_mod`	proper noun list of proper noun modifiers
`obj`	object
`subj`	subject
`a_obj`	(indirect) à object
`de_obj`	(indirect) de object
`xcomp`	subordinate clause whose subject is a reentrancy
`comp`	subordinate clause with a subject (not reentrancy)
`coord`	coordination
`obl`	indirect complement
`obl_agt`	indirect complement of passive expression (thematic subject)
`obl_compar`	comparative phrase
`topic_rel`	topic of relative clause
`rel_mod`	relative clause
`focus`	focus
`topic`	topic
`cleft_subj`	non-thematic subject of clefted phrase
`spec`	
`det`	(in)definite determiner
`poss`	possessive determiner
`quant`	quantitative determiner

Appendix C: Head-Finding Rules for the MFT

In this appendix, we give the head-finding rules used in all experiments requiring them in experiments presented in this thesis. One set of head-finding rules is presented for MFT-norm (Figure 6.2) and the other set is presented for MFT-fct (Figure 6.1). Both are presented in the Lisp format required by Bikel's parser, where `tt` and `ss` are used as special strings, inserted where normally a space character or some other special character that is a reserved character for Bikel's parser would normally be.

The rules should be read as follows, for

$$(\text{LHS } (d_1 \text{ ConstList}_1) \cdots (d_n \text{ ConstList}_n)). \tag{6.1}$$

For the root of a depth-one subtree, LHS, for $i \in \{1, \ldots, n\}$ scan for the ith time, the children of LHS in the direction $d_i \in \{l, r\}$ for the constituents in the list (in order of priority) ConstList$_i$. As soon as the head is found the search process stops. The direction l means from left to right and the direction r means from right to left.

```
((SENT* (l VNttfinite) (l COORDttSint) (l COORDttNC) (l))
 (Sint* (l VNttfinite) (l COORDttSint) (l COORDttNC) (l))
 (Ssub* (l COORDttSsub) (l VNttfinite) (l NP* ) (l))
 (Srel* (l COORDttSrel) (l VNttfinite) (l))
 (VPpart* (l COORDttVPpart) (l VNttpart) (l))
 (VPinf* (l COORDttVPinf) (l VNttinf) (l))
 (VP* (l COORDttVP) (l VNttfinite) (l))
 (VNttfinite* (l COORDttVN_finite) (l Vssfinite) (l))
 (VNttinf* (l COORDttVN_inf)(l Vssinf) (l))
 (VNttpart* (l COORDttVN_part) (l Vsspart) (l))
 (NC* (l COORDttNC) (r))
 (CLP* (l CL*) (r))
 (CLseP* (l CL*) (r))
 (AP* (l COORDttAP) (l A Asscard Assrel Assint Assord) (l))
 (NP* (l COORDttNP) (l N Nssrel Nsscard Nssord Nssint PRO
                            PROssrel ET CL) (l))
 (AdP* (l COORDttAdP) (l ADV*) (l))
 (PP* (l COORDttPP) (l P Pssrel Pssint PROssrel) (r))
 (COORDttunary (l VNttfinite) (r))
 (COORD* (r))
 (* (l)))
```

Table 6.1: Head-Finding Rules (with function labels)

```
((SENT (l VNttfinite COORDttSint COORDttNC) (l))
 (Sint (l VNttfinite COORDttSint COORDttNC) (l))
 (Ssub (l COORDttSsub VNttfinite) (l NP AP AdP) (l))
 (Srel (l COORDttSrel VNttfinite) (l))
 (VPpart* (l COORDttVPpart*) (l VNttpart) (l))
 (VPinf (l COORDttVPinf) (l VNttinf) (l))
 (VP (l COORDttVP) (l VNttfinite) (l))
 (VNttfinite (l COORDttVN_finite) (l Vssfinite) (l))
 (VNttinf (l COORDttVN_inf)(l Vssinf) (l))
 (VNttpart (l COORDttVN_part) (l Vsspart) (l))
 (NC (l COORDttNC) (r))
 (AP (l COORDttAP) (l A) (l))
 (APrel (l COORDttAP*) (l Assrel) (l))
 (APint (l COORDttAP*) (l Assint) (l))
 (NP (l COORDttNP) (l N ET PRO Nsscard) (l))
 (NPttint (l COORDttNP) (l Nssint ETssint PROssint) (l))
 (NPttrel (l COORDttNP) (l Nssrel PROssrel) (l))
 (AdP (l COORDttAdP) (l ADV*) (l))
 (AdPttint (l COORDttAdP ADVssint) (l ADV*) (l))
 (PP (l COORDttPP) (l P) (r))
 (PPttrel (l COORDttPP*) (l Pssrel PROssrel) (r))
 (PPttint (l COORDttPP*) (l Pssint PROssint) (r))
 (COORDttNP* (r))
 (COORDttVP (r))
 (COORDttNC (r))
 (COORDttSint (r))
 (COORDttSsub (r))
 (COORDttSrel (r))
 (COORDttAP (r))
 (COORDttAdP* (r))
 (COORDttPP* (r))
 (COORDttVN_part (r))
 (COORDttVN_inf (r))
 (COORDttVN_finite (r))
 (COORDttVPinf (r))
 (COORDttVPpart* (r))
 (COORDttUC (r))
 (COORDttunary (l VNttfinite) (r))
 (* (l)))
```

Table 6.2: Head-Finding Rules (without function labels)

Bibliography

Abeillé, A. (2003). Guide des annotations fonctionnelles, *Technical report*, Université Paris 7.

Abeillé, A. and Barrier, N. (2004). Enriching a french treebank, *LREC Conference Proceedings*, Lisbon.

Abeillé, A. and Clément, L. (2003). Annotation morpho-syntaxique: Les mots simples–les mots composés. corpus *Le Monde*, *Technical report*, Université Paris 7.

Abeillé, A., Toussenel, F. and Chéradame, M. (2004). Corpus le monde: Annotations en constituants. guide pour les correcteurs, *Technical report*, LLF and UFRL and Université Paris 7.

Al-Raheb, Y., Akrout, A., van Genabith, J. and Dichy, J. (2006). An lfg gold standard resource for the arabic penn treebank, *Proceedings of The Challenge of Arabic for NLP/MT*, pp. 105–116.

Alsina, A. (1992). On the argument structure of causatives, *Linguistic Inquiry* **23**(4): 517–555.

Arun, A. and Keller, F. (2005). Lexicalization in crosslinguistic probablisitic parsing: The case of french, *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, MI, pp. 306–313.

Bick, E. (2004). Parsing and evaluating the french europarl corpus, *Méthodes et outils pour l'évaluation des analyseurs syntaxiques (Journées ATALA)*, Paris, pp. 4–9.

Bies, A., Ferguson, M., Katz, K. and MacIntyre, R. (1995). Bracketing guidelines for treebank ii style penn treebank project, *Technical report*, University of Pennsylvania.

Bies, A. and Maamouri, M. (2003). Penn arabic treebank guidelines.
URL: *http://www.ircs.upenn.edu/arabic/Jan03release/guidelines-TB-1-28-03.pdf*

Bikel, D. (2002). Design of a multi-lingual, parallel-processing statistical parsing engine, *Proceedings of the 2nd International Conference on Human Language Technology Research*, San Francisco.

Bikel, D. (2004). Intricacies of collins' parsing model, *Computational Linguistics* **30**: 479–511.

Brants, S., Dipper, S., Hansen, S., Lezius, W. and Smith, G. (2002). The tiger treebank.

Brown, P. F., Della, V. J., Desouza, P. V., Lai, J. C. and Mercer, R. (1992). Class-based n-gram models of natural language, *Computational Linguistics* **18**: 467–479.

Burke, M. (2006). *Automatic Treebank Annotation for the Acquisition of LFG Resources*, PhD thesis, School of Computing, Dublin City University, Dublin 9, Ireland.

Burke, M., Lam, O., Cahill, A., Chan, R., O'Donovan, R., Bodomo, A., van Genabith, J. and Way, A. (2004). Treebank-based acquisition of a chinese lexical-functional grammar, *Proceedings of the PACLIC-18 Conference*, Waseda University, Tokyo, Japan, pp. 101–121.

Butt, M., King, T., no, M.-E. N. and Segond, F. (1999). *A Grammar Writer's Cookbook*, CSLI Publications, Stanford.

Cafferkey, C., Hogan, D. and van Genabith, J. (2007). Multi-word units in treebank-based probabilistic parsing and generation, *Proc. of RANLP 2007*, Boverets, Bulgaria.

Cahill, A. (2004). *Parsing with Automaticaly Acquired, Wide-Coverage, Robust, Probabalistic LFG Approximations*, PhD thesis, School of Computing, Dublin City University, Dublin 9, Ireland.

Cahill, A., Burke, M., Forst, M., Odonovan, R., Rohrer, C., van Genabith, J. and Way, A. (2005). Treebank-based acquisition of multilingual unification grammar resources, *Journal of Research on Language and Computation* **3**(2): 247–279.

Cahill, A., Burke, M., O'Donovan, R., Riezler, S., van Genabith, J. and Way, A. (2008). Wide-coverage deep statistical parsing using automatic dependency structure annotation, *Computational Linguistics* **34**(1): 81–124.

Cahill, A., Burke, M., O'Donovan, R., van Genabith, J. and Way, A. (2004). Long-distance dependency resolution in automatically acquired wide-coverage pcfg-based lfg approximations, *Proc. of ACL04*, Barcelona, Spain, pp. 21–26.

Cahill, A., McCarthy, M., van Genabith, J. and Way, A. (2002a). Automatic annotation of the penn treebank with lfg f-structure information, *in* A. Lenci, S. Montemagni and V. Pirelli (eds), *Proceedings of the LREC 2002 workshop on Linguistic Knowledge Acquisition and Representation*, ELRA, Paris.

Cahill, A., McCarthy, M., van Genabith, J. and Way, A. (2002b). Evaluating automatic f-structure annotation for the penn-ii treebank, *in Proceedings of the Treebanks and Linguistic Theories (TLT02) Workshop*, Kluwer Academic Publishers, Sozopol, Bulgaria, pp. 42–60.

Cahill, A. and van Genabith, J. (2006a). Robust pcfg-based generation using automatically acquired lfg approximations, *Proceedings of the 21st International Conference on Cmoputational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, pp. 1033–1040.

Cahill, A. and van Genabith, J. (2006b). Robust pcfg-based generation using automatically acquired lfg approximations, *Proc. of the joint conference of ICCL and ACL*, Sydney, Australia.

Cakici, R. (2005). Automatic induction of a ccg grammar for turkish, *Proceedings of the ACL Student Research Workshop*, Ann Arbor, MI, USA.

Candito, M.-H. and Crabbé, B. (2009). Improving generative statistical parsing with semi-supervised word clustering, *Proceedings of the 11th International Conference on Parsing Technologies (IWPT 2009)*, Paris, France.

Candito, M.-H., Crabbé, B. and Denis, P. (2010). Statistical french dependency parsing, *Proceedings of LREC 2010*, Malta.

Candito, M.-H., Crabbé, B., Denis, P. and Guérin, F. (2009). Analyse syntaxique du français : des constituants aux dépendances, *Proceedings of TALN 2009*, Senlis, France.

Candito, M.-H., Nivre, J., Denis, P. and Anguiano, E. H. (2010). Benchmarking of statistical dependency parsers for french, *Proceedings of COLING'2010 (poster session)*, Beijing, China.

Chang, C. and Lin, C. (2001). *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics, *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, AAAI Press/MIT Press, Menlo Park.

Chen, J., Bangalore, S. and Vijay-Shanker, K. (2006). Automated extraction of tree-adjoining grammars from treebanks, *Natural Language Engineering* **12**(3): 251–299.

Chen, W., Zhang, Y. and Isahara, H. (2007). A two-stage parser for multilingual dependency parsing, *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLLL 2007*, Prague, pp. 1129–1133.

Chesley, P. and Salmon-Alt, S. (2006). Automatic extraction of subcategorization frames for french, *Proc. of LREC06*, Genoa, Italy.

Chrupała, G. and van Genabith, J. (2006a). Improving treebank-based automatic lfg induction for spanish, *Proc. of LFG06*, Konstanz, Germany.

Chrupała, G. and van Genabith, J. (2006b). Using machine-learning to assign function labels to parser output for spanish, *Proc. of COLING/ACL 2006 Main Conference Poster Sessions*, Sydney, Australia.

Collins, M. (1997). Three generative lexicalised models for statistical parsing, *Proceedings of the 35th Annual Meeting of the ACL*, Madrid, Spain, pp. 16–23.

Crouch, R., Kaplan, R., King, T. H. and Riezler, S. (2002). A comparison of evaluation metrics for a broad coverage parser, *Proc. of the LREC02 Workshop: Beyond PARSEVAL–Towards Improved Evaluation Measures for Parsing Systems*, Las Palmas, Spain, pp. 67–74.

Dalrymple, M. (2001). *Lexical Functional Grammar*, Vol. 34 of *Syntax and Semantics*, Academic Press, San Diego.

Dickinson, M. and Meurers, W. D. (2003a). Detecting errors in part-of-speech annotation, *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL-03)*, Budapest, Hungary, pp. 107–114.

Dickinson, M. and Meurers, W. D. (2003b). Detecting inconsistencies in treebanks, *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*, Växjö, Sweden, pp. 45–56.

Dickinson, M. and Meurers, W. D. (2005). Prune diseased branches to get healthy trees! how to find erroneous local trees in a treebank and why it matters, *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*, Barcelona, Spain.

Frank, A. (2000). Automatic f-structure annotation of treebank trees, *in* M. Butt and T. King (eds), *Proceedings of the LFG00 Conference*, CSLI Publications, Berkeley, California.

Frank, A. and Berman, J. (1996). *Deutsche und Französische Syntax im Formalismus der LFG*, number 344 in *Linguistische Arbeiten*, Niemeyer.

Gaifman, H. (1965). Dependency systems and phrase-structured systems, *Information and Control* **8**: 304–337.

Graham, Y., Bryl, A. and van Genabith, J. (2009). F-structure transfer-based statistical machine translation, *Proceedings of Lexical Functional Grammar Conference 2009*.

Graham, Y. and van Genabith, J. (2008). Packed rules for automatic transfer-rule induction, *Proceedings of the European Association of Machine Translation Conference 2008*, Hamburg, Germany, pp. 57–65.

Graham, Y. and van Genabith, J. (2009). An open source rule indution tool for transfer-based smt, *The Prague Bulletin of Mathematical Linguistics, Special Issue: Open Source Tools for Machine Translation* **91**: 37–46.

Guo, Y., van Genabith, J. and Wang, H. (2007). Treebank-based acquisition of lfg resources for chinese, *Proc. of LFG07*, Stanford, CA.

Guo, Y., van Genabith, J. and Wang, H. (2008). Dependency-based n-gram models for general purpose sentence realisation, *Proceedings of the 22nd International Conference on Computational Linguistics*, Manchester, UK, pp. 297–304.

Guo, Y., Wang, H. and van Genabith, J. (2007). Recovering non-local dependencies for chinese, *Proc. of 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Prague, Czech Republic, pp. 257–266.

Guo, Y., Wang, H. and van Genabith, J. (2008). Recovering non-local dependencies for chinese, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Prague, Czech Republic, pp. 257–266.

Hajič, J., Panevová, J., Buráňová, E., Urešová, Z. and Bémová, A. (1999). Annotations at analytical level. instructions for annotators, *Technical report*, Prague Dependency Bank.

Hajičová, E. and Sgall, P. (2003). Dependency syntax in functional generative description, pp. 570–592.

Hockenmaier, J. (2006). Creating a ccgbank and a wide-coverage ccg lexicon for german, *Proc. of ACL/COLING 2006*, Sydney, Australia.

Hockenmaier, J. and Steedman, M. (2007). Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank, *Computational Linguistics* **33**(3): 355–396.

Hogan, D., Cafferky, C., Cahill, A. and van Genabith, J. (2007). Exploiting multi-word units in history-based probabilistic generation, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Prague, Czech Republic.

Hudson, R. A. (1976). Conjuction reduction, gapping, and right-node raising, *Language* **52**(3): 535–562.

Johansson, R. and Nugues, P. (2007). Extended constituent-to-dependency conversion, *NODALIDA 2007 Conference Proceedings*, Tartu, Estonia, pp. 105–112.

Johnson, M. (1998). Pcfg models of linguistic tree representations., *Computational Linguistics* **24**(4): 613–632.

Kahane, S., Nasr, A. and Rambow, O. (1998). Pseudo-projectivity: a polynomially parsable non-projective dependency grammar, *Proceedings of the 17th international conference on Computational linguistics*, pp. 646–652.

Kaplan, R. M. and Zaenen, A. (1989). Long-distance dependencies, constituent structure, and functional uncertainty, pp. 17–42.

Kűbler, S. (2005). How do treebank annotation schemes influence parsing results? or how not to compare apples and oranges, *Proc. of RANLP*, Borovets, Bularia.

Kűbler, S. and Prokic, J. (2006). Is it really that difficult to parse german?, *Proc. of the Fifth International Workshop on Treebanks and Linguistic Theories*, Prague, Czech Republic.

King, T. H., Crouch, R., Riezler, S., Dalrymple, M. and Kaplan, R. M. (2003). The parc700 dependency bank, *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03*, pp. 1–8.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing, pp. 423–430.

Lin, D. (1995). A dependency-based method for evaluating broad-coverage parsers, *Proceedings of IJCAI-95*, pp. 1420–1425.

M. Marcus, G. Kim, M. M. R. M. A. B. M. F. K. K. and Schasberger, B. (1994). The penn treebank: Annotating predicate argument structure, *Proceedings of the Human Language Technology Workshop*, San Francisco, CA.

MacLeod, C., Grisham, R. and Meyers, A. (1994). The comlex syntax project: The first year, *Proc. of ARPA Workshop on Human Language Technology*, Princeton, NJ, pp. 669–703.

Maier, W. (2006). Annotation schemes and their influence on parsing results, *Proc. of the ACL-2006 Student Research Workshop*, Sydney, Australia.

Maxwell, J. T. and Manning, C. D. (1996). A theory of non-constituent coordination based on finite-state rules, *LFG Conference*, Grenoble, France.

McCarthy, M. (2003). *Design and evaluation of the linguistic basis of an automatic f-structure annotation algorithm for the penn-ii treebank*, Master's thesis, School of Computing, Dublin City University, Dublin 9, Ireland.

McDonald, R., Crammer, K. and Pereira, F. (2005). Online large-margin training of dependency parsers, *Proc. of ACL 2005*.

Mel'čuk, I. (1998). *Vers une linguistique Sens-Texte. Leçon inaugurale*, Collège de France, Paris.

Mel'čuk, I. (2001). *Communicative Organisation in Natural Language. The Semantic-Communicative Structure of Sentences*, Benjamins, Amsterdam and Philadelphia.

Mel'čuk, I. (2003). Levels of dependency in linguistic description: Concepts and problems, **1**: 188–229.

Messiant, C., Kohonen, A. and Poibeau, T. (2008). Lexschem: A large subcategorization lexicon for french verbs, *Proc. LREC 2008*, Marrakech, Morocco.

Miyao, Y. and Tsujii, J. (2005). Probabilistic disambiguation models for wide-coverage hpsg parsing, *Proc. of ACL 2005*, Ann Arbor, Michigan.

Nakanishi, H., Miyao, Y. and Tsujii, J. (2004). Using inverse lexical rules to acquire a wide-coverage lexicalised grammar, *Proc. of the Workshop "Beyond Shallow Analyses—Formalism and Statistical Modelling for Deep Analyses"(IJCNLP04)*, Hainan Island, China.

Nilsson, J., Nivre, J. and Hall, J. (2006). Graph transformations in data-driven dependency parsing, *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pp. 257–264.

Nivre, J. (2006). *Inductive Dependency Parsing*, Springer Verlag.

Nivre, J., Hall, J. and Nilsson, J. (2006). Malt-parser: A data-driven parser generator for dependency parsing, *Proceedings of LREC'06*.

Nivre, J. and Nilsson, J. (2005). Pseudo-projective dependency parsing, *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Ann Arbor, Michigan, pp. 99–106.

Nivre, J., Nilsson, J. and Hall, J. (2007). Generalizing tree transformations for inductive dependency parsing, *ACL '07: Proceedings of the 45th Annual Meeting on Association for Computational Linguistics*, Prague, Czech Republic, pp. 968–975.

O'Donovan, R., Burke, M., Cahill, A., van Genabith, J. and Way, A. (2005). Large-scale induction and evaluation of lexical resources from the penn-ii and penn-iii treebanks, *Computational Linguistics* **31**: 329–366.

O'Donovan, R., Cahill, A., van Genabith, J. and Way, A. (2005). Automatic acquisition of spanish lfg resources from the cast3lb treebank, *Proceedings of the Tenth International Conference on LFG*, Bergen, Norway.

Oya, M. and van Genabith, J. (2007). Automatic acquisition of lexical-functional grammar resources from a japanese dependency corpus, *Proceedings of the 21st Pacific Asia Conference on Language, Information and Computation*, Seoul, Korea.

Park, J. (2006). Extraction of tree adjoining grammars from a treebank for korean, *COLING ACL '06: Proceedings of the 21st International Conference on computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 73–78.

Petrov, S. and Klein, D. (2007). Improved inference for unlexicalized parsing, *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, Association for Computational Linguistics, Rochester, New York, pp. 404–411.

Rehbein, I. and van Genabith, J. (2006). German particle verbs and pleonastic preposition, *Proc. of the Third ACL-SIGSEM Workshop on Prepositions*, Trento, Italy.

Rehbein, I. and van Genabith, J. (2007). Treebank annotation schemes and parser evaluation for german, *Proc. of the EMNLP-CoNLL 2007*, Prague, Czech Republic.

Rehbein, I. and van Genabith, J. (2009). Automatic acquisition of lfg resources for german - as good as it gets., *Proceedings of the 14th International LFG Conference*, Cambridge.

Robinson, J. J. (1970). Dependency structure and transformational rules, **46**: 259–285.

Sadler, L., van Genabith, J. and Way, A. (2000). Automatic f-structure annotation from the ap treebank, *in* M. Butt and T. H. King (eds), *Proc. LFG00 Conference*, CSLI Publications, University of Berkeley, California.

Schluter, N. and van Genabith, J. (2007). Preparing, restructuring, and augmenting a french treebank: Lexicalised parsers or coherent treebanks?, *Proceedings of PACLING 2007*, Melbourne, Australia.

Schluter, N. and van Genabith, J. (2008). Automatic induction of probabilistic lfg resources for french, *Proceedings of LREC08*, Marrakesh, Morocco.

Schluter, N. and van Genabith, J. (2009). Dependency parsing resources for french: Converting acquired lexical functional grammar f-structure annotations and parsing f-structures directly, *Proceedings of NODALIDA 2009*, Odense, Denmark.

Schmid, H. (2004). Efficient parsing of highly ambiguous context-free grammars with bit vectors, *Proceedings of the 20th International Conference on Computational Linguistics*, Geneva, Switzerland.

Seddah, D., Candito, M. and Crabbé, B. (2009). Cross parser evaluation: a french treebanks study, *Proceedings of the 11th International Conference on Parsing Technologies (IWPT 2009)*, Paris, France.

Sgall, P., Panevová, J. and Hajičová, E. (2004). Deep syntactic annotation: Tectogrammatical representation and beyond, *in* A. Meyers (ed.), *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, Association for Computational Linguistics, Boston, MASS, pp. 32–38.

Tounsi, L., Attia, M. and van Genabith, J. (2009a). Automatic treebank-based acquisition of arabic lfg dependency structures, *Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages*, Athens, Greece.

Tounsi, L., Attia, M. and van Genabith, J. (2009b). Parsing arabic using treebank-based lfg resources, *In Proceedings of the 14th International LFG Conference*, Cambridge.

van der Beek, L. (2003). The dutch it-cleft constructions, *Proceedings of the LFG03 Conference*.

Žabokrtský, Z. (2005). Resemblances between meaning-text theory and functional generative description, *in* J. Apresjian and L. Iomdin (eds), *Proceedings of the 2nd International Conference of Meaning-Text Theory*, Slavic Culture Languages Publishers House, Moscow, Russia, pp. 549–557.

Xia, F. (1999). Extracting tree adjoining grammars from bracketed corpora, *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS)*, Beijing, China.

Yamada, H. and Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines, *Proceedings of the 8th International Workshop on Parsing Technologies*, Nancy, France, pp. 195–206.

Yates, N. (2002). French causatives: a biclausal account in lfg, *Proceedings of the LFG02 Conference*, Athens, Greece.

Yoshida, K. (2005). Corpus-oriented development of japanese hpsg parsers, *Proc. of the 43rd ACL Student Research Workshop*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 139–144.

Zaenen, A. and Dalrymple, M. (1996). Les verbes causatifs "polymorphiques": les prédicats complexes en français, *Langages* **30**: 79–95.