Data and Models for Statistical Parsing with Combinatory Categorial Grammar

Julia Hockenmaier

Doctor of Philosophy
Institute for Communicating and Collaborative Systems
School of Informatics
University of Edinburgh
2003

Abstract

This dissertation is concerned with the creation of training data and the development of probability models for statistical parsing of English with Combinatory Categorial Grammar (CCG).

Parsing, or syntactic analysis, is a prerequisite for semantic interpretation, and forms therefore an integral part of any system which requires natural language understanding. Since almost all naturally occurring sentences are ambiguous, it is not sufficient (and often impossible) to generate all possible syntactic analyses. Instead, the parser needs to rank competing analyses and select only the most likely ones. A statistical parser uses a probability model to perform this task. I propose a number of ways in which such probability models can be defined for CCG. The kinds of models developed in this dissertation, generative models over normal-form derivation trees, are particularly simple, and have the further property of restricting the set of syntactic analyses to those corresponding to a canonical derivation structure. This is important to guarantee that parsing can be done efficiently.

In order to achieve high parsing accuracy, a large corpus of annotated data is required to estimate the parameters of the probability models. Most existing wide-coverage statistical parsers use models of phrase-structure trees estimated from the Penn Treebank, a 1-million-word corpus of manually annotated sentences from the Wall Street Journal. This dissertation presents an algorithm which translates the phrase-structure analyses of the Penn Treebank to CCG derivations. The resulting corpus, CCGbank, is used to train and test the models proposed in this dissertation. Experimental results indicate that parsing accuracy (when evaluated according to a comparable metric, the recovery of unlabelled word-word dependency relations), is as high as that of standard Penn Treebank parsers which use similar modelling techniques.

Most existing wide-coverage statistical parsers use simple phrase-structure grammars whose syntactic analyses fail to capture long-range dependencies, and therefore do not correspond to directly interpretable semantic representations. By contrast, CCG is a grammar formalism in which semantic representations that include long-range dependencies can be built directly during the derivation of syntactic structure. These dependencies define the predicate-argument structure of a sentence, and are used for two purposes in this dissertation: First, the performance of the parser can be evaluated according to how well it recovers these dependencies. In contrast to purely syntactic evaluations, this yields a direct measure of how accurate the semantic interpretations returned by the parser are. Second, I propose a generative model that captures the local and non-local dependencies in the predicate-argument structure, and investigate the impact of modelling non-local in addition to local dependencies.

Acknowledgements

Mark Steedman has been a true Doktorvater. I am immensely grateful for his guidance, his time, his patience, his enthusiasm and support throughout these years. It is needless to say that without Mark, the research reported in this dissertation would have been impossible.

Stephen Clark has been a wonderful colleague. Apart from Mark, he has been the person I have worked with the closest for more than the last two years. This research has benefited enormously from all our discussions, and working on this project would not nearly have been as much fun, had Steve not come along. Both Mark and Steve have also provided invaluable feedback on the form and content of this dissertation. Chris Brew was also a wonderful supervisor in the first year of my PhD and before. My second supervisor, Henry Thompson, was also very helpful and encouraging whenever I sought his help. I would also like to thank my examiners, Ted Briscoe and Ewan Klein, for their feedback.

I probably would not have ended up with a dissertation on CCG, had I not been put in an office with Gann Bierner and Jason Baldridge. It was impossible to withstand their enthusiasm, and the time we have spent together in room 16 was among the happiest and most productive I had in Edinburgh. They have not just been great office mates, but also very good friends. I would also like to thank my fellow students (that are too many to name) and my other office mates for their company, in particular Bettina Braun (for bearing with me during the last stressful months in Edinburgh), Geert-Jan Kruijff, Ofir Zussman and Ambrose Nankivell. Miles Osborne was my next-door office neighbour for a long time, and I will miss our conversations and the coffee he offered me on an almost daily basis.

I could not have wished for a better place to do my PhD than Edinburgh, and it is impossible to name everybody who is responsible for making it such a stimulating and friendly environment. I am especially grateful to Bonnie Webber, whose energy and warmth have always been an inspiration. Bonnie has also been an invaluable resource of practical information on life in Philadelphia. I would also like to thank Johanna Moore, in particular for letting me tutor her course. Betty Hughes deserves a special thank-you, not only for helping me with submitting my thesis from across the Atlantic, and for all the chocolates and biscuits she would so often put out in the coffee room.

I was able to present some of the work presented in this dissertation at the Universities of Stuttgart, Cambridge and Pennsylvania as well as at a number of talks at Edinburgh and at the European Graduate College programme in collaboration with the University of the Saarland, and I am grateful for the invitations to these talks and the feedback I received on these occa-

sions. I am also grateful for the feedback I received from Aravind Joshi and Mark Johnson at the ACL 2001 Student Research Workshop in Toulouse.

I would like to thank the members of our team at the 2002 Johns Hopkins Summer Workshop – Mark, Steve, Miles, Rebecca Hwa (my parsing jiejie), Anoop Sarkar, Stephen Baker and Jeremiah Crim – for this summer, which was not just a lot of work, but also a lot of fun. I would also like to acknowledge and remember Paul Ruhlen, the other graduate student on the team, who tragically passed away in June 2003. Thanks also to Frederick Jelinek and Peng Xu for their interest in CCG.

I would like to thank Malte Gabsdil and Jon Herring for making our time on the EACL student board both fun and productive.

During the last weeks in which this dissertation was written, I was already in Philadelphia, and I am greatly indebted to Aravind Joshi for giving me this extra time without which this dissertation would have been much worse, and to Trisha Yannuzzi, Marisa Pagano and Laurel Sweeney for all the help they provided in making this transition as smooth as it has been. I would also like to thank Mitch Marcus for our conversations and his interest in this research. Thanks also to Dan Gildea for wanting to use my parser, and to Carlos Prolo, Alexandra Kinyon, David Chiang and Dan Bikel for interesting discussions. Rashmi Prasad was a great office mate during the very final weeks of this thesis.

Without the support of my friends and flatmates outside of ICCS, Aidan Slingsby, Andrew Oppenheim, Alastair Butler, Chris Heaton, Claudia Krause, Fiona McLay, Hayley Duffin, Katie Greenfield, Jenni Rodd, John Hale, Mark Symonds, Michelle Roots, Nicola van Rijsbergen, Niki Orfanou, Robert Weetman, Sebastian Grob and others, I would not have been able to make it through this PhD without losing my sanity. They have all, at one time or another, ensured that I do not forget that there is more to life than working on one's dissertation. I am also grateful to my friends back home, especially Anja Kleinknecht, Beate Waffenschmidt, Christine Hess and Sibylle Klein for the time we were able to spend together in Germany and Scotland.

My greatest thanks go to my parents for all their love, support and encouragement.

I gratefully acknowledge the financial support provided by EPSRC grant GR/M96889, an EPSRC studentship and the Edinburgh Language Technology Group.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Julia Hockenmaier*)

Table of Contents

Chapter 1

Introduction

> *Strange to say, at that time it never seriously occurred to me that there could be a third approach, namely the one attacking natural languages and ordinary speech with the best methods of theoretical and statistical linguistics, respectively. But then both these disciplines were still in a rather poor state in the late forties.* Bar-Hillel (1964)

Since Bar-Hillel wrote these words, both theoretical and statistical methods in linguistics have advanced greatly. In recent years, programs that use statistical models to analyze a wide range of natural language sentences with high accuracy have been created (eg. Collins (1999), Charniak (2000)). Since natural language is highly ambiguous, it is not sufficient (and often impossible) to return all analyses that a grammar provides for a sentence. Instead, we are only interested in the most likely analyses. Probabilistic models provide the means to formalize the notion of likelihood. However, in order to achieve high accuracy, the parameters of these models are generally estimated from labelled data, and it is only through the availability of large corpora of syntactically annotated sentences such as the Penn Treebank (Marcus *et al.*, 1993, 1994) that such approaches have become possible.

Syntactic analysis, or parsing, forms an integral part of any system which required natural language understanding, since it is a prerequisite for semantic interpretation. However, parsers such as Collins (1999) and Charniak (2000) are based on simple phrase-structure grammars whose syntactic analyses fail to capture long-range dependencies, and therefore do not correspond to directly interpretable semantic representations. More expressive grammar formalisms such as Head-driven Phrase Structure Grammar (Pollard and Sag, 1994) or Lexical Functional Grammar (Bresnan, 1982) require more complex kinds of statistical models (Abney, 1997) that can be difficult to estimate, and only recently have statistical methods (Riezler *et al.*, 2002) been applied to these formalisms.

Based on work by Ajdukiewicz (1935), Bar-Hillel (1953) proposed a grammar formalism in which every syntactic derivation corresponds to a semantic interpretation. This formalism, Categorial Grammar, has been developed further in many ways. One of its extensions, Combinatory Categorial Grammar (Steedman, 2000, 1996), forms the basis of the research reported in this dissertation. Combinatory Categorial Grammar (CCG) provides a particularly simple and semantically transparent treatment of extraction and coordination. The immediate availability of interpretable predicate-argument structure (or logical forms) even for those constructions is what makes CCG particularly attractive for any application which requires semantic interpretation. Despite Bar-Hillel's early comments, there has been very little work on applying statistical techniques to categorial grammars. Osborne and Briscoe (1998), Briscoe (2000) and Villavicencio (2002) consider statistical models for learning categorial grammars. However, the parsers that are presented here and in Clark *et al.* (2002) are the first wide-coverage statistical parsers for categorial grammar. The lexicon that we obtain is one of the largest available lexicons for CCG; the only other attempts at building wide-coverage lexicons for CCG that we are aware of are Doran and Srinivas (1994, 2000), who create a CCG lexicon from the XTAG grammar, a large hand-built Tree-Adjoining Grammar for English, and Villavicencio (1997).

The research presented in this dissertation was guided by the question of how to define the structure of a probability model that is appropriate for statistical parsing with CCG, and how to estimate the parameters of such a model given existing linguistic resources. The first part of this dissertation is concerned with the creation of training data for statistical CCG parsers from the Penn Treebank. The second part of this dissertation addresses the development of probability models for CCG and presents experimental results which demonstrate that a very simple kind of model, generative models over normal-form derivation trees, can achieve state-of-the-art performance (as measured in terms of the recovery of word-word dependencies) on this task.

1.1 The thesis proposed

The main thesis put forward in this dissertation is that generative models of canonical CCG derivations yield state-of-the art results on recovery of word-word dependencies on the standard wide-coverage parsing task for English. This demonstrates that the detailed linguistic information which is often hard to recover from the output of statistical parsers such as Charniak (2000) and Collins (1999), does not necessarily make the task of syntactic analysis significantly harder. On the contrary, it seems to be precisely this information which allows for simpler models to achieve comparable performance. More importantly, this information is what makes the output of a CCG parser directly amenable to semantic interpretation.

The research undertaken in order to arrive at this thesis consisted of the following steps:

- The creation of CCGbank, a corpus of CCG derivations from an existing resource, the Penn Treebank (Marcus *et al.*, 1993).
- The development and definition of probability models that are appropriate for CCG.
- An empirical investigation into the adequacy of the models estimated from this corpus.

CCGbank is a corpus of CCG derivations which is acquired from the Penn Treebank by an algorithm presented in chapter 3. This corpus is used to test and train the parsers developed in this dissertation. Deriving an annotated corpus from an existing resource is a cheap way to obtain a relatively large amount of high-quality labelled training and test data. Furthermore, by using the same sentences as the training and test data used for the standard parsing task for English, the performance of the parsers is easier to compare with parsers trained on the original corpus. Through such a comparison we hope to gain insight not only in the effectiveness of different parametrizations for probabilistic models of grammars, but also in the impact of the underlying representation of grammatical information. However, CCGbank is not only useful for the development of statistical parsers. From it we can obtain a large CCG lexicon, which can be used by any CCG parser. Moreover, since CCG derivations have a corresponding semantic interpretation, the creation of a corpus of CCG derivations can be seen as a first step towards a corpus annotated with logical forms.

The probability models developed in this thesis model CCG derivations. The standard assumption in CCG is that derivations are not a level of representation (Steedman, 2000), and therefore it might seem strange to base a probability model of CCG on derivations rather than the derived structure, such as the underlying predicate-argument structure. However, CCG itself is not a theory of the derived structure. Rather it is a theory of how to construct predicate-argument structure from the surface string, and the only restriction it poses on the derived structure is that it be constructible with the combinatory rules provided by CCG. A statistical model of parse selection that chooses analyses according to the plausibility of the derived structure (ie. the semantic interpretation), is in principle independent of the underlying grammar formalism. One of the aims of this research is to investigate how we can exploit the properties of CCG itself for parse selection. Therefore, modelling the derivations themselves is a natural choice.

One of the properties of CCG is that the same semantic interpretation of a given surface string can be arrived at by multiple syntactic derivations, and this is one of the main challenges for a model of CCG parse selection that is based on derivational structure. The models proposed in this dissertation all assume a particular kind of canonical derivation structure, namely one which prefers function application over composition and type-raising whenever possible. This is also the derivation structure used in CCGbank.

State-of-the-art parsers of phrase-structure grammars also model derivations. However, it is well known that the underlying tree representation plays a crucial role for the performance of a probabilistic model of grammar (Johnson, 1998). CCG derivation trees are very different from the trees of grammars extracted from the Penn Treebank: they are at most binary branching, not arbitrarily flat, and although there are many more category labels in CCG, their use is tightly constrained by the combinatory rules. How the kinds of features that have proven to be important for probabilistic models of Treebank grammars influence the performance of a probabilistic CCG parser is addressed by the experiments presented in chapter 5.

All state-of-the-art parsers incorporate models of word-word dependencies. However, standard Treebank parsers such as Collins (1999) or Charniak (2000) model only dependencies that can be expressed in terms of local trees. Their models do not capture the dependencies involved in extraction, control, and certain coordinate constructions, since such dependencies are difficult to represent in a simple phrase-structure grammar. We demonstrate in chapter 6 that CCG allows us to formulate a consistent, generative model of the word-word dependencies in the predicate-argument structure.

1.2 The structure of this dissertation

Chapter 2 provides an introduction to CCG as a grammar formalism, and reviews existing approaches to (non-statistical) parsing with CCG.

Chapter 3 presents an algorithm to translate the phrase-structure trees of the Penn Treebank to normal-form CCG derivation trees, and gives an overview of the syntactic analyses inherent in CCGbank, the resulting treebank of CCG derivations. Appendix A describes the changes on the original trees that are necessary before translation in order to obtain the desired CCG analyses.

Chapter 4 provides the reader who is not familiar with statistical approaches to parsing with a brief introduction to the necessary concepts of probability theory and summarizes relevant previous work on statistical parsing of English.

Chapter 5 develops a number of generative probability models for CCG derivations. The impact that certain features that have been found useful for phrase-structure based parsers is investigated. The experimental results presented here show that statistical parsers for CCG can obtain a level of performance similar to approaches that are based on less expressive grammars.

Chapter 6 presents a probability model for the word-word dependencies inherent in predicate-argument structure. This model captures directly the local and long-range dependencies inherent in the semantic representation. It performs almost as well as the dependency model presented in chapter 5, despite that fact that the current implementation suffers from an overly aggressive beam search. We argue that a model of this kind is required for languages with freer word order, such as Dutch, since the model of chapter 5 would fail to capture the correct dependencies.

Chapter 2

Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG, Steedman (1996, 2000)), is a grammatical theory which provides a completely transparent interface between surface syntax and underlying semantics. Each (complete or partial) syntactic derivation corresponds directly to an interpretable structure. This allows CCG to provide an account for the incremental nature of human language processing. The syntactic rules of CCG are based on the categorial calculus of Ajdukiewicz (1935) and Bar-Hillel (1953) as well as on the combinatory logic of Curry and Feys (1958). The main attraction of using CCG for parsing is that it facilitates the recovery of the non-local dependencies involved in constructions such as extraction, coordination, control and raising.

This chapter provides a brief introduction to the aspects of CCG that are necessary to understand the remainder of this dissertation. We focus in particular on the definition of categories and the combinatory rules, the role of derivations within CCG, the problem of so-called “spurious ambiguity”, and on parsing algorithms for CCG. The main references for CCG are Steedman (2000, 1996), and the reader is referred to these for further details on other aspects of CCG. Wood (1993) provides a good overview of different variants of categorial grammar.

2.1 Categories and the lexicon

In categorial grammar, words are associated with very specific categories which define their syntactic behaviour. A set of universal rules defines how words and other constituents can be combined according to their categories. Variants of categorial grammar differ in the rules they allow, but categories as defined below are the building blocks of any categorial grammar.

In general, the set of syntactic categories C is defined recursively as follows:

Atomic categories: the grammar for each language is assumed to define a finite set of atomic categories, usually $\mathsf{S}, \mathsf{NP}, \mathsf{N}, \mathsf{PP} ... \in C$

Complex categories: if $\mathsf{X}, \mathsf{Y} \in C$, then $\mathsf{X/Y}, \mathsf{X\backslash Y} \in C$

Complex categories $\mathsf{X/Y}$ or $\mathsf{X\backslash Y}$ are functors with an argument Y and a result X. Here we assume a directional variant of categorial grammar, which differentiates between arguments to the right of the functor (indicated by the forward slash $/$) and arguments to the left of the functor (indicated by the backslash $\backslash$). We follow Steedman's notation to represent complex functor categories. In this notation, the result category always precedes the argument. Hence the category $\mathsf{(S\backslash NP)/NP}$ for transitive verbs in English encodes the information that the verb takes a noun phrase to its right, and another noun phrase to its left to form a sentence. [1]

The lexicon specifies the categories that the words of a language can take. For instance, a categorial lexicon for English might contain the following entries:

(1) *John* $\vdash \mathsf{NP}$
shares $\vdash \mathsf{NP}$
buys $\vdash \mathsf{(S\backslash NP)/NP}$
sleeps $\vdash \mathsf{S\backslash NP}$
well $\vdash \mathsf{(S\backslash NP)\backslash(S\backslash NP)}$

Here, *John* and *shares* are noun phrases. *Buys* is a transitive verb. *Sleeps* is intransitive, as it only takes one NP argument. *Well* can modify *sleeps*, as it takes an intransitive verb (or a verb phrase) as argument to return a constituent of the same category.

As defined above, the set of categories is infinite. However, it is tacitly assumed that the lexicon of any specific language will only use a finite subset of these categories, and in CCG there is a strong interaction between the lexical categories of a language and the combinatory rules allowed in its grammar.[2]

Each syntactic category also has a semantic interpretation whose type must correspond to that of the syntactic category. If one chooses to represent semantic interpretations in the language of the λ-calculus (as is commonly done), then each (syntactic) argument of a complex functor category has a semantic counterpart in the form of a bound variable in the λ-term. Semantic interpretations can be arbitrarily complex objects (as long as there is a correspondence

[1] There is an alternative notation originating in Lambek (1958), which also uses forward slashes and backslashes, but puts the argument "underneath" the backslash. In this notation, a transitive verb becomes $\mathsf{(NP\backslash S)/NP}$

[2] A recent proposal by Baldridge (2002) presents a refinement of CCG, called Multi-Modal CCG, which assumes a universal set of combinatory rules, with all the language-specific variation being accounted for in the lexicon.

to the syntactic category). However, this dissertation is primarily concerned with syntactic analysis, and for our purposes it is sufficient to represent semantic interpretation as simple predicate-argument structures such as the following:

(2) *John* $\vdash \mathsf{NP} : john'$
shares $\vdash \mathsf{NP} : shares'$
buys $\vdash (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP} : \lambda x.\lambda y.buys'xy$
sleeps $\vdash \mathsf{S}\backslash\mathsf{NP} : \lambda x.sleeps'x$
well $\vdash (\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP}) : \lambda f.\lambda x.well'(fx)$

2.2 Function application – AB categorial grammar

The system defined by Ajdukiewicz (1935) and Bar-Hillel (1953) (hence: AB categorial grammar) forms the basis for CCG and all other variants of categorial grammar. In AB categorial grammar, categories can only combine through function application. In directional variants of categorial grammar, there are two versions of function application, respecting the directionality of the slash in the syntactic category. However, their effect on the semantic interpretation is the same:[3]

(3) *Forward Application*:
$\mathsf{X}/\mathsf{Y} : f \quad \mathsf{Y} : a \quad \Rightarrow \quad \mathsf{X} : f(a)$
Backward Application:
$\mathsf{Y} : a \quad \mathsf{X}\backslash\mathsf{Y} : f \quad \Rightarrow \quad \mathsf{X} : f(a)$

The rule of forward application states that if a constituent with category X/Y is immediately followed by a constituent with category Y, they can be combined to form a constituent with category X. Analogously, backward application allows a constituent $\mathsf{X}\backslash\mathsf{Y}$ that is immediately preceded by a constituent with category Y to combine with this constituent to form a new constituent of category X.

A string α is considered grammatical if each word in the string can be assigned a category (as defined by the lexicon) so that the lexical categories of the words in α can be combined (according to the rules of the grammar) to form a constituent. The process of combining constituents in this manner is called a derivation, although I will also sometimes refer to the syntac-

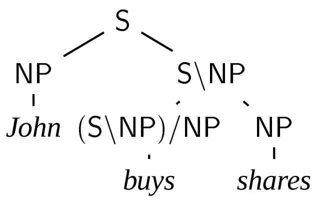
[3] Categorial grammar rule schemata are traditionally presented in a bottom-up fashion. In the remainder of this dissertation, rules that are presented in the form $\alpha \Rightarrow X$ are written in bottom-up manner.

tic structure constructed by this process as a derivation. Since derivations proceed bottom-up, starting from the words, they are usually represented in the following manner:

(4)
$$\begin{array}{ccc}
\textit{John} & \textit{buys} & \textit{shares} \\
\cline{1-1}\cline{2-2}\cline{3-3}
\mathsf{NP} : \textit{John}' & (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP} : \lambda x.\lambda y.\textit{buys}'xy & \mathsf{NP}\textit{shares}' \\
\cline{2-3}
 & \multicolumn{2}{c}{\mathsf{S}\backslash\mathsf{NP} : \lambda y.\textit{buys}'\,\textit{shares}'\,y \quad >} \\
\cline{1-3}
\multicolumn{3}{c}{\mathsf{S} : \textit{buys}'\,\textit{shares}'\,\textit{John}' \quad <}
\end{array}$$

But note that the constituent structure induced by this derivation corresponds to a tree:

(5)



In this dissertation, derivations will often be represented as trees, since this representation follows naturally from the translation algorithm of chapter 3 and the models of chapter 5 and 6. The correspondence to the more traditional notation is assumed to be understood.

2.3 The combinatory rules of CCG

CCG extends AB categorial grammar by a set of rule schemata based on the combinators of combinatory logic (Curry and Feys, 1958). These combinatory rules enable a succint analysis of the long-range dependencies involved in extraction and coordination constructions.

Syntactically, they allow analyses of extraction and coordinate constructions which use the same lexical categories for the heads of such constructions as in the canonical case. Semantically, they guarantee that non-local dependencies fill the same argument slots as local dependencies.

Vijay-Shanker and Weir (1994) demonstrate that both CCG and Tree-Adjoining Grammar (TAG, Joshi *et al.* (1975)) are weakly equivalent to Linear Indexed Grammar (LIG) and belong to a family of languages whose generative power they identify as “mildly context-sensitive”. Therefore, CCG is more expressive than AB categorial grammar, which has been shown by Bar-Hillel *et al.* (1960) to be context-free. Note that combinatory logic itself does not impose any restrictions on the generative power of such combinatory rules. However, Steedman (2000, p. 54) advocates the following principles to which all combinatory rules must adhere in order to keep the generative power of the grammar under control:

The Principle of Adjacency Combinatory rules may only apply to finitely many phonologically realized and string-adjacent entities

The Principle of Consistency All syntactic combinatory rules must be consistent with the directionality of the principal function.

The Principle of Inheritance If the category that results from the application of a combinatory rule is a function category, then the slash defining directionality for a given argument in that category will be the same as the one(s) defining directionality for the corresponding argument(s) in the input function(s).

Composition allows two functor categories to combine to form another functor, whereas type-raising is a unary rule which reverts the roles of functor and argument by allowing an argument category X to change into a functor category $\mathsf{T/(T\backslash X)}$ (or $\mathsf{T\backslash(T/X)}$), where $\mathsf{T\backslash X}$ can be instantiated by any functor category that takes X as argument.

(6) a. *Forward Composition*:

$$\mathsf{X/Y} : f \quad \mathsf{Y/Z} : g \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X/Z} : \lambda x.f(g(x))$$

b. *Forward Crossing Composition*:

$$\mathsf{X/Y} : f \quad \mathsf{Y\backslash Z} : g \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X\backslash Z} : \lambda x.f(g(x))$$

c. *Backward Composition*:

$$\mathsf{Y\backslash Z} : g \quad \mathsf{X\backslash Y} : f \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X\backslash Z} : \lambda x.f(g(x))$$

d. *Backward Crossing Composition*:

$$\mathsf{Y/Z} : g \quad \mathsf{X\backslash Y} : f \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X/Z} : \lambda x.f(g(x))$$

(7) a. *Forward Type-raising*:

$$\mathsf{X} : a \quad \Rightarrow_{\mathbf{T}} \quad \mathsf{T/(T\backslash X)} : \lambda f.f(a)$$

where $\mathsf{T\backslash X}$ is a parametrically licensed category for the language.

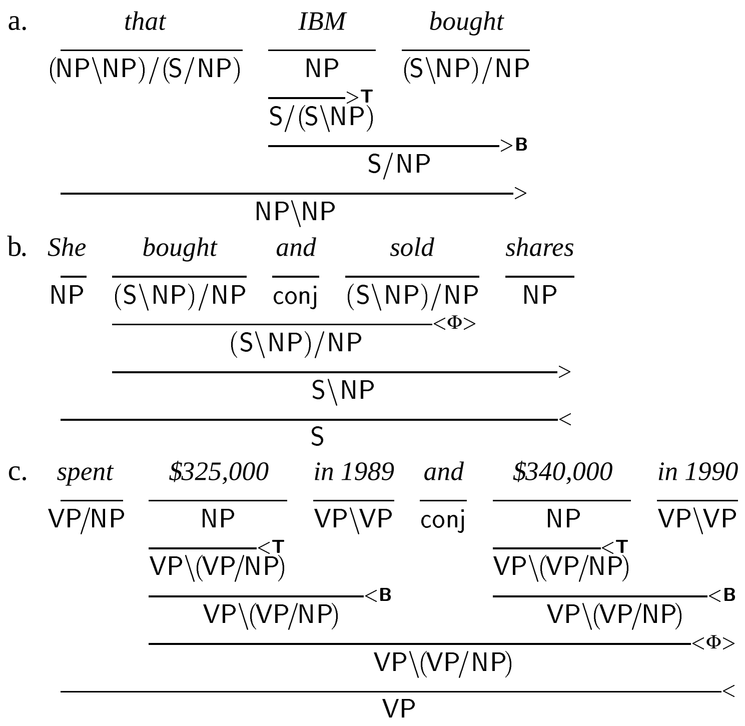
b. *Backward Type-raising*:

$$\mathsf{X} : a \quad \Rightarrow_{\mathbf{T}} \quad \mathsf{T\backslash(T/X)}\lambda f.f(a)$$

where $\mathsf{T/X}$ is a parametrically licensed category for the language.

Composition and Type-raising interact to capture the kinds of long-distance dependencies involved in extraction (8a) and right node raising (8b) as well as argument cluster coordinations (8c) among others:

(8)



Note that in all of these constructions the verbs, *bought* and *spent*, have the same lexical categories as when their arguments are in canonical position.

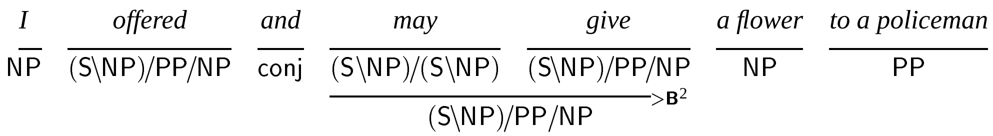
In fact, in CCG all bounded and unbounded dependencies are projected from the lexicon, something which is expressed by the following two principles (Steedman, 2000, p.32):

The Principle of Lexical Head Government: Both bounded and unbounded syntactic dependencies are specified by the lexical syntactic type of their head.

The Principle of Head Categorial Uniqueness: A single nondisjunctive lexical category for the head of a given construction specifies both the bounded dependencies that arise when its complements are in canonical position and the unbounded dependencies that arise when those complements are displaced under relativization, coordination, and the like.

As stated above, composition is only allowed into functions of one argument. Note that this is also the case in the argument cluster construction above, since the second functor has only one argument VP/NP, albeit a complex functor category in itself. However, generalized composition is required for sentences such as the following (Steedman, 2000, p.42):

(9)



Steedman defines a “\$ convention” which allows him to schematize over functor categories with a varying number of argument but the same target, or innermost result category.

(10) The \$ convention:
For a category α, $\alpha\$$ (respectively $\alpha/\$$, $\alpha\backslash\$$) denotes the set containing α and all functions (respectively leftward function, rightward functions) into a category in $\alpha\$$ (respectively $\alpha/\$$, $\alpha\backslash\$$).

Then generalized composition can be defined as follows:

(11) a. *Generalized Forward Composition*:

$$\mathsf{X/Y} : f \quad (\mathsf{Y/Z})/\$_1 : ...\lambda z.gz... \quad \Rightarrow_{>\mathbf{B}^n} \quad (\mathsf{X/Z})/\$_1 : ...\lambda z.f(g(z...))$$

b. *Generalized Forward Crossing Composition*:

$$\mathsf{X/Y} : f \quad (\mathsf{Y\backslash Z})\$_1 : ...\lambda z.gz... \quad \Rightarrow_{>\mathbf{B}^n_\times} \quad (\mathsf{X/Z})\$_1 : ...\lambda z.f(g(z...))$$

c. *Generalized Backward Composition*:

$$(\mathsf{Y\backslash Z})\backslash\$_1 : ...\lambda z.gz... \quad \mathsf{X\backslash Y} : f \quad \Rightarrow_{<\mathbf{B}^n} \quad (\mathsf{X\backslash Z})\backslash\$_1 : ...\lambda z.f(g(z...))$$

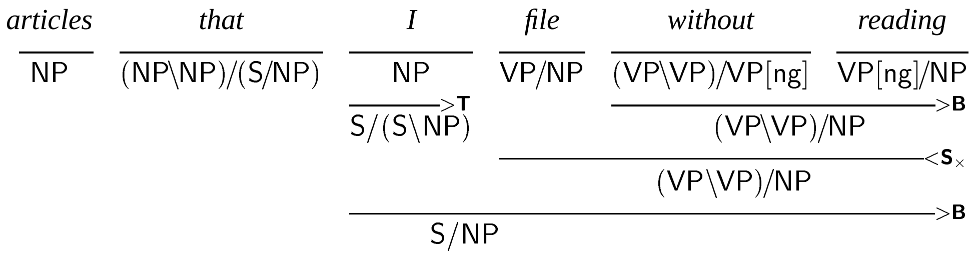
d. *Generalized Backward Crossing Composition*:

$$(\mathsf{Y\backslash Z})/\$_1 : ...\lambda z.gz... \quad \mathsf{X\backslash Y} : f \quad \Rightarrow_{<\mathbf{B}^n_\times} \quad (\mathsf{X\backslash Z})\$_1 : ...\lambda z.f(g(z...))$$

Each of these rules corresponds to a family of rules for each arity *n* of the secondary functor. Without any restriction on the arity of the secondary functor, full context-sensitivity would be obtained. However, there has not yet been any evidence that this is required to capture natural language syntax. Therefore, only schemata up to a bounded arity *n* (Steedman assumes 4 for English) are allowed in practice. The restrictions on the type-raising rules stated above serve a similar purpose. Together, they preserve mild context-sensitivity.

Another combinatory rule, substitution, is required for parasitic gaps, such as the following example:

(12)



Substitution is defined as follows:

(13) a. *Forward Crossing Substitution*:

$$(\mathsf{X/Y})\backslash\mathsf{Z} : f \quad \mathsf{Y\backslash Z} : g \quad \Rightarrow_{\mathbf{S}} \quad \mathsf{X\backslash Z} : \lambda x.fx(g(x))$$

b. *Backward Substitution*:

$$\mathsf{Y\backslash Z} : g \quad \mathsf{(X\backslash Y)\backslash Z} : f \quad \Rightarrow_{\mathbf{S}} \quad \mathsf{X\backslash Z} : \lambda x.fx(g(x))$$

c. *Backward Crossing Substitution*:

$$\mathsf{Y/Z} : g \quad \mathsf{(X\backslash Y)/Z} : f \quad \Rightarrow_{\mathbf{S}} \quad \mathsf{X/Z} : \lambda x.fx(g(x))$$

d. *Forward Substitution*:

$$\mathsf{(X/Y)/Z} : f \quad \mathsf{Y/Z} : g \quad \Rightarrow_{\mathbf{S}} \quad \mathsf{X/Z} : \lambda x.fx(g(x))$$

2.4 Spurious ambiguities and the status of derivations in CCG

Composition and type-raising lead to a number of syntactically distinct, but semantically equivalent alternative derivations even for ordinary sentences:

(14)

$$\begin{array}{ccc}
\textit{John} & \textit{buys} & \textit{shares}\\
\hline
\mathsf{NP} : \mathit{John}' & \mathsf{(S\backslash NP)/NP} : \lambda x.\lambda y.\mathit{buys}'xy & \mathsf{NP}\mathit{shares}'\\
\cline{1-1}
\multicolumn{1}{c}{\mathsf{S/(S\backslash NP)} : \lambda f.f\mathit{John}' \quad {>}\mathbf{T}} & & \\
\cline{1-2}
\multicolumn{2}{c}{\mathsf{S/NP} : \lambda x.\mathit{buys}'\ x\ \mathit{John}' \quad {>}\mathbf{B}} & \\
\hline
\multicolumn{3}{c}{\mathsf{S} : \mathit{buys}'\mathit{shares}'\mathit{John}' \quad {>}}
\end{array}$$

The difference between this derivation and the derivation in (4) is usually referred to as "spurious" ambiguity (Wittenburg, 1986). While this leads to a number of practical problems when it comes to parsing with CCG, the introduction of so-called spurious ambiguity is not regarded as a shortcoming of the theory, since this added flexibility allows CCG to account for the above-mentioned coordinate and extraction constructions, which are difficult to analyze if substrings such as *John buys* cannot be regarded as constituents. Furthermore, since each syntactic rule has a corresponding operation on the semantic interpretation, CCG can account for the fact that human language processing and understanding is incremental and operates on incomplete phrases and constituents. These observations lead Steedman (2000) to the assumption that syntactic structure itself is not a level of representation for the linguistic theory; instead the syntactic operations only provide the means to generate semantic interpretations. The syntactic derivation is then merely considered a trace of the algorithm that builds semantic interpretations. Steedman shows how the multiplicity of derivations can be used to account not only for incremental language understanding, but also the interaction of prosody and information structure, and quantifier scope alternations. Each of these analyses hinges on the assumption of a particular kind of derivation for a particular kind of interpretation.

2.5 Parsing with CCG

A system which processes language can be considered to consist of three parts (Steedman, 2000): a grammar which specifies the strings of the language and their possible analyses, an algorithm which applies the rules of the grammar to the strings to be analyzed, and an oracle which deals with nondeterminism or ambiguities. Most of this and the following chapter is concerned with presenting the grammar used by our system, whereas chapters 4–6 deal primarily with how probabilistic methods can be used to define oracles for parse disambiguation. Here, we briefly consider the algorithm which applies the grammar rules to the input strings. There is a large literature on different parsing algorithms; but we will focus only on one of these, the Cocke-Kasami-Younger (CKY) chart parsing algorithm, since it is the algorithm used in our own parser. Vijay-Shanker and Weir (1990, 1993) show how CKY can be used for mildly-context sensitive grammar formalisms such as CCG and TAG while still preserving polynomial parsing time. Previous parsing algorithms for CCG such as Pareschi and Steedman (1987), Wittenburg (1987) and Hepple and Morrill (1989) have worst-time exponential complexity.

Research on parsing algorithms for categorial grammar has concentrated on dealing with the problem of spurious ambiguity, and there have been a number of attempts to define so-called “normal-form” parsers, which aim to retain only one of multiple semantically equivalent analyses for each substring. Some of these approaches are briefly reviewed in section 2.5.2.

2.5.1 The Cocke-Kasami-Younger algorithm for CCG

```
for i = 1 to n:
  chart[i][i] = set of (pre)terminals expanding to w_i
for k = 2 to n:
  for i = 1 to (n - k)+ 1:
    for j = 1 to k - 1:
    for all A -> B C in G,
      such that there is a B in chart[i][k] and a C in chart[k+1][j],
      if there is no A in chart[i][j], insert A into chart[i][j]
```

Figure 2.1: The Cocke-Kasami-Younger algorithm

The Cocke-Kasami-Younger (CKY) algorithm for context-free grammars (Kasami (1965) and Younger (1967)) is a polynomial bottom-up chart parser. A parse chart for a sentence $s = w_1 ... w_n$ is a $n \times n$ (half-)table which is used to record analyses of substrings. A cell $chart[i][j]$ contains a list of constituents spanning the substring from words w_i to w_j.[4] The basic insight behind this algorithm is that analyses of shorter substrings can be recorded in the chart and reused for the analyses of increasingly larger substrings, which leads to recognition times that are polynomial instead of exponential in the length of the sentence. In its standard, left-to-right formulation, the algorithm proceeds as described in figure 2.1.

The CKY algorithm was originally developed for context-free grammars, which have a finite set of nonterminal categories. For these grammars, recognition time is $O(n^3)$, where n is the length of the string to be parsed. However, the set of categories in a CCG can be infinite, and in a naive implementation, CKY can have exponential recognition times. Vijay-Shanker and Weir (1993) present a structure-sharing technique to encode functor categories whose arity is greater than a fixed bound (determined by the lexicon and the greatest arity n allowed in generalized composition B^n). With this technique, recognition time for CCG is polynomial ($O(n^6)$). Although we use CKY (or rather, a probabilistic variant thereof) in our own parser, we do not employ Vijay-Shanker and Weir's structure-sharing technique, since the probability models described in chapters 5 and 6 constrain the set of categories and rule instantiations to those that occur in the training data. Therefore, the size of categories that our parser operates on is bounded, and Vijay-Shanker and Weir's technique does not apply. This reduces recognition complexity to $O(n^3)$.[5] Worst-case parsing complexity in a probabilistic parser depends on the probability model and the equivalence classes it defines. For example, the probability models underlying parsers such as Collins (1997) take word-word dependencies into account. This results in higher worst-case complexity. However, since probabilities can be computed for every entry in the chart, this information can be used to prune the search space, leading to much lower actual, average case parsing complexities.

Our parser uses a standard version of CKY as described above, not the incremental variant proposed by Steedman (2000) as a model of human sentence processing.

[4] There is an alternative presentation of this algorithm where string positions (spaces between words) are numbered as $0 w_1 1 w_2 2 ... (n-1) w_n n$, and where cell $chart[i][j]$ contains entries spanning words $w_{i+1} ... w_j$.

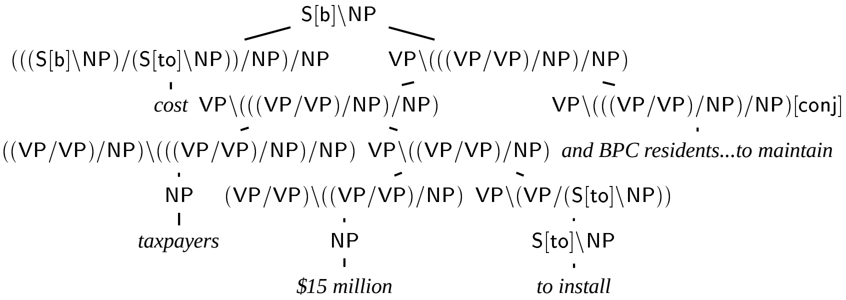
[5] In the actual implementation, even complex categories are simply represented as atomic symbols. We conjecture from the fact that the set of categories and rule instantiations used in our grammar is finite that its generative power is constrained to weak context-freeness.

2.5.2 Normal-form parsing for CCG

“Spurious” ambiguity arises through the availability of type-raising and composition. Depending on the lexical categories, these operations may make any binary branching derivation structure for a string available. Under these circumstances, the number of derivations grows as a function of the Catalan number[6] of the words in the string. However, Hepple and Morrill (1989) show for a fragment of CCG consisting of application, forward type-raising and forward composition that derivations can always be reduced to a normal form, in which composition and type-raising are only used when syntactically necessary. Vijay-Shanker and Weir (1990) demonstrate that spurious ambiguity that arises through composition can be detected and eliminated by comparing alternative derivations of the form $((\alpha\beta)\gamma)$ and $(\alpha(\beta\gamma))$ for substrings α, β and γ. If the right-branching derivation structure is considered normal-form, then it suffices to mark all derivations that have a left-branching structure and to prefer unmarked derivations whenever there is a choice. Although Vijay-Shanker and Weir propose this to be done in a stage following recognition, their method could also be used during parsing. Such a proposal is made by Eisner (1996), who considers a restricted version of CCG without type-raising. Like Vijay-Shanker and Weir, Eisner suggests to mark constituents that are not in normal form. However, he makes the slightly stronger proposal that it suffices to assign tags to categories that are produced by composition rules, so that no constituent which is the result of a forward (backward) composition can serve as the primary functor in another forward (backward) composition. But this is only true for a CCG without type-raising. For instance, in the following example of an argument cluster coordination involving more than two arguments (which is taken from the Penn Treebank), Eisner’s technique would not work, since it would fail to produce the correct derivation:

(15) a. *It could cost taxpayers $15 million to install and BPC residents $1 million a year to maintain.*

b.



[6] The Catalan number $C_n = \frac{(2n)!}{(n+1)!n!}$ gives the number of possible binary bracketings of a string of $n+1$ words.

Chapter 5 develops a probability model for normal-form derivations. There, our goal is similar to Eisner's in that we want to restrict the parser to produce only one normal-form parse per semantic equivalence class; however, this is done in a probabilistic framework. Instead of enforcing normal-form derivations by marking individual constituents, the aim within the probabilistic approach presented in chapter 5 is to assign non-zero probability only to normal-form derivations. Although we will see that we can only guarantee that normal-form derivations receive higher probability than other kinds of derivations, the beam search and dynamic programming strategies used by our probabilistic chart parser can eliminate low-probability analyses during parsing, which provides an alternative solution to the problem of spurious ambiguity.

Eisner's method only concerns chains of forward (or backward) compositions, such as X/X X/X X. Alternative derivations of the form X/X X X\X are not considered equivalent by his method. He uses examples such as *"softly knock twice"* and *"intentionally knock twice"* to argue that this decision cannot be taken on syntactic grounds alone. Karttunen (1989) proposes that the equivalence of derivations should be defined in terms of the equivalence of their semantic interpretations, so that of two derivations whose interpretations mutually subsume each other, only one is kept in the chart. This has been criticized as being computationally too expensive; however, Komagata (1999) uses this method in his CCG parser, and reports substantial improvements in parsing efficency, thus countering the criticisms levelled against Karttunen's approach.

The only semantic information that is considered by the probability models developed in this dissertation is the set of word-word dependencies defined in the predicate-argument structure. These dependencies are insensitive to scope, and therefore cannot distinguish between cases such as *"intentionally (knock twice)"* and *"(intentionally knock) twice"*. Instead, since both derivations use the same rules, they will be considered equivalent. We will return to this point in chapter 5, where the adequacy of probability models over derivations is discussed.

2.6 Predicate-argument structure in CCG

In this dissertation, the term predicate-argument structure is used to refer to the list of local and long-range dependencies between lexical items that are defined by a CCG derivation. Predicate-argument structure will become important in two ways: the performance of the probability models developed in chapters 5 and 6 will be evaluated according to the accuracy with which they recover predicate-argument structure. Furthermore, the model of chapter 6 captures exactly those word-word dependencies that are expressed by predicate-argument structure.

This section explains how predicate-argument structure is represented in CCG and gives a description of the data structures and operations that implement this representation in our parser. We represent categories and their lexical dependencies as feature structures, following previous work in categorial grammar such as Zeevat *et al.* (1987), Uszkoreit (1986) and Villavicencio (2002). However, these feature structures are merely an implementational device to express the information represented in CCG categories. General operations that might raise the generative power of the grammar (as discussed by Carpenter (1991)) are not allowed.

After some introductory terminology, lexical entries for proper nouns, intransitive verbs and adverbs are given, and function application will be used to exemplify the *unify*-operation which is necessary to implement the combinatory rules. Then, implementations of coordination and the combinatory rules of type-raising, composition and substitution are presented. In CCGbank, the translation of the Penn Treebank to CCG described in chapter 3, a number of non-combinatory rules are used to deal with complex adjuncts and extraposed phrases and to represent coordination. Although these rules are only motivated in detail in the following chapter, their effect on predicate-argument structure is described here.

In CCG, bounded and unbounded dependencies are projected from the lexicon. For instance, the lexical catgory for a relative pronoun is $(\mathsf{NP}\backslash\mathsf{NP})/(\mathsf{S}\backslash\mathsf{NP})$ or $(\mathsf{NP}\backslash\mathsf{NP})/(\mathsf{S}/\mathsf{NP})$: the relative pronoun takes a sentence missing a subject or an object to its right and a noun phrase to its left. The entire constituent is also a noun phrase. The noun phrase argument of the relative pronoun is also the missing object or subject of its sentential argument. Informally, this identity relation can be represented by co-indexing the two NPs, eg. $(\mathsf{NP}\backslash\mathsf{NP}_\mathsf{i})/(\mathsf{S}\backslash\mathsf{NP}_\mathsf{i})$. I will use this example to demonstrate how such identity relations can be implemented to obtain the correct predicate-argument dependencies.

The lexicon used by our parser is extracted from CCGbank. However, since CCGbank contains purely syntactic information, the lexical categories have to be augmented with appropriate co-indexation information in order to represent non-local dependencies. Therefore, this section concludes with a brief description of these dependencies in our lexicon.

2.6.1 Category objects

I will distinguish categories (the abstract types the grammar is defined over) from category objects (the data structures the parser operates on); however, this distinction is omitted when the difference is clear, so that a symbol like $\mathsf{S[dcl]}\backslash\mathsf{NP}$ can either refer to the abstract category or to a data structure.[7]

Atomic categories are categories without any arguments, eg. S[dcl] (declarative sentence), NP (noun phrase) etc. I assume a simple feature system such as the one described in section 3.4, whereby two atomic categories of the same type (eg. S) can unify if their features match. This is the case if either both carry the same feature (eg. S[dcl] and S[dcl]), or if only one of them carries a feature. Hence, S[dcl] and S match, but S[b] and S[dcl] do not match.

Complex categories are functor categories with one or more arguments, eg. NP[nb]/N, (S[dcl]\NP)/NP, (NP\NP)/(S[dcl]/NP). The arguments of complex categories are numbered from 1 to *n*, starting at the innermost argument, where *n* is the arity of the functor, eg. $\mathsf{(S[dcl]\backslash NP_1)/NP_2}$, $\mathsf{(NP\backslash NP_1)/(S[dcl]/NP)_2}$. Arguments can themselves be complex categories; for instance the second argument of (NP\NP)/(S[dcl]/NP) is S[dcl]/NP. Two complex categories of the same type match if each of their corresponding arguments match.

Category objects are the data structures that the parser operates on. They represent syntactic categories and the predicate-argument dependencies that correspond to these categories. An atomic category object **C** has three fields, ⟨CAT, HEADS, DEPS⟩:

- A category symbol CAT.
- A list of lexical heads HEADS.
- A list of (unfilled) dependency relations DEPS

Both HEADS and DEPS can be empty. A complex category object ⟨CAT, HEADS, DEPS, RES, ARG, DIR⟩ additionally has two elements RES (result) and ARG (argument), both of which are categories, as well as an attribute DIR, which indicates the directionality of the argument: *FW* (forward, /) or *BW* (backward, \). In principle, this representation could be extended; in particular, a semantic interpretation (or logical form) which corresponds to the syntactic category could be provided. However, since the current implementation of our parser does not construct logical forms, this is omitted in the present description. Similarly,

[7] In the current implementation, the edges in the chart are not category objects; category objects only form the part of the representation of edges that is necessary to build predicate-argument structure. The models described in chapter 5 represent categories as atomic strings. When parsing with these models, it is not necessary to build predicate-argument structure for all edges in the chart, since this information is not used by the models. Instead, predicate-argument structure is built after parsing, and only for the derivation with the highest probability. However, the model in chapter 6 requires the information represented in predicate-argument structure, and there all edges in the chart have an associated category object.

in the current implementation, filled dependency relations are not stored within the categories themselves, but in another data structure that forms part of the representation of edges in the chart. Therefore, filled dependency relations are also omitted here.

The lexicon L is a set of category-word pairs (lexical entries) $\langle c, w\rangle$. If $\langle c, w\rangle$ is a lexical entry, then c is a lexical category of w. Throughout this dissertation, we consider words to be fully inflected word forms, eg. *buys*, *shares*. A derived category is a category object that arises from one or more lexical categories through the application of zero or more combinatory rules. Each derived category has a list of lexical heads. A lexical head is a $\langle c, w\rangle$ pair, where w is a word and c a symbol denoting a lexical category of w.

If the lexical category of a word is complex, the lexicon specifies dependency relations that hold between the heads of the arguments of the category and the head of the lexical category itself. Formally, we represent a **dependency relation** as a 3-tuple $\langle\langle c, w\rangle, i, \langle c', w'\rangle\rangle$, where c is a functor category with arity $\geq i$, and $\langle c', w'\rangle$ is a lexical head of the ith argument of c.

A dependency relation $\langle\langle c, w\rangle, i, \langle c', w'\rangle\rangle$ holds for a derived category C if $\langle c', w'\rangle$ is a lexical head of C, and C is the ith argument of $\langle c, w\rangle$. The attribute DEPS is a list of all dependency relations that hold for this particular category. I will use the abbreviation $Arg(i, \langle c, w\rangle)$ to indicate that the lexical heads of this category are the ith argument of $\langle c, w\rangle$. If HEAD has more than one element, this relation holds for each of its elements.

If an argument of a lexical category is complex, the lexicon can also specify **identity relations** between arguments of the complex argument and other arguments of the same category. Such identity relations encode certain kinds of (bounded and unbounded) non-local dependencies. For instance, in the lexical category for object extraction relative pronouns, $(\mathsf{NP}\backslash\mathsf{NP})/(\mathsf{S[dcl]}/\mathsf{NP})$, the head of the first argument is identical to the head of the NP argument of the second argument. We indicate this identity by indices on the categories, eg.: $(\mathsf{NP}\backslash\mathsf{NP_i})/(\mathsf{S[dcl]}/\mathsf{NP_i})$. Section 2.6.6 explain this mechanism in detail.

2.6.2 Some lexical entries

Here is the atomic category object representing the NP *John*. The lexical head of this category is $\langle\mathsf{NP}, John\rangle$. No dependency relations are specified.

$$\begin{bmatrix} \text{CAT:} & \mathsf{NP} \\ \text{DEPS:} & \langle\ \rangle \\ \text{HEAD:} & \langle\langle\mathsf{NP}, John\rangle\rangle \end{bmatrix}$$

This is the complex category object for the intransitive verb *resigned*:[8]

$$\begin{bmatrix} \textsc{cat}: & \mathsf{S[dcl]\backslash NP} \\ \textsc{dir}: & \mathit{BW} \\ \textsc{head}: & \langle\langle \mathsf{S[dcl]\backslash NP}, \mathit{resigned}\rangle\rangle_{\boxed{1}} \\ \textsc{arg}: & \begin{bmatrix} \textsc{cat}: & \mathsf{NP} \\ \textsc{deps}: & \langle \mathit{Arg}(\mathit{1}, \langle \mathsf{S[dcl]\backslash NP}, \mathit{resigned}\rangle)\rangle \\ \textsc{head}: & \langle\ \rangle \end{bmatrix} \\ \textsc{res}: & \begin{bmatrix} \textsc{cat}: & \mathsf{S[dcl]} \\ \textsc{deps}: & \langle\ \rangle \\ \textsc{head}: & \boxed{1} \end{bmatrix} \end{bmatrix}$$

Its lexical head is $\langle \mathsf{S[dcl]\backslash NP}, \mathit{resigned}\rangle$. The head of the NP argument is empty; however, a dependency relation between the NP and the lexical head of the category is established. The head of the result is the same as the head of the entire category (indicated by the index $\boxed{1}$).

The category object of a modifier, such as the adverb $\langle \mathsf{(S\backslash NP)\backslash(S\backslash NP)}, \mathit{yesterday}\rangle$, is different from that of a verb or a noun phrase. When *yesterday* is combined with a verb or verb phrase, such as *resigned*, the head of the resulting constituent *resigned yesterday* is not the adverb, but the head of the verb phrase, *resigned*. There is a dependency relation between the lexical head of the modifier and the lexical head of the argument. The result category is unified with the argument category, so that its head is the same as the head of the argument:

$$\begin{bmatrix} \textsc{cat}: & \mathsf{(S\backslash NP)\backslash(S\backslash NP)} \\ \textsc{dir}: & \mathit{BW} \\ \textsc{head}: & \langle\langle \mathsf{(S\backslash NP)\backslash(S\backslash NP)}, \mathit{yesterday}\rangle\rangle \\ \textsc{arg}: & \boxed{1}\begin{bmatrix} \textsc{cat}: & \mathsf{S\backslash NP} \\ \textsc{deps}: & \langle \mathit{Arg}(\mathit{2}, \langle \mathsf{((S\backslash NP)\backslash(S\backslash NP))}, \mathit{yesterday}\rangle\rangle \\ \textsc{head}: & \langle\ \rangle \end{bmatrix} \\ \textsc{res}: & \boxed{1} \end{bmatrix}$$

We assume that the head of a noun phrase is the head of the noun. Therefore, the lexical category of a determiner such as $\langle \mathsf{NP[nb]/N}, \mathit{the}\rangle$ specifies that the head of its results is the same as the head of its argument:[9]

[8] In CCGbank, categories are represented as strings, eg. S[dcl]\NP. Features such as [dcl] (declarative) or [b] (bare infinitive) are used to distinguish different kinds of sentences and verb phrases (see section 3.4 for a description of these features). These features ought to be represented directly as attribute-value pairs in the representation used in this chapter. However, for simplicity's sake, and in order to maintain a uniform presentation throughout the dissertation, these features are also represented on the category strings in this chapter. Since the grammar underlying CCGbank does not have agreement features, this is also omitted in the examples in this chapter.

[9] In the current implementation, this treatment was not adopted for lexical categories of determiners which have other categories, such as that of temporal modifiers (eg. *"that year"*).

$$\begin{bmatrix} \text{\textsc{cat}:} & \mathsf{NP[nb]/N} \\ \text{\textsc{dir}:} & \mathit{FW} \\ \text{\textsc{head}:} & \langle\langle \mathsf{NP[nb]/N}, \mathit{the}\rangle\rangle \\ \text{\textsc{arg}:} & \begin{bmatrix} \text{\textsc{cat}:} & \mathsf{N} \\ \text{\textsc{deps}:} & \langle \mathit{Arg}(\mathit{1}, \langle \mathsf{NP[nb]/N}, \mathit{the}\rangle)\rangle \\ \text{\textsc{head}:} & \langle\ \rangle\,\boxed{1} \end{bmatrix} \\ \text{\textsc{res}:} & \begin{bmatrix} \text{\textsc{cat}:} & \mathsf{NP} \\ \text{\textsc{deps}:} & \langle\ \rangle \\ \text{\textsc{head}:} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

The lexical category for possessive $\langle(\mathsf{NP[nb]/N})\backslash\mathsf{NP}, \textit{'s}\rangle$ is defined analogously.

2.6.3 Function application and the *unify*-operation

When a functor category (such as the $\mathsf{S[dcl]\backslash NP}$ *resigned*) is applied to an argument (eg. to the noun phrase *John*), its argument is unified with the category it is applied to:

$$\begin{bmatrix} \text{\textsc{cat}:} & \mathsf{S[dcl]\backslash NP} \\ \text{\textsc{dir}:} & \mathit{BW} \\ \text{\textsc{head}:} & \langle\langle \mathsf{S[dcl]\backslash NP}, \mathit{resigned}\rangle\rangle\,\boxed{1} \\ \text{\textsc{arg}:} & \begin{bmatrix} \text{\textsc{cat}:} & \mathsf{NP} \\ \text{\textsc{deps}:} & \langle \mathit{Arg}(\mathit{1}, \langle \mathsf{S[dcl]\backslash NP}, \mathit{resigned}\rangle)\rangle \\ \text{\textsc{head}:} & \langle\langle \mathsf{NP}, \mathit{John}\rangle\rangle \end{bmatrix} \\ \text{\textsc{res}:} & \begin{bmatrix} \text{\textsc{cat}:} & \mathsf{S[dcl]} \\ \text{\textsc{deps}:} & \langle\ \rangle \\ \text{\textsc{head}:} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

This establishes the dependency relation $\langle\langle \mathsf{S[dcl]\backslash NP}, \mathit{resigned}\rangle, 1, \langle \mathit{John}, \mathsf{NP}\rangle\rangle$ between *John* and *resigned*. The result of this operation is the result category of the functor (recall that in the current implementation filled dependency relations are stored elsewhere):

$$\begin{bmatrix} \text{\textsc{cat}:} & \mathsf{S[dcl]} \\ \text{\textsc{deps}:} & \langle\ \rangle \\ \text{\textsc{head}:} & \langle\langle \mathit{resigned}, \mathsf{S[dcl]\backslash NP}\rangle\rangle \end{bmatrix}$$

We can define a unification operation *unify* $C, C \rightarrow C$ over category objects as follows:

- Two atomic category objects $C' = \langle \text{CAT}', \text{HEADS}', \text{DEPS}' \rangle$ and $C'' = \langle \text{CAT}'', \text{HEADS}'', \text{DEPS}'' \rangle$ can unify if the values of CAT$'$ and CAT$''$ match. The result category $C = \langle \text{CAT}, \text{HEADS}, \text{DEPS} \rangle$ is defined as follows:

  - CAT: unify CAT$'$, CAT$''$.
  - HEADS: concatenate HEADS$'$ and HEADS$''$.
  - DEPS: concatenate DEPS$'$ and DEPS$''$.

  If one of CAT$'$ and CAT$''$ carries a feature (such as $\mathsf{S[dcl]}$), the result of unifying CAT$'$ and CAT$''$ also carries this feature. If HEADS and DEPS are both non-empty, then all $dep \in$ DEPS must hold for all elements h of HEADS. Hence, the dependencies filled by this unification operation are given as the elements of the Cartesian product of HEADS and DEPS.

- Two complex category objects C' and C'' with $C' = \langle \text{CAT}', \text{HEADS}', \text{DEPS}', \text{ARG}', \text{RES}' \rangle$ and $C'' = \langle \text{CAT}'', \text{HEADS}'', \text{DEPS}'', \text{ARG}'', \text{RES}'' \rangle$ can unify if CAT$'$ and CAT$''$ match.
  The result category $C = \langle \text{CAT}, \text{HEADS}, \text{DEPS}, \text{ARG}, \text{RES} \rangle$ is defined as follows:

  - CAT: unify CAT$'$ and CAT$''$.
  - HEADS: concatenate HEADS$'$ and HEADS$''$.
  - DEPS: concatenate DEPS$'$ and DEPS$''$.
  - ARG: unify ARG$'$ and ARG$''$
  - RES: unify RES$'$ and RES$''$

  Here, unification of category strings is CAT$'$ and CAT$''$ is extended in the obvious manner to deal with features on either CAT$'$ or CAT$''$.

The concatenation of two lists L_1, L_2 is defined in the usual manner: if L_2 is empty, return L_1. Otherwise, append all elements of L_2 to the end of L_1, and return the result.

Function application is then defined as follows:

(16) A functor category X'/Y' can be applied to an argument category Y'' if Y' and Y'' unify to Y. If X'/Y' is applied to Y'', unify Y' and Y'' to yield a category X/Y, where X is identical to X' except for any possible instantiation of variables that might have taken place through the unification of Y' and Y''. Return the result X of X/Y.

Above, it was stated informally that in the lexical entry of the adverb $(\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})$ *yesterday*, the result $\mathsf{S}\backslash\mathsf{NP}$ is unified with the argument $\mathsf{S}\backslash\mathsf{NP}$. Here is the intermediate result of applying *yesterday* to *resigned*; the result category is co-indexed with the argument category.

$$\begin{bmatrix} \textsc{cat}: & (\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP}) \\ \textsc{dir}: & \mathit{BW} \\ \textsc{head}: & \langle\langle(\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP}),\mathit{yesterday}\rangle\rangle \\ \textsc{arg}: & \boxed{1}\begin{bmatrix} \textsc{cat}: & \mathsf{S[dcl]}\backslash\mathsf{NP} \\ \textsc{dir}: & \mathit{BW} \\ \textsc{head}: & \langle\langle\mathsf{S[dcl]}\backslash\mathsf{NP},\mathit{resigned}\rangle\rangle\,\boxed{2} \\ \textsc{arg}: & \begin{bmatrix} \textsc{cat}: & \mathsf{NP} \\ \textsc{deps}: & \langle \mathit{Arg}(\mathit{1},\langle\mathsf{S[dcl]}\backslash\mathsf{NP},\mathit{resigned}\rangle)\rangle \\ \textsc{head}: & \langle\ \rangle \end{bmatrix} \\ \textsc{res}: & \begin{bmatrix} \textsc{cat}: & \mathsf{S[dcl]} \\ \textsc{deps}: & \langle\ \rangle \\ \textsc{head}: & \boxed{2} \end{bmatrix} \end{bmatrix} \\ \textsc{res}: & \boxed{1} \end{bmatrix}$$

Thus, the result of this function application is like the category of *resigned* itself:

$$\begin{bmatrix} \textsc{cat}: & \mathsf{S[dcl]}\backslash\mathsf{NP} \\ \textsc{dir}: & \mathit{BW} \\ \textsc{head}: & \langle\langle\mathsf{S[dcl]}\backslash\mathsf{NP},\mathit{resigned}\rangle\rangle\,\boxed{1} \\ \textsc{arg}: & \begin{bmatrix} \textsc{cat}: & \mathsf{NP} \\ \textsc{deps}: & \langle \mathit{Arg}(\mathit{1},\langle\mathsf{S[dcl]}\backslash\mathsf{NP},\mathit{resigned}\rangle)\rangle \\ \textsc{head}: & \langle\ \rangle \end{bmatrix} \\ \textsc{res}: & \begin{bmatrix} \textsc{cat}: & \mathsf{S[dcl]} \\ \textsc{deps}: & \langle\ \rangle \\ \textsc{head}: & \langle\langle\mathsf{S[dcl]}\backslash\mathsf{NP},\mathit{resigned}\rangle\rangle\,\boxed{1} \end{bmatrix} \end{bmatrix}$$

When this verb phrase combines with a subject such as *John*, the subject is an argument of the verb, not the verb phrase modifier.

2.6.4 The combinatory rules

This section explains how the combinatory rules type-raising, composition and substitution are implemented.

Type-raising Type-raising a constituent with category X and lexical head H_X to $\mathsf{T}/(\mathsf{T}\backslash\mathsf{X})$ results in a category whose head is H_X. The two T categories unify, and have the same head as the argument $\mathsf{T}\backslash\mathsf{X}$. Backward typeraising is defined in a similar fashion.

Here is the NP *John* typeraised to $\mathsf{S/(S\backslash NP)}$:

$$\begin{bmatrix}
\text{\textsc{cat}:} & \mathsf{S/(S\backslash NP)} \\
\text{\textsc{dir}:} & \mathit{FW} \\
\text{\textsc{head}:} & \langle\langle \mathsf{NP}, \mathit{John}\rangle\rangle\,\boxed{2} \\
\text{\textsc{arg}:} & \begin{bmatrix}
\text{\textsc{cat}:} & \mathsf{S\backslash NP} \\
\text{\textsc{dir}:} & \mathit{BW} \\
\text{\textsc{head}:} & \boxed{1} \\
\text{\textsc{arg}:} & \begin{bmatrix} \text{\textsc{cat}:} & \mathsf{NP} \\ \text{\textsc{deps}:} & \langle\ \rangle \\ \text{\textsc{head}:} & \boxed{2} \end{bmatrix} \\
\text{\textsc{res}:} & \boxed{3}\begin{bmatrix} \text{\textsc{cat}:} & \mathsf{S} \\ \text{\textsc{deps}:} & \langle\ \rangle \\ \text{\textsc{head}:} & \boxed{1} \end{bmatrix}
\end{bmatrix} \\
\text{\textsc{res}:} & \boxed{3}
\end{bmatrix}$$

This can then be applied to the $\mathsf{S[dcl]\backslash NP}$ *resigned* (note that the $\mathsf{S\backslash NP}$ and $\mathsf{S[dcl]\backslash NP}$ match). Here is intermediate result of unifiying the argument of the type-raised category with *resigned*:

$$\begin{bmatrix}
\text{\textsc{cat}:} & \mathsf{S/(S\backslash NP)} \\
\text{\textsc{dir}:} & \mathit{FW} \\
\text{\textsc{head}:} & \langle\langle \mathsf{NP}, \mathit{John}\rangle\rangle\,\boxed{2} \\
\text{\textsc{arg}:} & \begin{bmatrix}
\text{\textsc{cat}:} & \mathsf{S[dcl]\backslash NP} \\
\text{\textsc{dir}:} & \mathit{BW} \\
\text{\textsc{head}:} & \boxed{1} \\
\text{\textsc{arg}:} & \begin{bmatrix} \text{\textsc{cat}:} & \mathsf{NP} \\ \text{\textsc{deps}:} & \langle \mathit{Arg}(\mathit{1}, \langle \mathsf{S[dcl]\backslash NP}, \mathit{resigned}\rangle)\rangle \\ \text{\textsc{head}:} & \boxed{2} \end{bmatrix} \\
\text{\textsc{res}:} & \boxed{3}\begin{bmatrix} \text{\textsc{cat}:} & \mathsf{S[dcl]} \\ \text{\textsc{deps}:} & \langle\ \rangle \\ \text{\textsc{head}:} & \langle\langle \mathsf{S[dcl]\backslash NP}, \mathit{resigned}\rangle\rangle\,\boxed{1} \end{bmatrix}
\end{bmatrix} \\
\text{\textsc{res}:} & \boxed{3}
\end{bmatrix}$$

And here is the result category of this application:

$$\begin{bmatrix}
\text{\textsc{cat}:} & \mathsf{S} \\
\text{\textsc{deps}:} & \langle\ \rangle \\
\text{\textsc{head}:} & \langle\langle \mathsf{S[dcl]\backslash NP}, \mathit{resigned}\rangle\rangle\,\boxed{1}
\end{bmatrix}$$

Composition Function composition is defined in a similar manner to function application. Here I only give the definition of forward composition, since backward composition and the crossing variants are defined analogously.

(17) A (primary) functor category $\mathsf{X'/Y'}$ can be composed with a (secondary) functor category $\mathsf{Y''/Z''}$ if $\mathsf{Y'}$ and $\mathsf{Y''}$ unify to Y.[10] If $\mathsf{X'/Y'}$ is composed $\mathsf{Y''/Z'}$, unify $\mathsf{Y'}$ and $\mathsf{Y''}$ to yield categories $\mathsf{X/Y}$ and $\mathsf{Y/Z}$, where X is identical to $\mathsf{X'}$ and Z is identical to $\mathsf{Z''}$, except for any possible instantiation of variables that might have taken place through the unification of $\mathsf{Y'}$ and $\mathsf{Y''}$. The result of this composition is a category $\mathsf{X/Z}$.

Let us consider an example. A transitive verb such as *buys* has the following lexical entry:

$$\left[\begin{array}{ll}
\textsc{cat}: & \mathsf{(S[dcl]\backslash NP)/NP} \\
\textsc{dir}: & \mathit{FW} \\
\textsc{head}: & \langle\langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathit{buys}\rangle\rangle\ \boxed{1} \\
\textsc{arg}: & \left[\begin{array}{ll}
\textsc{cat}: & \mathsf{NP} \\
\textsc{deps}: & \langle \mathit{Arg}(\mathit{2}, \langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathit{buys}\rangle)\rangle \\
\textsc{head}: & \langle\ \rangle
\end{array}\right] \\
\textsc{res}: & \left[\begin{array}{ll}
\textsc{cat}: & \mathsf{S[dcl]\backslash NP} \\
\textsc{dir}: & \mathit{BW} \\
\textsc{head}: & \langle\langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathit{buys}\rangle\rangle\ \boxed{1} \\
\textsc{arg}: & \left[\begin{array}{ll}
\textsc{cat}: & \mathsf{NP} \\
\textsc{deps}: & \langle \mathit{Arg}(\mathit{1}, \langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathit{buys}\rangle)\rangle \\
\textsc{head}: & \langle\ \rangle
\end{array}\right] \\
\textsc{res}: & \left[\begin{array}{ll}
\textsc{cat}: & \mathsf{S[dcl]} \\
\textsc{deps}: & \langle\ \rangle \\
\textsc{head}: & \langle\langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathit{buys}\rangle\rangle\ \boxed{1}
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

When the $\mathsf{S/(S\backslash NP)}$ *"John"* is composed with this category, its argument $\mathsf{S\backslash NP}$ is instantiated with the result $\mathsf{S[dcl]\backslash NP}$ of the $\mathsf{(S[dcl]\backslash NP)/NP}$ *"buys"*, which in turn instantiates the subject NP of *buys* with *John*. The S in the type-raised $\mathsf{S/(S\backslash NP)}$ (*"John"*) is instantiated with an $\mathsf{S[dcl]}$ headed by *buys*:

[10] We make the usual assumption that the arity of $\mathsf{Y'}$ and $\mathsf{Y''}$ is bounded.

$$\begin{bmatrix}
\text{CAT:} & \mathsf{S/(S\backslash NP)} \\
\text{DIR:} & \mathit{FW} \\
\text{HEAD:} & \langle\langle \mathsf{NP}, \mathit{John}\rangle\rangle\,\boxed{2} \\
\text{ARG:} & \begin{bmatrix}
\text{CAT:} & \mathsf{S[dcl]\backslash NP} \\
\text{DIR:} & \mathit{BW} \\
\text{HEAD:} & \boxed{1} \\
\text{ARG:} & \begin{bmatrix}
\text{CAT:} & \mathsf{NP} \\
\text{DEPS:} & \langle \mathit{Arg}(\mathit{1}, \langle \mathit{buys}, \mathsf{(S[dcl]\backslash NP)/NP}\rangle)\rangle \\
\text{HEAD:} & \boxed{2}
\end{bmatrix} \\
\text{RES:} & \boxed{3}\begin{bmatrix}
\text{CAT:} & \mathsf{S[dcl]} \\
\text{DEPS:} & \langle\ \rangle \\
\text{HEAD:} & \langle\langle \mathit{buys}, \mathsf{(S[dcl]\backslash NP)/NP}\rangle\rangle\,\boxed{1}
\end{bmatrix}
\end{bmatrix} \\
\text{RES:} & \boxed{3}
\end{bmatrix}$$

Unifying the $\mathsf{S[dcl]\backslash NP}$ of the transitive $\mathsf{(S[dcl]\backslash NP)/NP}$ *"buys"* with the $\mathsf{S\backslash NP}$ of the type-raised noun phrase does not instantiate any variables in the object NP of *"buys"*. Therefore, the resulting $\mathsf{S[dcl]/NP}$ (*"John buys"*) is as follows:

$$\begin{bmatrix}
\text{CAT:} & \mathsf{S[dcl]/NP} \\
\text{DIR:} & \mathit{FW} \\
\text{DEPS:} & \langle\ \rangle \\
\text{HEAD:} & \langle\langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathit{buys}\rangle\rangle\,\boxed{1} \\
\text{ARG:} & \begin{bmatrix}
\text{CAT:} & \mathsf{NP} \\
\text{DEPS:} & \langle \mathit{Arg}(\mathit{2}, \langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathit{buys}\rangle)\rangle \\
\text{HEAD:} & \langle\ \rangle
\end{bmatrix} \\
\text{RES:} & \begin{bmatrix}
\text{CAT:} & \mathsf{S[dcl]} \\
\text{DEPS:} & \langle\ \rangle \\
\text{HEAD:} & \boxed{1}
\end{bmatrix}
\end{bmatrix}$$

So-called argument cluster coordination is another example where composition and type-raising are required. This is a construction where, unlike in the case considered above, composition does not result in any filled dependencies. Consider the following derivation:[11]

(18)

$$\begin{array}{cccccc}
\textit{give} & \textit{a dog} & \textit{a bone} & \textit{and} & \textit{a policeman} & \textit{a flower} \\
\overline{\mathsf{TV/NP}} & \overline{\mathsf{NP}} & \overline{\mathsf{NP}} & \overline{\mathsf{conj}} & \overline{\mathsf{NP}} & \overline{\mathsf{NP}} \\
 & \overline{\mathsf{TV\backslash(TV/NP)}}{<}\mathbf{T} & \overline{\mathsf{VP\backslash TV}}{<}\mathbf{T} & & \overline{\mathsf{TV\backslash(TV/NP)}}{<}\mathbf{T} & \overline{\mathsf{VP\backslash TV}}{<}\mathbf{T} \\
 & \multicolumn{2}{c}{\overline{\qquad\mathsf{VP\backslash(TV/NP)}\qquad}{<}\mathbf{B}} & & \multicolumn{2}{c}{\overline{\qquad\mathsf{VP\backslash(TV/NP)}\qquad}{<}\mathbf{B}} \\
 & \multicolumn{5}{c}{\overline{\qquad\qquad\qquad\mathsf{VP\backslash(TV/NP)}\qquad\qquad\qquad}{<}\Phi{>}} \\
\multicolumn{6}{c}{\overline{\qquad\qquad\qquad\qquad\qquad\mathsf{VP}\qquad\qquad\qquad\qquad\qquad}{<}}
\end{array}$$

[11] I use TV to abbreviate the category of a transitive verb, $\mathsf{(S\backslash NP)/NP}$.

This is the type-raised category for *a bone*:

$$\left[\begin{array}{ll}
\textsc{cat}: & (\mathsf{S}\backslash\mathsf{NP})\backslash((\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}) \\
\textsc{dir}: & \mathit{BW} \\
\textsc{arg}: & \left[\begin{array}{ll}
\textsc{cat}: & (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP} \\
\textsc{dir}: & \mathit{FW} \\
\textsc{arg}: & \left[\begin{array}{ll}
\textsc{cat}: & \mathsf{NP} \\
\textsc{head}: & \langle\langle \mathsf{N}, \mathit{dog}\rangle
\end{array}\right] \\
\textsc{res}: & \boxed{1}
\end{array}\right] \\
\textsc{res}: & \boxed{1}\left[\begin{array}{ll}
\textsc{cat}: & \mathsf{S}\backslash\mathsf{NP} \\
\textsc{dir}: & \mathit{BW} \\
\textsc{arg}: & \left[\textsc{cat}: \mathsf{NP}\right] \\
\textsc{res}: & \left[\textsc{cat}: \mathsf{S}\right]
\end{array}\right]
\end{array}\right]$$

And here is the (simplified) type-raised category for *a dog*:

$$\left[\begin{array}{ll}
\textsc{cat}: & ((\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP})\backslash(((\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP})/\mathsf{NP}) \\
\textsc{dir}: & \mathit{BW} \\
\textsc{arg}: & \left[\begin{array}{ll}
\textsc{cat}: & ((\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP})/\mathsf{NP} \\
\textsc{dir}: & \mathit{FW} \\
\textsc{arg}: & \left[\begin{array}{ll}
\textsc{cat}: & \mathsf{NP} \\
\textsc{head}: & \langle\langle \mathsf{N}, \mathit{dog}\rangle
\end{array}\right] \\
\textsc{res}: & \boxed{1}
\end{array}\right] \\
\textsc{res}: & \boxed{1}\left[\begin{array}{ll}
\textsc{cat}: & (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP} \\
\textsc{dir}: & \mathit{FW} \\
\textsc{arg}: & \left[\textsc{cat}: \mathsf{NP}\right] \\
\textsc{res}: & \left[\textsc{cat}: \mathsf{S}\backslash\mathsf{NP}\right]
\end{array}\right]
\end{array}\right]$$

This is the result of composing *a bone* with *a dog* – note that no dependencies are filled:

$$\left[\begin{array}{ll}
\textsc{cat}: & ((\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP})\backslash(((\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP})/\mathsf{NP}) \\
\textsc{dir}: & \mathit{BW} \\
\textsc{arg}: & \left[\begin{array}{ll}
\textsc{cat}: & ((\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP})/\mathsf{NP} \\
\textsc{dir}: & \mathit{FW} \\
\textsc{arg}: & \left[\begin{array}{ll}
\textsc{cat}: & \mathsf{NP} \\
\textsc{head}: & \langle\langle \mathsf{N}, \mathit{bone}\rangle
\end{array}\right] \\
\textsc{res}: & \left[\begin{array}{ll}
\textsc{cat}: & (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP} \\
\textsc{dir}: & \mathit{FW} \\
\textsc{arg}: & \left[\begin{array}{ll}
\textsc{cat}: & \mathsf{NP} \\
\textsc{head}: & \langle\langle \mathsf{N}, \mathit{dog}\rangle
\end{array}\right] \\
\textsc{res}: & \boxed{1}\left[\textsc{cat}: \mathsf{S}\backslash\mathsf{NP}\right]
\end{array}\right]
\end{array}\right] \\
\textsc{res}: & \boxed{1}
\end{array}\right]$$

Substitution Backward crossing substitution is used to account for parasitic gaps in English, such as the following:

(19)
$$\begin{array}{ccccc}
\textit{articles} & \textit{that} & \textit{I} & \textit{file} & \textit{without reading}\\
\overline{\mathsf{NP}} & \overline{(\mathsf{NP\backslash NP})/(\mathsf{S/NP})} & \overline{\mathsf{NP}} & \overline{(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}} & \overline{((\mathsf{S\backslash NP})\backslash(\mathsf{S\backslash NP}))/\mathsf{NP}}\\
 & & & \multicolumn{2}{c}{\overline{\hphantom{(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}\quad((\mathsf{S\backslash NP})\backslash(\mathsf{S\backslash NP}))/\mathsf{NP}}}{<}\mathbf{S}_{\times}}\\
 & & & \multicolumn{2}{c}{(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}}
\end{array}$$

Recall that backward crossing substitution is defined as follows:

(20) $\mathsf{Y/Z}\quad (\mathsf{X\backslash Y})/\mathsf{Z}\ \Rightarrow_{\mathbf{S}}\ \mathsf{X/Z}$

Here, the Ys and Zs are both unified. Substitution is similar to conjunction in that the dependencies that hold for the Z argument of the resulting $\mathsf{X/Z}$ are the union of the dependencies that hold for the Z argument of the first functor and those that hold for the Z argument of the second functor. This operation can be implemented as follows:

(21) A functor category $\mathsf{Y'/Z'}$ and a functor category $(\mathsf{X''\backslash Y''})/\mathsf{Z''}$ can be combined through substitution if $\mathsf{Y'}$ and $\mathsf{Y''}$ unify to Y and if $\mathsf{Z'}$ and $\mathsf{Z''}$ unify to Z. If $\mathsf{Y'/Z'}$ and $(\mathsf{X''\backslash Y''})/\mathsf{Z''}$ are combined through substitution, unify $\mathsf{Y'}$ and $\mathsf{Y''}$ and $\mathsf{Z'}$ and $\mathsf{Z''}$ to yield categories $\mathsf{Y/Z}$ and $(\mathsf{X\backslash Y})/\mathsf{Z}$, where X is identical to $\mathsf{X'}$ except for any possible instantiation of variables that might have taken place through the unification of $\mathsf{Y'}$ and $\mathsf{Y''}$ and $\mathsf{Z'}$ and $\mathsf{Z''}$. The result of this substitution is a category $\mathsf{X/Z}$.

In the parasitic gap example, the $\mathsf{Y/Z}$ of the rule schema is instantiated with $(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}$, and the $(\mathsf{X\backslash Y})/\mathsf{Z}$ with $((\mathsf{S\backslash NP})\backslash(\mathsf{S\backslash NP}))/\mathsf{NP}$. This is the category object for *without reading*:

$$\begin{bmatrix}
\textsc{cat}: & ((\mathsf{S\backslash NP})\backslash(\mathsf{S\backslash NP}))/\mathsf{NP}\\
\textsc{dir}: & \textit{FW}\\
\textsc{head}: & \\
\textsc{arg}: & \begin{bmatrix}\textsc{cat}: & \mathsf{NP}\\ \textsc{deps}: & \langle \textit{Arg}(2,(\mathsf{S[ng]\backslash NP})/\mathsf{NP},\textit{reading}\rangle))\rangle\\ \textsc{head}: & \langle\ \rangle\end{bmatrix}\\
\textsc{res}: & \begin{bmatrix}\textsc{cat}: & (\mathsf{S\backslash NP})\backslash(\mathsf{S\backslash NP})\\ \textsc{dir}: & \textit{BW}\\ \textsc{head}: & \\ \textsc{arg}: & \boxed{2}\begin{bmatrix}\textsc{cat}: & \mathsf{S\backslash NP}\\ \textsc{dir}: & \textit{BW}\end{bmatrix}\\ \textsc{res}: & \boxed{2}\end{bmatrix}
\end{bmatrix}$$

The $(\mathsf{S\backslash NP})\backslash(\mathsf{S\backslash NP})$ result of this category is similar to the adverb category given above for *yesterday* in that its result is unified with its argument. Hence, the X of the rule schema is unified with the Y. Therefore, substitution results in the following category for *file without reading*:

$$\begin{bmatrix} \textsc{cat}: & \mathsf{(S[dcl]\backslash NP)/NP} \\ \textsc{dir}: & \mathit{FW} \\ \textsc{head}: & \langle\langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathit{file}\rangle\rangle \boxed{1} \\ \textsc{arg}: & \begin{bmatrix} \textsc{cat}: & \mathsf{NP} \\ \textsc{deps}: & \langle \mathit{Arg}(\mathit{2}, \langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathit{file}\rangle), \mathit{Arg}(\mathit{2}, \mathsf{(S[ng]\backslash NP)/NP}, \mathit{reading}\rangle))\rangle \\ \textsc{head}: & \langle\ \rangle \end{bmatrix} \\ \textsc{res}: & \begin{bmatrix} \textsc{cat}: & \mathsf{S[dcl]\backslash NP} \\ \textsc{dir}: & \mathit{BW} \\ \textsc{head}: & \langle\langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathit{file}\rangle\rangle \boxed{1} \\ \textsc{arg}: & \begin{bmatrix} \textsc{cat}: & \mathsf{NP} \\ \textsc{deps}: & \langle \mathit{Arg}(\mathit{1}, \langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathit{file}\rangle)\rangle \\ \textsc{head}: & \langle\ \rangle \end{bmatrix} \\ \textsc{res}: & \begin{bmatrix} \textsc{cat}: & \mathsf{S[dcl]} \\ \textsc{deps}: & \langle\ \rangle \\ \textsc{head}: & \langle\langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathit{file}\rangle\rangle \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

2.6.5 Non-combinatory rules

The grammar underlying CCGbank (described in chapter 3) uses a number of non-combinatory rules in order to deal with punctuation, coordination, and certain types of complex adjuncts. This section describes how these rules can be implemented in the framework developed in this chapter.

Punctuation In CCGbank, punctuation marks such as commas, semicolons, full stops etc. do not have real CCG categories, but carry instead categories derived from their part-of-speech tags, such as ".", ";" etc. There are a number of rules dealing with punctuation, such as $\mathsf{S[dcl]}\ \ .\ \Rightarrow \mathsf{S[dcl]}$. We assume that there are no dependencies between words and punctuation marks, and that the result of such a rule is the same as the category of the non-punctuation category.

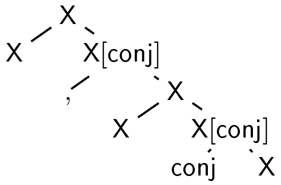
Here is the lexical category for the full stop:

$$\begin{bmatrix} \textsc{cat}: & . \\ \textsc{deps}: & \langle\ \rangle \\ \textsc{head}: & \langle\langle .,. \rangle\rangle \end{bmatrix}$$

Certain types of punctuation marks, such as opening brackets or dashes, can have specific lexical categories (such as $\langle(\mathsf{NP}\backslash\mathsf{NP})/\mathsf{S[dcl]},–\rangle$, see section 3.7.4) and are therefore treated like ordinary lexical items. The ampersand "&" is also treated like an ordinary lexical item. The plural possessive ' (as in *the parents'*) has also real categories (mainly $(\mathsf{NP}/\mathsf{N})\backslash\mathsf{NP}$), and is thus treated like an ordinary lexical item.

Coordination Coordination (see section 3.7.1) is encoded in CCGbank with an intermediate level $\mathsf{X[conj]}$, eg.:

(22)



This corresponds to rules of the form:

(23) a. $\mathsf{conj}\ \ \mathsf{X} \Rightarrow \mathsf{X[conj]}$

b. $,\ \ \mathsf{X} \Rightarrow \mathsf{X[conj]}$

c. $\mathsf{X}\ \ \mathsf{X[conj]} \Rightarrow \mathsf{X}$

Commas which occur in coordinate constructions have the same lexical category as described in the previous paragraph. Conjunctions have similar categories, eg.:

$$\begin{bmatrix} \textsc{cat}: & . \\ \textsc{deps}: & \langle\ \rangle \\ \textsc{head}: & \langle\langle \mathsf{conj}, \mathit{and}\rangle\rangle \end{bmatrix}$$

Coordination itself is implemented as follows: When coordinating two constituents X and X' with the same category, the result X'' is the unification of X and X' as defined above. Hence, the head and dependency relations of X'' and its subcategories are the concatenation of the corresponding head and dependency relations of X and X'. Here is the result of the coordination of two transitive verbs *buy* and *sell* with lexical category $(\mathsf{S[b]}\backslash\mathsf{NP})/\mathsf{NP}$:

$$\begin{bmatrix}
\textsc{cat:} & \mathsf{(S[b]\backslash NP)/NP} \\
\textsc{dir:} & \mathit{FW} \\
\textsc{head:} & \langle\langle \mathsf{(S[b]\backslash NP)/NP}, \mathit{buys}\rangle, \langle \mathsf{(S[b]\backslash NP)/NP}, \mathit{sells}\rangle\rangle\ \boxed{1} \\
\textsc{arg:} & \begin{bmatrix} \textsc{cat:} & \mathsf{NP} \\ \textsc{deps:} & \langle \mathit{Arg}(\mathit{2}, \langle \mathsf{(S[b]\backslash NP)/NP}, \mathit{buy}\rangle), \mathit{Arg}(\mathit{2}, \langle \mathsf{(S[b]\backslash NP)/NP}, \mathit{sell}\rangle)\rangle \\ \textsc{head:} & \langle\ \rangle \end{bmatrix} \\
\textsc{res:} & \begin{bmatrix} \textsc{cat:} & \mathsf{S[b]\backslash NP} \\ \textsc{dir:} & \mathit{BW} \\ \textsc{head:} & \boxed{1} \\ \textsc{arg:} & \begin{bmatrix} \textsc{cat:} & \mathsf{NP} \\ \textsc{deps:} & \langle \mathit{Arg}(\mathit{1}, \langle \mathsf{(S[b]\backslash NP)/NP}, \mathit{buy}\rangle), \mathit{Arg}(\mathit{1}, \langle \mathsf{(S[b]\backslash NP)/NP}, \mathit{sell}\rangle)\rangle \\ \textsc{head:} & \langle\ \rangle \end{bmatrix} \\ \textsc{res:} & \begin{bmatrix} \textsc{cat:} & \mathsf{S[b]} \\ \textsc{deps:} & \langle\ \rangle \\ \textsc{head:} & \boxed{1} \end{bmatrix} \end{bmatrix}
\end{bmatrix}$$

Note that now there are two dependency relations that hold for each of the arguments. Furthermore, this category has two lexical heads.

In CCGbank, the coordination of constituents that do not have the same category (as in "*is 69 years old and chairman*") is treated by the following rule schema (see section 3.7.2 for details):

(24) conj Y ⇒ X[conj]

This is implemented in the same way as the unary type-changing rules described in the next paragraph; the head of the resulting X[conj] is the same as the head of the Y conjunct.

Unary type-changing rules Apart from type-raising, there are several genuine type-changing rules in CCGbank. The simplest and most common one is N ⇒ NP. However, this is merely a change from a complete noun phrase to a bare noun phrase, which can be implemented by simply replacing the top-level category string N with NP. Another type of unary rule which appears once in every complete tree is of the form X ⇒ TOP. All other type-changing rules are of the form Y ⇒ X/X or Y ⇒ X\X, where a complement category Y is changed into an adjunct X/X or X\X (see section 3.8 for more details).

We assume that type-changing rules of the form Y ⇒ X/X or Y ⇒ X\X can project dependencies if Y is a complex category whose argument is of category X, as in the following rule for postmodifiers of nouns and noun phrases:

- $\mathsf{S[dcl]/NP} \Rightarrow \mathsf{NP\backslash NP}$
  "the shares John bought"

- $\mathsf{S[pss]\backslash NP} \Rightarrow \mathsf{NP\backslash NP}$
  "the shares bought by John"

Binary type-changing rules CCGbank contains a number of binary type-changing rules that are triggered by certain kinds of NP extraction (explained in section 3.7.5), such as:

(25) a. *No dummies, the drivers pointed out they still had space (...)*

b. *Factories booked \$236.74 billion in orders in September,*
$[_{NP}$ nearly the same as the \$236.79 billion in August]

The binary type-changing rules are as follows:

$$\begin{array}{llll}
\mathsf{NP} & , & \Rightarrow & \mathsf{S/S} \\
, & \mathsf{NP} & \Rightarrow & \mathsf{S\backslash S} \\
, & \mathsf{NP} & \Rightarrow & \mathsf{(S\backslash NP)\backslash(S\backslash NP)}
\end{array}$$

Note that this is equivalent to assigning the comma categories such as $\mathsf{(S/S)\backslash NP}$. Being essentially anaphoric, it is difficult to establish a clear semantic relation between these extracted noun phrases and the sentence or verb phrase they belong to. Therefore, we assume for the purpose of the model that there is no dependency between them.

2.6.6 Co-indexation and non-local dependencies

As mentioned above, certain types of bounded and unbounded non-local dependencies are captured in the predicate-argument structure by co-indexation within complex lexical categories. For example, relative pronouns, auxiliaries, modals and control verbs are all cases where one (NP) argument fills the argument slot of another $\mathsf{S\backslash NP}$ argument. I will use the example of relative pronouns and of auxiliaries to show in detail the effect of co-indexation. Section 2.6.7 lists the different classes of lexical categories that project long-range dependencies in the current implementation.

Relative pronouns Here is the category object corresponding to the object extraction relative pronoun *that*:

$$\begin{bmatrix}
\text{\textsc{cat}:} & \mathsf{(NP\backslash NP)/(S[dcl]/NP)}\\
\text{\textsc{dir}:} & \mathit{FW}\\
\text{\textsc{head}:} & \langle\langle \mathsf{(NP\backslash NP)/(S[dcl]/NP)}, \mathit{that}\rangle\rangle\,\boxed{1}\\
\text{\textsc{arg}:} & \begin{bmatrix}
\text{\textsc{cat}:} & \mathsf{S[dcl]/NP}\\
\text{\textsc{dir}:} & \mathit{FW}\\
\text{\textsc{deps}:} & \langle \mathit{Arg}(2,\langle \mathsf{(NP\backslash NP)/(S[dcl]/NP)}, \mathit{that}\rangle)\rangle\\
\text{\textsc{head}:} & \langle\ \rangle\\
\text{\textsc{arg}:} & \boxed{2}
\end{bmatrix}\\
\text{\textsc{res}:} & \begin{bmatrix}
\text{\textsc{cat}:} & \mathsf{NP\backslash NP}\\
\text{\textsc{dir}:} & \mathit{BW}\\
\text{\textsc{head}:} & \boxed{1}\\
\text{\textsc{arg}:} & \boxed{2}\begin{bmatrix}
\text{\textsc{cat}:} & \mathsf{NP}\\
\text{\textsc{head}:} & \langle\ \rangle\\
\text{\textsc{deps}:} & \langle \mathit{Arg}(1,\langle \mathsf{(NP\backslash NP)/(S[dcl]/NP)}, \mathit{that}\rangle)\rangle
\end{bmatrix}\\
\text{\textsc{res}:} & \boxed{2}
\end{bmatrix}
\end{bmatrix}$$

This entry specifies that the head of the NP argument of the relative clause be the same as the head of the NP it modifies. Note that no dependency relations are introduced for the NP argument of the relative clause, but that instead the dependencies are percolated up to the NP argument of the relative pronoun. Here is the category of the relative clause *which John likes*:

$$\begin{bmatrix}
\text{\textsc{cat}:} & \mathsf{NP\backslash NP}\\
\text{\textsc{dir}:} & \mathit{BW}\\
\text{\textsc{head}:} & \langle\langle \mathsf{(NP\backslash NP)/(S[dcl]/NP)}, \mathit{which}\rangle\rangle\\
\text{\textsc{arg}:} & \boxed{2}\begin{bmatrix}
\text{\textsc{cat}:} & \mathsf{NP}\\
\text{\textsc{deps}:} & \langle \mathit{Arg}(2,\langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathit{likes}\rangle), \mathit{Arg}(1,\langle \mathsf{(NP\backslash NP)/(S[dcl]/NP)}, \mathit{which}\rangle)\rangle\\
\text{\textsc{head}:} & \langle\ \rangle
\end{bmatrix}\\
\text{\textsc{res}:} & \boxed{2}
\end{bmatrix}$$

When this is applied to a noun phrase *books*, the intermediate result is as follows:

$$\begin{bmatrix}
\text{\textsc{cat}:} & \mathsf{NP\backslash NP}\\
\text{\textsc{dir}:} & \mathit{BW}\\
\text{\textsc{head}:} & \langle\langle \mathsf{(NP\backslash NP)/(S/NP)}, \mathit{which}\rangle\rangle\\
\text{\textsc{arg}:} & \begin{bmatrix}
\text{\textsc{cat}:} & \mathsf{NP}\\
\text{\textsc{deps}:} & \langle \mathit{Arg}(2,\langle \mathsf{S\backslash NP/NP}, \mathit{likes}\rangle), \mathit{Arg}(1,\langle\langle \mathsf{(NP\backslash NP)/(S/NP)}, \mathit{which}\rangle\rangle)\rangle\\
\text{\textsc{head}:} & \langle\langle \mathit{books}, \mathsf{N}\rangle\rangle
\end{bmatrix}\\
\text{\textsc{res}:} & \boxed{2}
\end{bmatrix}$$

Auxiliaries We assume that when an auxiliary verb such as *will* combines with an untensed verb such as *buy*, the subject NP is an argument of both verbs. However, the head of this constituent is the auxiliary. Therefore, the lexical entry for $\langle(\mathsf{S[dcl]\backslash NP})/(\mathsf{S[b]\backslash NP}), \mathit{will}\rangle$ is as follows:

$$\begin{bmatrix}
\textsc{cat}: & (\mathsf{S[dcl]\backslash NP})/(\mathsf{S[b]\backslash NP}) \\
\textsc{deps}: & \langle\ \rangle \\
\textsc{head}: & \langle\langle(\mathsf{S[dcl]\backslash NP})/(\mathsf{S[b]\backslash NP}), \mathit{will}\rangle\rangle_{\boxed{1}} \\
\textsc{arg}: & \begin{bmatrix}
\textsc{cat}: & \mathsf{S[b]\backslash NP} \\
\textsc{dir}: & \mathit{BW} \\
\textsc{head}: & \langle\ \rangle_{\boxed{2}} \\
\textsc{arg}: & \boxed{3} \\
\textsc{res}: & \begin{bmatrix}
\textsc{cat}: & \mathsf{S[dcl]} \\
\textsc{deps}: & \langle\ \rangle \\
\textsc{head}: & \boxed{1}
\end{bmatrix}
\end{bmatrix} \\
\textsc{res}: & \begin{bmatrix}
\textsc{cat}: & \mathsf{S[dcl]\backslash NP} \\
\textsc{dir}: & \mathit{BW} \\
\textsc{head}: & \boxed{1} \\
\textsc{arg}: & \begin{bmatrix}
\textsc{cat}: & \mathsf{NP} \\
\textsc{deps}: & \langle \mathit{Arg}(\mathit{1}, \langle(\mathsf{S[dcl]\backslash NP})/(\mathsf{S[b]\backslash NP}), \mathit{will}\rangle)\rangle \\
\textsc{head}: & \langle\ \rangle
\end{bmatrix} \boxed{3} \\
\textsc{res}: & \begin{bmatrix}
\textsc{cat}: & \mathsf{S[dcl]} \\
\textsc{deps}: & \langle\ \rangle \\
\textsc{head}: & \boxed{1}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}$$

Note that the subject NP argument of the untensed $\mathsf{S[b]\backslash NP}$ verb phrase argument is co-indexed with the subject NP of the auxiliary. When this category is applied to an $\mathsf{S[b]\backslash NP}$ argument (which in turn specifies dependencies on its subject NP), these dependencies also hold for the NP argument of the resulting tensed $\mathsf{S[dcl]\backslash NP}$ verb phrase.

2.6.7 Non-local dependencies in our system

Non-local dependencies projected by the lexicon This section lists the types of lexical categories in the current implementation that project non-local dependencies. As described in the previous section, non-local dependencies are encoded by co-indexing arguments of complex arguments of the lexical category with other arguments of the same lexical category (eg $(\mathsf{S[.]\backslash NP_i})/(\mathsf{S[.]\backslash NP_i})$, so that there is a dependency of the co-indexed argument both on the head of this lexical category and on the head of the argument of this lexical category. Note that expletive *it* noun phrases ($\mathsf{NP[expl]}$) are not co-indexed with other arguments.

In the current implementation, with the only exception of the subject control verb *promise*, it is assumed that all lexical categories of the same type project the same dependencies. While this generalization leads to the correct results in almost all cases, a more careful word-by-word examination of the co-indexation rules would be necessary to guarantee that the correct dependencies are obtained in all cases.

When evaluating the accuracy of the different parsing models in chapters 5 and 6, I will distinguish between three different kinds of dependencies in the predicate-argument structure. **Local** argument/adjunct dependencies are dependencies between a head and its immediate argument or adjunct when both are in canonical position, and when the dependency is not mediated through another lexical item. There are two kinds of **non-local** dependencies: **locally mediated** dependencies are dependencies that arise through raising, control and related constructions. They are not strictly local, since they are projected through co-indexation in the lexical category of a third lexical item (such as a modal, auxiliary or control verb). Locally mediated dependencies hold between two arguments of the same lexical item, and are therefore bounded. **Long-range** dependencies are unbounded dependencies, such as object extraction, topicalization, *tough* movement and right node raising. Long-range dependencies that arise through co-indexation in the lexicon differ from locally mediated dependencies in that the projected dependency need not hold between two arguments of the lexical item whose category projects the dependency. For example, object relative pronouns $((\mathsf{NP}\backslash\mathsf{NP}_\mathsf{i})/(\mathsf{S[dcl]}/\mathsf{NP}_\mathsf{i}))$ project a long-range dependency between the noun phrase they modify and an (unboundedly distant) element inside the relative clause $\mathsf{S[dcl]}/\mathsf{NP}$. However, the noun phrase itself need not be an argument of the head of the relative clause. According to this classification, unembedded subject relative pronouns project a locally mediated dependency between the noun phrase they modify and the head of the relative clause (whereas embedded subject relative clauses are analyzed like object relatives in CCG, and project an unbounded dependency). When evaluating the parser, the dependency between a noun phrase and the head of a non-embedded subject relative clause that modifies it will therefore be counted as a locally mediated dependency. However, in the context of discussing the Penn Treebank, this dependency will be included in the class of long-range dependencies, because movement-based analyses (such as those that the Penn Treebank is based on) assume the same kind of trace for subject relatives as for object relatives.

Auxiliaries, modals, raising verbs – $(\mathsf{S}[.]\backslash\mathsf{NP}_\mathsf{i})/(\mathsf{S}[.]\backslash\mathsf{NP}_\mathsf{i})$**:** In lexical categories that match this pattern, both NPs are co-indexed. Apart from auxiliaries, modals and raising verbs, this includes entries such as:

$$\langle \textit{expects}, (\mathsf{S}[\mathsf{dcl}]\backslash\mathsf{NP})/(\mathsf{S}[\mathsf{to}]\backslash\mathsf{NP})\rangle,$$
$$\langle \textit{began}, (\mathsf{S}[\mathsf{dcl}\backslash\mathsf{NP})/(\mathsf{S}[\mathsf{ng}]\backslash\mathsf{NP})\rangle,$$
$$\langle \textit{declined}, (\mathsf{S}[\mathsf{dcl}\backslash\mathsf{NP})/(\mathsf{S}[\mathsf{to}]\backslash\mathsf{NP})\rangle$$

These dependencies are locally mediated.

Control verbs – $((\mathsf{S}[.]\backslash\mathsf{NP})/(\mathsf{S}[.]\backslash\mathsf{NP}))/\mathsf{NP}$**:** in subject control verbs (forms of *promise*), the subject NP argument is co-indexed with the subject of the verb phrase argument:

$$((\mathsf{S}[.]\backslash\mathsf{NP}_\mathsf{i})/(\mathsf{S}[.]\backslash\mathsf{NP}_\mathsf{i}))/\mathsf{NP}$$

All other verbs with a category matching $((\mathsf{S}[.]\backslash\mathsf{NP})/(\mathsf{S}[.]\backslash\mathsf{NP}))/\mathsf{NP}$ (eg. forms of *persuade*) are assumed to be object control. In these cases, the object NP is co-indexed with the subject of the verb phrase argument:

$$((\mathsf{S}[.]\backslash\mathsf{NP})/(\mathsf{S}[.]\backslash\mathsf{NP}_\mathsf{i}))/\mathsf{NP}_\mathsf{i}.$$

Other cases of object control are categories that have both a verb phrase and a noun phrase argument, such as $\langle \textit{find}, ((\mathsf{S}[\mathsf{dcl}\backslash\mathsf{NP})/(\mathsf{S}[\mathsf{ng}]\backslash\mathsf{NP}))/\mathsf{NP}\rangle$ etc. Furthermore, if a verb has a verb phrase argument, but no other noun phrase argument than the subject, the subject of the verb phrase argument is co-indexed with the subject of the verb. Dependencies projected by control verbs are also locally mediated

Subject extraction verbs – $((\mathsf{S}[.]\backslash\mathsf{NP})/\mathsf{NP}_\mathsf{i})/(\mathsf{S}[.]\backslash\mathsf{NP}_\mathsf{i})$**:** lexical entries where the category matches this pattern (unless the $\mathsf{S}[.]\backslash\mathsf{NP}$ is $\mathsf{S}[\mathsf{adj}]\backslash\mathsf{NP}$), are special cases of subject extraction from embedded sentences. See section 3.9.7. Since this analysis is similar to object extraction, we classify these dependencies as long-range.

VP modifiers – $((\mathsf{S}\backslash\mathsf{NP}_\mathsf{i}) \mid (\mathsf{S}\backslash\mathsf{NP}_\mathsf{i}))/(\mathsf{S}[.]\backslash\mathsf{NP}_\mathsf{i})$**:** In categories of pre- and post-verbal modifiers that take an additional verb phrase argument, the subject of their verb phrase argument is co-indexed with the subject of the verb phrase they modify. This is a locally mediated dependency.

Small clause PPs – $(\mathsf{X}/(\mathsf{S}[.]\backslash\mathsf{NP}_\mathsf{i}))/\mathsf{NP}_\mathsf{i}$**:** in categories that match this pattern (where X can be one of $\mathsf{S}[\mathsf{for}]$, PP, or an adjunct category, such as $\mathsf{NP}\backslash\mathsf{NP}$ or $(\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})$), the noun phrase argument is co-indexed with the subject of the verb phrase argument. Examples: $\langle \textit{with}, (((\mathsf{S\backslash NP})\backslash(\mathsf{S\backslash NP}))/(\mathsf{S[ng]\backslash NP}))/\mathsf{NP}\rangle$, $\langle \textit{for}, (\mathsf{S[for]}/(\mathsf{S[to]\backslash NP}))/\mathsf{NP}\rangle$. This is also a locally mediated dependency.

Relative pronouns – $(\mathsf{NP_i\backslash NP_i})/(\mathsf{S[.]\backslash NP_i})$ **and** $(\mathsf{NP_i\backslash NP_i})/(\mathsf{S[.]/NP_i})$**:** The noun phrase argument (and result) of the lexical category of a relative pronoun is co-indexed with the noun phrase argument of the relative clause. Subject relative pronouns project locally mediated dependencies, whereas object relative pronouns project long-range dependencies.

Relative pronouns that take a noun argument – $(\mathsf{X}/(\mathsf{S[.]} \mid \mathsf{NP_i}))/\mathsf{N_i}$**:** The head of the noun argument is co-indexed with the head of the noun phrase argument of the relative clause. Examples: $\langle \textit{whose}, ((\mathsf{NP\backslash NP})/(\mathsf{S[dcl]/NP}))/\mathsf{N}\rangle$, $\langle \textit{whatever}, (\mathsf{NP}/(\mathsf{S[dcl]/NP}))/\mathsf{N}\rangle$. As before, we consider subject extraction a locally mediated dependency and object extraction long-range.

Yes-no question words – $(\mathsf{S[q]}/(\mathsf{S[.]\backslash NP_i}))/\mathsf{NP_i}$**:** The noun phrase argument is co-indexed with the subject of the verb phrase argument. This is a locally mediated dependency.

***Tough*-adjectives –** $(\mathsf{S[adj]\backslash NP_i})/((\mathsf{S[to]\backslash NP})/\mathsf{NP_i})$**:** eg. *tough*, *difficult*, *easy*, *impossible*, *reasonable*, *wrong*. These dependencies are long-range.

Non-local dependencies through coordination and type-changing rules Coordination of functor categories can also create unbounded long-range dependencies. If the direction of the argument of the left conjunct is forward (as in right node raising), this is a long-range dependency. Similarly, if the direction of the argument of the right conjunct is backward, then we classify this as a long-range dependency. This also includes VP coordination.

Unary type-changing rules can also create long-range dependencies. Type-changing rules that correspond to subject extraction are considered locally mediated, whereas type-changing rules that encode object extraction are considered long-range.

Distinguishing non-local from local dependencies In order to be able to distinguish locally mediated and long-range dependencies from local dependencies, each lexical functor category specifies for each of its arguments whether this dependencies are local or not, and if the dependencies are not local, whether they are locally mediated (bounded) or long-range (unbounded). For example, dependencies on the NP argument of the $\mathsf{S/NP}$ in the lexical category of a relative pronoun, $(\mathsf{NP}\backslash\mathsf{NP})/(\mathsf{S}/\mathsf{NP})$ are long-range dependencies, whereas the dependency between the relative pronoun and the head of the noun phrase it modifies is local. When the relative pronoun is applied to a relative clause, all dependencies that the relative clause S/NP defines for its argument are marked as unbounded long-range dependencies. Similarly, coordination of functor categories marks the dependencies on right arguments of left conjuncts and on left arguments of right conjuncts as unbounded long-range dependencies. This allows us to evaluate the parser's recovery of different classes of non-local dependencies.

Chapter 3

Creating a CCG corpus from the Penn Treebank

This chapter presents an algorithm for translating phrase-structure trees from the Penn Treebank (Marcus *et al.*, 1993, 1994) to CCG normal-form derivation trees, notes some of the changes to the Penn Treebank representation that are necessary for the translation procedure to yield adequate CCG analyses, and describes CCGbank, the corpus that results from applying this algorithm to the Penn Treebank.

The work presented in this chapter is an extension of an algorithm to extract categorial lexicons from the Penn Treebank, which was originally presented in (Hockenmaier *et al.*, 2002) and (Hockenmaier *et al.*, 2000). Hockenmaier and Steedman (2002a) also describes an earlier version of CCGbank.

3.1 Introduction

The Penn Treebank (Marcus *et al.*, 1993, 1994) is the largest available manually parsed corpus of English, and has been used as the standard test and training material for statistical parsing of English (see e.g. Collins, 1999; Charniak, 1999). This chapter presents an algorithm for translating Penn Treebank trees to normal-form CCG derivations. The resulting corpus, CCGbank, is used to test and train the parsers developed in this dissertation. However, CCGbank is not only useful for the development of statistical parsers. From it we can obtain a large CCG lexicon, which can be used by any CCG parser. Moreover, since CCG derivations have a corresponding semantic interpretation, the creation of a corpus of CCG derivations can be seen as a first step towards a corpus annotated with logical forms.

This chapter is organized in the following way: section 3.3 presents an algorithm for translating simple phrase structure trees to CCG. After an overview of the feature system used by the Treebank CCG (section 3.4), the analysis of basic sentence and noun phrase structure as well as the necessary modifications to the algorithm are shown (sections 3.5–3.7). Section 3.8 shows how unary type-changing rules for certain types of adjuncts can be introduced into the grammar to ensure a compact lexicon without augmenting the generative power of the system, demonstrating that a wide coverage CCG does not require a prohibitively large lexicon. The algorithm is then extended to deal with the various kinds of null elements in the Penn Treebank which encode long-range dependencies arising through extraction (section 3.9) and coordination (section 3.10). The analysis of specific syntactic constructions is covered in detail; therefore this chapter serves a double purpose in providing a concise overview over the coverage and analyses of our CCG for English, which is largely based on Steedman (2000, 1996). In order to obtain the desired categorial derivation trees, some changes on the original Treebank trees need to be performed before translation. Changes that are required because the Treebank analysis does not conform to the CCG account are noted in sections 3.5–3.7. However, a large number of changes are necessary to correct inconsistencies and annotation errors in the data. These are discussed in sections 3.12 and in more detail in appendix A. Section 3.14 examines the size and coverage of the acquired lexicon and grammar. In section 3.15 the algorithm developed here is compared with an alternative procedure for the acquisition of AB categorial grammar lexicons from a subcorpus of the Penn Treebank without null elements (Watkinson and Manandhar, 2001). A direct comparison with related algorithms for the extraction of Lexicalized Tree-Adjoining Grammars and Lexical Functional Grammars from the Penn Treebank is less straightforward, however.

Appendix A gives more implementational details, such as the head-finding rules, the rules for distinguishing complements from adjuncts, and the changes made to the Penn Treebank in the current implementation.

3.2 The Penn Treebank

The Wall Street Journal subcorpus of the Penn Treebank contains about 1 million words of parsed and tagged Wall Street Journal text collected in 1989.

The Treebank markup encloses constituents in brackets, with a label indicating the part of speech tag or syntactic category. A typical example is shown here:

```
(S (PP-TMP (IN In)
       (NP (DT the) (NN past) (NN decade)))
   (, ,)
   (NP-SBJ (JJ Japanese) (NNS manufacturers))
   (VP (VBD concentrated)
       (PP-CLR (IN on)
        (NP (NP (JJ domestic) (NN production))
            (PP (IN for)
                (NP (NN export))))))
    (. .))
```

In the following, part of speech tags and other irrelevant details of the trees will be omitted when presenting examples.

The Treebank markup is designed so that complements and adjuncts can in general be distinguished, except for certain difficult cases, such as prepositional phrases. However, the complement-adjunct distinction is not always marked explicitly. Instead, we use heuristic procedures which rely on the label of a node and its parent to make this distinction. Syntactic heads are also not indicated explicitly, but existing head-finding procedures such as those originally given by Magerman (1994) were adapted to our purposes (see appendix A.1).

The Treebank markup uses different types of null elements to encode long-range dependencies arising through coordination and extraction. The presence of these null elements is what makes it possible to translate the Treebank trees to the corresponding CCG derivations for relative clauses, wh-questions and coordinate constructions such as right node raising. The treatment of these null elements is discussed in sections 3.9–3.10.

3.3 The basic algorithm

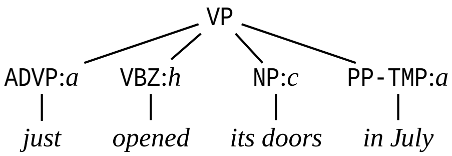
The basic algorithm for translating the Penn Treebank to CCG consists of three steps, each of which is a simple top-down recursive procedure:

(26)
```
foreach tree $\tau$:
       determineConstituentType($\tau$);
       makeBinary($\tau$);
       assignCategories($\tau$);
```

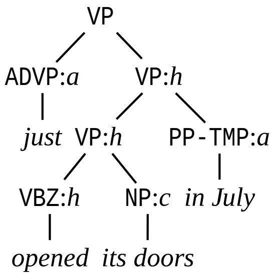
This algorithm presumes that the constituent structure of the original tree conforms to the desired CCG analysis. In practice, this is not always the case, and sections 3.5–3.7 give a detailed account of the reanalysis of specific constructions, and show how this algorithm can be adapted to deal with simple coordinate constructions. Sections 3.9 and 3.10 extend this algorithm to deal with the null elements in the Penn Treebank that encode long-range dependencies in constructions involving extraction and coordination.

This section explains the three steps of the basic algorithm.

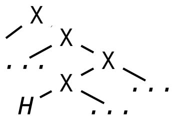
First, the constituent type of each node (head (*h*), complement (*c*), or adjunct (*a*)) is determined, using heuristics adapted from Magerman (1994) and Collins (1999) (for details, see appendix A):



Then the flat trees are transformed to binary trees.



This binarization process inserts dummy nodes into the tree such that all children to the left of the head branch off in a right-branching tree, and then all children to the right of the head branch off in a left-branching tree:



Categories are assigned to the nodes in a binary tree in the following manner (corresponding to a reverse CCG derivation):

The root node Each tree is rooted in a node labelled TOP. This node has one daughter, whose category is determined by the Treebank label of the root node of the Treebank tree (eg.$\{\mathtt{S},\mathtt{SINV},\mathtt{SQ}\} \rightarrow \mathsf{S}$, $\{\mathtt{VP}\} \rightarrow \mathsf{S\backslash NP}$). Section 3.4 explains the feature system used to distinguish different types of sentences (declarative, interrogative, embedded declarative, embedded interrogative) and verb phrases.

Complement nodes The category of a complement child is defined by a similar mapping from Treebank labels to categories.

Adjunct nodes Given a parent category X, the category of an adjunct child is a unary functor $\mathsf{X'/X'}$ if the adjunct child is the left daughter, or $\mathsf{X'\backslash X'}$ if it is the right daughter.

The category $\mathsf{X'}$ is determined by the parent category X. In order to avoid a proliferation of category types, adjunct categories do not carry any morphological features. This means for instance that VP adjuncts all have the category $\mathsf{(S\backslash NP)\backslash(S\backslash NP)}$ or $\mathsf{(S\backslash NP)/(S\backslash NP)}$ - that is, we do not distinguish between adjuncts appearing in declarative, infinitival, or passive verb phrases. *Quickly* is $\mathsf{(S\backslash NP)\backslash(S\backslash NP)}$ regardless of whether it modifies *buys*, *bought* or *buying*. In a version of categorial grammar without composition, $\mathsf{X'}$ would have to be equal to the current head category (without features). However, in the case of adjuncts of adjuncts, this leads to a proliferation of categories. But, in CCG, adjuncts can combine with the heads using (generalized) composition. Therefore, $\mathsf{X'}$ can be the current head catgory with the outermost arguments stripped off. Thus, the following strategy is used: A left adjunct, $\mathsf{X'/X'}$, can combine with the head using (generalized) forward non-crossing composition:

(27) *Forward Composition*:

$$\mathsf{X/Y} \quad \mathsf{Y/Z} \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X/Z}$$

Forward crossing composition is not permitted in English, since it would lead to greater freedom in word order than English allows:

(28) *Forward Crossing Composition*:

$$\mathsf{X/Y}: f \quad \mathsf{Y\backslash Z}: g \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X\backslash Z}$$

Hence, in the case of forward-looking (left) adjuncts ($\mathsf{X'/X'}$), $\mathsf{X'}$ is the parent category minus all outermost forward-looking arguments. $\mathsf{X'/X'}$ can then combine with the current head category through (generalized) forward non-crossing composition. If any backward-looking arguments were stripped off, $\mathsf{X'/X'}$ could only combine with the head through forward crossing composition.

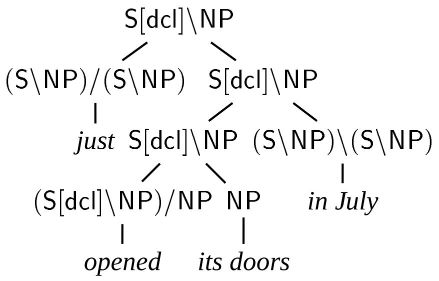
A CCG for English allows backward crossing as well as non-crossing composition. Therefore, in the case of backward-looking (right) adjuncts, $\mathsf{X'\backslash X'}$, all outermost arguments which have the same directionality as the last argument are stripped off from the parent category in order to obtain $\mathsf{X'}$ – that is, if the outermost argument of the current head category is forward-looking, then all outermost forward arguments are stripped off (corresponding to generalized backward crossing composition). If the outermost argument is backward-looking, all outermost backward arguments can be stripped off (generalized backward non-crossing composition).

In the case of VP-adjuncts, however, we stipulate that we do not generalize beyond the $\mathsf{S\backslash NP}$ level, since we want to distinguish verbal adjuncts from sentential adjuncts. Consider for instance the adjunct *in July*. Since its parent's category is $\mathsf{S\backslash NP}$ and it appears to the right of the head verb, it receives the category $\mathsf{(S\backslash NP)\backslash(S\backslash NP)}$.

Punctuation marks In general, the category of a punctuation mark is the POS tag assigned to the punctuation mark. Exceptions to this rule are discussed in section 3.7.4.

Head nodes The head child of a parent with category X has category X if the non-head child is an adjunct or a punctuation mark. If the non-head child is a complement with category Y, the category of the head child is $\mathsf{X/Y}$ if the head child is left, and $\mathsf{X\backslash Y}$ if the head child is right.

Here is the previous tree annotated with CCG categories:



The category assignment procedure corresponds to a reverse derivation which always uses function application, except for adjuncts, where composition can be used in order to provide a more general analysis. The extensions to this algorithm described below use type-raising and composition only when syntactically necessary. Therefore, the derivations in CCGbank are in a normal form.

3.4 Atomic categories and features in CCGbank

We assume the atomic categories S, NP, N and PP, and employ features to distinguish between declarative sentences (S[dcl]), wh-questions (S[wq]), yes-no questions (S[q]), embedded declaratives (S[emb]) and embedded questions (S[qem]).[1] We also distinguish different kinds of verb phrases ($S\backslash NP$), such as bare infinitives, to-infinitives, past participles in normal past tense, present participles, and past participles in passive verb phrases. This information is encoded as an atomic feature on the category, eg. $S[pss]\backslash NP$ for a passive VP, or S[dcl] for a declarative sentence. Predicative adjectives have the category $S[adj]\backslash NP$.

The sentential features are as follows:

(29) a. S[dcl]: for declarative sentences
b. S[wq]: for wh-questions
c. S[q]: for yes-no questions
d. S[qemb]: for embedded questions
e. S[emb]: for embedded declaratives
f. S[frg]: for sentence fragments (derived from the Treebank label FRAG)
g. S[for]: for small clauses headed by *for*
h. S[intj]: for interjections
i. S[inv]: for elliptical inversion

These are the verb phrase features:

(30) a. $S[b]\backslash NP$: for bare infinitives, subjunctives and imperatives
b. $S[to]\backslash NP$: for to-infinitives
c. $S[pass]\backslash NP$: for past participles in passive mode
d. $S[pt]\backslash NP$: for past participles used in active mode
e. $S[ng]\backslash NP$: for present participles

The main purpose of these features is to specify subcategorization information—for instance, the verb *doubt* takes both embedded declaratives (*doubt that*) and questions (*doubt whether*) as argument, whereas *think* can only take embedded declaratives. The complementizer *that*

[1] In order to improve readability, I will sometimes abbreviate the verb phrase category $S\backslash NP$ as VP in derivations. A category such as VP[dcl] always stands for $S[dcl]\backslash NP$. In some of the derivations given here, features are omitted.

takes a declarative sentence, and yields an embedded declarative: *that* ⊢ S[emb]/S[dcl]. These features are treated as atomic, and do not indicate the full morphosyntactic information. For instance, the infinitival particle *to* has the category (S[to]\NP)/(S[b]\NP), since it takes a bare infinitive (S[b]\NP) as its argument and yields a to-infinitival verb phrase S[to]\NP in both *”to give”* and *”to be given”*. Since the information whether the verb phrase is active of passive becomes available during the derivation, there is no need for separate lexical entries. The [dcl], [b], [ng], [pt] and [pss] features are determined from the POS tags of the head verbs and the presence of passive traces. [em], [qem], [q], [frg] and [tpc] are determined from the nonterminal labels of the trees. Adjunct categories other than modifiers of adjectives do not carry features.

Being derived from the POS tags in the Penn Treebank, this feature system is rather coarse; for instance it does not encode differences in verb mood: subjunctive and imperative appear as bare infinitive, whereas conditional and indicative appear both as declarative.

Following Bierner (2001), we also distinguish bare and non-bare noun phrases; however, we abbreviate non-bare noun phrases as N. Determiners, such as *the*, are functions from bare to non-bare noun phrases: NP[nb]/N. Plural nouns such as *researchers* are always bare (N). Apart from determiners, no other categories in the acquired lexicon specify the bareness of their noun phrase arguments.

3.5 Basic clause structure

This section shows how standard CCG analyses of basic clause types can be obtained from the corresponding Treebank annotation. We look in turn at simple declarative sentences, infinitival and participial verb phrases, passive, control and raising, small clauses, yes-no questions, inverted sentences, ellipsis and fragments. Wh-questions are dealt with in section 3.9.2.

3.5.1 Simple declarative sentences

Declarative sentences are annotated as S in the Treebank. The annotation is flat, with modifiers and punctuation marks appearing at the same level as the subject noun phrase and main verb phrase:

```
(TOP (S (NP-TMP Today)
        (, ,)
        (NP-SBJ PC shipments)
        (ADVP-TMP annually)
        (VP (VBP total)
            (NP some $38.3 billion)
            (ADVP-LOC world-wide))
        (. .)))
```

Since end-of-sentence punctuation marks modify the entire sentence, a separate sentence level needs to be inserted which includes everything but the end-of-sentence punctuation mark:

```
(TOP (S (S (NP-TMP Today)
           (, ,)
           (NP-SBJ PC shipments)
           (ADVP-TMP annually)
           (VP (VBP total)
               (NP some $38.3 billion)
               (ADVP-LOC world-wide)))
        (. .)))
```

Note that otherwise the binarization procedure would analyze the full stop as a modifier of S\NP. In order to improve readability, end-of-sentence punctuation marks will generally be omitted in the categorial derivation trees shown in this dissertation.

A sentence carries the feature [dcl] if its head verb has one of the following POS tags: VBZ (3rd person singular present), VBP (non-3rd-person singular present), VBD (past tense).

3.5.2 Infinitival and participial VPs, gerunds

In the Treebank, participial phrases, gerunds, imperatives and infinitival verb phrases are annotated as sentences with a * null subject (which can be co-indexed with another NP in the sentence, depending on the construction), whereas auxiliaries and modals take a VP argument:

(31) a.
```
(NP (NP the policy)
    (S (NP-SBJ (-NONE- *))
       (VP (TO to)
           (VP (VB seduce)
               (NP socialist nations)
               (PP-CLR into the capitalist sphere)))))
```

b.
```
(S (NP-SBJ-1 The banks)
   (VP (VBD stopped)
       (S (NP-SBJ (-NONE- *-1))
          (VP (VBG promoting)
              (NP the packages))))))
```

c.
```
(S (NP-SBJ Both sides)
   (VP (VBP are)
       (VP (VBG taking)
           (NP action))))
```

Any S with a null subject receives the category $\mathsf{S \backslash NP}$ (with appropriate verbal feature), which is also the category assigned to VP arguments. The feature $\mathsf{[b]}$ is used for bare infinitives (VB). It is also used for imperatives and the subjunctive, since they cannot be distinguished from bare infinitives. Present participles carry the feature $\mathsf{[ng]}$.

3.5.3 Passive

The surface subject of a passive sentence is co-indexed with a `*` null element which appears in the direct object position after the past participle, for example:

```
(S (NP-SBJ-1 John)
   (VP (VBD was)
       (VP (VBN hit)
           (NP (-NONE- *-1))
           (PP (IN by)
               (NP-LGS a ball)))))
```

In this case, the null element does not indicate an argument which should be reflected in the category of the participial. Instead, the correct lexical categories are as follows:

(32) a. *was* $\vdash \mathsf{(S[dcl]\backslash NP)/(S[pss]\backslash NP)}$

b. *hit* $\vdash \mathsf{S[pss]\backslash NP}$

The algorithm uses the presence of the `*` null element to distinguish past participles in passive verb phrases from past participles in active verb phrases such as the following example:

(33)
```
(S (NP-SBJ-1 John)
   (VP (VBZ has)
       (VP (VBN hit)
           (NP the ball))))
```

In this case, *hit* has the following lexical entry:

(34) *hit* $\vdash \mathsf{(S[pt]\backslash NP)/NP}$

We analyze the *by*-PP in passive verb phrases as an adjunct rather than as an argument of the passive participle. The reason for this is that the *by*-PP is optional, so the category (S[pss]\NP)/PP[by] does not have to be acquired for all passive verbs.

In the case of verbs like *pay for*, which subcategorize for a PP, the null element appears within the PP:

```
(S (NP-SBJ-30 (PRP$ Its)
     (NN budget))
   (VP (VBZ is)
       (VP (VBN paid)
           (PP-CLR (IN for)
                   (NP (-NONE- *-30)))
           (PP (IN by)
                   (NP-LGS (PRP you)))))
           (. .)))
```

In this example, the correct lexical categories are as follows:

(35) a. *is* ⊢ (S[dcl]\NP)/(S[pss]\NP)

b. *paid* ⊢ (S[pss]\NP)/(PP/NP)

c. *for* ⊢ PP/NP

Note that the preposition has its ordinary category PP/NP, and that the past participle subcategorizes for the preposition alone, instead of the saturated PP. This means that in passive verb phrases with passive traces in PPs in object position, the passive trace must be taken into account as an argument to the preposition, but it must also be percolated up to the PP level in order to assign the correct category to the past participle.

3.5.4 Control and raising

In the Treebank, raising and subject control both have a co-indexed *-trace in the subject position of the embedded clause, for instance:

(36) a.

```
(S (NP-SBJ-1 Mr. Stronach)
   (VP (VBZ wants)
       (S (NP-SBJ (-NONE- *-1))
          (VP (TO to)
              (VP (VB resume)
                  (NP a more influential role))))))
```

```
     b. (S (NP-SBJ-1 Every Japanese under 40)
           (VP (VBZ seems)
               (S (NP-SBJ (-NONE- *-1))
                  (VP (TO to)
                      (VP (VB be)
                          (ADJP-PRD fluent in Beatles lyrics))))))
```

Since an S with an empty subject NP has category $\mathsf{S\backslash NP}$, we obtain the correct syntactic category $\mathsf{(S[dcl]\backslash NP)/(S[to]\backslash NP)}$ for both *seems* and *wants*.

In the case of object control (37a), the controlled object appears as a separate argument to the verb and is co-indexed with a *-trace in subject position of the complement. Object raising (37b) is analyzed as a small clause in which the verb takes a sentential complement:

```
(37) a. (S (NP-TMP Last week)
           (, ,)
           (NP-SBJ housing lobbies)
           (VP (VBD persuaded)
               (NP-1 Congress)
               (S (NP-SBJ (-NONE- *-1))
                  (VP (TO to)
                      (VP raise the ceiling to $124,875))))

     b. (S (NP-SBJ Czechoslovakia)
           (ADVP-TMP still)
           (VP (VBZ wants)
               (S (NP-SBJ-1 the dam)
                  (VP (TO to)
                      (VP (VB be)
                          (VP (VBN built)
                              (NP (-NONE- *-1))))
```

However, as explained in more detail in section 3.5.5, the CCG account of these constructions (Steedman, 1996) assumes that both elements of the small clause are arguments of the verb, and we modify the tree so that we obtain the same lexical category $\mathsf{(((S[dcl]\backslash NP)/(S[to]\backslash NP))/NP)}$ for both verbs.

3.5.5 Small clauses

The Treebank adopts a small clause analysis for constructions such as the following:

```
(38) a. (S (NP-SBJ The country)
            (VP (VBZ wants)
                (S (NP-SBJ-2  half the debt)
                   (VP (VBN forgiven)
                       (NP (-NONE- *-2))))))
     b. (S (NP-SBJ that volume)
           (VP (VBZ makes)
               (S  (NP-SBJ it)
                   (NP-PRD (NP the largest supplier))
                        (PP of original TV programming)
                        (PP-LOC in Europe))))
```

If these verbs occur in the passive, they are analyzed as taking a small clause, with a passive NP null element as subject (see section 3.5.3):

```
(39) (S (NP-SBJ-1 The refund pool)
        (VP (MD may) (RB not)
            (VP (VB be)
               (VP (VBN held)
                   (S (NP-SBJ (-NONE- *-1))
                      (NP-PRD (NN hostage)))))))
```

Steedman (1996) argues against this kind of small clause analysis on the basis of extractions like "*what does the country want forgiven*", which suggest that these cases should rather be treated as involving two complements. We therefore eliminate the small clause, and transform the trees such that the verb takes the children of the small clause as complements. This corresponds to the following analyses:

```
(40) a. (S (NP-SBJ the country)
           (VP (VBZ wants)
               (NP half the debt)
               (VP (VBN forgiven)
                   (NP (-NONE- *-2)))))
     b. (S (NP-SBJ that volume)
           (VP (VBZ makes)
               (NP it)
               (NP-PRD (NP the largest supplier)
                       (PP of original TV programming)
                       (PP-LOC in Europe))))
```

c.
```
(S (NP-SBJ-1 The refund pool)
   (VP (MD may) (RB not)
       (VP (VB be)
           (VP (VBN held)
               (NP (-NONE- *-1))
               (NP-PRD hostage)))))
```

The other case where small clauses are used in the Treebank includes absolute *with* constructions, which are analyzed as adverbial SBAR:

(41) a.
```
(S (SBAR-ADV (IN With)
             (S (NP-SBJ the limit)
                (PP-PRD in effect)))
   (, ,)
   (NP-SBJ members)
   (VP would be able to execute trades at the limit price)
   (. .))
```

b.
```
(S (SBAR-ADV (IN Though)
             (S (NP-SBJ (-NONE- *-1))
                (ADJP-PRD (JJ modest))))
        (, ,)
        (NP-SBJ-1 the change)
        (VP (VBZ reaches)
            (PP-LOC-CLR  beyond the oil patch)
            (, ,)
            (ADVP too))
        (. .)))
```

We use the same approach for these cases, and assume that the subordinating conjunction (*with* or *though*, in these examples), takes the individual constituents in the small clause as complements. In the examples above, this gives the following lexical categories:

(42) a. *with* $\vdash ((\mathsf{S}/\mathsf{S})/\mathsf{PP})/\mathsf{NP}$

b. *though* $\vdash (\mathsf{S}/\mathsf{S})/(\mathsf{S[adj]}\backslash\mathsf{NP})$

3.5.6 Yes-no questions

The Treebank gives yes-no questions (labelled SQ) a flat structure, in which the inverted auxiliary/copula is a sister of both the subject NP and the VP or predicative NP:

(43) a.
```
(SQ (VBZ Did)
    (NP-SBJ I)
    (VP (VB buy)
        (NP it)
    (. ?))
```

b.
```
(SQ (VBZ Is)
    (NP-SBJ this)
    (NP-PRD the future of chamber music)
    (. ?)))
```

This flat structure corresponds to the categorial analysis of yes-no questions given eg. in Carpenter (1992). Here is the CCG derivation corresponding to example 43a:

(44)
```
       Did            I         buy        it?
 ---------------     ---   ---------------  ---
 S[q]/(S[b]\NP)/NP   NP    (S[b]\NP)/NP     NP
 ---------------------->   ---------------------->
     S[q]/(S[b]\NP)              S[b]\NP
 ------------------------------------------------->
                     S[q]
```

3.5.7 Inversion

The Treebank analyzes as `SINV` a number of constructions that do not follow the normal SVO pattern of English:

- verb fronting
- predicative fronting
- locative inversion
- negative inversion
- elliptical inversion
- conditional inversion
- direct speech inversion

Verb fronting The Treebank analyses predicate fronting and verb fronting in terms of movement, similar to topicalization:

```
(SINV (VP-TPC-1 (VBG Following)
                (NP the feminist and population-control lead))
      (VP (VBZ has)
          (VP (VBN been)
              (VP (-NONE- *T*-1))))
      (NP-SBJ a generally bovine press)
      (. .))
```

However, the noun phrase labelled as NP-SBJ here is actually in accusative case:

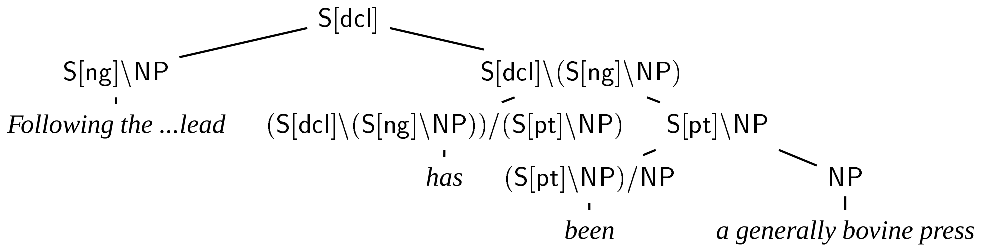
(45) a. **Following the lead has been I.*

b. *Following the lead has been me.*

Therefore, we assume that the NP is an object, and the verb phrase subject. The trees are changed so that the noun phrase appears as an object to the innermost verb. The VP-trace is removed:

```
(SINV (VP-TPC-1 (VBG Following)
                (NP the feminist and population-control lead))
      (VP (VBZ has)
          (VP (VBN been)
              (NP a generally bovine press)))
      (. .))
```

Here is the CCG derivation:

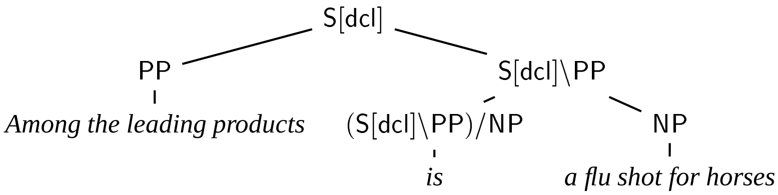


Predicative inversion Predicative inversion around the copula is also analyzed as topicalization within an SINV:

```
(SINV (PP-LOC-PRD-TPC-1 (IN Among)
                        (NP the leading products)))
```

```
      (VP (VBZ is)
          (PP-LOC-PRD (-NONE- *T*-1)))
      (NP-SBJ (NP a flu shot)
              (PP for horses))
      (. .))
```

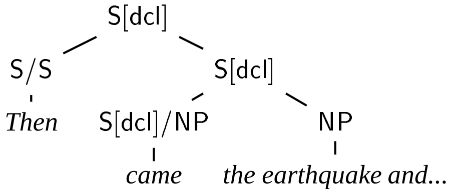
Both the fronted adverb and the subject are arguments of the verb. Since this construction occurs mainly in present tense or simple past, a modification of the tree like in the case of verb fronting was not implemented:



Locative inversion With certain verbs of movement the subject can appear to the right of the verb if there is a modifier to the left of the verb:

```
(SINV (ADVP-TMP Then)
      (VP came)
      (NP-SBJ (NP the earthquake))
              (CC and)
              (NP (NP a damaging delay)
                  (PP of 11 days)))
      (. .))
```

In those cases, both the adverb and subject should be arguments of the verb. This would require a modification to the adjunct-complement distinction which has not yet been implemented. Therefore, *came* receives category S[dcl]/NP in the above sentence:



Negative inversion In negative inversion, the negative element is often part of a sentential modifier, and hence difficult to identify:

```
(SINV (ADVP-TMP Never once)
      (VBD did)
      (NP-SBJ (PRP she))
      (VP (VP gasp for air)
          (CC or)
          (VP mop her brow))
      (. .))
```

The only case where negative inversion can be identified easily is after *nor*:

```
(S (S (NP-SBJ Michael)
      (VP won't confirm the identities of any Easy Egg customers))
   (, ,)
   (CC nor)
   (SINV (MD will)
         (NP-SBJ (PRP it))
         (VP (VB say)
             (NP much of anything else)))
   (. .)))
```

The word order following the negative item is like a yes-no question. This could be captured in CCG by letting the negative item subcategorize for a yes-no question:

(46)
$$\begin{array}{cc} \textit{nor} & \textit{will it say ...} \\ \hline (S\backslash S)/S[q] & S[q] \\ \hline \multicolumn{2}{c}{S\backslash S \quad >} \end{array}$$

This analysis was only implemented for negative inversion with *nor*, since in other cases, the negative item is difficult to detect given the Treebank markup.

Elliptical inversion Certain words, such as *so*, *than*, *as*, take an inverted elliptical sentence as argument. The resulting constituent is either a declarative sentence or an adjunct:

(47) *I attend, and so does a television crew from New York City.*

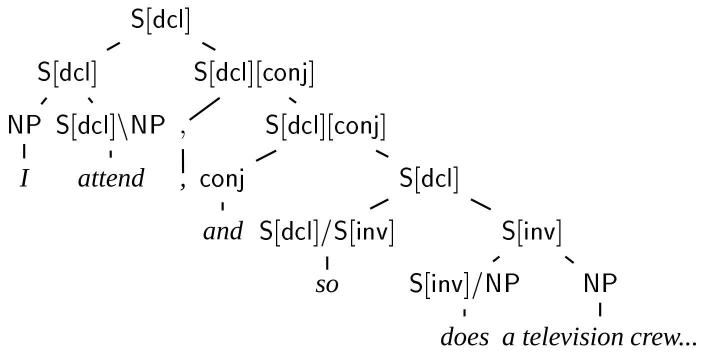
The Treebank annotation is flat:

```
(SINV (ADVP-PRD-TPC-1 (RB so))
      (VP (VBZ does)
          (ADVP-PRD (-NONE- *T*-1)))
      (NP-SBJ a television crew from New York City))
```

However, we change this to include a special kind of inverted sentence:

```
(SINV (ADVP-PRD-TPC-1 (RB so))
      (SINVERTED (VP (VBZ does)
                     (ADVP-PRD (-NONE- *T*-1)))
                 (NP-SBJ a television crew from New York City)))
```

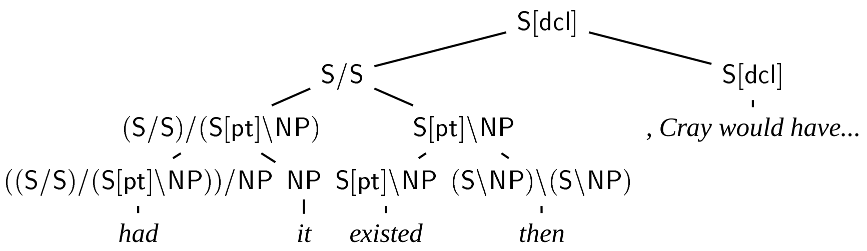
This inverted elliptical sentence consists of a do-form, auxiliary or modal, followed by the subject. A special feature [inv] on the CCG categories is used to distinguish those constructions from ordinary declaratives:



Conditional inversion Certain auxiliaries and modals (*had*, *should*, etc.) can introduce an inverted sentence which acts as a conditional clause:

```
(S (PP On the other hand)
   (, ,)
   (SBAR-ADV (SINV (VBD had)
                   (NP-SBJ (PRP it))
                   (VP (VBN existed)
                       (ADVP-TMP (RB then)))))
        (, ,)
        (NP-SBJ Cray Computer)
        (VP (MD would)
            (VP (VB have)
                (VP (VBN incurred)
                    (NP a $20.5 million loss))))
        (. .)))
```

Since this construction is triggered by auxiliaries, we treat the auxiliary as head of the adjunct, leading to the following derivation:

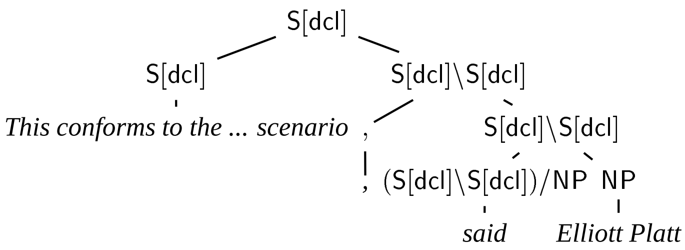


Recall that the features in our grammar do not distinguish conditional from indicative mood; therefore we cannot encode the fact that this construction requires the conditional in the main clause.

Direct speech inversion The Treebank analyses sentences where direct speech occurs before the verb as SINV:

```
(SINV (‘‘ ‘‘)
      (S-TPC-1 This conforms to the ‘soft landing’ scenario)
      (’’ ’’)
      (VP (VBD said)
           (S (-NONE- *T*-1)))
      (NP-SBJ Elliott Platt)
      (. .)))
```

However, since this is clearly a lexical phenomenon that can only occur with verbs of direct speech, the corresponding CCG analysis treats this word order as being determined in the lexicon, rather than being triggered by a topicalization rule:



In fact, verbs which take direct speech as argument can have at least three categories, one for the untopicalized case, and two for the sentence-topicalized cases with and without subject inversion:

(48) a. *Elliott Platt said: “This conforms to the ‘soft landing’ scenario”*

b. *“This conforms to the ‘soft landing’ scenario”, Elliott Platt said.*

c. *“This conforms to the ‘soft landing’ scenario”, said Elliott Platt.*

As explained below (section 3.12), quotation marks are cut out by the translation algorithm.

3.5.8 Ellipsis

The Treebank uses the null element `*?*` as placeholder "for a missing predicate or a piece thereof" (Marcus *et al.*, 1993). `*?*` is used for VP ellipsis, and can also occur in conjunction with a VP pro-form *do*, or in comparatives:

(49) a.
```
(S (NP-SBJ No one)
        (VP (MD can)
            (VP (VB say)
                (SBAR (-NONE- *?*))))
        (. .))
```
b.
```
(S (NP-SBJ The total of 18 deaths)
   (VP (VBD was)
       (ADJP-PRD (ADJP far higher))
            (SBAR (IN than)
                  (S (NP-SBJ (-NONE- *))
                     (VP (VBN expected)
                         (S (-NONE- *?*))))))))
```
c.
```
(S (S (NP-SBJ You)
      (RB either)
      (VP (VBP believe)
          (SBAR Seymour can do it again)))
   (CC or)
   (S (NP-SBJ you)
      (VP (VBP do)
          (RB n't)
          (VP (-NONE- *?*)))))
```

Although the `*?*` null element indicates a semantic argument of the head of the `VP` under which it appears (*e.g.* of *expected* or *do* in the examples above), we do not reflect this argument in the syntactic category of the heads. We follow the analysis of VP ellipsis under conjunction given in Steedman (2000), which argues that both conjuncts in examples such as (49c) are complete sentences. Therefore, the syntactic category of *do* is $\mathsf{S\backslash NP}$, not $\mathsf{(S\backslash NP)/VP}$. See section 3.5.7 for our treatment of elliptical constructions after words such as *as* or *than*.

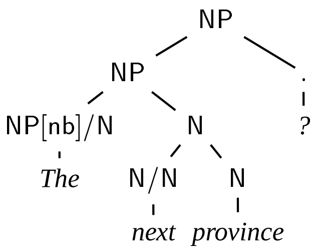
3.5.9 Fragments in the Treebank

The Treebank labels any constituent for which no proper analysis could be given as `FRAG` (for fragment). This can be an entire tree, or part of another constituent:

```
(50) a. (FRAG (NP The next province) (. ?))
     b. (SBARQ (WRB how) (RP about) (FRAG (NP the New Guinea Fund)) (. ?)
     c. (TOP (FRAG (FRAG (IN If)
                         (RB not)
                         (NP-LOC (NNP Chicago))
                         (, ,)
                         (RB then)
                         (PP-LOC (IN in) (NP New York)))
                   (: ;)
                   (FRAG (IN if)
                         (RB not)
                         (NP-LOC (the U.S.))
                         (, ,)
                         (RB then)
                         (NP-LOC overseas)))
                   (. .)))
```

These constituents are often difficult to analyze, and the annotation is not very consistent. Appendix A.4 lists some heuristics that were used to infer additional structure.

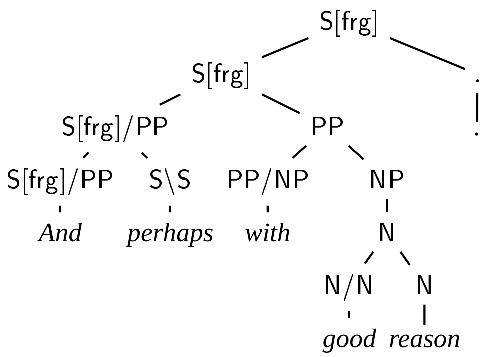
If an node is labelled FRAG, and there is only one daughter (plus, if this node is the root of the entire tree, optionally, a punctuation mark), we treat the tree as if it was labelled with the label of its daughter:



If the first daughter of a FRAG is a conjunction, we analyze it as head. The entire constituent is a fragment (S[frg]):

```
(TOP (FRAG (CC And)
           (ADVP (RB perhaps))
           (PP (IN with)
               (NP (JJ good)
                   (NN reason)))
           (. .)))
```

Here is the CCG translation:



3.6 Basic noun phrase structure

3.6.1 Noun phrases and nouns

The Treebank assumes a flat internal structure with no separate noun level:

(51) `(NP (DT the) (NNP Dutch) (VBG publishing) (NN group))`

Following the analysis of Bierner (2001), the implementation distinguishes non-bare noun phrases (NP[nb]) from nouns, or bare noun phrases, (N). Determiners, such as *the*, are functions from bare to non-bare noun phrases: *the* ⊢ NP[nb]/N. Other prenominal modifiers are functions from bare to bare noun phrases, eg. *Dutch* ⊢ N/N. Plural nouns are always bare: *researchers* ⊢ N. Nouns are not marked for number. The grammar includes a unary projection from NP to N, so that verb or other elements do not specify the bareness of their noun phrase arguments.

3.6.2 Compound nouns

Compound nouns in the Treebank have in general no internal structure:

```
(NP (JJR lower) (NN rate) (NNS increases))
(NP (JJ only) (JJ French) (NN history)  (NNS questions))
```

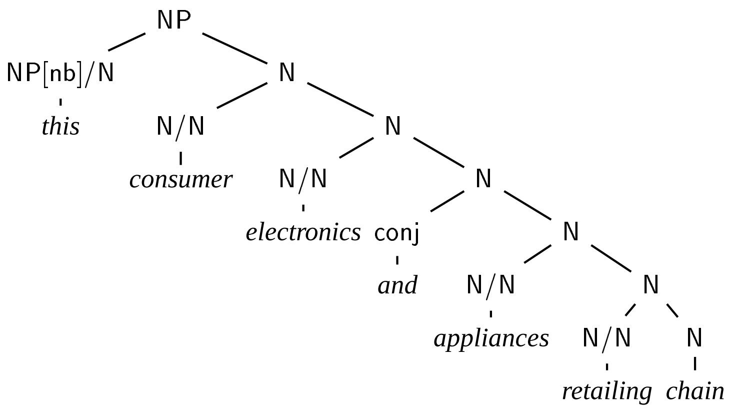
In order to obtain the correct analysis, manual re-annotation would be required, which was not deemed feasible within the current project. Therefore, compound nouns are simply translated into strictly right-branching trees. This is particularly problematic for conjunctions within compound nouns:

```
(NP (DT this)
    (NN consumer)
    (NNS electronics)
    (CC and)
    (NNS appliances)
    (NN retailing)
    (NN chain))
```

We include the following non-standard rule in our grammar:

(52) conj N ⇒ N

This rule allows us to translate the above tree as follows:



3.6.3 Appositives

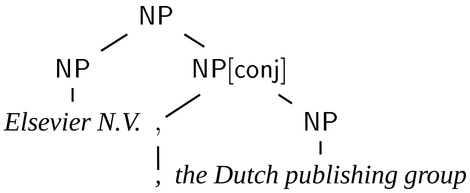
Appositives (53) are not indicated as such in the Treebank:

(53)
```
(NP (NP Elsevier N.V.)
    (, ,)
    (NP the Dutch publishing group))

(NP-SBJ-1 (NP his son)
          (, ,)
          (NP Zwelakhe)
          (, ,)
          (NP a newspaper editor)
          (, ,))
```

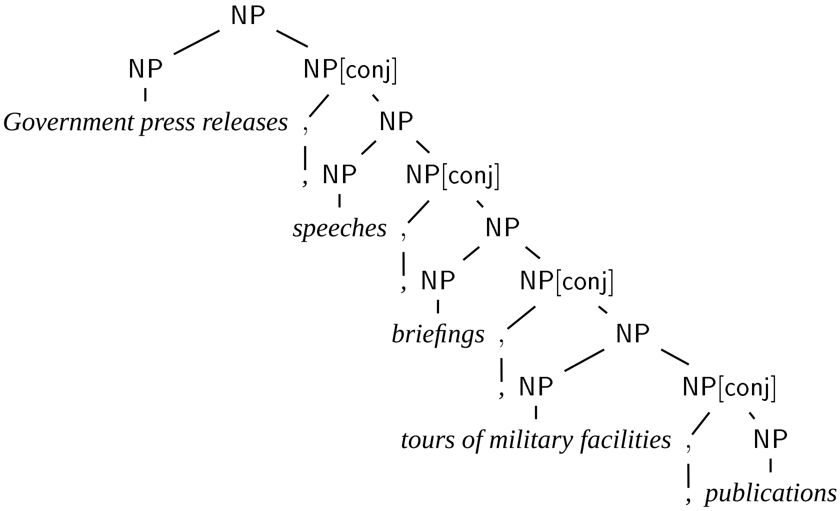
Their markup is indistinguishable from that of NP coordinations such as example (54)

```
(54)  (NP-SBJ (NP Government press releases)
              (, ,)
              (NP speeches)
              (, ,)
              (NP briefings)
              (, ,)
              (NP tours of military facilities)
              (, ,)
              (NP publications))
```

Therefore, our CCG does not distinguish between NP appositives and NP coordination, even though appositives should really be analyzed as modifiers:



Here is the “true” coordinate construction:



This leads to a reduction of ambiguity in the grammar, but is semantically not desirable.

3.6.4 Possessive *’s*

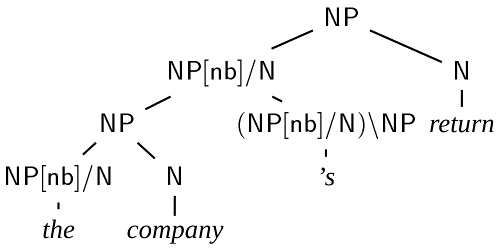
Possessive NPs in the Treebank have a flat structure in which the possessive *‘s* or ‘ is the last daughter:

```
(NP (NP (DT the) (NN company) (POS ’s))
    (NN return))
```

In CCG, the possessive *’s* and *’* are analyzed as functors from NPs to determiners. In order to obtain this analysis, we insert a new constituent POSSDT, which consists of the innermost NP and the possessive *’s* itself, so that the final structure (before assigning categories) is as follows:

```
(NP (POSSDT (NP the company)
            (POS ’s))
    (NOUN return))
```

Within a POSSDT, the POS is head, and the NP its argument; otherwise a POSSDT is like an ordinary determiner.



3.6.5 Quantifier phrases

The Treebank assumes a flat internal structure for QPs (“quantifier phrases”):

```
(QP (IN between) (CD 3) (NN %) (CC and) (CD 5) (NN %))
```

We use a number of heuristics to identify the internal structure of these constituents, for instance to detect conjuncts and prepositions. The above example is then re-bracketed as

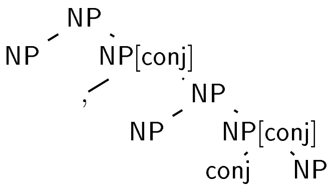
```
(QP (IN between)
    (QP (QP (CD 3) (NN %))
        (CC and)
        (QP (CD 5) (NN %)))
```

See appendix A.4.6 for details.

3.7 Other constructions

3.7.1 Coordination

Note that in the case of coordination (or lists), there is more than one head child, and we need to modify the translation algorithm accordingly. Coordinate constructions (or lists) are transformed into strictly right-branching binary trees. We assume that in those cases, all head children have the same category (see section 3.7.2 for a discussion of unlike coordinate phrases UCP). The inserted dummy nodes receive the same category as both conjuncts, but additionally carry a feature [conj]. A node with category X[conj] always has its head daughter (with category X) on the right, and the left daughter is either a conjunction (conj), or a punctuation mark, such as a comma or semicolon:



Therefore, coordination is captured in CCGbank by the following binary rule schemata:

(55) a. conj X ⇒ X[conj]

b. , X ⇒ X[conj]

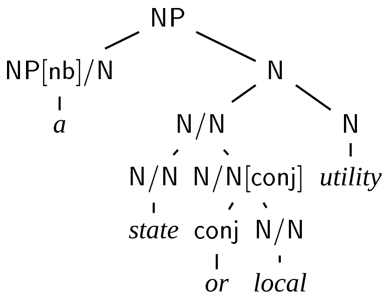
c. X X[conj] ⇒ X

3.7.2 ”Unlike” coordinate phrases

The Treebank annotates as UCP (“unlike coordinate phrase”) coordinate constructions where the conjuncts do not belong to the same syntactic category, eg:

```
(NP (DT a)
    (UCP (NN state)
         (CC or)
         (JJ local))
    (NN utility))
```

A UCP as head of a modifier is not a problem, because the CCG categories of modifiers only depend on the CCG category of the head:



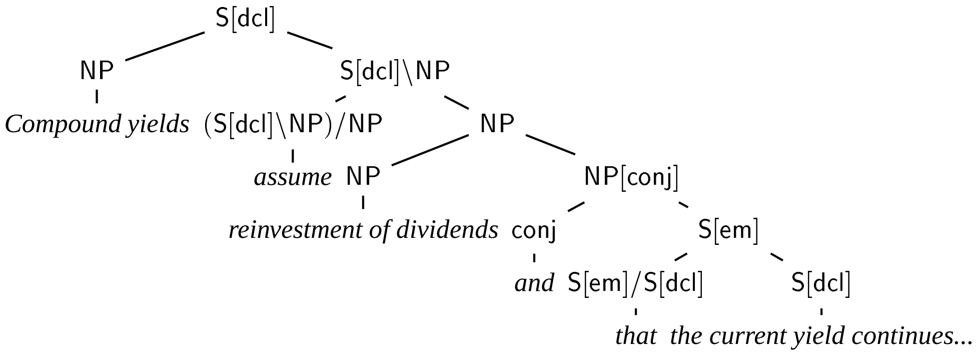
Such constructions only pose a problem for the translation algorithm if they occur as arguments, since the CCG category of arguments is determined by their Treebank label:

```
(S (NP-SBJ Compound yields)
   (VP (VBP assume)
       (UCP (NP reinvestment of dividends)
            (CC and)
            (SBAR (IN that)
                  (S (NP-SBJ the current yield)
                     (VP continues for a year)))))
   (. .))
```

In order to account for such cases, we modify the grammar slightly, and add special coordination rules of the following form:

(56) $\mathsf{conj}\quad \mathsf{Y} \Rightarrow \mathsf{X[conj]}$

This enables us to analyze the previous sentence:

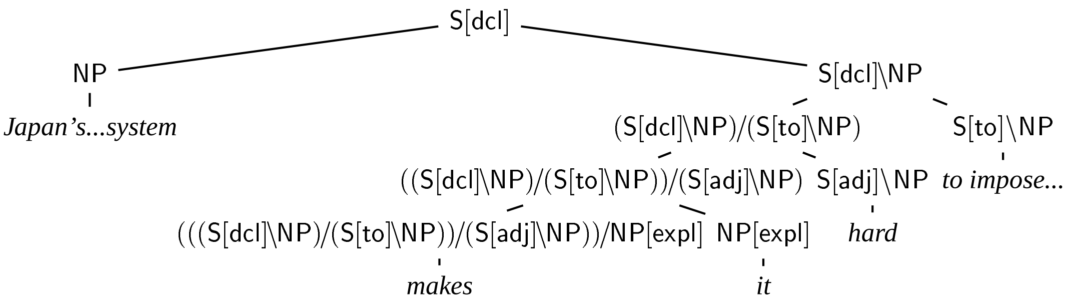


3.7.3 Expletive *it*

If *"it"* is annotated with an expletive trace `*EXPL*` or is the subject of a cleft sentence (see section 3.9.8), its category is NP[expl], and the verb subcategorizes for a NP[expl].

```
(TOP (S (NP-SBJ Japan ’s management system)
        (VP (VBZ makes)
            (S (NP-SBJ (NP (PRP it))
                       (S (-NONE- *EXP*-1)))
               (ADJP-PRD (JJ hard))
               (S-1 (NP-SBJ (-NONE- *))
                    (VP to impose a single system corporatewide))))
        (. .)))
```

The *EXPL* trace is cut out, resulting in the following CCG derivation:[2]

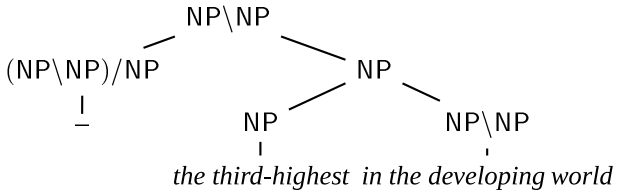


3.7.4 Parentheticals

The label PRN indicates a parenthetical element. A large number of these constituents constitute of a dash followed by one constituent or a constituent enclosed in parentheses, eg:

```
(PRN (: --)
     (NP (NP the third-highest)
         (PP-LOC (IN in)
                 (NP the developing world))))
```

We treat opening parentheses, dashes or colons as functors which take the following constituent and yield a modifier:

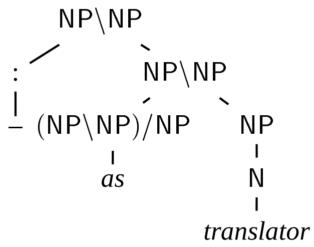


This is a slightly over-simplified analysis, since it does not take into account that dashes are balanced (see eg. (Nunberg, 1990)). However, if the enclosed constituent is of a type that could be a modifier within the local tree under which the PRN appears, the enclosed constituent is the head of the parenthetical and receives the same category as if it did not appear underneath a PRN:

```
(PRN (: -)
     (PP (IN as)
         (NP (NN translator))))
```

[2] An alternative analysis might treat the to-infinitive as an argument of the adjective; however, in the Treebank analysis, the to-infinitive is a sister, not a daughter of the adjectival phrase. Therefore the presen analysis is obtained.

Here, the PP is a modifier category; hence, the corresponding CCG derivation is as follows:



The dash is treated like an ordinary punctuation mark, and its category is determined from its POS tag, :.

3.7.5 Extraposition of appositives

Appositive noun phrases can be extraposed out of a sentence or verb phrase. However, in written English, a comma is required before or after the appositive:

(57) a. *No dummies,* the drivers pointed out they still had space.

b. Factory inventories fell 0.1% in September, *the first decline since February 1987.*

We analyze these constructions with special binary type changing rules that take into account that these appositives can only occur adjacent to commas:

$$\begin{array}{llll} \mathrm{NP} & , & \Rightarrow & \mathrm{S/S} \\ , & \mathrm{NP} & \Rightarrow & \mathrm{S\backslash S} \\ , & \mathrm{NP} & \Rightarrow & \mathrm{(S\backslash NP)\backslash(S\backslash NP)} \end{array}$$

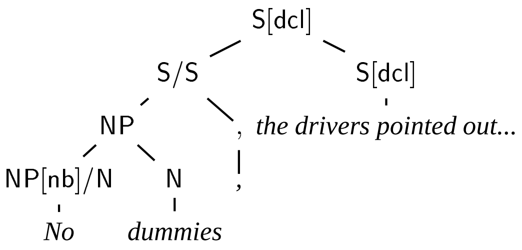
The Treebank analysis does not usually group the noun phrase and comma together, eg.:

```
(S (S-ADV (NP-SBJ (-NONE- *-1))
          (NP-PRD No dummies))
   (, ,)
   (NP-SBJ-1 the drivers)
   (VP pointed out they still had space ...).
```

Therefore, we insert a new constituent XNP which comprises the NP and the comma:

```
(S (XNP (S-ADV (NP-SBJ (-NONE- *-1))
          (NP-PRD No dummies))
   (, ,))
   (NP-SBJ-1 the drivers)
   (VP pointed out ...)).
```

This yields the following CCG analysis:



How these rules are acquired from the Treebank is explained in detail in appendix A.4.

3.7.6 Multi-word expressions

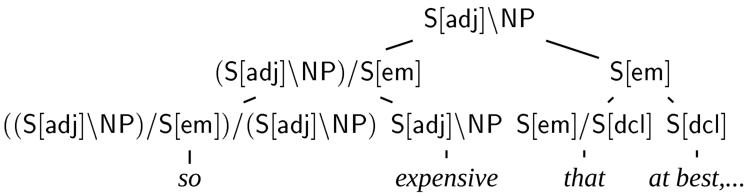
Multi-word expressions pose a particular problem for a lexicalist framework.With a few exceptions, explained below, an analysis for multi-word expressions is currently not attempted. This includes phrasal verbs, which are difficult to spot in the Treebank since particles can be found as PRT, ADVP-CLR and ADVP. Therefore, phrasal verbs do not subcategorize for particles in our grammar. An analysis was attempted for some common expressions that are either very frequent or where the multi-word expression has a different subcategorization behaviour from the head word of the expression.

because of...,instead of... We analyze *because* as the head of this expression. The prepositional phrase headed by *of* that follows it is its argument. When *instead of* appears within a PP, it is analyzed in the same way as *because of*.

as if..., as though Two special categories, S[as] and S[poss] are used for this construction. SBARs whose specifier is *if* and *though* have category S[poss] as a complement. SBARs whose specifier is *as* and which contain another SBAR with category S[poss] have category S[as] as complement.

so X that... In adjective phrases of the form *"so ADJ that..."*, we analyze *so* as a modifier of the adjective which also takes the embedded declarative as argument:

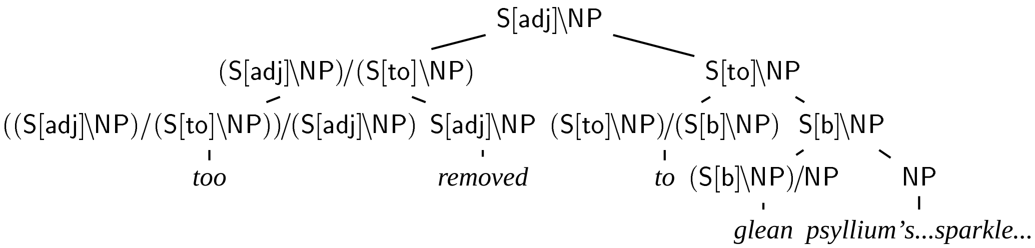
```
(ADJP-PRD (ADJP (RB so)
                (JJ expensive))
          (SBAR (IN that)
                (S (, ,)
                   (PP at best)
                   (, ,)
                   (NP-SBJ only the largest exchanges)
                   (VP can afford it))))
```



too ADJ to Similarly, constructions of the form *"too ADJ to"*, where the *to*-VP is a sister of *too* is changed to an analysis where *too* takes the *to*-VP as argument:

```
(ADJP (RB too)
      (VBN removed)
      (S (NP-SBJ (-NONE- *))
         (VP to glean psyllium 's new sparkle in the West)))
```

This is the resulting derivation:



This modification presumes that if the to-VP is an argument of the adjective (as in *"happy to do something."*), the to-VP is embedded within an ADJP, which in turn is modified by *"too"*.

at least/most X In this construction, we analyze *at* as taking a superlative adjective (with category S[asup]\NP) as argument.

Monetary expressions Amounts of currency (eg. *$ 1.5 billion*, *$ 3.85*) are among the most frequent type of multi-word expression in the Wall Street Journal. However, their internal structure is completely regular: it is always a currency symbol followed by a sequence of numbers (tagged CD), and syntactically, they behave like one item. Therefore, we replace these expressions by a string DOLLARS.

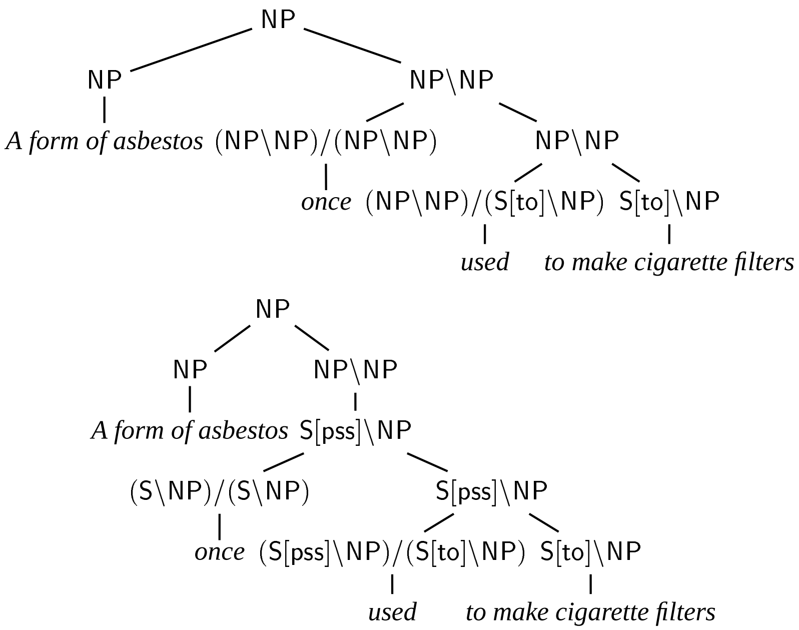


Figure 3.1: The effect of type-changing rules on lexical categories of adjuncts

3.8 Clausal adjuncts in a lexicalist framework

Figure 3.1 illustrates how the basic algorithm described in section 3.3 leads to a proliferation of adjunct categories. For example, a past participle such as *used* receives different categories depending on whether it occurs in a reduced relative or a main verb phrase. As a consequence, modifiers of *used* will also receive different categories depending on what occurrence of *used* they modify. This is undesirable, since we are only guaranteed to acquire a complete lexicon if we have seen participles (and all their possible modifiers) in all their possible surface positions. Similar regularities have been recognized and given a categorial analysis by Carpenter (1992), who advocates lexical rules to account for the use of predicatives as adjuncts.

It seems more economical to us to put such type-changing rules in the grammar. Such an approach has been taken by Aone and Wittenburg (1990) (also in a categorial framework) to encode morphological rules, and for reduced relatives and other syntactic constructions. Aone and Wittenburg show that these type-changing rules can be derived from zero morphemes in the grammar. Carpenter (1991, 1992) demonstrates that, in general, the inclusion of lexical rules can lead to a shift in generative power from context-free to recursively enumerable. However, this does not hold true if these lexical rules cannot operate on their own output and hence generate an infinite set of category types. Like Aone and Wittenburg, we only consider a finite number of instantiations of these type-changing rules, namely those which arise when we extend the category assignment procedure in the following way: for any sentential or verb phrase modifier (an adjunct with label S,SBAR with null complementizer, or VP) to which the original algorithm assigns category $\mathsf{X|X}$, apply the following type-changing rule (given in bottom-up notation) in reverse:

(58) $\mathsf{S\$} \Rightarrow \mathsf{X|X}$

where $\mathsf{S\$}$ is the category that this constituent obtains if it is treated like a head node by the basic algorithm. $\mathsf{S\$}$ has the appropriate verbal features, and can be $\mathsf{S\backslash NP}$ or $\mathsf{S/NP}$. Some of the most common type-changing rules are:

(59) a. $\mathsf{S[pss]\backslash NP} \Rightarrow \mathsf{NP\backslash NP}$
“workers [exposed to it]”

b. $\mathsf{S[adj]\backslash NP} \Rightarrow \mathsf{NP\backslash NP}$
“a forum [likely to bring attention to the problem]”

c. $\mathsf{S[ng]\backslash NP} \Rightarrow \mathsf{NP\backslash NP}$
“signboards [advertising imported cigarettes]”

d. $\mathsf{S[ng]\backslash NP} \Rightarrow \mathsf{(S\backslash NP)\backslash(S\backslash NP)}$
“become chairman, [succeeding Ian Butler]”

e. $\mathsf{S[dcl]/NP} \Rightarrow \mathsf{NP\backslash NP}$
“the millions of dollars [it generates]”

In written English, certain types of NP-extraposition require a comma before or after the extraposed noun phrase:

(60) *Factories booked \$236.74 billion in orders in September, [$_{NP}$ nearly the same as the \$236.79 billion in August]*

We make use of this fact in the following binary type-changing rules:

$$\begin{array}{llll} \mathsf{NP} & , & \Rightarrow & \mathsf{S/S} \\ , & \mathsf{NP} & \Rightarrow & \mathsf{S\backslash S} \\ , & \mathsf{NP} & \Rightarrow & \mathsf{(S\backslash NP)\backslash(S\backslash NP)} \end{array}$$

These rules are used whenever a NP-adjunct without functional tags, such as -TMP, appears at the periphery of a sentence or verb phrase and is immediately followed or preceded by a comma.

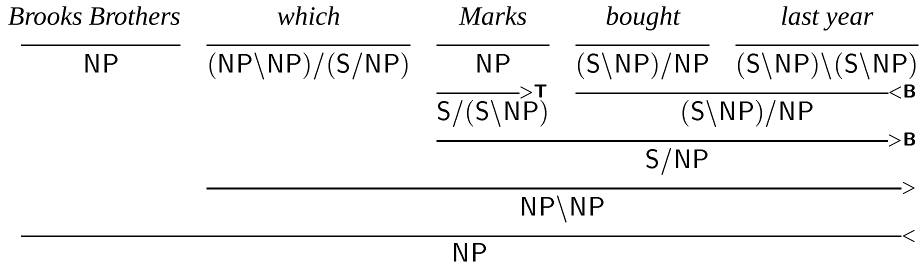
3.9 Long-range dependencies through extraction

The Treebank represents wh-questions, relative clauses, topicalization of complements, *tough* movement and parasitic gaps in terms of movement. The "moved" constituent is co-indexed with a trace (*T*) which is inserted at the extraction site:

```
(NP-SBJ (NP Brooks Brothers))
        (, ,)
        (SBAR (WHNP-1 (WDT which))
              (S (NP-SBJ NNP Marks))
                 (VP (VBD bought)
                     (NP (-NONE- *T*-1))
                     (NP-TMP last year)))))
        (, ,))
```

CCG has a different, but similarly uniform treatment of these constructions. What in transformational terms is described as the moved constituent is analyzed in CCG as a functor over a sentence missing a complement. For instance, the relative pronoun in the following examples has the category $(\mathsf{NP}\backslash\mathsf{NP})/(\mathsf{S}/\mathsf{NP})$, while the verb *bought* maintains its respective canonical categories $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$:

(61)



CCG allows the subject noun phrase and the incomplete verb phrase to combine via type-raising and forward composition to form a constituent with the category S/NP, which can in turn be taken as an argument of the relative pronoun. As the relative clause itself is a noun phrase modifier, the relative pronoun has the category $(\mathsf{NP}\backslash\mathsf{NP})/(\mathsf{S}/\mathsf{NP})$. This treatment of "movement" in terms of functors over "incomplete" constituents allows CCG to keep the same category for the verb even when its arguments are extracted.

The *T* traces in the Treebank help us in two ways to obtain the correct categorial analysis: firstly, their presence indicates a complement which needs to be taken into account in order to assign the correct category to the verb, and secondly, we can use a mechanism very similar to slash-feature passing in GPSG (Gazdar *et al.*, 1985) to obtain the correct categories for the wh-word and the incomplete sentence.

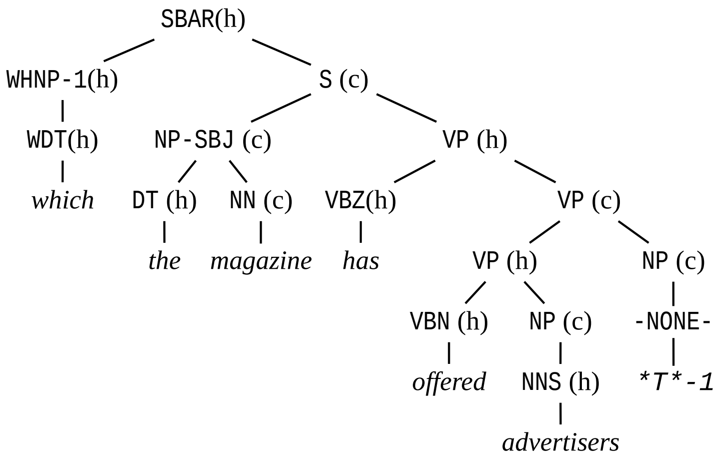
The following sections show our treatment of various constructions that involve long range dependencies arising through extraction, such as relative clauses, wh-questions, *tough* movement, parasitic gaps, topicalization, pied piping, subject extraction from embedded sentences and clefts. We use the example of relative clauses to explain in detail how the algorithm deals with `*T*` traces.

3.9.1 Relative clauses

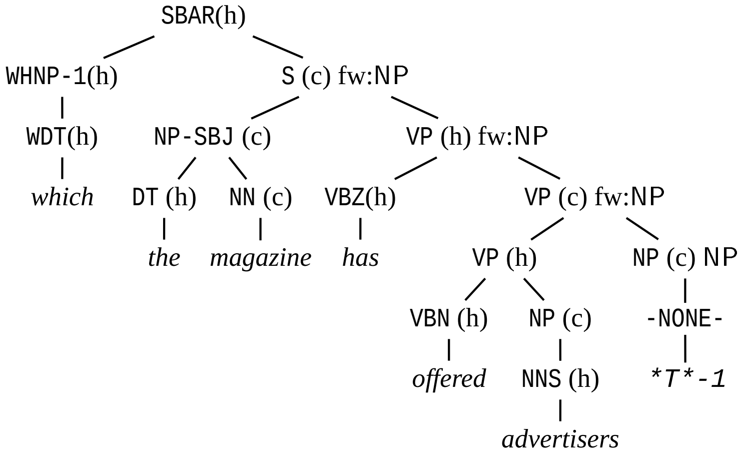
Here is the Treebank analysis of a relative clause:

```
(SBAR (WHNP-1 (WDT which))
  (S (NP-SBJ (DT the) (NN magazine))
     (VP (VBZ has)
         (VP (VBN offered)
            (NP (NNS advertisers))
            (NP (-NONE- *T*-1)))))))
```

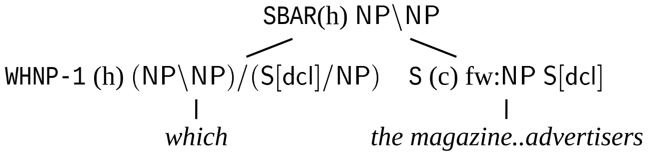
This is the same tree binarized and marked up with the constituent type:



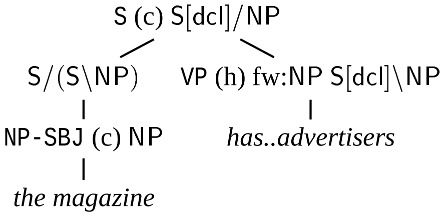
The NP-node with a `*T*` null element underneath it is a complement trace. The category of a complement trace (here NP) is determined (from its label) before the recursive category assignment described above. This category is then percolated up the head path of the parent of the trace to the next clausal projection. Depending on the position of the trace node within its local tree, this category is marked as a forward (fw) or backward (bw) argument:



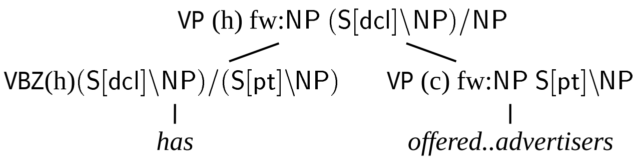
The S node is a complement, and receives category $S[dcl]$. However, since the S node also has a (forward) trace NP which is not percolated further to its parent, the head daughter (WHNP) subcategorizes for the unsaturated $S[dcl]/NP$. Assuming that the category of the SBAR parent is $NP\backslash NP$, the WHNP has category $(NP\backslash NP)/(S[dcl]/NP)$:



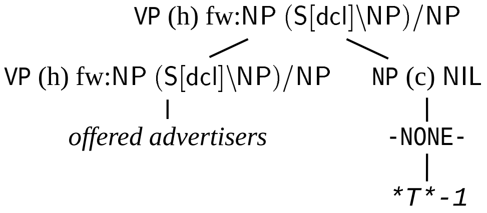
In the next step, the children of the S-node are assigned categories. The NP-SBJ is a backward complement with category NP, but since the S-node has a forward trace, it is type-raised to $S/(S\backslash NP)$. (Like adjunct categories, type-raised categories do not carry features). The forward trace is also appended to the category of the parent S node to yield $S[dcl]/NP$:



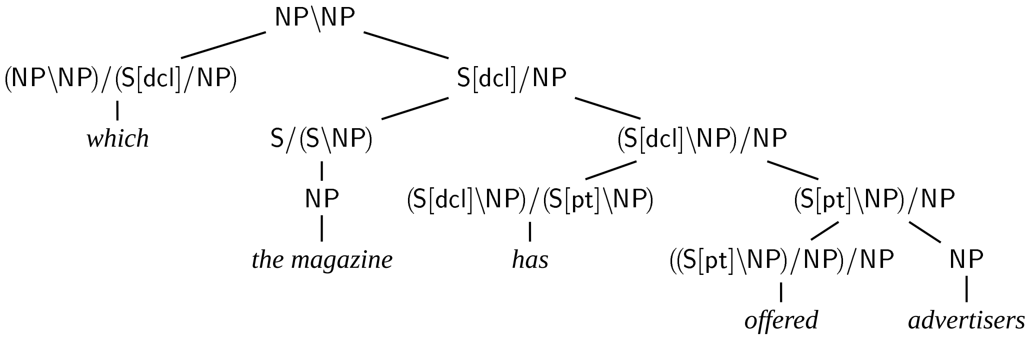
Then, categories are assigned to the children of the VP-node. The VP daughter is a complement with category $S[pt]\backslash NP$. Since the forward trace is carried up to the parent VP, the head daughter VBZ subcategorizes for $S[pt]\backslash NP$ and hence receives category $(S[dcl]\backslash NP)/(S[pt]\backslash NP)$. The forward trace is also appended to the category of the parent VP:



Going down to the children of the VP daughter, the NP trace is a forward argument. This yields the transitive verb category (S[pt]\NP)/NP for the VBN node. However, since the NP is a trace, this node will be cut out of the tree in a postprocessing stage. The category of the VP parent node is now also changed to take its forward trace into account.



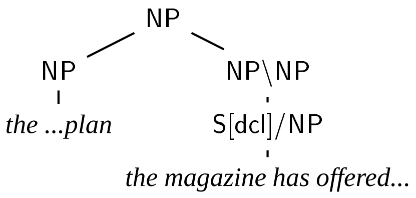
After cutting out the trace and all unary projections X ⇒ X, the subtree of the relative clause is as follows:



Reduced and zero relative clauses do not have a relative pronoun:

```
(NP-PRD (NP the second incentive plan))
        (SBAR (WHNP-1 (-NONE- 0))
              (S (NP-SBJ the magazine))
                 (VP (VBZ has)
                     (VP (VBN offered)
                         (NP advertisers)
                         (NP (-NONE- *T*-1))
                         (PP-TMP in three years))))
```

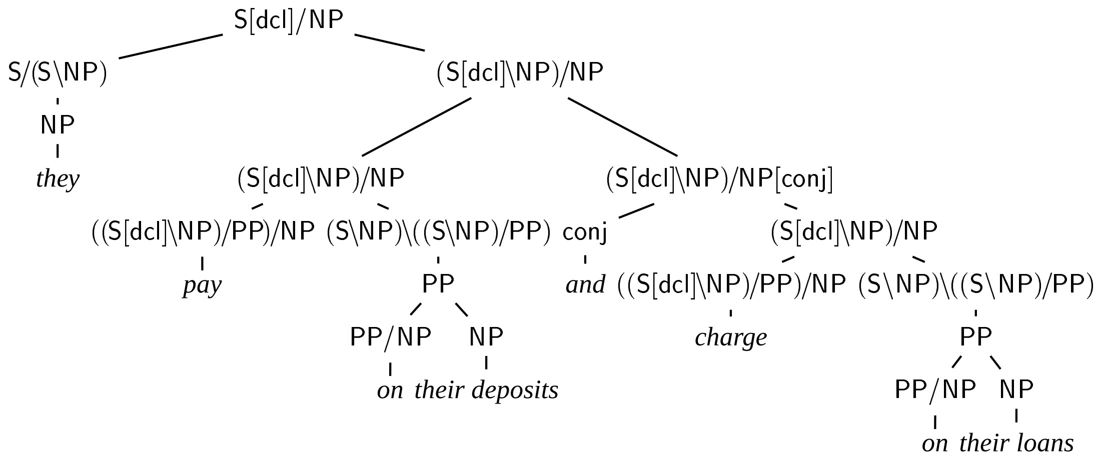
As already mentioned in section 3.8, we use unary type-changing rules (corresponding to the zero morphemes of Aone and Wittenburg (1990)) to account for this:



This algorithm works also if there is a coordinate structure within the relative clause such that there are two *T* traces:

```
(NP (NP the interest rates))
    (SBAR (WHNP-1 (-NONE- 0))
          (S (NP-SBJ (PRP they))
             (VP (VP (VBP pay)
                     (NP (-NONE- *T*-1))
                     (PP-CLR (IN on)
                             (NP their deposits))))
                 (CC and)
                 (VP (VB charge)
                     (NP (-NONE- *T*-1))
                     (PP-CLR (IN on)
                             (NP their loans))))))))
```

This results in the following CCG derivation for the relative clause:

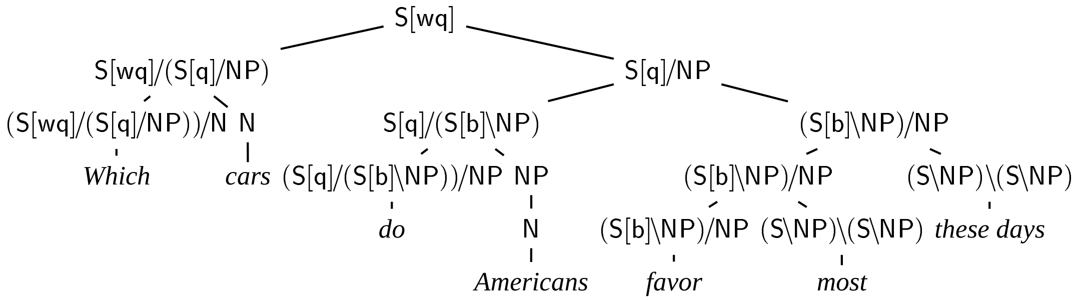


3.9.2 Wh-questions

T-traces are also used for wh-questions (labelled SBARQ):

```
(TOP (SBARQ (WHNP-1 (WDT Which)
                    (NNS cars))
            (SQ (VBP do)
                (NP-SBJ (NNP Americans))
                (VP (VB favor)
                    (NP (-NONE- *T*-1))
                    (ADVP (RBS most))
                    (NP-TMP (DT these)
                            (NNS days))))
            (. ?)))
```

The Treebank analyses wh-questions as SBARQ, with the wh-word in specifier position. It assumes a further level of structure, SQ, which includes the rest of the question. By percolating the *T*-trace up to the SQ-level, we obtain the desired CCG analysis:



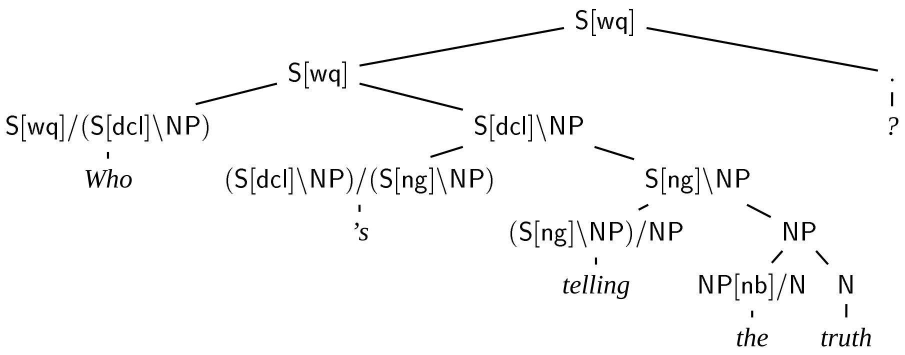
However, in subject questions, such as the following example, the following two kinds of analyses can be found:[3]

```
(62)    a. (SBARQ (WHNP-1 (WP Who))
                  (SQ (VBZ ’s)
                      (NP-SBJ (-NONE- *T*-1))
                      (VP (VBG telling)
                          (NP (DT the)
                              (NN truth))))
                  (. ?))
```

[3] Erratum: In Hockenmaier *et al.* (2002), we used an example where the trace was in the wrong place.

```
b. (SBARQ (WHNP-1 (WP Who))
          (S (NP-SBJ (-NONE- *T*-1))
             (VP (VBZ ’s)
                 (VP (VBG telling)
                     (NP (DT the)
                         (NN truth)))))
          (. ?))
```

For our purpose, a sentence with a trace in subject position is the same as a verb phrase; and the CCG analysis assumes in fact the the wh-word subcategorizes for a declarative verb phrase:



However, in order to obtain the correct CCG derivation from the first Treebank analysis, the inverted subject trace underneath the SQ has to be cut out, and the label SQ has to be changed to VP:

```
(SBARQ (WHNP-1 (WP Who))
       (VP (VBZ ’s)
           (VP (VBG telling)
               (NP (DT the)
                   (NN truth))))
       (. ?))
```

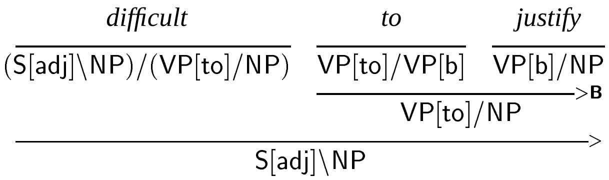
3.9.3 *Tough* movement

Tough movement is also annotated using *T*-traces:

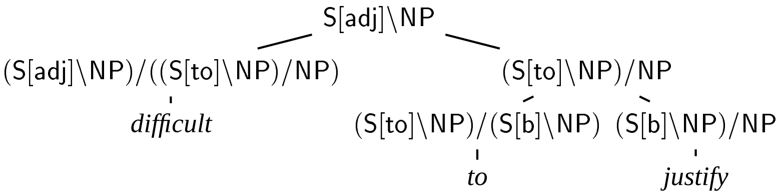
```
(S (NP-SBJ (PRP It))
   (VP (VBZ is)
       (ADJP-PRD (JJ difficult)
                 (SBAR (WHNP-1 (-NONE- 0))
                        (S (NP-SBJ (-NONE- *))
(VP (TO to)
    (VP (VB justify)
        (NP (-NONE- *T*-1))))))))
```

Following Steedman (1996), the adjective phrase has the following categorial analysis:

(63)



We obtain this analysis from the Treebank by percolating the forward NP slash category to the SBAR-level. As explained in section 3.5.2, the empty subject in infinitival verb phrases is cut out.



3.9.4 Parasitic gaps

Steedman (1996) analyzes parasitic gaps such as the following example with the combinatory rule of backward crossing substitution:

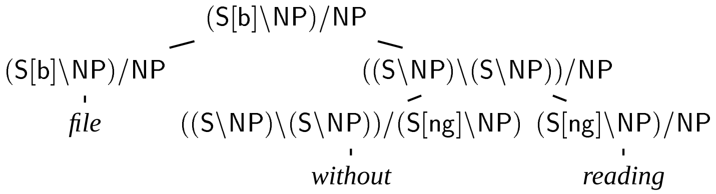
(64) *Which papers$_i$ did you file* t$_i$ *without reading* t$_i$*?*

This construction does not seem to occur in the Treebank. However, if it did, and if the annotation did contain the appropriate traces in both locations, our algorithm would be able to acquire the correct analysis.

Imagine the previous example was annotated as follows:

```
(SBARQ (WHNP-1 (WDT Which) (NNS papers))
   (SQ (VBD did)
      (NP-SBJ-2 (NN you))
      (VP (VB file)
          (NP (-NONE- *T*-1))
          (PP (IN without)
             (S-NOM (NP-SBJ (-NONE- *-2))
                  (VP (VBG reading)
                      (NP (-NONE *T*-1))))))))
```

There are two traces that are percolated up; however, each constituent can have at most one trace. Therefore, we would obtain the following analysis:

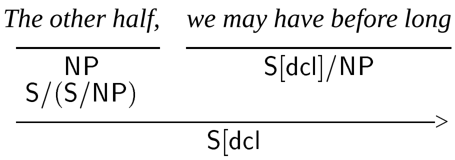


3.9.5 Topicalization

Following Steedman (1987), we assume that English has the following schema of a non order-preserving type-raising rule to account for topicalization of noun phrases, adjective phrases and prepositional phrases:

(65) $\mathsf{X} \Rightarrow \mathsf{S/(S\backslash X)}$
with $\mathsf{X} \in \{\mathsf{NP}, \mathsf{PP}, \mathsf{S[adj]\backslash NP}\}$

(66)



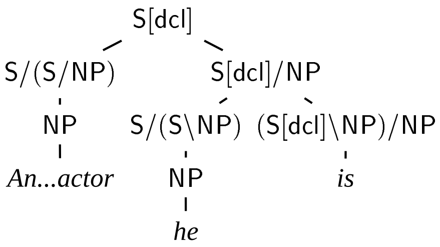
The Penn Treebank uses `*T*`-traces also for topicalization. If a constituent is topicalized, or fronted, it receives the tag `-TPC`, and is placed at the top level of the sentence. A co-indexed `*T*`-trace is inserted at the canonical position of that constituent:

```
(TOP (S (NP-TPC-1 An excellent environmental actor))
        (NP-SBJ (PRP he))
        (VP (VBZ is)
            (NP-PRD (-NONE- *T*-1)))
        (. .)))
```

We stipulate that categories of the form $\mathsf{S/(S\backslash X)}$ which can only be derived by non-order-preserving type-raising, can be assigned by the parser to any constituent of category X in sentence initial position.[4] Therefore, this category need not appear in the lexicon. Topicalised

[4] This assumption is not implemented in the parser or the probability model presented in the following chapters.

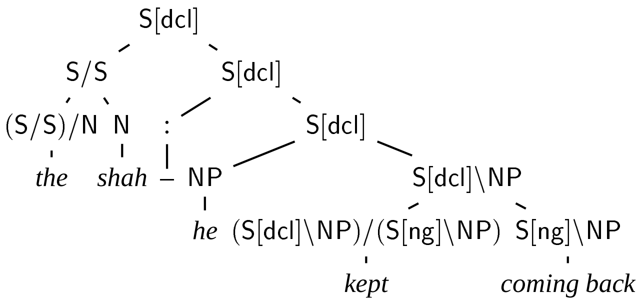
noun phrases (NP-TPC) receive the category NP, but in order to assign the correct category to the verb, an NP with tag -TPC is not considered a complement:



If there is a resumptive pronoun (that is, the fronting is an instance of left-dislocation), there is no coreference between the fronted element and the pronoun:

```
(S (NP-TPC the shah)
   (: --)
   (NP-SBJ-1 he)
   (VP (VBD kept)
       (S (NP-SBJ (-NONE- *-1))
          (VP coming back))))
```

In these cases, we obtain the correct lexical entry for the verb in the same manner, since it suffices to treat the (NP-TPC) as adjunct:

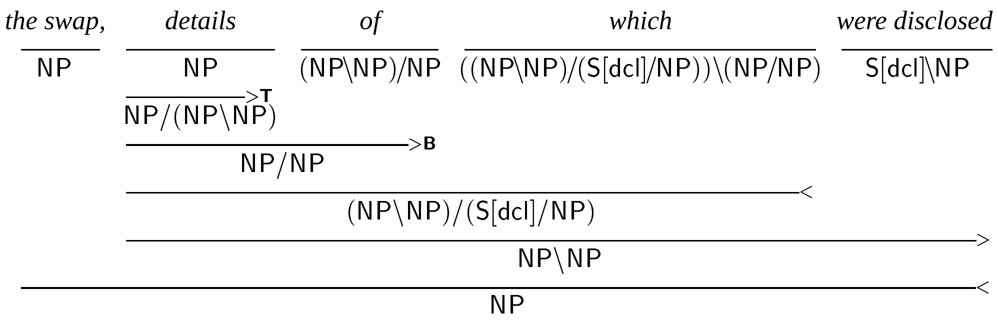


See 3.5.7 for the treatment of S-TPC and VP-TPC.

3.9.6 Pied piping

We follow the analysis of Steedman (1996) for pied piping:

(67)



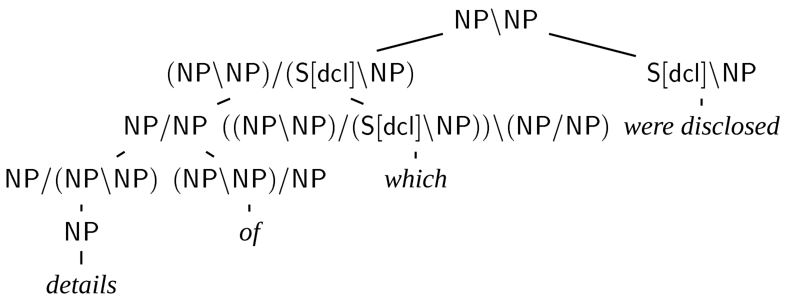
In the corresponding Treebank tree the preposition and relative pronoun form a PP which modifies the head noun.

```
(NP-SBJ (NP the swap)
        (, ,)
        (SBAR (WHNP-2 (WHNP (NNS details))
                      (WHPP (IN of) (WHNP (WDT which))))
              (S (NP-SBJ-1 (-NONE- *T*-2))
                 (VP were disclosed))))
```

However, in the CCG analysis, the preposition *"of"* combines first with the noun phrase *"details"*, and the relative pronoun *"which"* takes this constituent as argument. Therefore, a new constituent `piedPipeNP` is inserted, which consists of the noun phrase and preposition:

```
(NP-SBJ (NP the swap)
        (, ,)
        (SBAR (WHNP (piedPipeNP (WHNP (NNS details))
                                (IN of))
                    (WHNP (WDT which)))
              (S (NP-SBJ-1 (-NONE- *T*-2))
                 (VP were disclosed))))
```

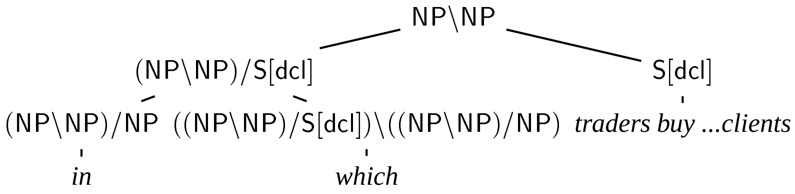
The preposition is an adjunct of the noun phrase; however, a special rule during category assignment guarantees that it is assigned the ordinary category of an NP-modifying preposition, $(\mathrm{NP}\backslash\mathrm{NP})/\mathrm{NP}$, and that the noun phrase is type-raised and composed with the preposition. This constituent is then treated as an argument of the relative pronoun.



Similarly, there are cases of pied piping with only the PP:

```
(NP (NP commodities markets)
    (, ,)
    (SBAR (WHPP (IN in)
                (WHNP (WDT which)))
      (S (NP-SBJ traders)
         (VP buy and sell both for their own account and for clients))))
```

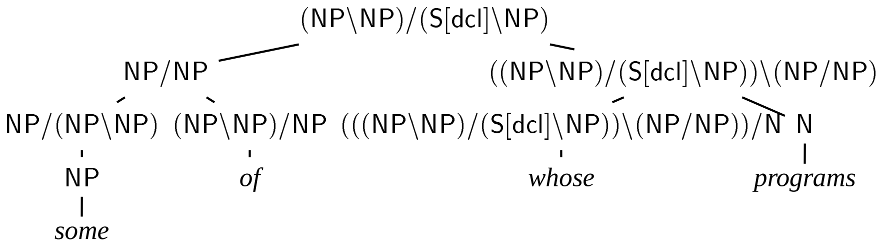
Here, we leave the tree as it is, since its constituent structure already corresponds to the CCG analysis. In this case the `WHNP` is head. The preposition is an argument and receives the NP-modifying category (NP\NP)/NP:



In the case of pied piping with *"whose"*, the relative pronoun also takes a noun as argument:

```
(NP-PRD (NP Ruth Messinger)
   (, ,)
   (NP a Manhattan city councilwoman)
   (, ,)
   (SBAR (WHNP-1 (NP (DT some))
                 (WHPP (IN of)
                       (NP (WHNP (WP$ whose)
                           (NNS programs))
                 (, ,)
                 (ADJP such as commercial rent control)
                 (, ,))))
          (S (NP-SBJ (-NONE- *T*-1))
             (VP have made their way into Mr. Dinkins 's position papers)
```

This is the CCG derivation:



3.9.7 Subject extraction from embedded sentences

Steedman (1996) assumes that the verb which takes a bare sentential complement from which a subject is extracted has category $((S\backslash NP)/NP)/(S\backslash NP)$. This category is only required when the subject is extracted.

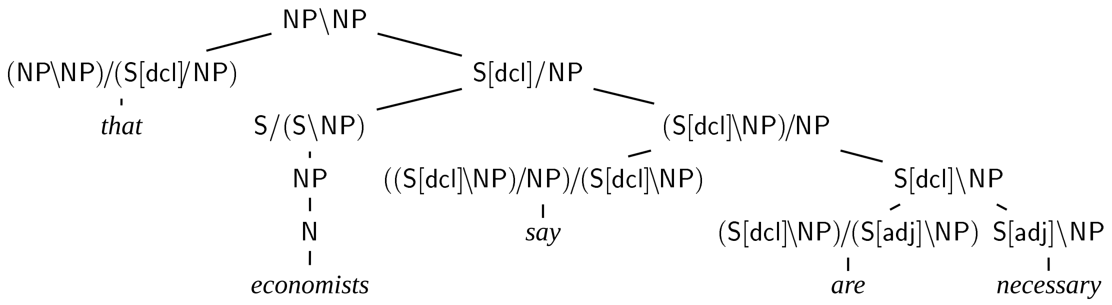
This constituent structure does not correspond to the Treebank analysis, where subject traces are in the ordinary position:

```
(NP (NP the sort)
    (PP of measures)
    (SBAR  (WHNP-1 (IN that))
           (S (NP-SBJ (NNS economists))
              (VP (VBP say)
                  (SBAR (-NONE- 0)
                        (S (NP-SBJ (-NONE- *T*-1))
                           (VP (VBP are)
                               (ADJP-PRD necessary))))))))
```

However, in order to obtain the desired analysis, we assume that the verb takes the VP and the NP argument in reversed order. Therefore, the tree is changed as follows:

```
(NP (NP the sort)
    (PP of measures)
    (SBAR  (WHNP-1 (IN that))
           (S (NP-SBJ (NNS economists))
              (VP (VBP say)
                  (VP (VBP are)
                       (ADJP-PRD (JJ necessary)))
                  (NP-SBJ (-NONE- *T*-1))))))
```

This then leads to the following CCG analysis:

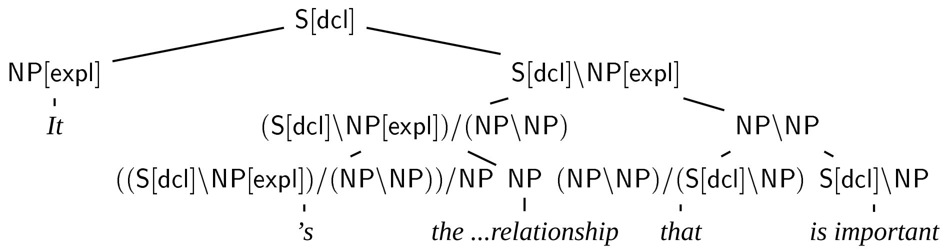


3.9.8 Clefts

Clefts are annotated as S-CLF in the Treebank:

```
(TOP (S-CLF (NP-SBJ (PRP It))
            (VP (VBZ 's)
                (NP-PRD the total relationship)
                (SBAR (WHNP-2 (WDT that))
                      (S (NP-SBJ (-NONE- *T*-2))
                         (VP (VBZ is)
                             (ADJP-PRD (JJ important))))))
            (. .)))
```

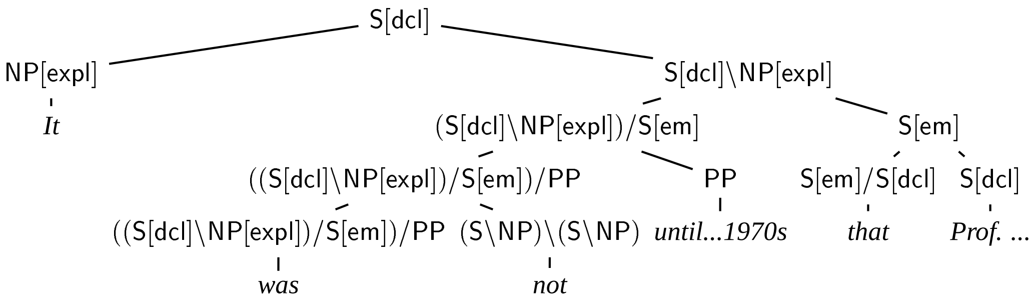
Although there is no *EXP*-trace, all subject noun phrases under S-CLF are assigned the category NP[expl]. We treat the SBAR as argument of the verb. If the focus of the cleft is a NP, as in the above example, its category is that of a noun phrase modifier NP\NP:



If the focus of the cleft is an adverb, the SBAR is simply an embedded declarative, eg:

```
(TOP (S-CLF (NP-SBJ (PRP It))
            (VP (VBD was)
                (RB not)
                (PP-PRD until the early 1970s)
                (SBAR (IN that)
                      (S (NP-SBJ-2 Prof. Whittington and two graduate students)
                         (VP began to publish ...))))
            (. .)))
```

Here is the corresponding CCG translation:

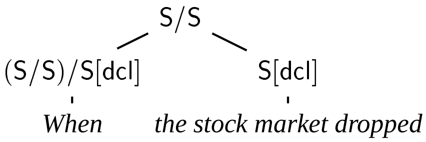


3.9.9 Extraction of adjuncts

T-traces can also stand for an adjunct (here indicated by the label ADVP-TMP):

```
(TOP (S (SBAR-TMP (WHADVP-1 (WRB When))
                  (S (NP-SBJ the stock market)
                     (VP (VBD dropped)
                         (ADVP-TMP (-NONE- *T*-1)))))
        (S (NP-SBJ the Mexico fund)
           (VP plunged about 18%))
        (. .)
```

Adjunct traces do not indicate missing complements, and are therefore simply ignored by the translation procedure.



3.9.10 Heavy NP shift

In English, some noun phrase arguments can be shifted to the end of the sentence if they become too “heavy”:

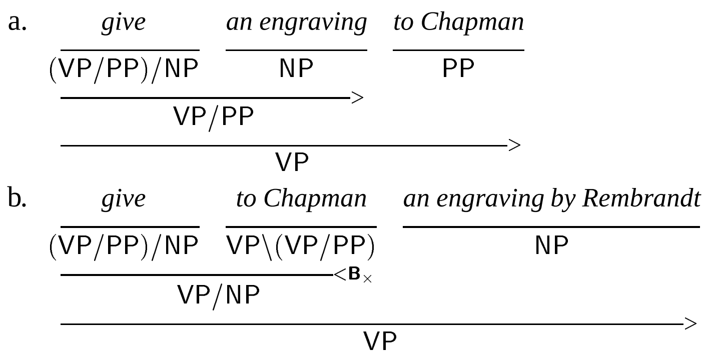
(68) a. *give an engraving to Chapman*

b. *give to Chapman an engraving by Rembrandt*

This kind of construction has been studied extensively by Ross (1967).

In order to provide an analysis according to which *give* has the same lexical category in the second case, Steedman (1996) uses backward-crossing composition:

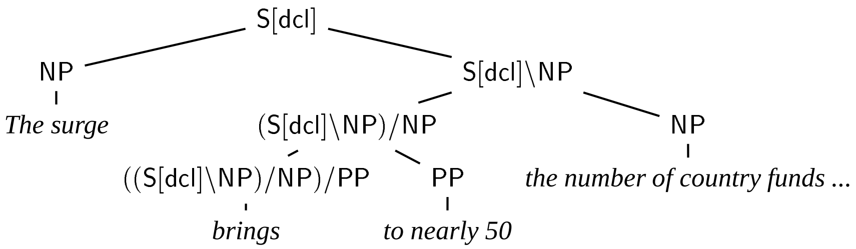
(69)



In the Penn Treebank, heavy NP shift is not marked:

```
(S (NP-SBJ The surge))
   (VP (VBZ brings)
       (PP-CLR to nearly 50)
       (NP (NP the number)
           (PP (IN of)
               (NP country funds
                   that are or soon will be listed in New York or London))))
        (. .)))
```

From this tree, the following CCG translation is obtained, which does not conform to Steedman's analysis:



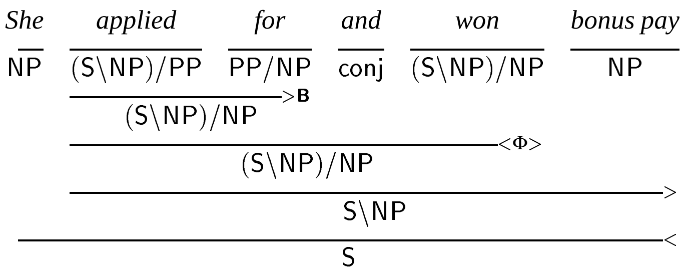
Backward crossing composition is also used in Steedman (1996, 2000) to account for certain preposition stranding phenomena in English. However, in its unrestricted form, this rule leads to overgeneralization. In order to avoid this overgeneralization, Steedman advocates the use of two features, *SHIFT* and *ANT*, on the arguments of lexical functor categories. At present, such features are not induced from the Penn Treebank. The grammar which underlies CCGbank only consists of particular rule instantiations, and might therefore not be prone to the overgeneration problem which motivated Steedman's features. We leave the question of whether such features would be necessary for a CCG that consists only of the rule instantiations found in a corpus, and how they could be obtained, open to further investigation.

3.10 Long-range dependencies through coordination

3.10.1 Right node raising

Right node raising constructions such as (70) can be analyzed in CCG using the same lexical categories as if the shared complement was present in both conjuncts (Steedman, 1996):

(70)



In order to assign the correct lexical categories to such sentences, we need to know where the canonical location of the shared complement is, ie. that the shared constituent is interpreted as a complement of both verbs, and that sentence 70 means the same as:

(71) *She applied for bonus pay and won bonus pay.*

The Treebank adopts an analysis of this construction in which the shared constituent is co-indexed with two `*RNR*`-traces that occur in the canonical position of the shared element:

```
((S (NP-SBJ She)
    (VP (VP (VBD applied)
        (PP-CLR (IN for)
           (NP (-NONE- *RNR*-1))))
      (CC and)
      (VP (VBD won)
        (NP (-NONE- *RNR*-1)))
      (NP-1  bonus pay)
      (PP-LOC (IN under) (NP the reform law)))
     (. .)))
```

In order to assign correct lexical categories to sentences with right node raising, we need to alter the translation algorithm slightly. The category assignment proceeds in three steps for sentences which contain `*RNR*`-traces:

1. When determining the constituent type of nodes: Identify all nodes which are co-indexed with `*RNR*`-traces (eg. `NP-1`). These constituents are neither heads, complements nor adjuncts, and hence will get ignored in the category assignment. `*RNR*`-traces themselves (or their maximal projections, here `NP`s) are treated like ordinary constituents, and thus they can be either heads, complements or adjuncts.

2. Assign categories to the nodes as before. Nodes which are co-indexed with `*RNR*`-traces (eg. the `NP-1` above) will be ignored because they are neither heads, complements nor adjuncts. `*RNR*`-traces themselves will receive the category of an ordinary constituent in this canonical position. If an `*RNR*`-trace is a complement, its category is percolated up to the topmost level of this coordination. This trace category is treated like a trace arising through extraction.

3. If the `*RNR*`-traces with the same index do not have the same category, this sentence cannot be processed, as the CCG analysis predicts that both constituents in canonical position have the same category.[5] Otherwise, copy the category of the `*RNR*`-traces, and assign it to the co-indexed node. Then assign categories in the usual top-down manner to the subtree beneath the co-indexed node.

Ignoring the co-indexed constituent *bonus pay* in the first pass guarantees that *applied* is assigned $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{PP}$, not $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}/\mathsf{PP}$. Considering the `*RNR*`-traces as ordinary constituents guarantees that *for* is assigned PP/NP, not PP, and *won* $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$, not $\mathsf{S}\backslash\mathsf{NP}$.

In the above example, the shared constituent is a complement. The same algorithm works if the shared constituent is an adjunct, although in that case it is not strictly necessary to use this two-pass procedure.

In English it is also possible for two conjoined noun phrases to share the same head:

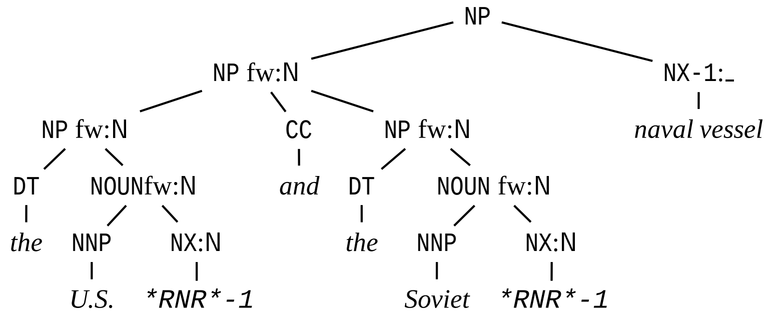
(72) *a U.S. and a Soviet naval vessel*

This is also annotated with `*RNR*`-traces.

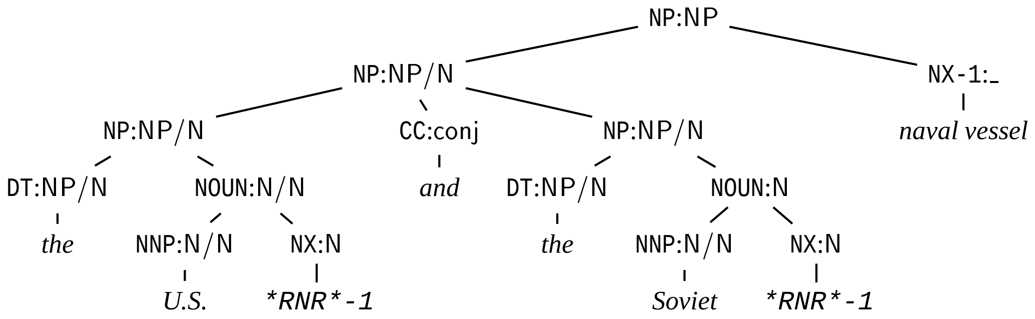
[5] This would arise if two `*RNR*`-trace complements did not have the same Treebank label, say if one was a `PP` and the other a `NP`, but in practice this does not happen. Note that in the adjunct case it is also conceivable that the categories of the two `*RNR*`-traces differ because they might be attached at different levels in the tree, but again, this does not seem to happen in practice.

```
(NP (NP (DT a)
        (NNP U.S.)
        (NX (-NONE- *RNR*-1)))
    (CC and)
    (NP (DT a)
        (JJ Soviet)
        (NX (-NONE- *RNR*-1)))
    (NX-1 (JJ naval)
          (NN vessel)))
```

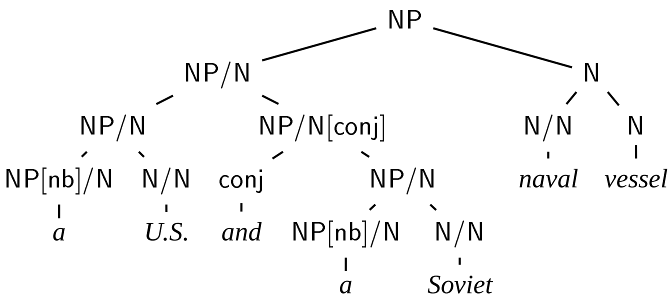
Our algorithm works just as well with this case: First, the category of the traces is determined and percolated up the tree (here shown with noun level inserted):



During category assignment, the NX-1 is ignored. Categories are assigned to all other nodes, resulting in the following tree:



Then we assign NP to the shared constituent NX-1, and assign the corresponding categories to its daughters. Finally, the *RNR* traces are cut out:

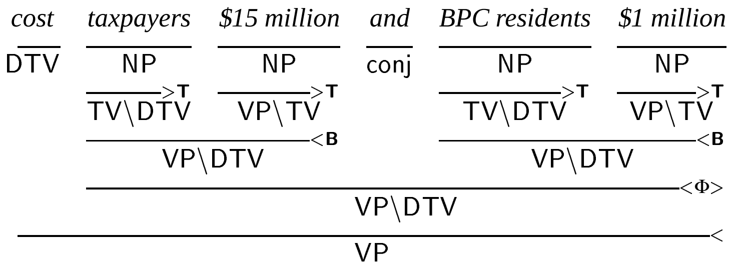


3.10.2 Argument cluster coordination

If two VPs with the same head are conjoined, the second verb can be omitted:

(73) *It could cost taxpayers $15 million and BPC residents $1 million.*

In the CCG account of this construction, *taxpayers $15 million* and *BPC residents $1 million* form constituents (“argument clusters”), which are then coordinated. Argument clusters are obtained by typeraising and composing the constituents of each argument cluster, such that the resulting category is a functor which takes a verb of the right category to its left to yield a verb phrase (cf. Steedman, 2000, chapter 7). Then the argument clusters are conjoined, and combine with the verb via function application:[6]

(74)


The Treebank encodes these constructions like a VP-coordination in which the second VP lacks a verb. Also, the daughters of the second conjunct are co-indexed with the corresponding elements in the first conjunct using the = notation (referred to in the Treebank manual as template gapping):

```
(S (NP-SBJ It)
   (VP (MD could)
       (VP (VP (VB cost)
               (NP-1 taxpayers)
```

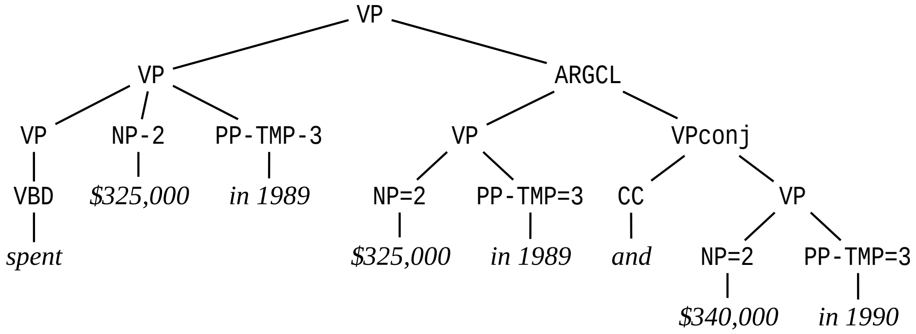
[6] We use the following abbreviations: VP for S\NP, TV for transitive (S\NP)/NP and DTV for ditransitive ((S\NP)/NP/NP)

```
               (NP-2 $ 15 million))
           (CC and)
           (VP (NP=1 BPC residents)
               (NP=2 $ 1 million))))
        (. .))
```

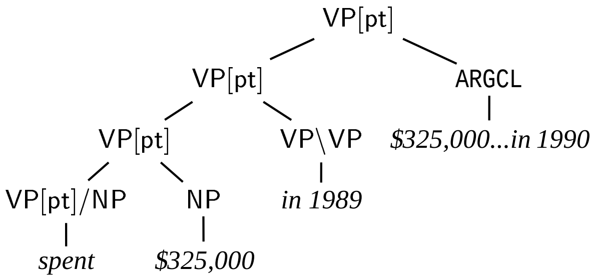
If the second VP contains constituents which do not correspond to constituents in the first VP, a null element (marked *NOT*) with the same label is inserted in the appropriate place in the first VP. This null element is co-indexed with the corresponding constituent in the second VP:

```
(S-ADV (NP-SBJ (-NONE- *-1))
       (VP (VP (VBG increasing)
              (PP-DIR-2 to 2.5 %)
              (PP-TMP-3 in February 1991)
              (ADVP-TMP-4 (-NONE- *NOT*)))
          (, ,)
          (CC and)
          (VP (PP-DIR=2 to 3 %)
              (PP-TMP=3 at six month intervals)
              (ADVP-TMP=4 thereafter))))
```

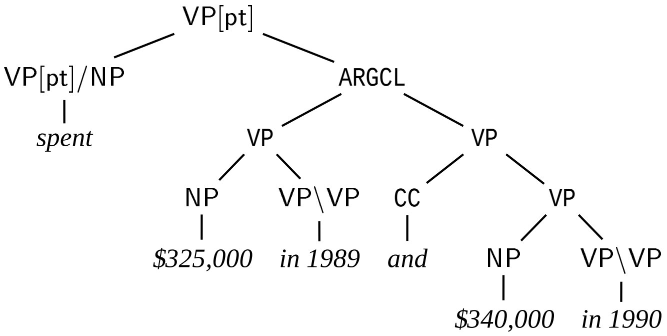
Since the Treebank constituent structure does not correspond to the CCG analysis, we need to transform the tree before we can translate it. We obtain the CCG constituent structure by creating a new node ARGCL consisting of a VP consisting of copies of the co-indexed elements of the first conjunct, the conjunction and the second conjunct (again using VP to abbreviate S\NP):



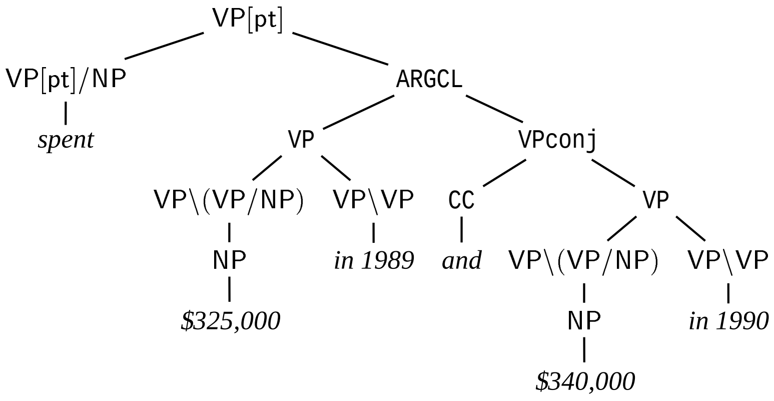
Category assignment now proceeds in two phases: first we assign categories in the normal fashion, ignoring the ARGCL tree:



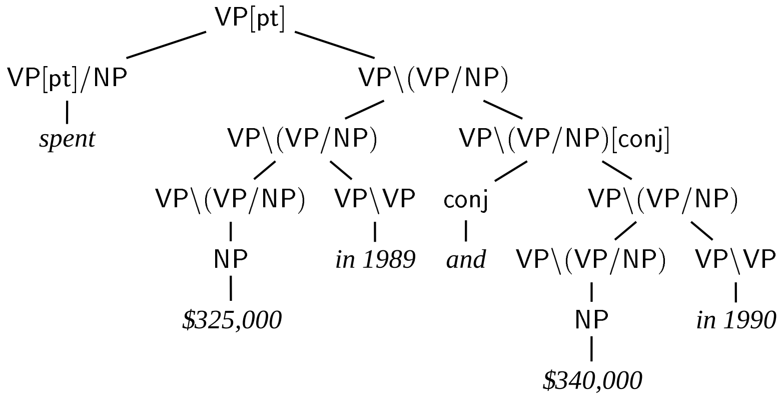
Then the constituents which are co-indexed with the elements in the first tree are assigned the same categories as their antecedents, and all nodes in the first conjunct apart from the verb are cut out:



Then category assignment proceeds underneath the co-indexed nodes (not shown here), as well as above them. Category assignment within an argument cluster is a bottom-up, right-to-left process. The leftmost node (PP-TMP=3) is an adjunct and does not need to be type-raised. However, the object noun phrase is backward-typeraised to VP\(VP/NP) (instantiating the T in the type-raising rule with the category of the parent of the argument cluster). If there was another object to the left of this noun phrase, with category Y, its type-raised category would be (VP/NP)\((VP/NP)/Y), instantiating the T with the argument category of the previous complement.



Then category assignment proceeds, bottom-up:



NOT-null elements, which are inserted in the first conjunct for elements in the second conjunct that do not have a counterpart in the first conjunct, are treated like ordinary constituents and are then later cut out.

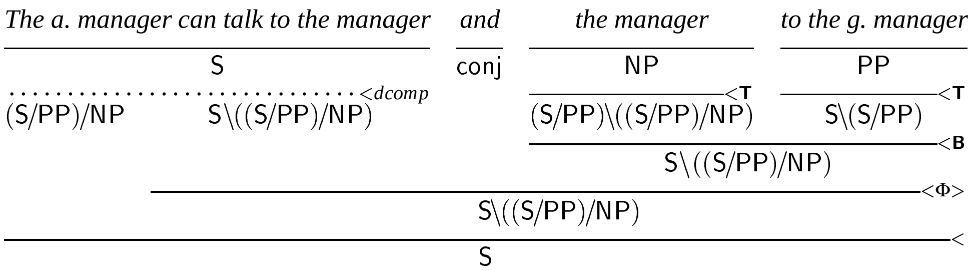
3.10.3 Gapping

A similar annotation to argument cluster coordination is used for sentential gapping:

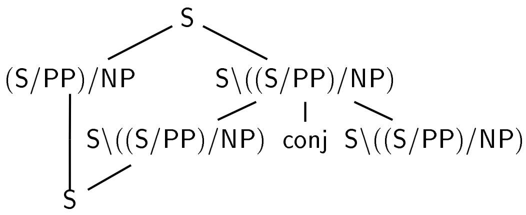
```
(S (S (NP-SBJ-1 Only the assistant manager)
      (VP (MD can)
          (VP (VB talk)
              (PP-CLR-2 (IN to)
                  (NP the manager)))))
   (CC and)
   (S (NP-SBJ=1 the manager)
      (PP-CLR=2 (TO to)
          (NP the general manager))))
```

This construction cannot be handled with the standard combinatory rules of CCG that are assumed for English. Instead, Steedman (2000) proposes an analysis of gapping that uses decomposition, which is a rule that is not based on combinatory logic. Decomposition allows a constituent to be split apart into two subparts, and is used to yield an analysis of gapping that is very similar to that of argument cluster coordination:

(75)



However, this analysis is problematic for our current purposes. Since the derivation is not a tree anymore, and since the decomposed constituents do not correspond to actual constituents in the surface string, this derivation is difficult to represent in a treebank:



3.11 Other null elements in the Treebank

Besides the null elements discussed above, the Treebank contains further kinds of null elements, all of which we ignore when extracting a lexicon from the Treebank.

`*ICH*` ("Insert constituent here") is used for extraposition of modifiers. When there is intervening material between a modifier and the constituent it modifies, and if the intervening material causes a difference in attachment height, a `*ICH*` null element is inserted as adjunct to the modified constituent:

(76) a.
```
(S (NP-SBJ (NP Areas of the factory)
           (SBAR (-NONE- *ICH*-2)))
   (VP (VBD were)
       (ADJP-PRD particularly dusty)
       (SBAR-2 where the crocidolite was used))
       (. .))
```

b.
```
(S (NP-SBJ There)
   (VP (VBZ 's)
       (NP-PRD (NP (DT an)
                   (NN understanding)
                   (SBAR (-NONE- *ICH*-1)))
          (PP on the part of the US)
          (SBAR-1 that Japan has to expand its functions in Asia))))
```

Like in the case of ellipsis, this is a case of a semantic dependency which should not be reflected in the syntactic category. Note that a constituent which is co-indexed with an `*ICH*` null element is not a complement. We therefore treat all constituents which are co-indexed with an `*ICH*` null element as adjuncts. In example (76a), for instance, *were* has category (S[dcl]\NP)/(S[adj]\NP), not ((S[dcl]\NP)/S[emb])/(S[adj]\NP).

The null element `*PPA*` ("Permanent Predictable Ambiguity") is used for genuine attachment ambiguities:

```
(S (NP-SBJ He)
   (ADVP-TMP already)
   (VP (VBZ has)
       (VP (VBN finagled)
           (NP (NP a $ 2 billion loan)
               (PP (-NONE- *PPA*-1)))
           (PP-CLR-1 from the Japanese government))
        (. .))
```

Since the Treebank manual states that the actual constituent should be attached at the more likely attachement site, we chose to ignore any `*PPA*` null element.

Another null element, `*U*`, is used to "mark the interpreted position of a unit symbol" (Marcus *et al.*, 1993). This null element is ignored. See section 3.7.6 for the treatment of monetary expressions.

```
(NP (QP ($ $) (CD 1.5) (CD billion))
    (-NONE- *U*)))
```

3.12 Preprocessing the Treebank

The previous sections presented the general translation procedure. Some necessary preprocessing steps, such as the insertion of a noun level and our reanalysis of small clauses, were mentioned. However, other problems remain. For instance, usually `PP` daughters of `VP` are adjuncts, but the functional tag `-CLR` on a `PP` daughter can be used to indicate that there is a closer relationship between the PP and the verb. For instance, a `-CLR` tag on a PP can indicate that this is the second object of a ditransitive verb. However, it is well known that this tag has not been used in a very consistent manner, and there is an estimated POS tagging error rate of 3% (Ratnaparkhi, 1996). The translation algorithm is sensitive to these errors and inconsistencies: POS tagging errors can lead to incorrect categories or to incorrect features on verbal categories (eg. when a past participle is wrongly tagged as past tense); the omission or addition of functional tags causes errors in the complement-adjunct distinction; and certain types of coordinate constructions are not recognized as such if the bracketing is not correct. Additionally, the algorithm presented in section 3.3 requires the constituent structure of the phrase-structure tree before binarization to conform to the desired CCG analysis. Some systematic changes that are necessary have already been mentioned, but there are other cases that require modification. For instance, if the flat tree of a coordinate construction contains any adjuncts or arguments to the conjuncts, a separate level has to be inserted before binarization can proceed. This is true for constituents which are co-indexed with a right node raising trace (`*RNR*`), but there are also other cases which are either not explicitly marked as right node raising constructions, or where adjuncts to one of the conjuncts appear as sisters rather than daughters of the constituent they modify. Some heuristics were developed to correct POS tag errors that are likely to lead to errors in the translation process. For instance if a simple past tense form occurs in a verb phrase which itself is the daughter of a verb phrase whose head is an inflected verb, it is highly likely that it should be a past participle instead. Using the verb form itself and the surrounding context, we attempt to correct such errors automatically. Originally, quotation marks (“ and ”) also caused a number of problems for the translation algorithm, since they can appear in various positions in the tree and surface string. We therefore decided to eliminate them in the preprocessing step.

3.13 Summary – the complete algorithm

Here is the complete translation algorithm, including the preprocessing step described in section 3.12 and the modifications necessary to deal with traces and argument clusters:

```
foreach tree $\tau$:
    preprocessTree($\tau$);
    preprocessArgumentClusters($\tau$);
    determineConstituentType($\tau$);
    makeBinary($\tau$);
    percolateTraces($\tau$);
    assignCategories($\tau$);
    treatArgumentClusters($\tau$);
    cutTracesAndUnaryRules($\tau$);
```

preprocessTree: Correct tagging errors, ensure the constituent structure conforms to the CCG analysis for noun phrases, coordinate constructions and small clauses. Eliminate quotes and preprocess monetary units.

preprocessArgumentClusters: Create the constituent structure that corresponds to an argument cluster in argument cluster coordination. This argument cluster contains a copy of the first conjunct. Within this copy, constituents are co-indexed with the original constituents in the first conjunct.

determineConstituentType: For each node, determine its constituent type. We distinguish the following constituent types: heads, complements, adjuncts, conjunctions, constituents that are co-indexed with a `*RNR*`-trace, spurious traces, and argument clusters.

makeBinary: Binarize the tree.

percolateTraces: Determine the category of `*T*` and `*RNR*` traces in complement position, and percolate them up to the appropriate level in the tree.

assignCategories: Assign categories to nodes in the tree, starting at the root node. Nodes that are co-indexed with `*RNR*` traces are ignored at first, and then receive the category of the corresponding traces. Argument clusters are also ignored in this step.

treatArgumentClusters: Determine the category of constituents within an argument cluster that are co-indexed with elements in a first conjunct. Assign categories to the nodes within these constituent in the ordinary top-down fashion. Use type-raising and composition to assign categories to the intermediate nodes within the argument cluster.

cutTracesAndUnaryRules: At this point, the tree contains certain constituents that are not part of the CCG derivation, such as traces and the copy of the first conjunct within an argument cluster. These are all cut out. Resulting unary projections of the form $\mathsf{X} \Rightarrow \mathsf{X}$ are eliminated.

There are certain types of trees that are not processed by this algorithm: we do not deal with gapping, unlike coordinate phrases (`UCP`) where the two conjuncts cannot be identified, or trees rooted in `X`. This amounts to 306, or 0.76%, of the 39,832 sentences in sections 02-21 of the Wall Street Journal subcorpus of the Treebank.

3.14 Size and coverage of the acquired lexicon and grammar

A lexicon acquired from sections 02-21 of the WSJ has 72,143 entries for 42,300 word types. There are 1224 lexical category types. 417 of these appear only once. Each word type has on average 1.7 entries. However, if we were to parse sections 02-21 with this lexicon, each token would on average have 19.2 categories, since a lot of high-frequency words such as certain closed class items have a high number of categories.

The grammar for this corpus has 3241 rule instantiations. Out of these rules, 1220 occur only once.

Token frequency f	category types
$f = 1$	417
$1 < f < 10$	398
$10 \leq f < 100$	243
$100 \leq f < 1,000$	103
$1,000 \leq f < 10,000$	48
$10,000 \leq f$	15

Table 3.1: Frequency distribution of lexical category types in WSJ02-21

Number of entries n	Words
$n = 1$	29,642
$1 \leq n < 10$	12,192
$10 \leq n < 20$	362
$20 \leq n < 30$	53
$30 \leq n < 40$	21
$40 \leq n$	30

Table 3.2: Distribution of number of entries per word (with features)

A lexicon extracted from section 00 has 11,060 entries for 7707 word types. On average a word has 1.4 categories. There are 377 category types. Figure 3.3 shows the coverage of the lexicon extracted from sections 02-21 on section 00. 77.8% of the unknown words are either nouns (N) or prenominal modifiers (N/N).

Figure 3.2 examines the growth of the set of lexical category types as sections 02-21 are

Entries also in reference lexicon:	94.1%
Entries not in reference lexicon:	5.9%
- Known words, known categories:	2.1%
- Known words, unknown categories:	0.03%
- Unknown words, known categories:	3.75%
Out of these: N or N/N	77.8%
- Unknown words, unknown categories:	-

Table 3.3: Coverage of a CCG lexicon from sec. 02-21 (= the reference lexicon) on sec. 00.

processed. The topmost curve indicates that new category types are continually added to the lexicon. Even after 40,000 sentences, there is no convergence yet. However, if only category types that occur more than 2 or 3 times in the entire corpus are considered, the set of lexical category types seems to have reached convergence. This, together with an inspection of the category types which occur only once, leads us to believe that the set of category types in our lexicon is indeed complete, and that category types which occur with low frequency ought to be considered noise.

3.15 Related work

3.15.1 An alternative algorithm

Watkinson and Manandhar (2001) present an alternative algorithm for the extraction of AB categorial lexicons from the Penn Treebank. However, they do not present a way of dealing with the various null elements in the Treebank, which means that they can only process a small subset of the sentences in the corpus.[7] Furthermore, unlike ours, their algorithm does not correspond to a reverse derivation, and therefore it is unclear how the correctness of their translation can be guaranteed unless categories assigned in the initial step can later be modified.

In particular, without such a correction, it would be possible for their method to assign lexical categories to a sentence which cannot be combined to derive a sentential category. Their algorithm proceeds in four stages:

[7] A search with tgrep showed that out of the 49,298 sentences in the Treebank, 34,318 contain a null element matching the regular expression /*/

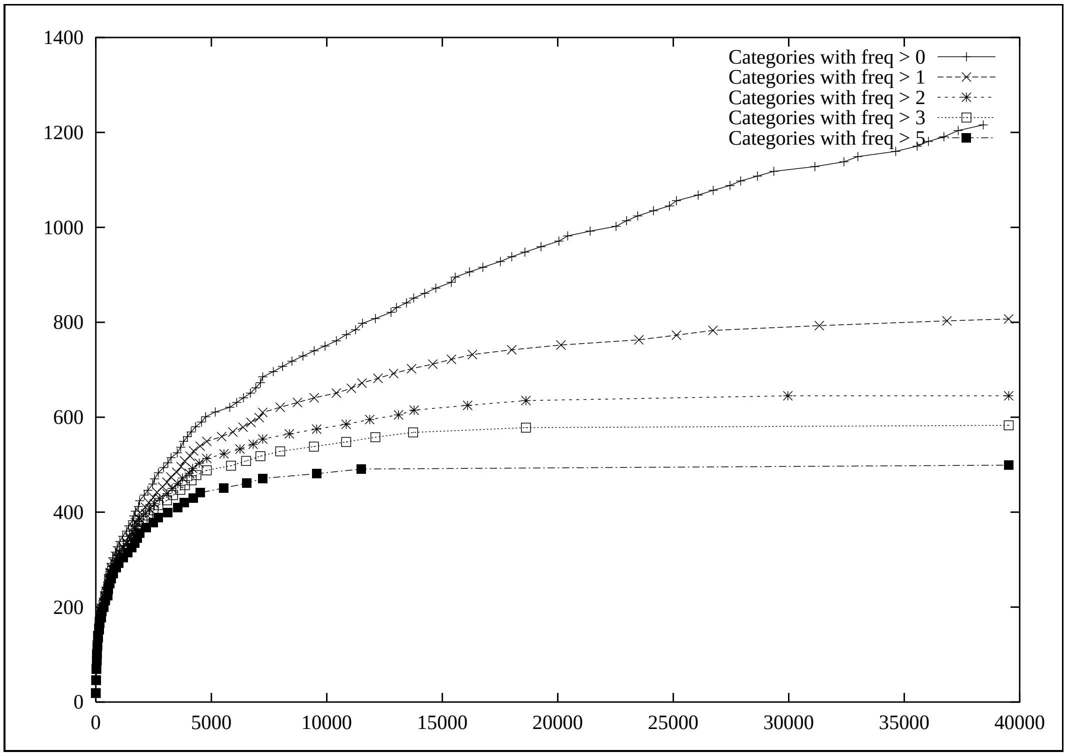


Figure 3.2: Growth of lexical category types (sections 02-21). The x-axis indicates the number of sentences. The y-axis gives the number of lexical category types. The topmost curve contains all category types; the other curves only count lexical category types that occur at least *n* times in sections 02-21.

1. Map some POS tags to categories.

2. Look at the surrounding subtree to map other POS tags to categories

3. Annotate subtrees with head, complement and adjunct information using heuristics similar to Collins (1999).

4. Assign categories to the remaining words in a bottom-up fashion.

In the first step, Watkinson and Manandhar map some part-of-speech tags deterministically to CG categories. The example they give is `DT` → $\mathsf{NP/N}$. However, this analysis is only correct for determiners appearing in noun phrases in complement position. For instance, it is not the correct analysis for determiners in temporal NPs.[8] Consider the `NP-TMP` in the following example:

(77)
```
(S (NP-SBJ South Korea)
   (VP (VBZ has)
       (VP (VBN recorded)
           (NP a trade surplus)
           (NP-TMP (DT this)
                   (NN year))))
    (. .)))
```

Here is the derivation of the embedded verb phrase:

(78)
$$\begin{array}{ccccc}
\textit{recorded} & \textit{a trade surplus} & \textit{this} & \textit{year} & \\
\hline
\mathsf{(S[pt]\backslash NP)/NP} & \mathsf{NP} & \mathsf{((S\backslash NP)\backslash(S\backslash NP))/N} & \mathsf{N} & \\
\cline{1-2}\cline{3-4}
\multicolumn{2}{c}{\mathsf{S[pt]\backslash NP}} & \multicolumn{2}{c}{\mathsf{(S\backslash NP)\backslash(S\backslash NP)}} & > \\
\hline
\multicolumn{4}{c}{\mathsf{S[pt]\backslash NP}} & <
\end{array}$$

In step 3, they use very similar heuristics to ours (both are based on Collins (1999)) to identify heads, adjuncts and complements. Thus, the `NP-TMP` would be identified as an adjunct, and either the analysis given in the first step would have to be modified, or the categories cannot combine:

(79)
$$\begin{array}{ccccc}
\textit{recorded} & \textit{a trade surplus} & \textit{this} & \textit{year} & \\
\hline
\mathsf{(S[pt]\backslash NP)/NP} & \mathsf{NP} & \mathsf{NP/N} & \mathsf{N} & \\
\cline{1-2}\cline{3-4}
\multicolumn{2}{c}{\mathsf{S[pt]\backslash NP}} & \multicolumn{2}{c}{\mathsf{NP}} & >
\end{array}$$

[8] It is also not the correct analysis for `NP`s which only consist of one `DT` daughter, such as the following: `(NP (DT those))`, `(NP (DT some))`, `(NP (DT all))`

Assuming that such cases can be detected and are corrected, it is not clear that their bottom-up translation procedure yields different results from our top-down method if the same heuristics are used to identify heads and distinguish between complements and adjuncts. In step 4, they assign variables to the lexical categories of words which have not been assigned categories yet, then traverse the whole tree bottom-up and instantiate these categories using the head/complement/adjunct information already available to instantiate these variables. However, some of this information will only be available at the top of the (sub)tree, and will thus presumably be percolated down the tree through the variables. In such cases, the resulting categories should be the same.

As Watkinson and Manandhar use AB categorial grammar, which only has function application, it is also not clear how they could extend their algorithm to deal with the `*T*` and `*RNR*` traces in the Treebank. Furthermore, they could not strip off the outer arguments of the head category when determining the categories of adjuncts, because AB categorial grammar does not allow composition. We would expect this to lead to a larger, less compact lexicon.

3.15.2 Related work using other grammar formalisms

The algorithm presented here is similar to algorithms which extract Lexicalized Tree-Adjoining Grammars from the Penn Treebank (Xia *et al.*, 2000; Xia, 1999; Chen and Vijay-Shanker, 2000; Chiang, 2000). All of these algorithms rely crucially on head-finding procedures and heuristics to distinguish complements from adjuncts; therefore different implementations of the same algorithm can yield very different lexicons (see Chen and Vijay-Shanker (2000) for the impact of different complement/adjunct heuristics). Cahill *et al.* (2002) present an algorithm to annotate the Penn Treebank with Lexical Functional Grammar F-structures.

3.16 Conclusion

This chapter presented an algorithm which translates the Penn Treebank into a corpus of CCG normal-form derivations, and outlined the kinds of changes that need to be performed on the trees before this algorithm can applied in order to obtain correct CCG analyses. Since every CCG derivation corresponds to the construction of a semantic interpretation, the work presented in this chapter can also be seen as a first step towards the creation of a treebank annotated with semantic interpretations.

As explained in this chapter, a number of obstacles for the translation of Penn Treebank trees to a linguistically richer formalism such as CCG remain. The flat noun phrase structure is one of them: although it is possible to introduce a separate noun level, compound nouns still have a flat internal structure, which is semantically undesirable, and leads us to assume a strictly right-branching (but often incorrect) analysis. The Treebank markup also makes appositives difficult to distinguish from noun phrase lists. Moreover, postnominal modifiers are always attached at the NP-level which is also undesirable from a semantic point of view.

Within the verb phrase and sentential structure, there are other problems. For instance, the acquired CCG lexicon does not have a proper analysis of phrasal verbs. The distinction between complements and adjuncts is similarly difficult to draw, especially for constituents annotated with the CLR-tag, which is known to be used fairly inconsistently across the corpus. Kinyon and Prolo (2002) describe a different set of heuristics for distinguishing complements from adjuncts than those used here, and it would be interesting to compare the two approaches. In a future version of CCGbank, it might be possible to make use of the complement-adjunct distinction that is implicit in the semantic role annotation of the Penn Treebank that is currently being developed within the Proposition Bank project (Kingsbury *et al.* (2002); Kingsbury and Palmer (2002)). There are certain types of constituents, such as fragments, where it is not even obvious what the correct linguistic analysis should be. Multi-word expressions are equally problematic, and further work is required. Sentential gapping, which in CCG is analyzed with decomposition, is also problem for the algorithm presented here; it is not clear how decomposition could be represented in a treebank.

Despite these shortcomings, the grammar that underlies CCGbank covers a wide range of syntactic phenomena of English. An analysis of the growth of lexical category types indicates that there is still some noise in the lexicon; however, if we only consider categories that occur more than once or twice, the set of lexical category types used in CCGbank seems to have converged.

Chapter 4

A brief introduction to statistical parsing

This chapter provides a brief introduction to statistical parsing. It reviews the elementary concepts of probability theory which form the mathematical basis for the models used in statistical parsing, and gives the definition of basic probability models for context-free grammars, since a large number of state-of-the-art models build on and extend this approach. Current approaches to parsing the Penn Treebank which are of particular relevance to the research reported in the following chapters are reviewed and discussed.

4.1 Why *statistical* parsing?

A parser is a program which assigns syntactic analyses (parses) to strings. Syntactic analysis is a necessary prerequisite for semantic interpretation; therefore parsing is an essential part of any system that requires natural language understanding. However, depending on the nature of the underlying grammar, a large proportion of naturally occurring sentences are highly ambiguous. This is a problem, as in most cases different syntactic analyses correspond to different semantic readings. Usually we are interested in finding one, the most likely or plausible, analysis for an input string, which the parser needs to select from among a potentially very large set of proposed analyses. Probability theory allows us to formalize the notion of likelihood. Therefore, statistical parsing is concerned with the definition and estimation of probability models over grammatical analyses, and with the search for the most probable analysis (under a particular model) for a given sentence.

Slightly more formally, given a grammar G that defines possible analyses (parses) τ for strings α, our aim is to define a function ϕ which assigns numerical scores to these parses, such that competing analyses can be ranked. Parse selection then reduces to selecting the analysis τ^* among the set of parses for α, $parses(\alpha)$, which has the highest score under ϕ :

$$\tau^* = \arg \max_{\tau \in parses(\alpha)} \phi(\tau)$$

In probabilistic models of parse selection, ϕ is a probability, and the task is to select the parse that has the highest conditional probability given that the string is α:

$$\tau^* = \arg \max_{\tau} P(\tau|\alpha)$$

In order to use a probabilistic model of parse selection, one has to complete two tasks: first one has to define how the probability of a parse is computed. Trees are complex, recursive objects, and grammars used for natural language define an infinite number of kinds of trees and strings, so $P(\tau|\alpha)$ can usually not be computed directly. The structure of the probability model determines how the probability of a parse is computed. Given the general structure of a model, one has to instantiate the probability distributions. This is the task of parameter estimation or training. The difficulty in parameter estimation is that the amount of available labelled training material is limited, and very often not sufficient to obtain reliable estimates for all parameters of the model.

Finally, the goodness of a particular model can be evaluated. Following standard methodology in machine learning, this is usually done by splitting the available annotated data in three parts, so that one part (the training set) can be used one part to estimate the parameters of the model, another part (the development test set) to tune certain aspects of the system, and a third part (the final test set), unseen during training and development, to compare the model's predictions against the manual annotations. Distinguishing the final test set from the development set helps to prevent adjusting the model too much to the idiosyncrasies of one particular set of data. There are a number of measures to compare the output of parsers, either based on the bracketings of parse trees, or on the recovery of dependencies between the words in the sentence.

This chapter gives first a brief introduction to the relevant concepts of probability theory and machine learning, as well as to probabilistic context-free grammars, a very simple class of probability models for context-free languages. Since all empirical approaches to statistical parsing require data to estimate the parameters of their models, most research on parsing for English has focussed on using the Penn Treebank (Marcus *et al.*, 1993, 1994), and section 4.5

reviews and compares some of the more recent approaches to this task. Our own research on the development of probability models for statistical parsing with CCG builds largely on the findings of this body of previous work, and I will emphasize those points that are of particular relevance to the work reported in the following chapters.

4.2 A brief review of probability theory

In order to understand the models used in statistical parsing, an understanding of the basic concepts of probability theory is necessary. Probability theory formalizes the notion of (un)certainty or likelihood of events. Traditionally, a process whose outcome is not clear is referred to as an experiment. Throwing a die is an experiment; generating a parse tree is also an experiment. We assume that there is a set Ω of possible elementary outcomes of the experiment. Ω is also called the sample space. For the die experiment, $\Omega = \{1,2,...,6\}$. For the parse tree experiment, Ω is the (possibly infinite) set of all trees generated by the grammar. An event E is a subset of Ω. A function $P(E)$ over subsets of Ω is a probability distribution if it satisfies the Kolmogorov axioms:

(80) **Kolmogorov axioms**:

a. $\forall E \subseteq \Omega : 0 \leq P(E) \leq 1.$

b. $P(\Omega) = 1.$

c. For events $E_1,...,E_n$, such that $E_i \cap E_j = \emptyset$:

$$P(\bigcup_i E_i) = \sum_i P(E_i)$$

Since events are subsets of Ω, the following relations hold:

(81) a. $P(E_1 \text{ or } E_2) = P(E_1 \cup E_2)$

b. $P(E_1 \text{ and } E_2) = P(E_1 \cap E_2)$

c. $P(\text{not } E) = P(\overline{E}) = 1 - P(E)$

In particular, the joint probability of two events, $P(E_1 \text{ and } E_2)$ or $P(E_1E_2)$, is defined as the probability of the intersection of $E1$ and E_2. From this follows the definition of conditional probability of one event (E_1) given another (E_2), $P(E_1|E_2)$:

(82) **Conditional Probability of E_1 given E_2**:

$$P(E_1|E_2) := \frac{P(E_1E_2)}{P(E_2)}$$

The task of parse selection can be phrased in terms of conditional probability: Recall that we are given a sentence α, and want to find the most likely parse τ^* for α. This amounts to searching for the parse τ^* that has the highest conditional probability given α:

$$\tau^* = \arg\max_{\tau} P(\tau|\alpha)$$

Models which estimate $P(\tau|\alpha)$ directly are called conditional models. In practice, such models are difficult to estimate, since they need to be defined for each sentence α in the language. Examples of conditional models used in statistical parsing are Collins (1996) and Riezler *et al.* (2002) (both discussed in this chapter). Chapter 5 discusses a conditional model for CCG proposed by Clark *et al.* (2002).

There is an alternative way of defining models of $P(\tau|\alpha)$. From the definition of conditional probability, $P(\tau|\alpha) = P(\tau\alpha)/P(\alpha)$. In parsing, α, the string, is already given. Also, if τ is a parse tree with yield α, $P(\tau,\alpha) = P(\tau)$. Hence, it suffices to define a probability model over parse trees τ to find the most likely parse for a sentence α. Among the parses of α, the most likely parse is the parse which receives highest probability under $P(\tau)$. Generative models estimate $P(\tau)$. The underlying assumption of generative models is that a complex object such as a parse tree is generated by a stochastic process, so that the probability of the entire object can be expressed in terms of the individual steps of this process. This is possible since the definition of conditional probability, $P(E_1E_2) = P(E_1|E_2)P(E_2)$, can be generalized to the following chain rule:

$$P(E_1....E_n) = P(E_1)P(E_2|E_1)P(E_3|E_2E_1)...P(E_n|E_1...E_{n-1})$$

Here, each subevent E_i corresponds to (the outcome of) one step in a process which generates $E_1...E_n$. For instance, a parse tree τ is composed of a number of local trees (trees of depth 1) $\tau_1...\tau_n$, and therefore its probability can be expressed as $P(\tau) = P(\tau_1...\tau_n)$, which in turn can be decomposed using the chain rule:

$$P(\tau) = P(\tau_1...\tau_n) = P(\tau_1)P(\tau_2|\tau_1)P(\tau_3|\tau_2\tau_1)...P(\tau_n|\tau_1...\tau_{n-1})$$

This is essentially the decomposition used in probabilistic context-free grammars (described below). Note that τ could have been decomposed in many different ways – using local trees is just one of many possibilities.

Another important concept is independence. Two events E_1, E_2, are independent of each other if their joint probability is the same as the product of the probability of the individual events. Given the definition of conditional probability, this is equivalent to saying that the conditional probability of E_1 given E_2 is the same as the (unconditional) probability of E_1.

(83) Two events E_1 and E_2 are **independent** if:
$P(E_1E_2) = P(E_1)P(E_2)$
or, equivalently:
$P(E_1|E_2) = P(E_1)$

Related is the notion of conditional independence. Formally, it is defined as follows:

(84) Events E_1 and E_2 are **conditionally independent** given event E_3 if:
$P(E_1|E_2,E_3) = P(E_1|E_3)$

When defining probability models over complex structures, it is often very difficult to estimate parameters for conditional probabilities with a large number of conditioning variables. Hence, it is convenient to make independence assumptions. That is, by assuming that events E_1 and E_2 are conditionally independent given E_3, we allow ourselves to replace $P(E_1|E_2,E_3)$ with $P(E_1|E_3)$, which might be easier to estimate. However, this is a modelling assumption which might or might not be true of the observed data.

The next sections present a number of generative probability models over parse trees, starting with the most basic model, probabilistic context-free grammars (PCFGs). Section 4.5 reviews the models underlying some state-of-the-art parsers for English. In section 4.6 we consider the performance of a particular statistical Tree-Adjoining Grammar parser on the standard Penn Treebank parsing task. Section 4.7 discusses the log-linear (conditional) models used for Lexical Functional Grammar and their performance on the Penn Treebank.

4.3 Probabilistic context-free grammars (PCFGs)

Probabilistic context-free grammars (Booth and Thompson, 1973) are a particularly simple kind of generative probability model for context-free languages. This section gives the definition of PCFGs. In section 4.3.2, we look at certain conditions that have to hold for a PCFG to constitute a probability model over the language defined by the grammar, and in section 4.5.2 the practical performance of PCFGs is investigated.

4.3.1 Definition

Recall the formal definition of a language L as a set of strings s defined over symbols of an alphabet Σ. A context-free grammar G for a language L is a tuple $\langle N,T,R,S\rangle$, consisting of a set of nonterminals N, a set of terminals T, a set of rules R of the form $X \rightarrow \alpha$, where $X \in N$ (a nonterminal symbol) and $\alpha \in (N \cup T)^*$ (a string over nonterminals and terminals). $S \in N$ is a designated start symbol. When applied to natural language, a further distinction is frequently drawn, and a set of preterminals (such as part-of-speech tags or other lexical types) is defined. In this case, the preterminals are the terminal symbols of the grammar, and a lexicon is required which maps the preterminals to the set of words in the language.

A probabilistic context-free grammar (PCFG) is a probability model over the language defined by a context-free grammar. The main insight behind PCFGs is that we can use the chain rule of joint probabilities to define the probability of a tree τ generated by G, $P(\tau)$, as the product of the probabilities to the individual steps of the derivation of τ. Recall that in each derivation step, exactly one nonterminal symbol $X \in N$ is rewritten by a rule $X \rightarrow \alpha$. Assuming that the probability of X rewriting to α is independent of the context surrounding X, the probability of a derivation step is simply the probability that rule $X \rightarrow \alpha$ is applied given a nonterminal symbol X. Therefore, PCFGs associate with each nonterminal symbol $X \in N$ a conditional probability distribution P defined over the rules of G in which X is the left-hand side symbol. These probability distributions fulfill the following conditions:

$$\forall X \rightarrow \alpha \in R : 0 \leq P(X \rightarrow \alpha | X) \leq 1$$
$$\forall X \in N : \sum_{X \rightarrow \alpha \in R} P(X \rightarrow \alpha | X) = 1$$

A PCFG $\langle N,T,R,P,S\rangle$ is then a grammar G together with a family of distributions P defined over the rules of G. The probability of a derivation in G is the product of all the rule applications in this derivation. The probability of a string α in L is the sum of the probabilities of all its derivations.

4.3.2 Tightness of PCFGs

Not every possible distribution $P(X \rightarrow \alpha | X)$ over the rules of a context-free grammar yields a probability model of the language defined by the grammar. A PCFG only defines a probability model over the strings of L if the probability mass of all strings s sums to one:

$$\sum_{s \in L} P(s) = 1$$

A probability model over a language is said to be tight (or consistent) if and only if it meets this condition. An inconsistent probability model, ie. one in which the probabilities of all sentences of the language sum to less than 1, is deficient. In such a model, some of the probability mass is assigned to strings which cannot be generated, or "lost" to infinite derivations. The following PCFG (from Manning and Schütze (1999)) over the language $L = \mathtt{rhubarb}^{+}$ is not tight:

$$\begin{array}{lcl} \mathtt{S} \rightarrow & \mathtt{S\ S} & (\mathtt{p} = 2/3) \\ \mathtt{S} \rightarrow & \mathtt{rhubarb} & (\mathtt{p} = 1/3) \end{array}$$

Booth and Thompson (1973) relate probabilistic context-free grammars to the theory of Galton-Watson branching processes (Harris, 1963), which allows them to state conditions which guarantee the consistency of a given PCFG. The theory of Galton-Watson branching processes is a model for recursive processes. A process is a Galton-Watson branching process if the following conditions hold:

1. There is a finite set of kinds of objects.
2. The initial set of objects in the 0th generation, Z_0, is given.
3. Each generation Z_i produces with some probability a following generation Z_{i+1}.
4. This probability is time-invariant (not dependent on i).
5. What children are born to one object does not depend on what happens to the object's sisters.

It is easy to see that these conditions hold for PCFGs: the set of kinds of objects $(N \cup T)$ is finite, and the initial set of objects is given as the singleton containing the start symbol S. The next generation Z_{i+1} is created from an existing Z_i by expanding all nonterminals in Z_i with one rule. Since the probabilities are defined in terms of the grammar rules, they are also time-invariant. The context-freeness guarantees furthermore that the evolution of one object does not depend on the evolution of one of its siblings.

Booth and Thompson (1973) derive the conditions which have to hold for a PCFG to be tight. Recall that a PCFG is tight if the probabilities of all terminal strings s in L sum to 1. Each terminal string is derived in a finite number of steps (= generations). Therefore, if the probability mass of all strings generated after n generations converges to 1 as n tends to infinity, the PCFG is tight. Booth and Thompson show that it is sufficient to define a stochastic expectation matrix E of size $|N|$ whose elements e_{ij} represent the expected number of occurrences of the nonterminal A_j in the set of productions with left-hand-side symbol A_i. If $lim_{i\to\infty}E^i = 0$, then no probability mass is lost to infinite derivations. E^i converges to zero if none of the eigenvalues of E are greater than 1. Give such an expectation matrix E, Sarkar (1998) gives an efficient algorithm to check the consistency of the model. He cites a theorem by Greršgorin, according to which the greatest eigenvalue of a square matrix is smaller than one if there is an $n \geq 1$ such that the sum of absolute values of each row of E^n is less than 1. If this is the case, it is also true for any $n' > n$ – therefore, it is sufficient to check E, E^2,E^4, etc., until the condition is met.

4.3.3 Parameter estimation

Given a context-free grammar $G = \langle N, T, R, S\rangle$, how do we determine the rule probabilities (ie. estimate the parameters) of the PCFG corresponding to G? Assuming that we have a sample C (= a corpus) of sentences belonging to the language defined by G, there are two methods, depending on whether the sentences in our sample are annotated with their derivation or not. If the corpus C contains the derivations of the sentences – that is, if it is a treebank, the probability of a rule $X \to \alpha$ can be obtained as the number of times X expands to α in the corpus (we will denote the frequency of a particular rule in C as $f(X \to \alpha)$), divided by the number of times X expands by any rule in R with X on its left-hand-side:

$$\hat{P}(X \longrightarrow \alpha) = \frac{f(X \to \alpha)}{\sum_{\alpha' \text{s.t.} X \longrightarrow \alpha' \in R} f(X \to \alpha')}$$

This is called the relative frequency estimator for PCFGs. It is well known that, within the space of possible PCFGs that can be defined over G, this estimator yields a model which assigns the highest probability to the sentences s observed in C. Therefore, the relative frequency estimate $\hat{P}$ is also called the maximum likelihood estimate (MLE) of a PCFG over G. Maximum likelihood estimates are usually considered optimal, for the following reasoning: If we are interested in finding the best model $\tilde{P}$ to account for some given data, a reasonable assumption is that the best among the class of models p under consideration is the model which is the most likely given the data. Lacking any prior knowledge about the likelihood of different ps, we can furthermore assume that all instantiations of p are equally likely. Under these assumptions, the MLE yields the most likely, and therefore the best, model out of the models in the class under consideration (see e.g. Mitchell (1997)).

If C contains only raw sentences (that is, only the yields, but not the parse trees themselves), the rule frequency cannot be determined directly, since, in general, sentences in C can have more than one derivation in G. Instead, this needs to be replaced by the expected rule frequency $E_P(f(X \rightarrow \alpha))$. In this case, the maximum likelihood estimate cannot be determined directly, and an iterative hill-climbing scheme, the Inside-Outside algorithm (Baker, 1979), can be used. However, this algorithm is not guaranteed to converge to the maximum likelihood solution, and in practice, at least a partial description of syntactic structure is required to obtain models which produce linguistically plausible analyses (Pereira and Schabes, 1992; Hwa, 1999).

Chi and Geman (1998) show that a PCFG estimated from labelled or unlabelled data using either relative frequency estimation or the Inside-Outside algorithm is always consistent.

4.4 Dealing with sparse data

PCFGs are a very simple class of models; however, information about the non-local structure of the parse tree or the words of the sentence is usually required to determine the correct analysis for a given sentence. Therefore, state-of-the-art parsers use more complicated models with additional features capturing such structural or lexical information. The more features a model contains, the more parameters need to be estimated. At the same time, a model which distinguishes many features will induce a more fine-grained partitioning of the training data. For instance, in an unlexicalized PCFG, the rule probabilities are estimated from all occurrences of the left-hand-side nonterminal. In a lexicalized model, we might want to condition the expansion of a nonterminal also on the head word of the constituent, thus enabling the model to capture, for example, subcategorization preferences of particular verbs. This means that rule probabilities can now only be estimated from co-occurrences of non-terminals and individual words. Therefore the counts on which the relative frequency estimate is based will be much lower (and zero in many cases). In a sample drawn from a particular distribution, the relative frequency estimate is only guaranteed to converge to the true probability in the limit, but if the sample is very small, the estimate can be very inaccurate.

In machine learning, this problem is often referred to as the tradeoff between bias and variance (Geman *et al.*, 1992). Bias is a measure of how well the particular model class in question is able to capture the desired target function. Geman *et al.* consider the problem of regression, where the task is to construct a (numerical) function $f(\mathbf{x};D) = y$ from a training set $\mathbf{D} = \{(\mathbf{x}_1, y_1), ...(\mathbf{x}_n, y_n)\}$. They define bias as the squared difference between the expectation (over all possible training sets $\mathbf{D}$) of the model function $f(\mathbf{x};D)$ and the expectation (also over all $\mathbf{D}$) of y given $\mathbf{X}$. Since bias is defined as the expected error of all possible models from a particular class, it is therefore a property of model classes, not of particular instantiations of that model class. Section 4.5.2 presents results by Johnson (1998), who investigates the impact of the linguistic representation embodied in a particular grammar on the bias of PCFGs estimated from this grammar. The variance of $f(\mathbf{x};D)$ (the expectation of the squared difference between $f(\mathbf{x};D)$ and the mean of $f(\mathbf{x};D)$) measures how sensitive the model class is to the data. In general, complex models have higher variance, since for such models there will be a large number of unseen events or events with very low frequency in the training data, which results in less accurate estimates. Different training sets contain different low frequency events, resulting in different estimates. Given that we are concerned with defining probability models of natural language, the sparse data problem is particularly acute, if only for the simple reason that not all words (let alone word-word combinations) of the language will ever occur in the training data.

Geman *et al.* argue that the purposeful introduction of bias can significantly reduce variance if the bias is specifically tailored to the particular target function that is to be learned. In particular, bias is "harmless" if the target function is still included in the space of possible models. Since bias is the expected error of all possible models, When defining a probability model over complex structures, one needs to decide on how to represent these structures, and (at least in the kinds of models explored in this dissertation) make explicit independence assumptions between various aspects of this representation. For instance, in defining a lexicalized PCFG, we need to determine for each rule in the grammar which of the nonterminal symbols on the right hand side is the head; and we would assume that a linguistically informed choice would yield a higher performance than a random, or linguistically counter-intuitive, choice. When designing a probability model, one also has to decide how to assign probabilities to unseen events and low-frequency events. There are a number of standard smoothing techniques, which adjust probability estimates to assign probability to unseen events. These techniques vary in their effectiveness (which can be measured by how well the smoothed models can account for unseen data). A standard method for smoothing distributions with many conditioning variables is to interpolate one estimate (usually the relative frequency estimate $\hat{P}$) with an estimate of a distribution with fewer conditioning variables:

$$\tilde{P}(y|x_1...x_i...x_n) = \lambda\hat{P}(y|x_1...x_n) + (1-\lambda)\tilde{P}(y|x_1...x_i) \quad (0 \leq \lambda \leq 1)$$

As the resulting distribution is a linear combination of $\hat{P}(y|x_1...x_n)$ and $\tilde{P}(y|x_1...x_i)$, this method is often referred to as linear interpolation. Note that one has to make at least three decisions here: what independence assumptions to make in the second distribution (ie. which of the $x_1...x_n$ to drop), how to compute the smoothing weight λ, and how to estimate the second distribution (for instance, whether a relative frequency estimate is good enough, or whether further smoothing is required). There are a number of general smoothing techniques to obtain the smoothing weights λ (for a comprehensive introduction to various smoothing techniques see Jelinek (1997)). Many of the best performing techniques require additional held-out data to optimize the values of the λ parameters.

Chen and Goodman (1996) examine the performance of different smoothing techniques for language modelling (the task of predicting the next word in a string given a sequence of previous words). They investigate *n*-gram models, which assume that the probability of a word depends only on the $n-1$ previous words. Unfortunately, there has not yet been a comparable study for the grammar-based models used in statistical parsing, so it is not clear whether their results apply to our domain. In our own parser, we have used the smoothing technique described by Collins (1999) (discussed below), which is based on linear interpolation. Since it does not require any held-out data, it is particularly easy to implement; however, more sophisticated alternatives might work better in practice.

4.5 Parsing the Penn Treebank

This section presents a review of some recent statistical parsers which are trained and tested on the Penn Treebank, as well as investigations of the performance of PCFGs on the standard Penn Treebank task. Wherever findings are made that are relevant to our own research, this will be pointed out.

Since the Penn Treebank is the largest manually annotated corpus of English, it has become the standard training and test corpus for statistical parsers of English in recent years. It has become customary to use sections 02 to 21 as training data, section 00 as development data, and section 23 as unseen test data. The experiments reported in this dissertation follow this practice.

Penn Treebank parsers are usually evaluated according to the so-called Parseval measures (Black *et al.*, 1991), a set of constituent-based evaluation metrics for parse trees. Since these are often the only evaluation measures reported for the parsers considered here, possible alternatives to Parseval are ignored in this section. Alternative measures consider the correctness of dependencies or grammatical relations between words (eg. Lin (1998), Carroll *et al.* (1999)). These are often considered superior to Parseval, since they do not depend as much on the particular representation of the underlying grammar. In chapter 5, we will return to dependency-based evaluation metrics when reporting the results of our own models, and use them to compare our parsers against others.

One of the assumptions underlying the description of natural language with a formal grammar is that the syntactic categories of that language correspond to the (nonterminal and terminal) symbols defined by that grammar, and, if the grammar is context-free, that constituents belonging to the same category can expand in the same manner regardless of the surrounding context. PCFGs make the even stronger assumption that the probability of constituents of the same category expanding in the same manner is the same regardless of context. However, inspection of the Penn Treebank reveals that the probability of a nonterminal expanding by a particular rule is not independent of the surrounding context of the nonterminal[1].

Johnson (1998) investigates the effect of different tree representations and certain nonlocal features on the parsing performance of unlexicalized PCFGs estimated from the Penn Treebank. For example, he shows that the performance of such a PCFG can be increased substantially if the label of the parent node is attached to the label of each node. State-of-the-art statistical parsers, such as Collins (1997), Charniak (2000) or Bod (2001), include additional features encoding structural properties of the parse tree that go beyond the scope of local rule applications. Such features include distance measures, the labels of the node above the root of the local tree, or entire tree fragments of a depth greater than 1.

Starting with the earliest statistical parsers such as Magerman (1994), it has become clear that the words which occur in a sentence are extremely important predictors of syntactic structure. Therefore, current statistical parsers use so-called “lexicalized” models, that is models in which events can be conditioned on the presence of particular words in the sentence. In general, we can distinguish two kinds of lexical information used in statistical parsing: lexicalized structural information, where words are used to predict syntactic structure, and bi-lexical dependencies, where words are conditioned on each other, depending on the dependencies between them as defined by the parse tree.

The necessity of additional structural and lexical information leads to a substantial increase in the size (ie. number of parameters) of the models. Work in language modelling (eg. (Chen and Goodman, 1996)) has shown that, with limited amounts of available labelled data, appropriate smoothing techniques are of paramount importance to the performance of statistical models. Unfortunately, to date there is only anecdotical evidence supporting this finding for statistical parsing, but there is no reason to believe that this is not the case.

It is in general very difficult to draw meaningful conclusions about the appropriateness of the underlying models from the comparison of Parseval scores of two parsers, since actual

[1] For instance, NPs in subject position behave very differently from NPs in object position, and NPs in first object position behave differently from NPs in second position (Manning and Schütze, 1999, pp.419.)

performance is not only determined by the model structure (ie. the conditioning variables and their dependencies), but also by a large number of implementational details, such as the search strategy employed by the parser, the smoothing technique, or extraneous features such as specific rules dealing with unknown words, punctuation marks or coordination. However, some general conclusions seem to emerge from almost all reported research on parsing. These include the importance of additional structural, lexical and grammatical information, and we will return to these points in the following chapter.

Recall from chapter 3 that the linguistic representation of the Penn Treebank is fairly rich, with functional tags indicating the role of constituents and empty nodes for long-range dependencies. Unlike in the CCG translation of the Penn Treebank used in this dissertation, this information is discarded by most existing parsers, either already in the model, or in the output. This can make it difficult to give a semantic interpretation to the analyses provided by these parsers. However, there have been some attempts (Blaheta and Charniak (2000) and Johnson (2002b)) to recover this information in a post-processing stage, and we will briefly review these approaches.

4.5.1 Evaluating parser performance – the Parseval measures

The parsers discussed in this section are all evaluated according to the so-called Parseval measures (Black *et al.*, 1991). These measures evaluate the correctness of the constituents defined by parse trees. The most frequently used measures are labelled precision (LP) and labelled recall (LR), which count a proposed constituent as correct if the gold standard contains a constituent with the same span and label. By contrast, the less strict measures bracketed precision (BP) and bracketed recall (BR) consider a constituent as correct if the gold standard contains a constituent with the same span, even if both constituents do not have the same label.

Precision is the ratio of correctly proposed constituents over the number of all proposed constituents, whereas recall is the ratio of correctly proposed constituents over the number of constituents in the gold standard.

Following the terminology of information retrieval (where the measures precision and recall originate, see in particular Van Rijsbergen (1979)), correctly proposed constituents are referred to as true positives (TP), and incorrectly proposed constituents are called false positives (FP), whereas those constituents in the gold standard that are not proposed by the parser are false negatives (FN). The F-score is a combined measure of precision and recall. Since in statistical parsing, precision and recall are usually considered equally important, a balanced F-score is usually given, which assigns equal weight to precision and recall. These three measures are related as follows:

$$\begin{array}{ll} \text{Precision} & P = \frac{TP}{TP+FP} \\ \text{Recall} & R = \frac{TP}{TP+FN} \\ \text{F-score} & F = \frac{2PR}{P+R} \end{array}$$

In comparison to the original Penn Treebank annotation, the output of parsers such as Collins (1997) or Charniak (1999) is linguistically impoverished – in particular, these parsers do not annotate non-terminal nodes with function tags such as `-SBJ` or `-TMP`, and do not generate empty nodes corresponding to the null elements indicating movement and extraction in the Penn Treebank. Therefore, standardly reported Parseval scores ignore this information. Section 4.5.10 reviews two approaches which use simple classification techniques in order to enhance the output of the parser of Charniak (1999) with this linguistic information.

4.5.2 The performance of PCFGs on the Penn Treebank

Charniak (1996) and Johnson (1998) investigate the performance of PCFGs without any lexicalized features estimated from the Penn Treebank.

Charniak (1996) reports that a PCFG trained on about 300,000 words achieves bracketed precision and recall of 78.8% and 80.4% when tested on about 30,000 words of sentences ≤ 40 words, assuming the POS tags of the Penn Treebank (but not the words themselves) as input to the parser. The grammar which underlies this model has over 10,000 rules. However, it suffers from extreme overgeneration, as well as from a lack of grammatical coverage. Charniak argues that any sequence of POS tags is a possible prefix to a sentence of the language generated by this grammar.[2] He therefore emphasizes the need for efficient search in the face of massive ambiguity and overgeneration. However, this overgeneration does not entail that the grammar contains all the rules necessary to generate the correct analyses for sentences in the test corpus.

When tested on unseen data, this PCFG model attains higher recall than precision on sentences of all lengths. However, when tested on the training corpus, it parses longer sentences with higher precision than recall. This seems to indicate that flatter rules (producing shallower trees with fewer constituents) are preferred when the part-of-speech tag sequence is already observed, whereas less flat, and perhaps more general, rules are preferred when the part-of-speech tag sequence is unseen. Charniak also shows that a modification of the actual distributions of

[2] The overgeneration of the grammar which underlies the Penn Treebank has been analyzed in depth by Klein and Manning (2001).

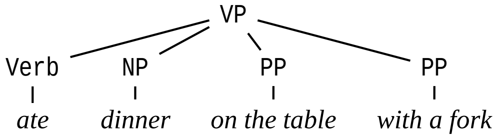
the model which favours right-branching trees achieves a higher performance than the original PCFG, despite the fact that this new model is not a maximum likelihood estimate, nor is it necessarily the case that it is consistent.

Johnson (1998) investigates the impact of different tree representations in the training corpus on the performance of a parser using a PCFG model estimated from this corpus. He shows that under certain conditions, the estimated likelihood of particular constructions according to the model can differ significantly from the frequency in the training corpus.

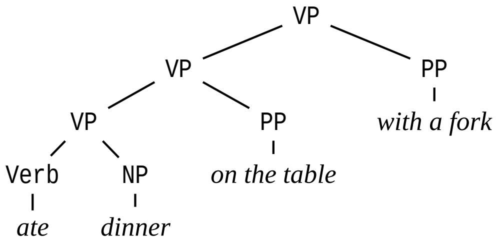
Johnson argues that if the underlying grammar produces shallow trees with few nodes, the statistical model embodies weaker independence assumptions than the model of a grammar with deeper, but narrower, trees. This is the case because a PCFG approximates the probability of an entire tree as the product of the probabilities of each local tree, regardless of its context. The fewer siblings and children a node has, the more independent it is from its surrounding context. Johnson relates the impact of the underlying tree representation to the "bias versus variance" problem in machine learning. Models with weak independence assumptions are said to have a lower bias than models with strong independence assumptions, since they are capable of modelling a larger class of distributions more accurately. However, models with weak independence assumptions typically require more parameters to estimate. For instance, grammars with very flat rules tend to generalize less well, and hence require more rules. In order to estimate these parameters accurately, more training data might be required than for the model with fewer parameters. Given that the amount of available training data is limited, this leads to an increase in the variance (ie. loss of accuracy) of the model.

Johnson looks in particular at the representation of modification by prepositional phrases. He considers the following three kinds of representation (examples from Johnson (1998)):

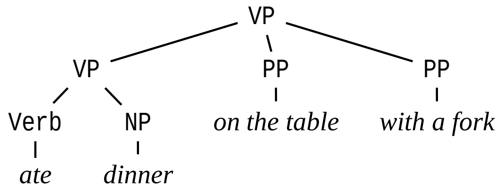
- **Flat representation** (all modifiers and arguments appear on the same level):



- **Chomsky adjunction** (each modifier appears on a separate level):



- **Two-level representation** (all modifiers appear together on a separate level):



Whereas only a grammar using Chomsky adjunction can generate an unbounded number of PP modifiers, it is clear that a grammar using the two-level representation which allows single PP modifiers also generates analyses using Chomsky adjunction. Johnson shows that this is the case for PP modification in the grammar underlying the Penn Treebank, and that a PCFG estimated from sections 02-21 of the Penn Treebank will always assign higher probability to the adjunction analysis of noun phrases with multiple PP modifiers than to the two-level representation which corresponds to the Penn Treebank analysis. In fact, the Treebank analysis will never be returned by the parser, since the Chomsky adjunct analysis has higher probability. Note, however, that this “misanalysis” will only be reflected in the precision score, but not in the recall or crossing brackets score (which might even be slightly inflated due to the increased number of constituents). By contrast, if the labelling is changed so that the label of each non-terminal node is appended to all its non-terminal children, a grammar using the two-level representation will no longer subsume the Chomsky adjunction representation.

A number of experiments are reported in Johnson (1998) which investigate the impact of a number of tree transformations on the performance of a PCFG parser estimated from the Penn Treebank. Here, a PCFG estimated from sections 02-21 of the Penn Treebank yields labelled precision and recall of 73.5% and 69.7%. The “parent transformation” described above significantly increases the labelled parseval scores to 80.0% precision and 79.2% recall, with recall now being much closer to precision than before. By contrast, if the trees in the training corpus are transformed to use the Chomsky adjunction representation for PP modification of NPs and VPs, the (de-transformed) parse trees of the test corpus obtain only 73.0% precision and 70.5% recall, which is, as Johnson demonstrates, not a statistically significant difference to the baseline model.

Johnson’s findings are of relevance to our purposes in that they highlight the importance of the linguistic representation for the performance of a statistical model. However, when using phrase-structure grammars, the syntactician has a much higher degree of freedom in designing the rules for a particular language than in CCG. By using CCG, we are already committed to a particular kind of tree structure. We might still disagree about the analyses of particular constructions, or about whether to model a canonical derivation structure or not (and if so, what kind of canonical derivation structure), but in any case we deal with at most binary-branching derivation trees that are labelled with very expressive categories. Therefore, CCG presents an interesting point of comparison to the grammars Johnson considers. On the one hand, the binary tree structure results in very strong independence assumptions if we model only local trees, but on the other hand, the information encoded in CCG categories captures certain non-local structural properties of the tree which are hard to represent in Penn Treebank style grammars.

4.5.3 The conditional model of Collins (1996)

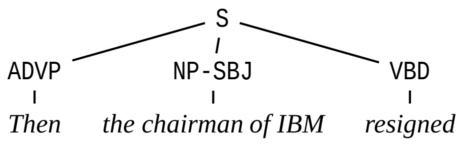
Collins (1996) presents a conditional model of parse selection, in which the probability of a parse tree T given a sentence S is estimated directly. Although this model has no direct relation to the approach taken in this dissertation, it is nevertheless important, since an alternative proposal for statistical parsing with CCG, Clark *et al.* (2002), which we will discuss in section 5.13.1, uses a model which is very similar to Collins' conditional model.

Here, a parse tree is represented as a sequence of non-recursive noun phrases (so-called base NPs) B and a set of dependency relations D between the headwords of the constituents of the parse tree. Given a sentence S (S is POS-tagged: $S = \langle (w_1, t_1), ..., (w_n, t_n) \rangle$), the probability of a parse tree T is determined as the product of the probability of the base NP sequence defined by T and the probability of the set of dependency relations defined by T:

$$P(T|S) = P(B,D|S) = P(B|S)P(D|S,B)$$

The model for the probability of the base NP sequence considers for each gap between two consecutive words in S the probability that this gap is outside of, the start of, in the middle of, or at the end of a base NP.

The dependency relations are determined by the parse tree in two ways: first, for each local tree, the head child is identified heuristically. The head word of a node is defined as the head word of its head child. Therefore, each local tree defines a set of dependency relations between the head word of its head child and the head words of its non-head children. Consider the following tree:



This local tree defines two dependencies, one between *chairman*, the head word of the subject NP, and *resigned*, and one between *Then* and *resigned*. Collins (1996) defines a dependency relation as the triple of the label of the non-head child, the label of the parent node and the label of the head child. Hence, the relation between *Then* and *resigned* is $\langle \mathtt{ADVP}, \mathtt{S}, \mathtt{VP} \rangle$ and that between *chairman* and *resigned* is $\langle \mathtt{NP}, \mathtt{S}, \mathtt{VP} \rangle$. Furthermore, if this S node is the root of the tree, there is a unary "dependency" relation $\langle S \rangle$ which holds for *resigned*, the head word of the tree. Since each constituent in a parse tree is either a non-head child or the head child of another constituent, each head word w_i of a constituent is either the head word of a non-head child of another constituent or the head word of the entire tree. For each head word w_j of a constituent in T, Collins denotes as "arrow from" w_j, or $AF(j)$, the ordered pair $\langle R_j, h_j \rangle$ consisting of a word h_j, and the dependency relation R_j which holds between w_j and h_j, such that w_j is either the head word of a non-head child of some constituent headed by h_j or the head word of the entire tree (in which case h_j is a dummy word "0").

The model for the probability of the dependency relations considers for each head word w_i of a constituent in T the probability of the dependency arrow from w_i, as defined by T. The probability of the set of dependency relations is then defined as the product of these individual dependency relations:

$$P(D|S,B) = \prod_{j=1}^{m} P(AF(j)|S,B)$$

However, the set of dependency relations that are captured by the model are only those that hold between head words of base NPs or other constituents, but not between the words within individual base NPs.

The probabilities of the dependency relations $P(AF(j)|S,B)$ are difficult to estimate directly, since, in general, S is unseen. Instead, Collins observes that they can be approximated in the following manner: For each pair of words w_i, w_j in the language, there is a distribution $P(AF(i) = \langle w_j, R_i \rangle | w_i, w_j)$, which can be estimated using the relative frequencies from the training corpus and appropriate smoothing techniques. Then, $P(AF(j)|S,B)$ can be computed as follows:

$$P(AF(j) = \langle h_j, R_j \rangle | S,B) \approx \frac{P(AF(j) = \langle h_j, R_j \rangle | w_j, h_j)}{\sum_{k=1..m, p \in P} P(AF(j) = \langle w_k, p \rangle | w_j, w_k)}$$

But, as noted by Collins (1999), there are a number of problems with this model: first, the way in which the dependency probabilities are estimated means that the model is no longer a maximum likelihood estimate. Furthermore, the model is deficient: by treating all dependency relations as independent, probability mass is wasted on sets of dependency structures that cannot be licensed by a single parse tree (such as sets containing crossing dependencies).

4.5.4 Goodman (1997b)'s probabilistic feature grammars

The probabilistic feature grammars of Goodman (1997b) are a general framework for generative probability models over binary-branching trees in which additional features such as distance measures or lexical head words can be incorporated. Given that CCG derivation trees are also (at most) binary-branching, they are of particular relevance to our own work.

Like in other generative models, such as PCFGs, trees are generated top-down, with the probability of child nodes conditional on the already generated ancestors. The main insight behind probabilistic feature grammars is that the probability of generating a local tree (ie. a rule expansion) can be factorized as the product of generating individual features of this tree. These features can also represent properties of the surrounding context, such as the label of the grandparent node, the lexical heads of the head and non-head child, etc. For instance, in his experiments, Goodman uses features such as the labels of the parent and daughter nodes, the lexical heads of the head child and the non-head child, distance measures between the lexical heads of the head child and the non-head child as well as between the lexical heads of the children and the left and right frontier of the children.

If local trees are represented as feature vectors $\langle f_1, ..., f_n \rangle$, where features $f_1..f_i$ represent the features of the ancestors, then instead of generating the entire feature vector at once with $P(\langle f_1, ..., f_n \rangle | f_1..f_i)$, we can apply the chain rule to generate individual features (or subsets thereof) one at a time, conditioned on previously generated features:

$$P(\langle f_1, ..., f_n \rangle | f_1..f_i) = P(f_{i+1} | f_1..f_i) P(f_{i+2} | f_1..f_i, f_{i+1}) ... P(f_n | f_1..f_{n-1})$$

This factorization is still equivalent to generating the entire feature vector at once, but it is straightforward to introduce independence assumptions between features by only conditioning on a subset of the previously generated features, eg.:

$$P(f_j | f_1, ..f_k, ..f_{j-1}) \approx P(f_j | f_k, f_{j-1})$$

Such distributions can in turn be smoothed, for instance by deleted linear interpolation:

$$P(f_j | f_k, f_{j-1}) \approx \lambda \hat{P}(f_j | f_k, f_{j-1}) + (1-\lambda) \hat{P}(f_j | f_{j-1})$$

with $0 \leq \lambda \leq 1$, and $\hat{P}(...|...)$ the relative frequencies obtained from the training data.

Thus, in this framework, the modelling task consists of the choice of suitable features, their ordering, and independence assumptions between them.

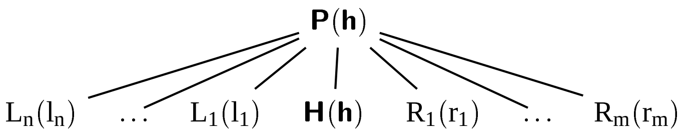
However, the experimental results given in Goodman (1997b) with LP=85.3%, LR=84.8% (sentences $\leq$ 40 words, section 23) show that on the standard Penn Treebank parsing task, a probabilistic feature grammar with bilexical dependencies and a distance measure performs slightly worse than the best instantiation of the model of Collins (1996), which achieves LP=86.3% and LR=85.8% for the same set of sentences. However, Collins' results are obtained with a special punctuation rule. This punctuation rule alone raises the performance of the baseline model of Collins (1996) from LP=84.9% and LR=84.9% (ie. slightly below Goodman) to LP=85.5% and LR=85.4% (slightly above Goodman).

4.5.5 The generative models of Collins (1997)

The models of Collins (1997) are of particular relevance to our own research, since a number of implementational details were adapted from this parser. Three main findings emerge from this work: first, a method to generalize over the rules of a context-free grammar, and hence to deal with the unconstrained, flat trees of the Penn Treebank. This method, called Markov grammar, was subsequently used by Charniak (2000). The second and third finding concern the use of grammatical information in models for statistical parsing. Collins (1997) shows that the inclusion of subcategorization information (or valency), conditioned on the words in question, can yield a substantial improvement in performance. On the other hand, he was unable to show such an effect for the information encoded by the null elements of the Penn Treebank which model "movement", or extraction. Another conclusion which was initially drawn from Collins' results, regarding the importance of bilexical dependencies, was disproved by Gildea (2001), at least for the model that does not make use of subcategorization information.

This section will first introduce the three models of Collins (1997) and their performance; then the beam search, the smoothing technique and the treatment of unknown words will be presented, since the methods used by our own parser are derived from these.

I will employ the following notation and terminology to talk about phrase-structure trees: a *node* is a triple $\langle X, t, w \rangle$, consisting of a label, a lexical tag and a head word. A *local* tree is a tree of depth 1, P is the non-terminal root label of a local tree (the *parent*). The head tag is t, and the head word is w. The head child has label H. The *i*th non-head child to the left of the head child has label $\mathsf{L_i}$, head tag $\mathsf{l_i}$, and head word $\mathsf{w_i}$. The *j*th non-head child to the right of the head child has label $\mathsf{R_j}$, head tag $\mathsf{r_j}$, and head word $\mathsf{w_j}$:



Model 1 - the basics This model contains three families of probability distributions: the initial probability of generating a sentence with root node $\langle \texttt{S}, t, w \rangle$ from the start symbol TOP, the head probabilities of generating a head child **h** given a root node $\langle P, t, w \rangle$, and the left and right probabilities of generating the ith non-head sister $\langle L_i, l_i, w_i \rangle$ to the left or right of the head child. The insertion of stop symbols $STOP$ as outermost non-head daughters is necessary in order to allow the Markov process to model the sequences properly. The model also incorporates the following two distance variables: adjacency(i) (= *true* or *false*), indicating whether the surface string between the head and the *i*th child is empty, and verb(i) indicating whether there is a verb in the surface string between head and non-head child.

Here are the unsmoothed probabilities used in Model 1:

$$
\begin{array}{ll}
\text{Top node :} & P_{top}(\mathrm{H}, \mathrm{h} | \mathrm{TOP}) \\
\text{otherwise :} & P_h(\mathrm{H} | \mathrm{P(h)}) \\
 & \cdot \prod_{i=1}^{n+1} P_l(\mathrm{L_i}(\mathrm{l_i}) | \mathrm{P(h)}, \mathrm{H}, \mathrm{adjacency(i)}, \mathrm{verb(i)}) \\
 & \cdot \prod_{j=1}^{m+1} P_r(\mathrm{R_j}(\mathrm{r_j}) | \mathrm{P(h)}, \mathrm{H}, \mathrm{adjacency(i)}, \mathrm{verb(i)})
\end{array}
$$

Model 2: Subcategorization Model 2 extends the basic model by incorporating the notion of subcategorization. This also presumes that the model distinguishes between complements and adjuncts. The generation of the non-head children is altered as follows: first, two subcategorization lists (*LC* and *RC*) of expected complements to the left and right of the head child are selected. Then the non-head children are generated with a probability that is also conditioned on the appropriate subcat list. After a complement child L_i (or R_i) is generated, the corresponding nonterminal is removed from the subcat list LC_i (RC_i), so that the following non-head child L_{i+1} (RC_{i+1}) is conditioned on the updated subcategorization list LC_{i+1} (RC_{i+1}).

$$
\begin{gathered}
P_h(\mathrm{H}|\mathrm{P(h)}) \\
\cdot P_{LC}(LC|P(h),H) \\
\cdot P_{RC}(RC|P(h),H) \\
\cdot \prod_{i=1}^{n+1} P_l(\mathrm{L_i(l_i)|P(h),H,adjacency(i),verb,}LC_\mathrm{i}) \\
\cdot \prod_{j=1}^{m+1} P_r(\mathrm{R_j(r_j)|P(h),H,adjacency(i),verb,}RC_\mathrm{i})
\end{gathered}
$$

Model 3: Movement Model 2 ignores the null elements which in the Treebank indicate long-distance dependencies. This means that different subcategorization frames have to be selected whenever there is a long-distance dependencies. In model 3, null elements are also predicted, and hence the same subcategorization frame can be used for local and long-distance dependencies. In order to do this, a gap-passing mechanism is incorporated in the model: heads also generate a gap-passing parameter $G \in \{Head, Left, Right\}$, which specifies where the $+gap$ feature is generated by or passed on to (H, LC or RC). An `NP` can, for instance, expand to an `NP` followed by a relative clause, `SBAR(+gap).` This gap feature is then percolated down to the `VP` level inside the relative clause, where generation of a `TRACE` removes the $+gap$ feature and one complement `NP` from the subcat frame. This analysis of relative clauses assumes that relative pronouns are adjuncts to the embedded clause. Hence, the gap feature cannot be generated by the relative pronoun, but must originate at the `NP` level.

$$
\begin{gathered}
P_h(\mathrm{H}|\mathrm{P(h)}) \\
\cdot P_G(\mathrm{G}|P(h),H) \\
\cdot P_{LC}(LC|P(h),H) \\
\cdot P_{RC}(RC|P(h),H) \\
\cdot \prod_{i=1}^{n+1} P_l(\ \mathrm{L_i(l_i)|P(h),H,adjacency(i),verb,}LC_\mathrm{i}) \\
\cdot \prod_{j=1}^{m+1} P_r(\ \mathrm{R_j(r_j)|P(h),H,adjacency(i),verb,}RC_\mathrm{i})
\end{gathered}
$$

The beam search With grammars as ambiguous as those underlying the Penn Treebank and models such as those used by Collins' parser, exhaustive parsing is next to impossible; therefore strategies which effectively prune down the search space without incurring too high a loss in accuracy are essential. Beam search strategies aim to identify those constituents within a cell that are likely to lead to a high probability parse. These constituents are kept, and the rest are discarded.

In generative models, the inside probability of a constituent τ (with label C and lexical head $h = \langle w,t \rangle$) is the probability that a node with label C expands to a tree τ. In lexicalized models which generate the lexical heads of constituents at their maximal projection, the inside probability of a constituent is not only conditioned on the label C, but also on the lexical head of the constituent. The outside probability of τ is the probability of generating any tree which contains a node labelled C (and its lexical head). If the probability model only takes local trees into account, the inside probability can easily be computed during parsing. However, the inside probability alone is not necessarily a good indicator of whether a constituent will form part of a high probability parse. Instead, Caraballo and Charniak (1998) and Goodman (1997a) advocate the use of a prior probability which indicates how likely a constituent with label C and lexical head h is. This prior probability is an approximation of the outside probability of the constituent. The "figure of merit" which is used to rank constituents within the same cell of the chart is then the product of the prior probability and the inside probability:

$$FigMerit(C) = P_{inside}(C)P_{prior}(C,w,t)$$

The prior probability is estimated in the following way:

$$\tilde{P}_{prior}(C,w,t) = \hat{P}(t,w)[\lambda\hat{P}(C|t,w) + (1-\lambda)\hat{P}(C|t)]$$

where $\hat{P}$ is a relative frequency estimate obtained from all constituents in the training data and the smoothing parameter λ is computed as described below.

The treatment of rare and unknown words Words which occur less than 5 times in the training corpus, and unknown words in the test corpus, are replaced by an ”UNKNOWN” token. The tagger of Ratnaparkhi (1996) is used to assign part-of-speech tags to unknown words in the test data.

Smoothing in Collins' models A very important factor for the performance of a statistical parser is the smoothing technique which is used in the probability model. Simple maximum likelihood estimation does not yield reliable parameters for events which are rare in the training data, and assigns zero probability to anything that does not occur in the training data. As explained in section 4.4, smoothing attempts to overcome this problem. Collins (1999) uses a

Back-off	$P_H(H\|\dots)$	$P_G(G\|\dots)$ $P_{LC}(LC\|\dots)$ $P_{RC}(RC\|\dots)$	$P_{L1}(L_i,l_i,c,p\|\dots)$ $P_{R1}(R_i,r_i,c,p\|\dots)$	$P_{L2}(lw_i\|\dots)$ $P_{R2}(rw_i\|\dots)$
1	P, w, t	P, H, w, t	P, H, w, t, Δ, *LC*	L$_i$, lt$_i$,c , p, P, H, w, t, Δ, *LC*
2	P, t,	P, H, t	P, H, t, Δ, *LC*	L$_i$, lt$_i$,c , p, P, H, t, Δ, *LC*
3	P	P, H	P, H, Δ, *LC*	lt$_i$

Table 4.1: The conditioning variables for the different back-off levels in Collins' Model 3: Δ is the distance variable, *p* and *c* are the slash and coordination features

linear combination of back-off estimates. Table 4.1 shows the conditioning variables for the different back-off levels. The estimation *e* of a conditional probability is computed as follows:

$$e = \lambda_1 e_1 + (1-\lambda_1)(\lambda_2 e_2 + (1-\lambda_2)e_3)$$

where *e*1, *e*2, *e*3 are maximum likelihood estimates with the conditioning variables at back-off levels 1, 2 and 3, and λ_i is a smoothing parameter with $0 \le \lambda_i \le 1$. These smoothing parameters are computed as follows: for $e_i = \frac{n_i}{f_i}$, and u_i = the number of unique outcomes in the distribution,

$$\lambda_i = \frac{f_i}{f_i + 5u_i}$$

Collins reports that the constant 5 was determined empirically. When estimating a conditional probability $P(X|Y)$, the number of unique outcomes in the distribution, u, is the number of distinct events X that co-occur with event Y. If u_i is high, the corresponding λ_i is small, thus reflecting an assumption that if Y co-occurs with many different types of events X in the training data, it is likely that the training data does not contain all possible events that Y can co-occur with, and that it will co-occur with unseen events in the test data.

Our own models use the same smoothing technique, although we would expect that a more sophisticated smoothing technique might yield better results. However, this method has the practical advantage that no held-out data is required, so we can use all available manually annotated data to train the model, and do not need to parse any unlabelled data. The question of how the estimation of the smoothing parameters influences parsing performance is beyond the scope of this dissertation.

4.5.6 Charniak (2000)'s Maximum-Entropy-inspired model

Charniak (2000, 1999) presents another statistical Treebank parser, which was the first to outperform Collins (1997). Apart from providing a slightly higher benchmark than Collins (1997) for Parseval, the main relevance of this parser to our own research lies in providing evidence that linear interpolation can lead to a decrease in performance if additional features are incorporated into an already complex model, but that with a different estimation technique a higher performance can be obtained from these features. However, like Collins (1997), the model itself is specifically tuned to account for the flat trees of the Penn Treebank, and most of its features are therefore of little immediate importance for our own purposes.

Like the parser of Collins (1997), the underlying probability model is based on a generative Markov grammar. Charniak generates the left and right siblings of the head node in the same order as Collins, but conditions the label of non-head daughters on up to three already generated non-head siblings that are adjacent to the node to be generated. There are no separate distance measures in this model – presumably this information is captured by the other non-head siblings. There are also no features corresponding to subcategorization frames, since this model does not distinguish between complements and adjuncts. Unlike Collins, Charniak uses a simple probabilistic chart parser to generate candidate parses, which he then ranks by his model. This allows him to take non-local features such as the labels of the parent and the left sibling of the root of the local tree into account. However, the individual distributions are not linear interpolations of relative frequencies. Instead, Charniak makes the observation that the conditional probability $P(Y|X_1,...,X_n)$ can be expressed in the following way:

$$P(Y|X_1,...X_n) = P(Y|X_1)\frac{P(Y|X_1,X_2)}{P(Y|X_1)}\frac{P(Y|X_1,X_2,X_3)}{P(Y|X_1,X_2)}\cdots\frac{P(Y|X_1,X_2,X_3,...,X_n)}{P(Y|X_1,X_2,X_{n-1})}$$

In the expression on the right-hand side, the first term uses minimal conditioning information, and each subsequent term is a number from zero to positive infinity which is greater than one if the added conditioning variable increases the probability of y, and less than one if it decreases the probability of Y. Note that with the exception of $P(Y|X_1,...,X_n)$, each numerator on the right hand side of the above equation is cancelled out by one of the denominators – therefore, this equation expresses strict equality. However, Charniak argues further that when Y is conditioned on a large number of conditioning variables $X_1,...,X_n$, it is convenient to assume that Y only depends on a subset of these conditioning variables, and is conditionally independent of the remaining conditioning variables given that subset. By making fewer independence assumptions for the terms in the numerators than for their counterparts in the denominator, the strict equality of the above equation is lost, and the resulting estimate of $P(Y|X_1,...,X_n)$ really differs from its relative frequency estimate. Again, each of the fractions are now terms greater or smaller than 1, depending on the contribution of the additional conditioning variables in the numerator to the probability of Y. In this sense, these fractions are similar to the features of a log-linear, or Maximum Entropy, model. However, this model is computationally simpler, in that there is no normalization factor, or optimization of the feature weights – in fact, Charniak comments that his model corresponds to a log-linear model where the weights have been initialized but not yet optimized. Unfortunately, he does not report whether the performance of his parser improves if his model is treated like a true Maximum Entropy model and the weights are optimized. Charniak demonstrates the effectiveness of his estimation technique by showing that including the label of the grandparent of the node to be generated as well as the label of the left sibling of its parent gives an improvement of 0.45% in labelled precision/recall when this estimation technique is used, whereas with standard linear interpolation, performance drops by 0.6% when these features are included. Note that the individual terms $P(Y|X_i,...,X_j)$ in the above equation still have to be smoothed. Here, Charniak uses a variant of deleted interpolation.

Charniak stresses the importance of part-of-speech tags. He reports that a word's POS tag is a better predictor of the word than the maximal projection of the constituent of which this word is head, and that the POS tag of the head of a constituent can also be a very good predictor of the expansion of a constituent, even when backing off from the head word. Conditioning word and rule probabilities on POS tags in this manner yields an improvement of 2% in his model.

4.5.7 Data-oriented parsing

Recently, the parser of Bod (2001) has claimed the highest reported Parseval scores for a single-model parser on section 23 of the Penn Treebank with labelled precision and recall of 89.7% on sentences $\leq$ 100 words and 90.8%/90.6% on sentences $\leq$ 40 words, although it is not clear whether an improvement of 0.2%/0.1% for sentences $\leq$ 100 words on the results of Charniak (2000) is a statistically significant difference. This parser is based on the Data Oriented Parsing (DOP) model (see also Bod (1998)), which computes the probability of a parse tree by considering the probability of all its subtrees, and hence can take arbitrary structural and lexical dependencies into account. However, as noted by Johnson (2002a), the relative frequency estimation procedure standardly used to estimate DOP models leads to biased and, more importantly, inconsistent models.

Like Charniak’s parser, Bod’s parser operates in two stages: first the *n*-best parse trees are generated using a chart parsing algorithm (although details are omitted); then the most likely parse tree is chosen from a set of 1,000 derivations. Bod shows that it is helpful to estimate the model over subtrees up to a depth of 14, and containing up to 12 words on their frontier. However, it is not clear what structural or lexical dependencies (other than perhaps certain kinds of collocations or fixed expressions) require this amount of lexical information, and it would be interesting to see how the specificity of the model impacts on its performance on data from another, more diverse, domain. It would also be interesting to know to what extent Bod’s parser benefits from using Good-Turing smoothing (instead of linear interpolation) to estimate the probabilities of subtrees with unknown lexical anchors.

4.5.8 Combining multiple parsers

Henderson and Brill (1999) investigate ways of combining the predictions of three existing parsers (Collins (1997), Charniak (1997) and Ratnaparkhi (1998)), and find that schemes with consider each constituent in isolation yield better results than deciding between three complete parse trees for each sentence. Their method gives the currently best known performance with 92.1% LP and 89.2% LR on section 23 (which they used as development set), and 92.4% LP and 90.1% LR on section 22 (their test set).

Henderson and Brill (2000) show that more efficient use can be made of the available training data. They use bagging and boosting to create different instantiations of the same model (they use Model 2 of Collins (1997)). The training data for these models is in both cases created by sampling with replacement from the standard training corpus. Boosting differs from bagging in that it iteratively re-samples the training data according to a distribution which is skewed to give more weight to samples where the previous model made an incorrect prediction. Like Henderson and Brill (1999), the final parse is selected by a voting procedure among the individual parsers. The best results of Henderson and Brill (2000) are 89.5% LP / 88.8% LR for bagging and 89.2% LP / 87.2% LR for boosting, both of which are an improvement on the performance of the individual parsers.

According to Dietterich (2000), for simple classification tasks ensemble learning methods can outperform single classifiers for three different kinds of reasons. Computational reasons are important for approaches which require hill-climbing searches (such as EM), but are probably less of a problem for estimators using labelled data. The statistical reason Dietterich gives is essentially an argument that ensemble methods can reduce the variance of an estimator if there is not enough training data available. The third reason is representational – even if the target function lies outside the class of functions considered by each individual model, it is still possible that it is in the space of the combined models. Therefore, ensemble methods can also reduce the bias of an estimator (which might explain why the combination of different parsers yields better results than the combination of different instantiations of the same model).

It is not clear to what extent the success of the constituent-based voting technique in Henderson and Brill (1999) is a result of the flat tree representation used in the Penn Treebank. Note that they give higher precision than recall scores, suggesting that their voting algorithm returns even flatter trees.

Since CCG derivation trees are binary and their labelling is constrained by the combinatory rule schemata, different derivations could not easily be combined by considering individual constituents in isolation – instead, constituent-based voting would have to take place during parsing, which would require a much higher integration of the different models as can be done with off-the-shelf implementations.

4.5.9 Re-ranking parsers

Collins (2000) and Collins and Duffy (2002) present two approaches which take the output of the parser of Collins (1999) (run in *n*-best mode) and re-rank the parses using a different model, thereby improving performance from 88.1% LP/88.3%LR to 89.6%LP/89.9% LR (Collins, 2000), using additional non-local features and a variant of boosting, and to 88.6% LP / 88.9 % LR (Collins and Duffy, 2002) using a voted perceptron which, like the DOP model, has access to all possible subtrees.

4.5.10 Enhancing the output of a Treebank parser

As noted above, Treebank parsers such as Charniak (1999) yield a linguistically impoverished output. The trees produced by these parsers lack the null elements and the function tags of the original Penn Treebank annotation. Function tags are important to distinguish certain types of adjuncts from complements (eg. temporal modifiers, such as NP-TMP). Null elements indicate extraction, right node raising, ellipsis and passive mood in the original Treebank.

Blaheta and Charniak (2000) give an statistical model which labels the nodes of parse trees with the function tags of the Penn Treebank. They give an F-measure of 87% on the output of Charniak's parser, ignoring nodes without function tags (since they constitute by far the majority). However, only correctly predicted constituents are considered, whereas false positives (nodes in the proposed parse tree that do not correspond to constituents in the gold standard) and false negatives (nodes in the gold standard that are not proposed by the parser) are ignored. Given that this parser has 89% labelled precision and recall, this corresponds to approximately 11% of the constituents. It is unclear how in a real application correctly predicted nodes could be distinguished from incorrect ones. In section 00, there are 38,128 non-terminal constituents that are neither punctuation marks nor null elements. Out of these, 9,645, or 25.3% carry at least one function tag. Therefore, we could expect the overall performance of this algorithm to be about two to three percent lower than 87%.

Johnson (2002b) presents an algorithm akin to memory-based learning which inserts null elements into phrase structure trees and co-indexes them with existing constituents in the tree. This algorithm attains an f-measure of 79% on restoring empty nodes in the trees proposed by the parser of Charniak (1999), and 88% on the original trees in section 23 stripped bare of their null elements. Co-indexation is harder, with an f-measure of 68% on the parser output and 75% on the original Treebank trees. For movement traces (`*T*` under `NP`) that are co-indexed with a relative pronoun (`WHNP`), this algorithm gives an f-measure of 90%. Most of these correspond to subject extraction relative clauses.

4.6 Generative models of Tree-Adjoining Grammar

Chiang (2000) presents experimental results on statistical parsing with a lexicalized Tree Insertion Grammar (LTIG) acquired from the Penn Treebank. His probability model is based on the Stochastic Tree-Adjoining Grammars of Resnik (1992) and Schabes (1992), which defined two (equivalent) generative probability models for languages defined by Tree-Adjoining Grammars (TAGs). These models are characterized by three families of distributions: the probabilities of selecting an initial tree, substitution probabilities and adjunction probabilities. Chiang extracts his grammar from the Penn Treebank, using the head-finding rules and complement-adjunct distinctions of Collins (1999). Like the approach taken in this dissertation, parameters are estimated directly from the translation of the Penn Treebank. Since the derived trees of TAG look like phrase-structure trees, Chiang uses standard Parseval measures to evaluate the output of his parser. With labelled precision and recall of 86.2% and 85.8% on sentences $\leq$ 100 words, it performs better than the model of Collins (1996), and only slightly worse than Collins (1997). This demonstrates that the use of an expressive grammar in statistical parsing does not need to lead to a decline in parsing accuracy. It should be noted that Chiang's parser is not as tuned to the representation of the Penn Treebank as Collins (1997), and does not use the heuristics for coordination and punctuation employed by Collins. Furthermore, the space of possible TAGs that can be extracted from the Penn Treebank is very large, and it is quite likely that the head-finding heuristics used here are not ideal for this grammar formalism.

4.7 Conditional models – the log-linear framework

Formalisms such as HPSG or LFG (as well as some variants of categorial grammar), use feature structures to represent linguistic information. Abney (1997) demonstrates that re-entrancies in these feature structures correspond to DAG-like dependencies which can lead to problems for standard relative frequency estimation methods. Instead, he proposes the use of log-linear Markov Random Field models. In such models, the conditional probability $P(Y|X)$ is expressed as follows:

$$P(Y|X) = \frac{e^{\sum_i \lambda_i f_i(X,Y)}}{\sum_Y \sum_i \lambda_i f_i(X,Y)}$$

Here, the $f_i(X,Y)$ are features, or indicator functions. These are usually binary-valued (0 and 1). The λ_i are (real-valued) feature weights which represent the contribution of the individual features. The modelling task consists of two steps: choosing appropriate indicator functions (feature selection) and setting their weights (parameter estimation). It can be shown that among all models of this parametric form, the model which maximizes the conditional entropy $H(Y|X)$ also maximizes the likelihood of the training data (Berger *et al.*, 1996). Therefore, such models are often referred to as Maximum Entropy or MaxEnt models. There is a sizable, and steadily growing, literature on log-linear models for NLP applications. Since these models are not of immediate relevance to the research presented in this dissertation, the interested reader is instead referred to introductions such as Berger *et al.* (1996).

Johnson *et al.* (1999) argue that a direct estimation of log-linear models which maximize the likelihood of the training data is computationally intractable for wide-coverage grammars, and propose an alternative estimation method which maximizes the conditional likelihood of the parses in the training data given their yields. Riezler *et al.* (2002) present experimental results for a log-linear model of a large, manually developed, Lexical Functional Grammar (LFG), whose parameters were estimated using a subset (10,000 sentences) of the Penn Treebank. This model is evaluated in terms of (labelled) predicate-argument relations between words. When tested on 700 sentences of section 23, an F-score of 79% is obtained.

4.8 Conclusion

This chapter has presented a brief overview of the relevant concepts of probability theory and machine learning as well as a review of important work in statistical parsing.

The success of so-called lexicalized models in statistical parsing has demonstrated that words (and, at least to some extent, the relations between words) are extremely important predictors of syntactic structure. Interestingly, and contrary to received wisdom in probabilistic modelling, there are no guarantees for some of the currently best performing statistical parsers on the Penn Treebank, such as Charniak (2000) and Bod (2001), that they are based on consistent models. In fact, for the model of Bod (2001), there exists even a proof to the contrary. It seems likely that with the current levels of performance, the choice of appropriate features is still more important than whether a probability model is consistent or not. The currently best performing single-model parsers on the Penn Treebank (Charniak (2000), Bod (2001), Collins (2000)) are all based on a re-ranking approach, where the parsing stage is separated from the selection stage, which makes it easy to use non-local features that capture information about the global structure of the tree. On the other hand, Henderson and Brill (1999) show that the combination of three parsers which all use only local features (albeit in slightly different manners) yields significantly better results than the individual models – and their performance has not yet been rivalled by re-ranking approaches. Charniak (2000) and Bod (2001) use more involved smoothing techniques than Collins (1999), which might have a significant impact on performance. Unfortunately, there has not been a study comparable to Chen and Goodman (1996) on the effectiveness of different smoothing techniques for statistical parsing, and it is difficult to know to what extent the existing models benefit from the smoothing technique employed.

Phrase-structure grammars are a general rewriting system, but do not constitute a linguistic theory by itself, in the sense that there are no restrictions on the category labels or specific instances of rewrite rules as long as they not violate the principles of the formal system. Given that all implementations of statistical parsers require data to estimate the parameters of their models, research on English parsing has focused on the representation embodied in the Penn Treebank, as the main available labelled resource. However, it is important to bear in mind that it is not clear whether the findings for this particular representation carry over to other representations.

The next chapter investigates how established techniques can be adapted to and perform on CCG, using CCGbank as the test and training data. We hope that this provides an interesting point of comparison to existing models, since the underlying linguistic representation differs from that of the original Penn Treebank in many important respects. The aim of the research reported in the next chapter is also to provide at least a baseline model and an implementation of a simple statistical parser for CCG, thus establishing the infrastructure required for more sophisticated approaches such as re-ranking or model combination.

Chapter 5

Generative models of CCG derivations

This chapter defines a number of generative models over CCG normal-form derivations, and presents empirical results which show that even a very simple dependency model can attain state-of-the-art performance in recovery of word-word dependencies. This model compares favourably with an alternative approach proposed by Clark *et al.* (2002), both in terms of the modelling technique and parsing accuracy. In terms of recovery of long-range dependencies, our approach also seems to achieve a higher performance than that of Treebank parsers (Collins, 1999; Johnson, 2002b).

Similar experiments on an earlier version of CCGbank were reported in Hockenmaier and Steedman (2002b).[1]

5.1 Introduction

As described in the previous chapter, generative models over phrase-structure trees such as Collins (1997) have achieved a high parsing performance on the Penn Treebank. This chapter shows how CCG derivation trees can be treated in a similar manner, and defines a number of generative models over CCG normal-form derivations in order to investigate how features which have been shown to improve the performance of Penn Treebank parsers influence the performance of a CCG parser. Our experimental results demonstrate that state-of-the-art per-

[1] The version of CCGbank used in Hockenmaier and Steedman (2002b), has slightly less coverage on the Penn Treebank. For instance, we did not cover UCPs in argument position. The grammar itself has also been refined since. The results reported in this chapter largely confirm the findings in our earlier publication. However, we obtain slightly higher results for the baseline model, and slightly lower results for Δ_{Pct} and the grandparent transformation on this version of CCGbank. We did not report results for **LexCatDist** in Hockenmaier and Steedman (2002b).

formance (in terms of recovery of unlabelled word-word dependencies) can be achieved by a very simple dependency model. We conjecture that this simple model performs so well because of the amount of linguistic information encoded in a CCG derivation.

However, our experimental results fail to show that the performance of the dependency model can be improved by including additional structural features such as distance measures. We argue that this is not because such features are redundant given what is represented in a CCG derivation. Instead, it is more likely that we are faced with a bias/variance problem that is more acute than for standard Treebank parsers, and exacerbated by the smoothing technique (taken from Collins (1999)). Penn Treebank parser often assume a very overgenerating grammar, which makes them very robust, at the expense of linguistic expressiveness. By contrast, our grammar is defined by the rule instantiations and word-category pairs observed in the training corpus. While we cannot rule out overgeneration, our experiments indicate that we suffer rather from the opposite problem, namely undergeneration, which manifests itself most clearly in a lack of lexical coverage. This, together with our current inability to benefit from structural and lexicalized knowledge at the same time, leads us to believe that one of the main obstacles towards higher parsing performance with our current models is the fact that they do not generalize beyond the category and rule instantiations observed in the data. What is needed is a linguistically sound way of generalizing from the behaviour of one category to another, coupled with an appropriate way of assigning probabilities to these generalizations.

Steedman (2000) argues that in CCG syntactic derivations should be considered merely a record of the construction of semantic interpretations, since the same interpretation can usually be derived in multiple ways. The following section provides a number of arguments why we nevertheless chose to model derivations. Then, a very simple generative model for CCG derivations is described. Next, the question of how to evaluate a CCG parser is addressed. After a description of the experimental setup, the simple model is extended to capture additional structural and lexical information, and the experimental results are presented and analyzed. Finally, the models presented in this chapter are compared with the (conditional) model of Clark *et al.* (2002).

5.2 What model for CCG?

In categorial grammar, semantic representations (or logical forms) are constructed together with the syntactic derivation (or surface structure). Steedman (2000, p.87) argues that in CCG “Surface Structure is not a grammatical ‘level of representation’ at all”, and that logical form is the only level of structure built by CCG. Instead, surface structure is considered "strictly a record for the process of deriving such Logical Forms." This warrants the question whether a probability model for CCG should be defined over surface structures or semantic interpretations. The approach presented in this and the following chapter models syntactic derivations, for a number of reasons:

The first argument concerns the status of derivational structure in CCG as a linguistic framework. Despite the fact that syntactic structure is not considered a level of representation, CCG itself is a theory of syntax, not of the derived semantic interpretation. The combinatory rules and principles of CCG state how semantic interpretations can be constructed from surface strings. However, this is the only restriction on the form of the semantic interpretations, which could be simple predicate-argument structures, or expressions of the λ-calculus (as in most examples given in Steedman (2000)), but need not be (see Baldrige and Kruijff (2002) for a proposal to use hybrid modal logic).

The second argument is about the modelling task itself. There need not be a one-to-one correspondence between semantic interpretation and surface string – depending on what is represented in the semantics, the same semantic interpretation may be expressed by multiple surface strings. If we restrict ourselves to rank competing analyses by the likelihood of their semantic interpretation without taking the syntactic structure into account, we run the risk of losing the connection between the surface string and its interpretation. In particular, if it is the case that the same logical form (*LF*) can be expressed by multiple surface strings (*S*), then we cannot assume independence between the the surface string and its interpretation, since $P(LF|S) \propto P(S|LF)P(LF) \neq P(LF)$ unless $P(S|LF) = 1$. Related to this is the following point: We have seen in the previous chapter that models of word-word dependencies can perform particularly well. The models of the previous chapter capture only dependencies between pairs of words. However, in the semantic interpretation, one word can depend on several other words, inducing a DAG-like structure. As shown by Abney (1997), such structures can easily lead to inconsistent (= deficient) models. Therefore, we first chose to investigate simpler models. Chapter 6 proposes a model that captures the word-word dependencies in the predicate-argument structure but does not suffer from the problems highlighted by Abney.

The next argument is more practical, in that it addresses the question of how to build an efficient wide-coverage parser for CCG. As discussed in chapter 2, formalisms like CCG permit multiple syntactic derivations of the same logical form from a given surface string. Wittenburg (1986) coined the term "spurious ambiguities" to refer to this problem of multiple, equivalent derivations. Therefore, a model of syntactic structure which treats semantically equivalent derivations as distinct objects might not seem an obvious choice. However, spurious ambiguity is a potentially serious problem for the efficiency of any parser, and appropriate search strategies need to be implemented to overcome this problem. Although the models presented in this chapter do assign non-zero probability to non-normal-form derivations, they will in practice always prefer derivations that are normal-form. Since these models can be incorporated directly into the parsing process, dynamic programming and a beam search can be used to make efficient parsing possible, even with a large lexicon and grammar.

The last argument reinforces the first, in that it is about CCG as a linguistic theory. A statistical model of parse selection that chooses analyses according to the plausibility of the derived structure (ie. the semantic interpretation), is in principle independent of the underlying grammar formalism. One of the aims of this research is to investigate how we can exploit the properties of the linguistic representation stipulated by CCG for parse selection. The information encoded in a CCG derivation is very different from that of a Penn-Treebank style parse tree. We have seen in section 4.5 of the previous chapter that state-of-the-art parsers for the Penn Treebank benefit from extraneous structural and grammatical features. This chapter demonstrates that a CCG parser can achieve a similar performance with a much simpler model, and argues that this is the case because in CCG much of the information captured by the extraneous features of parsers such as Collins (1999) is directly encoded in the linguistic representation.

5.3 A baseline model for CCG

This section defines a basic generative model over CCG derivation trees. This model does not incorporate the notion of combinatory rules. Instead, it is a general model over binary and unary trees, and it is only by virtue of the fact that it is trained on CCG derivation trees that it becomes a model of CCG.

Recall that in parse selection we are interested in finding the most likely parse τ^* for a given sentence α, according to the conditional probability $P(\tau|\alpha)$:

$$\tau^* = \arg\max_{\tau} P(\tau|\alpha)$$

In general, it is difficult to define the structure and to estimate the parameters of a model which conputes this probability directly for all sentences of the language. However, using Bayes Rule, this conditional probability can be expressed as follows:

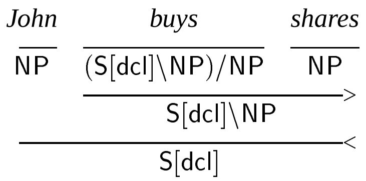
$$\begin{aligned}\tau^* &= \arg\ \max_{\tau} P(\tau|\alpha)\\ &= \arg\ \max_{\tau} \frac{P(\alpha|\tau)P(\tau)}{P(\alpha)}\\ &= \arg\ \max_{\tau} P(\alpha|\tau)P(\tau)\end{aligned}$$

If the parses τ are trees with yield α, $P(\alpha|\tau) = 1$, and what remains to be defined is a probability measure over such trees, $P(\tau)$. We have seen in the previous chapter that, since parse trees are complex, recursive objects, their probability is usually factorized into a product of probabilities of sub-events, such as the choice of head-words or rule applications. The sequence of these sub-events can be seen as the steps of a (stochastic) process which constructs such parse trees; therefore, such models are usually referred to as generative.

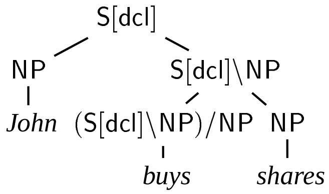
When defining generative models for natural language, the process is normally tied to the rules of an underlying grammar, as this guarantees that only strings generated by this grammar receive a non-zero probability. We will do the same, and define a generative process in terms of unary and binary branching trees corresponding to the local trees, or individual rule applications in a CCG derivations.

In standard expositions of categorial grammar, derivations are usually presented bottom-up:

(85)



However (as done already in chapter 3), derivations can also be presented top-down, emphasizing their tree structure:



In the following we will assume that CCG derivations are generated by a top-down process, starting at the root of the tree. This assumption forms the basis for all the models presented in this dissertation. Before investigating the impact of incorporating additional features such as bilexical dependencies or distance measures into the model, we describe a simple baseline model, which is defined in terms of probability distributions over local trees (trees of depth 1).

Leaf nodes	**Unary trees**	**Binary trees, head left:**	**Binary trees, head right:**
(S\NP)/NP **likes**	S/(S\NP) NP	S\NP S\NP/NP NP	S NP S\NP

Figure 5.1: The four different kinds of expansion

A node N with category C can expand in different ways: either it is a leaf node (in which case the generation process for this branch of the tree halts), or it has one or two daughters. Examples are given in figure 5.1. This is expressed by the **expansion probability** $P(exp \mid \mathsf{C})$. Since we want to extend the model to include lexical heads, we assume that a node with two children has one head daughter and one non-head daughter, and distinguish binary trees where the head daughter is on the left side from binary trees where the head daughter is on the right side:

(86) The expansion probabilities: $P(exp|\mathsf{C})$
with $exp \in \{left, right, unary, leaf\}$ and $\mathsf{C} \in \mathcal{C}$

In general, non-head daughters will be complements or adjuncts of the head daughter, or punctuation marks. However, this information does not need to be made explicit in the model, as it is captured by the CCG categories themselves.

If N is a leaf node ($exp = leaf$), a word w is generated (with the **lexical probability** $P(w|\mathsf{C}, exp = leaf)$):

(87) The lexical probabilities: $P(w|\mathsf{C}, exp = leaf)$
with $w \in \mathcal{T}$.

After the generation of a word, the generative process halts.

If N is not a leaf node, we have to generate its daughters. Instead of generating both daughters of a binary-branching tree simultaneously, we assume that the head daughter is generated first. This will allow us later on to condition features of the non-head daughter on properties of the head daughter. Thus, we first generate the category of the head daughter H with the **head probability** $P(\mathsf{H}|\mathsf{C}, exp)$. If N has two daughters (exp is *left* or *right*), we also generate the category of the non-head daughter D (with the **non-head probability** $P(\mathsf{D}|\mathsf{C}, exp, \mathsf{H})$). Finally, H and D are expanded themselves.

To summarize, the model contains the following families of probability distributions:

Expansion probability $P(exp|\mathsf{C})$**:**

Possible values of *exp*: $\{leaf, unary, left, right\}$.

Head probability $P(\mathsf{H}|\mathsf{C}, exp)$**:**

The category H of the head daughter.

Only defined for $exp \neq leaf$

Non-head probability $P(\mathsf{D}|\mathsf{C}, exp, \mathsf{H})$**:**

The category D of the sister daughter.

Only defined for $exp = left$ and $exp = right$

Lexical probability $P(\mathbf{w}|\mathsf{C}, exp = leaf)$**:**

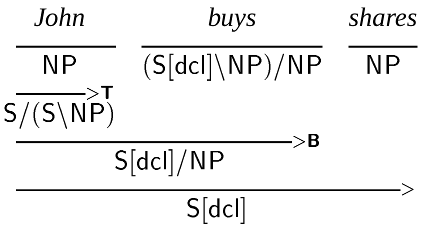
The word $\mathbf{w}$. Only defined for $exp = leaf$

The probability of a local tree is the product of the relevant probabilities, and the probability of a tree is the product of the probabilities of each local tree within this tree.

Note that this model does not explicitly contain the notion of combinatory rule schemata. Instead, it captures the likelihood of specific categories expanding according to specific instantiations of these schemata. In this manner, the model captures the constituent structure of "canonical" or normal-form derivations, where rules such as composition and type-raising are only used when required by syntax, and where the likelihood of these rules is therefore highly dependent on the specific categories involved.

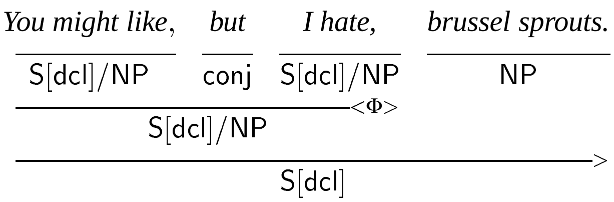
Does this model generate "canonical" CCG derivations? Because of CCG's treatment of extraction, even simple sentences, such as *John buys shares* can have multiple derivations; in addition to the normal form derivation, which uses only function application, the following is also allowed:

(88)



As the translation procedure described in chapter 3 maintains the original Treebank bracketing wherever possible, derivations in CCGbank correspond to normal form analyses which use function application wherever possible. Type-raising and composition, which are required for these “non-standard” analyses, are only used when required by syntax, ie. in the presence of extraction or so-called “non-constituent” coordination. Although these such constructions are not uncommon, they are not as frequent as “ordinary” sentences which can be dealt with using only function application. Therefore, a model estimated from CCGbank will in general always prefer a normal-form derivation which only uses composition and type-raising when necessary. Note, however, that it might assign some probability mass to derivations that are not normal-form. In the above (non-normal form) derivation, type-raising and composition are used to combine the subject with a transitive verb. Note that the probability model can only assign non-zero probability to this derivation if the training corpus contains at least one instance of the rule $\mathsf{S[dcl]} \rightarrow \mathsf{S[dcl]/NP}\ \mathsf{NP}$, and one instance of the rule $\mathsf{S[dcl]/NP} \rightarrow \mathsf{S/(S\backslash NP)}\ \mathsf{(S[dcl]\backslash NP)/NP}$. The second rule is fairly common, as it is required in object extraction relative clauses. But, given that the corpus contains only normal-form derivations, the first rule appears only if there is a right node raising construction at the sentential level, such as the following:

(89)

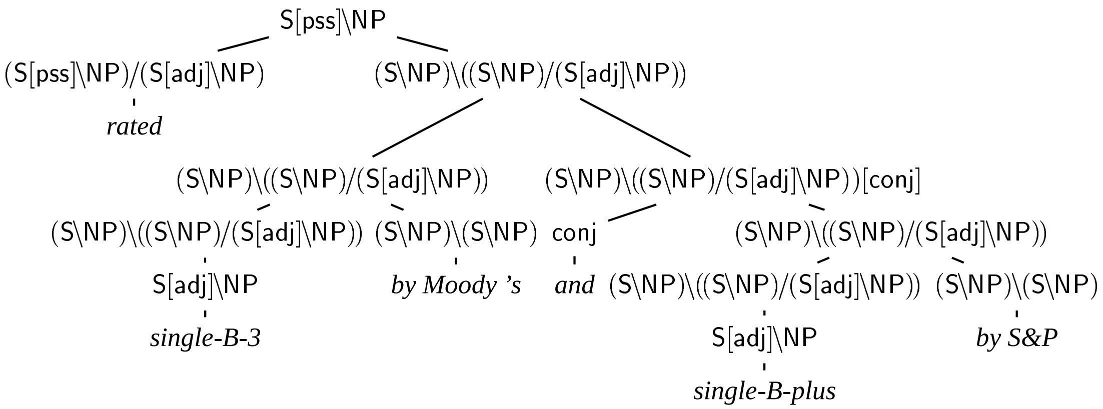


In order to guarantee that this rule is only applied in the case of right node raising, we would need to condition the expansion of a node on an additional feature indicating whether the head child contains a coordination or not. However, this particular construction is so rare that it does not even occur in the Wall Street Journal part of the Penn Treebank.

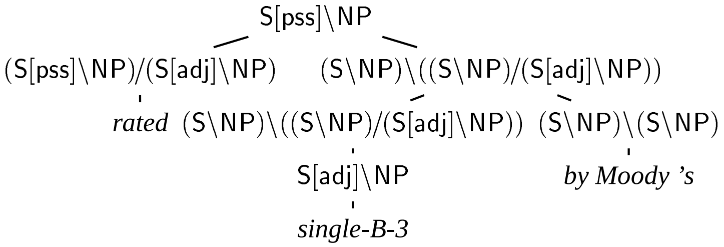
Similarly, the presence of argument cluster coordination and related constructions in the training corpus means that a non-zero probability is assigned to an analysis of transitive verbs where both arguments are combined with type-raising and composition to form a constituent, or to transitive verbs where the object is followed by a modifier:

(90) *The issue is rated single-B-3 by Moody’s and single-B-plus by S&P.*

The coordinate construction in this sentence has the following CCG analysis:


S[pss]\NP
(S[pss]\NP)/(S[adj]\NP)
(S\NP)\((S\NP)/(S[adj]\NP))
rated
(S\NP)\((S\NP)/(S[adj]\NP))
(S\NP)\((S\NP)/(S[adj]\NP))[conj]
(S\NP)\((S\NP)/(S[adj]\NP))
(S\NP)\(S\NP)
conj
(S\NP)\((S\NP)/(S[adj]\NP))
S[adj]\NP
by Moody 's
and
(S\NP)\((S\NP)/(S[adj]\NP))
(S\NP)\(S\NP)
single-B-3
S[adj]\NP
by S&P
single-B-plus

Note that the rule instantiations for this sentence can also produce the following analysis for the non-coordinate case:



Again, a feature indicating that the constituent with category (S\NP)\((S\NP)/(S[adj]\NP)) contains a coordinate construction would rule out (or at least strongly disfavour) this derivation. But since the overwhelming majority of sentences in the training corpus containing ditransitive verbs or transitive verbs with postverbal modifiers are analyzed with function application, the probability of this non-normal-form derivation will in general be vanishingly small.

5.4 Tightness of the model

As explained in section 4.3.2, a tight (or consistent) probability model over a language is a model which assigns a probability mass of 1 to the strings of the language, and does not assign any probability mass to infinite derivations or to strings that are not in the language. Models that are inconsistent are also sometimes referred to as deficient. The tightness of PCFGs and STAGs can easily be checked, since both have been shown to be instances of Galton-Watson branching processes (Booth and Thompson, 1973; Sarkar, 1998). Furthermore, is is known that PCFGs estimated from labelled or unlabelled data are tight (Chi and Geman, 1998). This section shows that the model defined above is also an instance of a Galton-Watson branching process, and provides similar checks for tightness.

In order to show that probabilistic CCG is a Galton-Watson process, the following conditions have to be met:

1. The set of kinds of objects has to be finite.
2. The initial generation is given.
3. Each generation Z_i produces with some probability a following generation Z_{i+1}.
4. This probability is time-invariant (not dependent on i).
5. What children are born to one object does not depend on what happens to the object's sisters

In the case of the baseline model, the kinds of objects are nodes labelled with categories – therefore there are as many kinds of objects as there are category types in G. Since we only deal with probability models estimated from labelled corpora here, we assume that the set of category types of G is finite. The extensions of the baseline model defined in section 5.8 assume larger, but still finite, sets of kinds of objects, since the additional features used by these models can only take a finite set of values.

Condition 2 states that the initial generation is given. This condition is trivial; we assume that the grammar has a designated start symbol TOP (which can expand to categories such as $\mathsf{S[dcl]}$, $\mathsf{S[q]}$, NP etc., depending on what has been found in the training corpus).

Condition 3 states that a probability distribution for the creation of a following generation can be defined. According to condition 4, this probability must not depend on i. Note that the ith generation in a given tree is the set of all nodes with distance i from the root node, plus the set of all leaf nodes with a distance smaller than i from the root. Generation Z_{i+1} is obtained from generation Z_i by generating children (but no further offspring) for all nonterminal nodes that were produced in generation Z_i. Since the sets of terminals and nonterminals are not disjoint in our grammar, we defined terminal nodes as nodes whose expansion $exp \neq leaf$. A node with category C is a terminal node with probability $P(exp = leaf|C)$. Therefore, the probability of generating Z_{i+1} from Z_i is given as the product of the probabilities of local tree expansions of all nodes generated in Z_i. These probabilities are given by the baseline model. They depend only on the category of the node itself, but not on the expansion of its siblings or its level in the tree – therefore conditions 4 and 5 are met.

Booth and Thompson (1973) define a stochastic expectation matrix E of size $|V|$, whose elements e_{ij} represent the expected number of occurrences of the jth nonterminal in direct expansions of the *i*th nonterminal, and show that a PCFG is tight if the largest eigenvalue of E, ρ_j, is < 1.

According to the baseline model described above, the expected number of occurrences of category A_j in direct expansions of a node with category A_i is as follows:

$$\begin{aligned} e_{ij} = \; & P_{exp}(left|H=A_i)P_{head}(H=A_j|A_i,left) \\ & + P_{exp}(left|H=A_i)\sum_k P_{head}(H=A_k|A_i,left)P(S=A_j|P=A_i,exp=left,H=A_k) \\ & + P_{exp}(right|H=A_i)P_{head}(H=A_j|A_i,right) \\ & + P_{exp}(right|H=A_i)\sum_k P_{head}(H=A_k|A_i,right)P(S=A_j|P=A_i,exp=right,H=A_k) \\ & + P_{exp}(unary|H=A_i)P_{head}(H=A_j|A_i,unary) \end{aligned}$$

This defines the expectation matrix E. If the absolute value of the largest eigenvalue of E is < 1, the model is tight.

5.5 Evaluating a CCG parser

Before presenting experimental results, we consider the question of how to evaluate a CCG parser. We previously discussed this in Hockenmaier (2001) and Clark and Hockenmaier (2002).

In general, two types of evaluation metrics for parsing are distinguished: constituent-based measures such as Parseval (Black *et al.*, 1991), or dependency-based measures, as proposed by Lin (1998) or Carroll *et al.* (1998) and Carroll *et al.* (1999), among others.

Since CCG produces unary and binary branching trees with a very fine-grained category set, CCG Parseval scores cannot be compared with scores of standard Treebank parsers. Measures such as crossing brackets penalize binary trees, which have many more bracketings than flat Treebank trees. Furthermore, the grammar underlying the Penn Treebank has only 48 POS tags and 14 non-terminal labels (excluding null elements) (Marcus *et al.*, 1993), whereas our grammar has over 1200 lexical category types. Therefore, our Parseval scores are not directly comparable to those of parsers which aim to reconstruct the original Treebank structures.

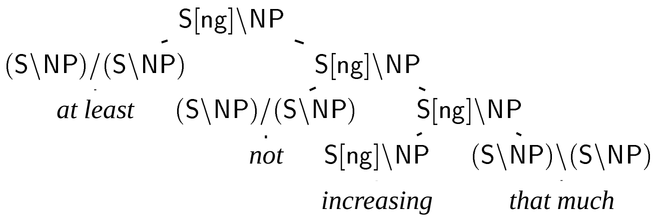
Despite the use of a normal-form grammar, there are situations where there are different binary bracketings. Leaving aside questions of semantic scope, these are equivalent, and it is only by chance that the model picks the one which occurs in the reference.[2] Consider the following example:

[2] Note that among the models presented in this chapter, only the distance models, described in section 5.8.1, are

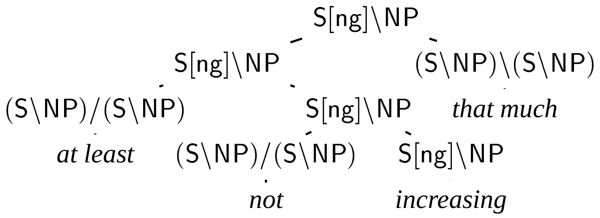
(91)
```
(VP (ADVP (IN at) (JJS least))
    (RB not)
    (VBG increasing)
    (ADVP (RB that) (RB much)))
```

There are three different, equivalent categorial derivations for the same string:

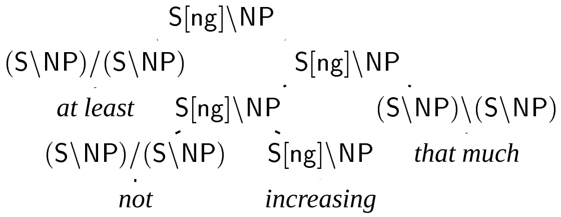
(92) a.



b.



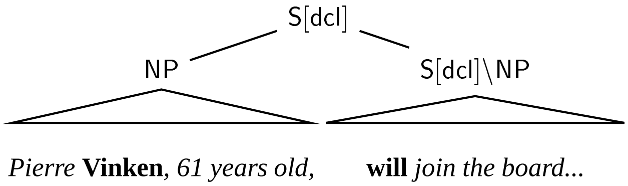
c.



The gold standard contains the first derivation. The other two trees would obtain precision and recall scores of only 33.3%, although they are assigned the same probability mass according to the present model. There are different ways in which this problem could be dealt with: we could incorporate a distance measure which favours the canonical bracketing (similar to the distance measure used by Collins (1999), which favours right-branching trees, but for binary trees). Another solution might be to produce flat trees (eg. by merging equivalent binary branchings) for which this problem does not arise. Or we can use an evaluation metric which does not penalize these spurious ambiguities. Collins (1999) evaluates the attachment accuracy of his parser by examining the recovery of word-word dependencies as defined by the parse trees. In this approach, a dependency relation *rel* is defined as a quadruple $\langle P, H, S, dir \rangle$, where P, H and S are the labels of the parent node, the head daughter, and the non-head daughter in a local tree, and *dir* indicates whether the non-head daughter is to the left or to the right of the

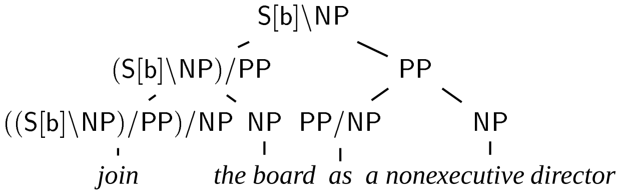
capable of capturing a difference between the three derivations.

head. $rel(w_i, w_j, dir)$ holds if w_i is the head word of H and w_j the head word of S, and S is to the dir side of H. We adapt this evaluation technique to our purpose. Given that in CCG local derivation trees are constrained by the combinatory rules, the directional parameter is not required. Hence, a local tree with parent node P, head daughter H and non-head daughter S (and position of S relative to P, ie. left or right, which is implicit in CCG categories) defines a $\langle P, H, S \rangle$ dependency between the head word of S, w_S, and the head word of H, w_H. In the following example, $\langle P, H, S \rangle = \langle \mathsf{S[dcl]}, \mathsf{S[dcl]\backslash NP}, \mathsf{NP} \rangle$, h_H =*will*, h_S =*Vinken*:

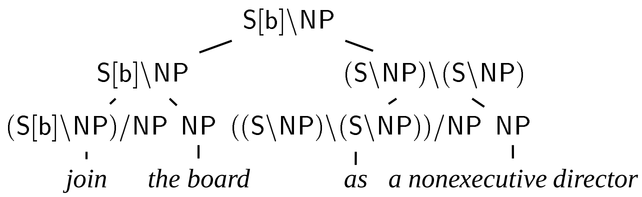

S[dcl]
NP
S[dcl]\NP
Pierre **Vinken**, *61 years old,*
will *join the board...*

A binary tree spanning a string of n words has $n-1$ non-leaf nodes, each of which determines one dependency relation. There is an additional relation $head(w_i)$ for the sentential head. Although not fully equivalent, we can therefore view this dependency evaluation measure on our binary trees as a metric which is neutral with respect to equivalent binary branchings.

However, the $\langle P, H, S \rangle$ labelled dependencies we report are also not directly comparable with Collins (1999): CCG categories encode subcategorization information, and are therefore more specific than the labelled dependencies for the Collins parser. This means that for a CCG derivation to be correct, the parser has also to identify for each constituent whether it is a complement or an adjunct. The following example demonstrates how mistaking the role of just one constituent can lead to a number of "errors" according to labelled Parseval and the labelled dependency evaluation $\langle P, H, S \rangle$. Assume that the gold standard contains the following derivation:

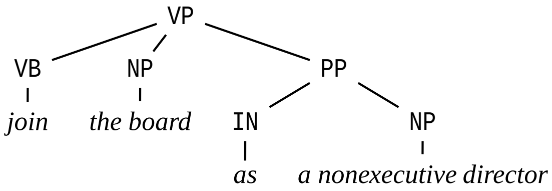


Here, the PP *as a nonexecutive director* is analyzed as an argument of *join*. If the parser analyzes this PP as an adjunct (which, in this case, is possibly a better analysis), it returns the following tree:



Arguably, this analysis contains just a single error. However, according to the fully labelled evaluation $\langle P,H,S\rangle$, none of the dependencies (with the exception of the identification of *join* as the head) in this tree are correct. By taking only the correctness of the non-head daughter label into account ($\langle S\rangle$), two points can be awarded for the dependencies between *join* and the NP *board* and between *as* and the NP *director*. According to labelled Parseval, this analysis would also only receive precision and recall scores of 40% (since two out of five nonterminal labels are correct).

By contrast, this kind of error is not reflected in the score for a Treebank parser. If only the correctness of the labels themselves is considered, a full score is awarded regardless of whether the PP is a complement or adjunct:



Although Collins provides an evaluation where he takes the correctness of the complement-adjunct distinction into account, this is also not comparable with our scores, since mistaking a complement for an adjunct or vice-versa results only in one, local, error for his parser, whereas the previous example demonstrated that it can be reflected in several errors in our parser. Also, Collins replaces all POS tags with the same token for the labelled dependency evaluation, so that tagging errors are not taken into account.

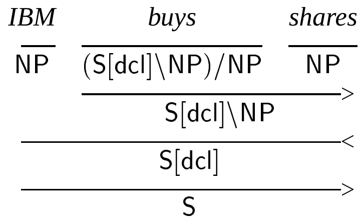
Therefore, only the unlabelled dependency evaluation $\langle\rangle$ (where it only matters whether word *a* is a dependent of word *b*, not what the labels of the nodes of the local tree are which define this dependency), scores can be compared across grammars with different sets of labels and different kinds of trees.

For our parser, the reported figures are also deflated by retaining verb features (see section 3.4). If this is done (by stripping off all verb features), an improvement of 0.4% on the $\langle P,H,S\rangle$ score for our best model is obtained.

Section 2.6 described how predicate-argument structure is represented in CCG, and the parser can also be evaluated according to how well it recovers the predicate-argument structure of unseen sentences. Like Clark *et al.* (2002), we measure this in terms of the recovery of the word-word dependencies that determine the predicate-argument structure. This measure allows us to evaluate how well the parser recovers dependencies that arise through raising, control, extraction, right node raising in addition to the local dependencies which are (mostly) captured by the Collins-style evaluation metric. We cannot compare our results directly with the evaluation given by Clark *et al.* (2002), since these experiments used an earlier version of the corpus and restricted the lexicon to frequent categories only, resulting in slightly lower coverage of the unseen data.

Like Clark *et al.* (2002), we define a word-word dependency in the predicate-argument structure in terms of the argument slots of lexical categories. Given a complex functor category, we number the arguments from 1 to *n*, starting at the innermost argument: $(\mathsf{S[dcl]}\backslash\mathsf{NP}_1)/\mathsf{NP}_2$. For instance, in the following example, *IBM* fills argument slot 1 of the lexical category $(\mathsf{S[dcl]}\backslash\mathsf{NP})/\mathsf{NP}$ of *buys*. This dependency relation can be denoted as the tuple $\langle \mathit{buys}, (\mathsf{S[dcl]}\backslash\mathsf{NP}_1)/\mathsf{NP}_2, 2, \mathit{shares}\rangle$:

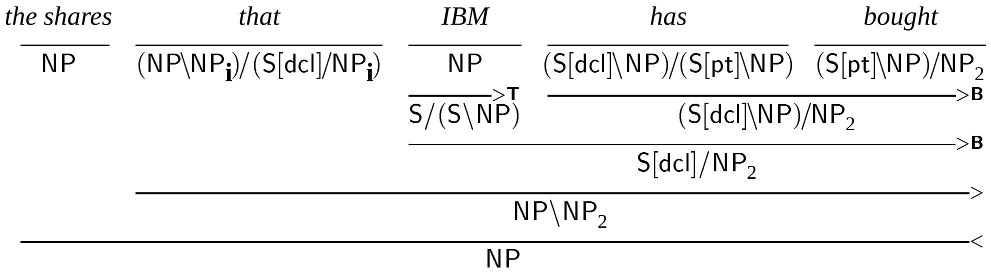
(93)



When reporting unlabelled precision and recall for dependencies in the predicate-argument structure, we also follow Clark *et al.* (2002) and only take into account whether there is a relation between two words. Here we allow the functor-argument relation between the words to be reversed, so that the parser is not penalized for misanalyzing a complement as an adjunct or vice versa.

Long-range dependencies can be projected through lexical categories. One example of a lexical category which projects an unbounded long-range dependency is that of a relative pronoun, $(\mathsf{NP}\backslash\mathsf{NP})/(\mathsf{S[dcl]}/\mathsf{NP})$. The extracted object can be co-indexed with the NP-argument in order to indicate their identity:

(94)



Thus, this example defines the following long-range dependency: ⟨*bought*, $(\mathsf{S[pt]}\backslash\mathsf{NP}_1)/\mathsf{NP}_2$, 2, *shares*⟩. The lexical categories which project long-range dependencies include wh-words, auxiliary, modal, control and raising verbs, *tough*-adjectives. We distinguish locally mediated (bounded) dependencies such as those expressed by raising and control verbs from unbounded long-range dependencies like those arising through object extraction, right node raising and tough movement. A detailed description of our classification of dependencies and of the way in which long-range dependencies are captured in our parser can be found in section 2.6.7.

5.6 Experimental setup

All the experiments reported in the following section were conducted using sections 02-21 of CCGbank as training corpus, and section 23 as test corpus. The gold standard test corpus against which the parser is evaluated has not been manually corrected. For unknown words, we do not take the standard approach of replacing all words in the training data which appear with a frequency below a predefined threshold with one token “UNKNOWN”, since this approach is impractical for a CCG parser with over 1200 lexical categories. Instead we replace all rare words in the training data with their POS tag. Unless indicated otherwise, the frequency threshold was set to $\leq$5. Like Collins (1999), we assume that the test data is POS-tagged, and can therefore replace unknown words in the test data with their POS tag, which is more appropriate for a formalism like CCG with a large set of lexical categories than one generic token for all unknown words. In most experiments we assume perfectly tagged input; however we also investigate the impact of tagging errors by testing our best model on input which was tagged by the POS tagger of Ratnaparkhi (1996).

We use a standard CKY chart parsing algorithm for probabilistic grammars. Dynamic programming and effective beam search strategies are essential to guarantee efficient parsing with highly ambiguous wide-coverage grammars. Both are used by our parser.

Dynamic programming exploits the fact that the probability model defines an equivalence relation between constituents, so that only of two equivalent constituents with the same span only the one with the higher inside probability needs to be retained, as the probability of the other constituent will always be lower. Note that each probability model defines different equivalence classes of constituents – for instance, in the baseline model, all constituents with the same category are equivalent, whereas in the models which take lexical heads into account, constituents are only equivalent if they have the same category and the same lexical heads.

Beam search strategies such as those advocated by Caraballo and Charniak (1998) or Collins (1999), go further than this in that they aim to identify the constituents within a cell that are most likely to lead to a high probability parse. However, the inside probability alone is not sufficient to predict this; therefore, the "figure of merit" which is used to compare constituents also includes the prior probability of a constituent. The constituent with the highest figure of merit within each cell is identified, and only constituents whose figure of merit lies within a certain fraction (the beam width) of the highest figure of merit within this cell are retained. Our parser uses a beam search as proposed by Collins (1999). The width of the beam was set to 0.0001.

5.7 The performance of the baseline model

The performance of the baseline model is shown in the top row of table 5.3. For six out of the 2401 sentences in our test corpus we do not get a parse. The reason is that a lexicon consisting of the word-category pairs observed in the training corpus does not contain all the entries required to parse the test corpus. We return to this problem in section 5.9, where a simple, but imperfect, solution is proposed, and in section 5.10. In comparison to Treebank PCFGs, our baseline model benefits from two factors – the lexical and non-local information encoded in CCG derivations. Experiments on unlexicalized PCFGs usually presume POS-tagged input, so that the terminals are the POS tags rather than the words themselves. By contrast, our baseline model can exploit the information given by the lexical probabilities $P(w|c)$ which capture subcategorization preferences of words. The row **Baseline (POS tags)** in table 5.3 gives the result of an experiment where the words in training and test data are replaced by their POS tags. Here, we obtain Parseval scores that are very similar to those of unlexicalized PCFGs estimated directly from the Penn Treebank, which have been given as 69.7% LP and 73.5% LR by Johnson (1998) (trained on sections 02-21, tested on section 22). As explained in section 5.5, Parseval is a harsher measure on the binary-branching categorial derivation trees of our grammar than on the flat trees of a CFG extracted from the Penn Treebank. This result seems to indicate that the additional structural information encoded in a CCG derivation is indeed useful for parsing. However, there are also a large number of parse failures. We are dealing with a much higher lexical ambiguity rate in this experiment than in the setup where words are used, and it is very likely that the high number parse failures is (at least in part) an artefact of the beam search.

We also report the results of an oracle experiment where words are assigned their correct lexical categories (**Baseline (LexCat oracle)**). The slightly higher number of parse failures in comparison to the ordinary Baseline experiment results from the fact that some of the lexical categories in the gold standard for the test data do not occur in the training data, and are therefore assigned zero probability by the model. This experiment indicates how much information is in the lexical categories alone, and what performance could be obtained if there was a way of guessing the correct lexical catgory for each word. Although no such oracle exists, the supertagger of Clark (2002) achieves an accuracy of 88.1% if it assigns one category to each word. Supertagging (Joshi and Srinivas, 1994) was originally proposed for Lexicalized Tree-Adjoining Grammars as a method of reducing the number of elementary trees assigned to each word by using techniques borrowed from part-of-speech tagging. This supertagger was also used in the parser of Clark *et al.* (2002) (discussed below), where it was run in a "multi-tagger" mode, returning on average 3.8 categories per word. Under this setting, it proposes the correct category for 98.4% of the words. This experiment shows that just by knowing the correct lexical category for each word, very high parsing accuracy can be obtained.

Since CCG categories are essentially unsaturated subcategorization frames, they encode a certain amount of non-local information which is lost in a PCFG estimated from the Penn Treebank. Johnson (1998) argues that a PCFG induced from a corpus with flat trees embodies weaker independence assumptions, and therefore has less bias than a PCFG estimated from a corpus with less flat trees. On the other hand, encoding non-local information in the labels also weakens the independence assumptions, and it seems that the non-local information represented in CCG categories more than compensates for the bias engendered by the binary-branching trees.

5.8 Extending the baseline model

State-of-the-art statistical parsers use many other features or conditioning variables, such as head words, subcategorization frames, distance measures and grandparent nodes. We too can extend the baseline model described in the previous section by including more features. Like the models of Goodman (1997b), the additional features in our model are generated probabilistically, whereas in the parser of Collins (1997) distance measures are assumed to be a function of the already generated structure and are not generated explicitly.

In order to estimate the conditional probabilities of our model, we recursively smooth empirical estimates $\hat{e}_i$ of specific conditional distributions with (possible smoothed) estimates of less specific distributions $\tilde{e}_{i-1}$, using linear interpolation:

$$\tilde{e}_i = \lambda \hat{e}_i + (1-\lambda)\tilde{e}_{i-1}$$

λ is a smoothing weight which depends on the particular distribution. λ is computed in the same way as Collins (1999, p.185), described here in section 4.5.5.

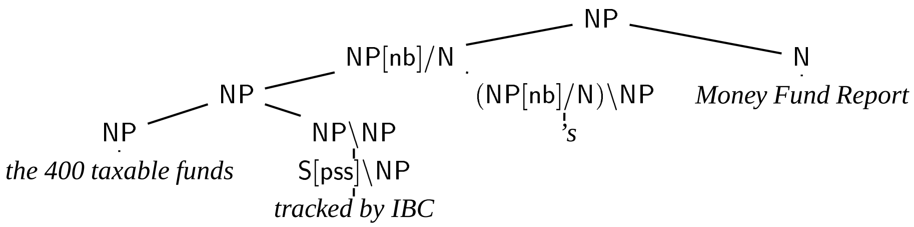
When defining models (eg. in table 5.1 and 5.2), we will indicate a backoff level with a # sign between conditioning variables, eg. A,B # C # D means that we interpolate the relative frequency estimate $\hat{P}(...|A,B,C,D)$ with $\tilde{P}(...|A,B,C)$, which in turn is an interpolation of $\hat{P}(...|A,B,C)$ and $\hat{P}(...|A,B)$ (both relative frequency estimates).

5.8.1 Adding non-lexical information

Features which represent certain non-local structural properties of the tree, such as the presence of coordination, the distance between the heads of constituents, or the label of the node above the local tree in question are commonly used, and are well known to improve the performance of standard Treebank parsers.

Although CCG categories encode some non-local information, a model of CCG which takes only local categories into account makes certain independence assumptions which are not warranted, and which can lead to incorrect analyses. Therefore additional features such as those found useful for Treebank parsers might prove beneficial as well. Section 5.3 has already argued for the importance of a feature indicating whether the head of a constituent is a coordinate construction or not, in order to guarantee that the model does not assign probability mass to non-normal form derivations. Furthermore, it is well known that the position of a constituent within a sentence influences how likely this constituent is to be modified. For instance, in the Penn Treebank, NPs in subject position are much more likely to be pronominalized than NPs in object position, whereas NPs in object position are much more likely to be modified by a prepositional phrase (Manning and Schütze, 1999, p.419ff.). However, in a model which takes only local trees into account, this difference cannot be captured. This is also true for a model of CCG derivations, since complement noun phrases always have the category NP, regardless of their position in the sentence. There are a number of other cases which also require non-local context. We use two different syntactic constructions, possessive *’s* and verb phrase modification, to demonstrate the kind of overgeneration that can occur if no non-local context is taken into account.

The possessive *’s* has the lexical category (NP[nb]/N)\NP in our grammar – it takes a noun phrase to its left to yield the category of a determiner (NP[nb]/N). The NP argument of the possessive is not very likely to be modified (unless, for instance, that modification is part of a name, as in *the Bank of America’s interest rates*). However, this information is not captured in the baseline model. Therefore, it can produce parses such as the following:



This analysis highlights another problem: according to the baseline model, this tree has the same probability as the correct analysis, since both use the same rules. Therefore, the baseline model does not only assign different probabilities to semantically equivalent derivations (all models proposed in this dissertation have this property), but it can also assign the same probability to semantically very different derivations. In principle, unlexicalized PCFGs can also suffer from this problem – but we would expect that PCFGs extracted from the Penn Treebank suffer from it to a lesser extent since the underlying tree representation is so flat. For instance, the possessive *’s* appears within the NP which according to the CCG analysis is its argument:

```
(NP-LGS (NP (NNP IBC) (POS ’s))
       (NNP Money) (NNP Fund) (NNP Report) ))))))
```

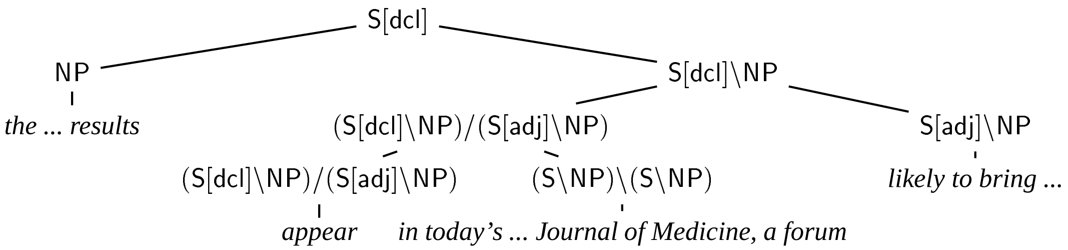
In English, a VP modifier can usually only appear between a verb and a direct object if the object noun phrase is “heavy”, as in the following example (see also section 3.9.10 in the previous chapter):

(95) *It has to weigh, on the other hand, the relatively high price of sugar it can earn on the export market*

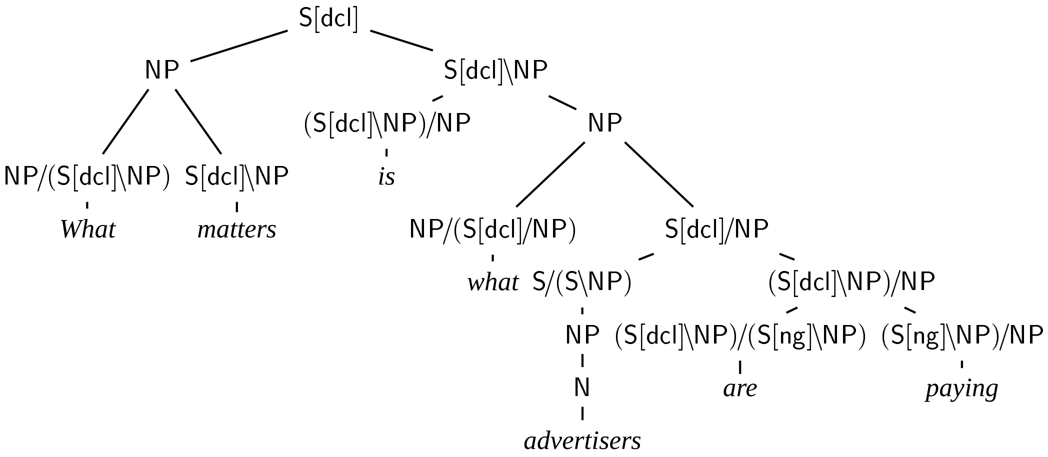
However, without a feature indicating the “heaviness” or length or the object, the parser can erroneously produce similar analyses in other cases as well. Consider the following sentence:

(96) *the latest results appear in today's New England Journal of Medicine,*
a forum likely to bring new attention to the problem

The baseline model analyzes *likely* as argument of *appears*, and the intervening PP as a modifier of the verb phrase:



A similar overgeneration in the grammar is caused by object relative clauses with VP modifiers:



This tree contains the following instance of backward crossing composition:

$$(S[ng]\backslash NP)/NP \quad (S\backslash NP)\backslash(S\backslash NP) \Rightarrow (S[ng]\backslash NP)/NP$$

If there are no non-local features in the probability model, this rule can be applied regardless of whether the object is extracted or not.

As mentioned in section 3.9.10, Steedman (1996, 2000) argues that, in English, backward crossing composition has to be restricted by a boolean feature *SHIFT*, which indicates whether an argument can undergo leftward extraction and heavy NP shift:

(97) *Backward Crossing Composition*:

$$Y/Z_{+SHIFT} : g \quad X\backslash Y : f \quad \Rightarrow_{\mathbf{B}} \quad X/Z_{+SHIFT}\lambda x. f(g(x))$$

	Expansion $P(exp\|\ldots)$	HeadCat $P(H\|\ldots)$	NonHeadCat $P(S\|\ldots)$
Baseline	P	P, exp	P, exp, H
+ Conj	$P, conj_P$	$P, exp, conj_P$	$P, exp, H, conj_P$
+ Grandparent	P, GP	P, GP, exp	P, GP, exp, H
+ Δ	$P\#\Delta_P^{L,R}$	$P, exp\#\Delta_P^{L,R}$	$P, exp, H\#\Delta_P^{L,R}$

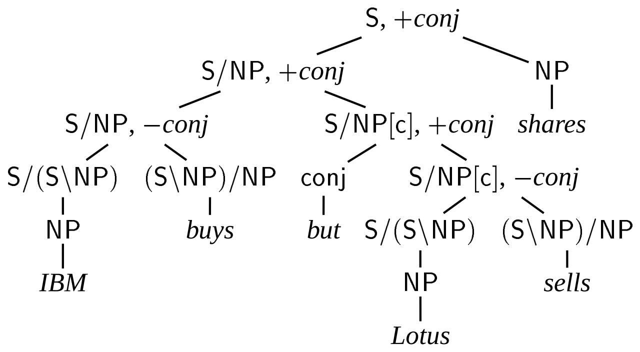
Table 5.1: The structure of the unlexicalized models

Our grammar and models currently lack such a feature, or a probabilistic equivalent thereof.[3] The function of this feature is to prevent specific instantiations of the rule schema for backward crossing composition which would lead to ungrammatical overgeneration. But this feature alone is insufficient to prevent misanalyses where "non-heavy" NPs are shifted, since the acceptability of heavy NP shift is not a question of binary-valued grammaticality, but of graded preference. Since the acceptability of heavy NP shift constructions depends on a structural property of the NP, namely its length or "heaviness", we argue that this ought to be captured by probabilistic features which indicate the length of constituents. As explained below, this is precisely what distance measures are intended for.

The remainder of this section investigates the impact of features which capture certain non-local structural properties of the derivation tree, such as distance measures, grandparent nodes and a coordination feature, on the performance of the CCG baseline model. Table 5.1 summarizes the structure of each of the models.

The coordination feature We define a boolean feature, *conj*, which is true for constituents which expand to coordinations on the head path. As explained in section 5.3, this feature can play an important role in ensuring that the parser only builds normal-form derivations. For instance, this features allows the model to capture the fact that, for a sentence without extraction or right node raising, a CCG derivation where the subject is type-raised and composed with the verb is very unlikely.

[3] Note that this is also true for the "lexicalized" models, which condition the expansion of a node on the lexical category of its head, since they can only capture such a regularity if the Z is really an argument of the head of $\mathsf{Y/Z}$, which is not the case if $\mathsf{Y/Z}$ is the result of composition, such as $\mathsf{Y/W}\ \ \mathsf{W/Z} \Rightarrow \mathsf{Y/Z}$.



This feature is generated at the root of the sentence with $P(conj|\text{TOP})$. For binary expansions, $conj_H$ is condtioned on H, S and $conj_P$:

$$P(conj_H|H,S,conj_P)$$

$conj_S$ is conditioned on S, P, exp_P, H, and $conj_P$. This is interpolated with $P(conj_S|S)$:

$$P(conj_S|S\ \#\ P,exp_P,H,conj_P)$$

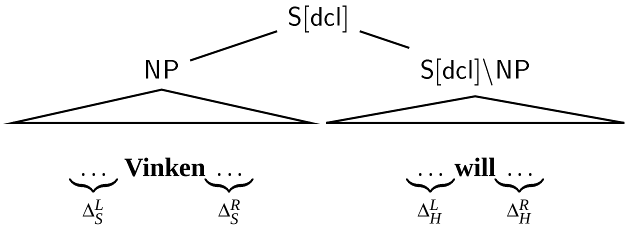
Table 5.1 shows how *conj* is used as a conditioning variable.

The grandparent feature As described in the previous chapter, Johnson (1998) showed that a PCFG estimated from a version of the Penn Treebank in which the label of a node's parent is attached to the node's own label yields a substantial improvement (LP/LR: from 73.5%/69.7% to 80.0%/79.2%). Charniak (1999) reports that the inclusion of an additional grandparent feature gives a slight improvement in his Maximum-Entropy inspired model – however, it results in a slight decrease in performance for a model that is based on relative frequency estimates. For simplicity's sake, we investigate here the impact of a grammar transformation like Johnson's, not that of a grandparent feature. Table 5.3 (**Grandparent**) shows that a grammar transformation like Johnson's does yield an improvement, but not as dramatic as in the Treebank-CFG case. At the same time coverage is reduced (which might not be the case if this was an additional feature in the model rather than a change in the representation of the categories). Both of these results are to be expected— since our category set is much larger, appending the parent node will lead to an even more fine-grained partitioning of the data, which then results in sparse data problems. Also, CCG categories encode more contextual information than Treebank labels, in particular about parents and grandparents; therefore the history feature might be expected to have less impact. On the other hand, there is some non-local information which is not directly represented in a CCG derivation, but can be captured by a grandparent feature in

Johnson's models, such as different distributions of noun phrases in subject of object position. However, in our models, this information might not be captured by a grandparent feature to the same extent: given that CCG derivation trees are binary-branching, a grandparent feature could only influence the probabilities of the first modifier of a constituent, but not those of subsequent modifiers. A more suitable feature for CCG might therefore be the category of the parent of the maximal projection of a constituent, but not the immediate grandparent.

Distance measures for CCG Features which measure the distance between head words of constituents have been shown to improve the performance of Penn Treebank parsers such as Collins (1996), Collins (1997) and Goodman (1997b), among others. These distance measures are a simple way of encoding non-local information: for instance, they do not only approximate subcategorization information, but can also bias the model toward the right-branching structure of English (Collins, 1999).

Our distance measures are related to those proposed by Goodman (1997b), which are appropriate for binary trees (unlike those of Collins (1997)). Every node has a left distance measure, Δ^L, measuring the distance from the head word to the left frontier of the constituent. There is a similar right distance measure Δ^R:



We implemented three different ways of measuring distance: $\Delta_{\text{Adjacency}}$ measures string adjacency (0, 1 or 2 and more intervening words); Δ_{Verb} counts intervening verbs (0 or 1 and more); and Δ_{Pct} counts intervening punctuation marks (0, 1, 2 or 3 and more). These Δs are generated by the model in the following manner: at the root of the sentence, generate the left distance measure Δ^L first (this is an arbitrary choice) with $P(\Delta^L|\text{TOP})$, and then the right distance measure Δ^R with $P(\Delta^R|\text{TOP},\Delta^L)$. Otherwise, if a node has only one daughter, the distance measure does not change. Therefore, $\Delta^L_H = \Delta^L_P$ and $\Delta^R_H = \Delta^R_P$ with a probability of 1. If a node has two daughters, the surface string can only grow in the direction of the non-head daughter. Hence, for the head child, only the Δ in the direction of its sister changes, with a probability of $P(\Delta^L_H|\Delta^L_P H\#P,S)$ if $exp = \texttt{right}$, and analogously for $exp = \texttt{left}$. However, we do not know at which position in the surface string the head word of S appears. So Δ^L_S

and Δ_S^R are conditioned on S and the Δ of H and P in the direction of S: $P(\Delta_S^L|S\#\Delta_P^R,\Delta_H^R)$ and $P(\Delta_S^R|S,\Delta_S^L\#\Delta_P^R,\Delta_H^R)$. The distance measures are then used as additional conditioning variables for the other distributions as shown in table 5.1.

These distance measures are slightly different from those used in Collins (1997). In Collins' models, the head child of a node is generated first, followed by its left and right siblings. The left and right siblings of the head are generated so that those immediately adjacent to the head are generated first. If we assume that constituents are fully expanded before their siblings are generated, the distance measure between the head and a modifier can be considered a function of the already generated surface string. The models presented here do not make this assumption.

Table 5.3 gives the Parseval and dependency scores obtained with each of these measures.[4]

5.8.2 Adding lexical information

Next we investigate how additional lexical information influences the performance of the parser.

We assume that each constituent has one lexical head (an assumption which is semantically inaccurate in the case of coordinate constructions), and that a lexical head consists of a word and its lexical category. In the models of Collins (1997) and Charniak (2000), lexical heads are word–POS tag pairs. We could in principle include POS tag information, but refrain to do so here, since we aim to investigate features that are directly represented in the CCG derivation.

The information associated with words can influence parse selection in two ways: on the one hand, certain words are more likely to occur with certain syntactic structures than others, and on the other hand, certain syntactic structures can be disfavoured because they represent unlikely relationships between words (so-called bi-lexical dependencies). However, Gildea (2001) shows that removing the bi-lexical dependencies in Model 1 of Collins (1997) (that is, not conditioning on w_h when generating w_s) decreases labelled precision and recall by only 0.5%. It can therefore be assumed that the main influence of lexical head features (words and preterminals) in Collins' Model 1 is on the structural probabilities.

Recall that a lexical category in CCG specifies the subcategorization frame, or valency, of a word. Lexical categories encode more information about the expansion of a nonterminal than Treebank POS tags and are therefore more constraining. To some extent, similar information is represented in Model 2 of Collins (1997), where additional features are required in order to

[4] Our results for Δ_{Pct} are different from Hockenmaier and Steedman (2002b). There, measuring distance in terms of punctuation marks, Δ_{Pct}, had a small, but positive effect.

	Expansion	HeadCat	NonHeadCat	LexCat		Head word	
	$P(exp\|\ldots)$	$P(H\|\ldots)$	$P(S\|\ldots)$	$P(c_S\|\ldots)$	$P(c_{\text{TOP}}\|\ldots)$	$P(w_S\|\ldots)$	$P(w_{\text{TOP}}\|\ldots)$
LexCat	P, c_P	P, exp, c_P	$P, exp, H\#c_P$	$S\#H, exp, P$	$P = \text{TOP}$	–	–
+ Δ	$P, c_P\#\Delta_P^{L,R}$	$P, exp, c_P\#\Delta_P^{L,R}$	$P, exp, H\#(c_P), \Delta_P^{L,R}$	$S\#H, exp, P$	$P = \text{TOP}$	–	–
LexCatDep	P, c_P	P, exp, c_P	$P, exp, H\#c_P$	$S\#H, exp, P\#c_P$	$P = \text{TOP}$	–	–
HeadWord	$P, c_P\#w_P$	$P, exp, c_P\#w_P$	$P, exp, H\#c_P\#w_P$	$S\#H, exp, P$	$P = \text{TOP}$	c_S	c_P
HWDep	$P, c_P\#w_P$	$P, exp, c_P\#w_P$	$P, exp, H\#c_P\#w_P$	$S\#H, exp, P$	$P = \text{TOP}$	$c_S\#P, H, S, w_P$	c_P
HWDepΔ	$P, c_P\#\Delta_P^{L,R}\#w_P$	$P, exp, c_P\#\Delta_P^{L,R}\#w_P$	$P, exp, H\#\Delta_P^{L,R}\#c_P\#w_P$	$S\#H, exp, P$	$P = \text{TOP}$	$c_S\#P, H, S, w_P$	c_P
HWDepConj	$P, c_P, conj_P\#w_P$	$P, exp, c_P, conj_P\#w_P$	$P, exp, H, conj_P\#c_P\#w_P$	$S\#H, exp, P$	$P = \text{TOP}$	$c_S\#P, H, S, w_P$	c_P

Table 5.2: The structure of the lexicalized models

			Parseval				Surface deps.					Pred-arg. deps.			
Model	NoParse	LexCat	LP	LR	BP	BR	$\langle P,H,S\rangle$	$\langle S\rangle$	$\langle\rangle$	CM on $\langle\rangle$	$\leq$2 CD	LP	LR	UP	UR
Rare word threshold: $\leq$5															
Baseline	6	87.9	73.8	73.5	79.3	78.9	77.2	82.4	85.8	24.9	56.2	76.4	76.9	87.2	87.8
(POS tags)	102	82.5	69.1	68.6	75.8	75.2	70.1	77.0	81.8	24.4	49.2	66.9	68.3	82.3	83.4
(LexCat oracle)	19	100.0	88.5	88.0	89.2	88.8	93.8	94.3	94.8	52.8	84.6	95.2	96.1	95.2	96.1
Coordination	6	88.1	74.3	74.1	79.8	79.6	77.7	82.8	86.3	25.9	57.8	77.2	77.0	87.8	87.6
Grandparent	147	88.1	74.7	74.8	80.2	80.3	78.2	83.4	86.8	30.8	61.0	77.2	77.4	87.7	87.9
Δ_{Pct}	9	88.0	73.3	72.8	78.7	78.2	77.0	82.1	85.6	24.5	55.2	76.3	76.9	86.7	87.4
Δ_{Verb}	8	87.9	75.6	75.1	81.2	80.8	77.5	82.8	86.1	26.5	57.6	76.6	77.2	87.2	87.9
$\Delta_{\text{Adjacency}}$	8	88.6	77.3	77.3	82.7	82.7	79.4	84.3	87.7	25.7	62.3	78.5	78.4	88.6	88.5
LexCat	11	88.5	75.6	75.3	81.1	80.8	78.7	83.7	87.2	28.6	60.6	77.5	78.4	87.8	88.8
LexCatDep	10	88.5	75.7	75.4	81.2	80.9	78.7	83.7	87.1	28.3	59.9	77.6	78.4	87.9	88.7
LexCatDist	13	88.0	77.0	76.9	82.7	82.6	78.9	84.1	87.4	28.4	60.9	77.6	77.6	88.3	88.3
HeadWord	9	89.5	77.7	77.9	82.7	82.9	80.7	85.4	88.8	31.9	64.6	79.6	79.8	89.4	89.6
HWDep	9	91.8	81.2	81.6	85.2	85.6	84.2	88.0	90.6	38.9	71.6	83.5	83.7	91.2	91.4
HWDepΔ	9	90.6	80.4	80.9	85.0	85.5	82.7	86.6	89.5	33.0	67.3	82.3	82.2	90.5	90.3
HWDepConj	12	91.6	80.4	80.9	84.5	85.0	83.6	87.5	90.2	36.8	70.1	83.2	82.7	91.0	90.5
Rare word threshold: $\leq$30															
HWDep	4	92.2	82.2	82.4	86.2	86.4	85.1	88.9	91.4	40.1	73.2	84.3	84.6	91.8	92.2
+POS tagger	**4**	**91.5**	**81.1**	**81.4**	**85.3**	**85.6**	**83.9**	**87.9**	**90.5**	**38.9**	**70.8**	**83.1**	**83.5**	**91.1**	**91.5**

Table 5.3: Performance of the models (sec. 23): LexCat indicates accuracy of the lexical categories; the Parseval scores are not commensurate with other Treebank parsers. $\langle P,H,S\rangle$, $\langle S\rangle$, and $\langle\rangle$ are as defined in section 5.5. CM on $\langle\rangle$ is the percentage of sentences with complete match on $\langle\rangle$, and $\leq$2 CD is the percentage of sentences with under 2 "crossing dependencies" as defined by $\langle\rangle$. All scores are rounded to one digit after the decimal point, and are only computed on the sentences that are parsed.

capture subcategorization information, but not in Collins' Model 1.

The baseline model can be extended in several ways to include increasing amounts of lexical information. In order to capture lexical information, the lexical head (the lexical category and the head word) needs to be generated at the maximal projection of a constituent, so that the expansion of this constituent can be conditioned on the lexical head. Furthermore, we can choose to capture dependencies between the lexical heads of constituents by conditioning the lexical head of non-head daughters on the lexical head of the head daughter (and parent).

Table 5.2 presents details on the structure of each of the models.

Generating a constituent's lexical category c at its maximal projection (ie. either at the root of the tree, TOP, or when generating a non-head daughter S), and using the lexical category as conditioning variable for the expansion of this node (**LexCat**) increases performance of the baseline model as measured by $\langle P,H,S \rangle$ by 1.5%.[5] In this model, c_S, the lexical category of S depends on the category S and on the local tree in which S is generated. However, about the same performance is obtained for **LexCatDep**, a model which is identical to the original **LexCat** model, except that c_S is also conditioned on c_H, the lexical category of the head node, which introduces a dependency between the lexical categories. **LexCatDist** is like **LexCat**, but includes a distance measure ($\Delta_{Adjacency}$).

Since there is so much information in the lexical categories, one might expect that this would reduce the effect of conditioning the expansion of a constituent on its head word w. However, we did find a substantial effect. Generating the head word at the maximal projection (**HeadWord**) increases performance by a further 2%. Finally, conditioning w_S on w_H, hence including word-word dependencies, (**HWDep**) increases performance even more, by another 3.5%, or 7% overall. This is in stark contrast to Gildea's findings for Collins' Model 1.

We conjecture that the reason why CCG benefits more from word-word dependencies than Collins' Model 1 is that CCG allows a cleaner parametrization of these surface dependencies.

In Collins' Model 1, w_S is conditioned not only on the local tree $\langle P,H,S \rangle$, c_H and w_H, but also on the distance Δ between the head and the modifier to be generated. However, Model 1 does not incorporate the notion of subcategorization frames. Instead, the distance measure was found to yield a good, if imperfect, approximation to subcategorization information.

[5] In Hockenmaier and Steedman (2002b), this was almost 3%. In our current experiments, we obtained a higher performance of the baseline model and a lower performance of the LexCat model.

Using our notation, Collins' Model 1 generates w_S with the following probability:

$$\begin{aligned}&P_{Collins1}(w_S|c_S,\Delta,P,H,S,c_H,w_H) =\\ &\lambda_1\hat{P}(w_S|c_S,\Delta,P,H,S,c_H,w_H)\\ &+(1-\lambda_1)\left[\lambda_2\hat{P}(w_S|c_S,\Delta,P,H,S,c_H)+(1-\lambda_2)\hat{P}(w_S|c_S)\right]\end{aligned}$$

—whereas the CCG dependency model generates w_S as follows:

$$\begin{aligned}&P_{CCGdep}(w_S|c_S,P,H,S,c_H,w_H) =\\ &\lambda\hat{P}(w_S|c_S,P,H,S,c_H,w_H)+(1-\lambda)\hat{P}(w_S|c_S)\end{aligned}$$

Since our P, H, S and c_H are CCG categories, and hence encode subcategorization information, the local tree generally identifies a specific argument slot (but see chapter 6 for counter-examples). Therefore it is not necessary for us to include a distance measure in the dependency probabilities.

5.8.3 Combining lexical and non-lexical information

Although the dependency model performs very well, there is nothing in it (or in the other lexicalized models) which prevents the sort of overgeneration alluded to in section 5.8.1. Therefore, a model which included both lexical and structural features could be expected to attain even higher performance. However, when incorporating the adjacency distance measure or the coordination feature into the dependency model (**HWDepΔ** and **HWDepConj**), overall performance is lower than with the dependency model alone.

We conjecture that this arises from data sparseness. It cannot be concluded from these results alone that the lexical dependencies make structural information redundant or superfluous. Instead, it is quite likely that we are facing an estimation problem similar to Charniak (1999), who reports that the inclusion of the grandparent feature worsens the performance of a model consisting of linear interpolations of relative frequency estimates, but improves the performance of the model if the individual distributions are estimated using his "Maximum-Entropy inspired" technique. This intuition is strengthened by the fact that, on casual inspection of the scores for individual sentences, it is sometimes the case that the lexicalized models perform worse than the unlexicalized models. Note that a lexicalized CCG model is much more prone to suffer from a sparse data problem than the models of Collins (1997). In CCG, knowing the lexical category that a nonterminal node expands to constrains the space of immediate daugh-

Cutoff	5	10	20	30	40	80
Known words	11,755	7,179	4,245	3,047	2,419	1,313
Parse failures	8	6	5	3	3	4
PredArg LP	82.6	83.2	83.5	83.7	83.7	83.3
PredArg LR	83.0	83.5	83.7	84.1	84.0	83.7
PredArg UP	90.8	91.3	91.6	91.7	91.6	91.5
PredArg UR	91.2	91.6	91.8	92.0	92.0	92.0
$\langle\rangle$, All	90.5	91.0	91.3	91.5	91.5	91.4
$\langle P,H,S\rangle$, All	83.4	84.0	84.3	84.7	84.7	84.4
# sent. with missing entries	409	342	265	222	199	161
$\langle P,H,S\rangle$ on sent. with missing	71.9	72.6	71.9	72.7	73.0	73.8
# sent. without missing entries	1492	1561	1639	1684	1707	1744
$\langle P,H,S\rangle$, on sent. without missing	87.5	87.3	86.9	86.7	86.5	85.6

Table 5.4: The impact of the frequency cutoff on lexical coverage (sec. 00)

ters of that node very strongly. Therefore, we do not back off from the lexical head category in our lexicalized models. Section 5.10 returns to this question.

5.9 The lexical coverage problem

A CCG lexicon pairs words with very specific categories. In our current setup, the lexicon contains only those categories that have been observed with a particular word form in the training corpus. However, not all word forms that appear in the training corpus occur there with all their possible lexical categories. Therefore, a lexicon that consists only of the word-category pairs observed in the training corpus is incomplete. Sentences in the test corpus may contain words whose correct lexical categories (according to the gold standard against which the parser is evaluated) are not in the lexicon, even though the lexicon does contain other entries for these words. This means that the derivations that the gold standard gives for these sentences are not in the search space of the parser; hence the overall parsing performance is lower on these sentences.

Recall that words below a certain frequency threshold in the training corpus are replaced by their POS tag. In this manner, lexical entries for unknown words with the same POS tag are acquired, so that unknown words in the test data can be dealt with by replacing them with their POS tags. This assumes that rare words with the same POS tag have similar categories, and that the lexicon is complete for all words which appear in the training corpus with a frequency that is at least as high as the threshold. Since the lexical entries for unknown words are obtained from all rare words with the same POS tag, unknown words are much less likely to suffer from a lexical coverage problem than known words. Instead, there is a risk that these words suffer from the opposite problem of having too many lexical categories. For example, an unknown verb can have all lexical categories that have been observed with any rare verb in the training corpus. Therefore, pooling the counts of rare words in this manner can increase the lexical ambiguity rate. This may have two adverse effects: it can slow down parsing, and it can decrease the accuracy of the parser, since erroneous analyses are included in the search space. On the other hand, low-frequency words are more likely to cause a lexical coverage problem, and increasing the frequency threshold might alleviate this problem to some extent. Here we investigate the impact of the tradeoff between incomplete lexical coverage for low frequency "known" words and lexical overgeneration for "unknown" words by examining the impact of the frequency threshold on parse accuracy. In the experiments reported above, the frequency threshold was set to 5, in line with standard practice in statistical parsing for the Penn Treebank. Here we use the dependency model, **HWDep**, and change the frequency threshold from 5 to 80. Since the frequency cutoff is an adjustable parameter, we report our results on the development corpus. For each setting, table 5.4 gives the number of known words, the overall accuracy in terms of dependencies in predicate-argument structure and the overall accuracy in terms of the Collins-style $\langle$P,H,S$\rangle$ evaluation. Furthermore, the overall performance on those sentences for which all correct lexical categories are in the lexicon and on those sentences for which at least one correct lexical entry is missing is given.

These experiments show that a frequency threshold of 5 is indeed not optimal for CCG. All the other frequency thresholds investigated here yield slightly better results. Even a frequency threshold of 80, which results in a known vocabulary whose size is only 11% of that obtained with a threshold of 5, gives slightly better accuracy and fewer parse failures.

As mentioned above, increasing the frequency threshold results in a higher lexical ambiguity rate. Therefore, all of the experiments reported in table 5.4 use a number of heuristics (explained below) which exploit the constraints on syntactic structure imposed by punctuation marks in order to improve parsing efficiency.[6]

[6] We did not use these heuristics in the experiments reported in table 5.3.

Punctuation rules Punctuation marks such as semicolons, parentheses or dashes impose certain constraints on the constituent structure of a sentence that are not captured by the grammar nor the model. Knowing these constraints might not only improve the performance of the parser, it also improves parsing efficiency, since they can narrow down the search space considerably, in particular for long sentences.

We have implemented the following heuristics, which are all used in the experiments reported below:

- **Semicolons:** A constituent that starts with a semicolon has to end adjacent to another semicolon or has to go until the end of the sentence. This is intended to capture the fact that semicolons separate what could otherwise be complete sentences.

- **Dashes:** A constituent that starts with a dash has to go until (but not including) a sentence-final punctuation mark or semicolon, or otherwise it cannot span another dash. The first case captures parentheticals at the end of a sentence, whereas the second case ensures that parenthetical within a sentence which begin with a dash can only go until the next dash.

- **Colons:** A constituent that starts with a colon has to go until the last word of a sentence. Usually, colons mark the start of direct speech or of lists.

- **Parentheses:** A constituent that ends with a closing parenthesis cannot span an opening parenthesis. In our grammar, opening parentheses take what comes inside the parentheses as argument, but do not "subcategorize" for a closing parenthesis. Instead, closing parentheses appear as modifiers to the parenthetical. This rule is inteded to deal with this shortcoming in the grammar.

The parser of Collins (1999) benefits also from a special constraint for punctuation marks which narrows down the search space of the parser, but is not part of the model. This constraint states that if two children of a node are separated by a comma, then the last word of the right child must be directly followed by a comma or it must be the last word of the sentence (Collins, 1999, p.154). Collins argues that 96% of the commas in the training data follow this rule. But since we are dealing with binary branching trees, we cannot directly adopt this heuristic.

5.10 Finding the right bias

From the experiments reported in section 5.8, a number of positive conclusions can be drawn about the effectiveness of the additional features that were investigated: all structural features (three distance measures, a coordination feature, to some extent even the grandparent-transformation) have been demonstrated to improve the performance of our parser. Despite the fact that CCG represents a lot of very specific lexical knowledge in its categories, lexical features such as head words and their lexical categories, as well as bilexical dependencies, have been shown to yield an even greater improvement. All of these results point to different overgeneralizations in the baseline model. It seems that these features have all biased the model towards more linguistically plausible structures, without incurring an overly high variance which might harm the performance of the model.

However, some negative conclusions can also be drawn from these experimental results. As shown above, there is a lexical coverage problem, especially for low frequency words, which do not appear often enough in the training data to co-occur with all their possible lexical categories. Although we have demonstrated that, at least in theory, non-structural features are essential to avoid certain kinds of overgeneration and misanalysis, the performance of the dependency model degrades when such features are included. Furthermore, there is a slight variation in parse coverage between the different models.

All of these findings relate to *under*generalizations in the various models. The lexical coverage problem is an obvious example of undergeneralization, as not all $\langle c, w \rangle$ pairs that are in the lexicon of the language are assigned non-zero probability. The differences in parse coverage and the lower performance of the dependency model with structural features point to undergeneralization in the structural part of the model. Recall that in the lexicalized models, the lexical category of the head word is generated at the maximal projection of a constituent. This is similar to the generation of the POS tag of the head word at the maximal projection of a constituent in standard Treebank parsers. But whereas probability models for Treebank grammars do not need to condition the expansion of a nonterminal on the POS tag of its head word, and can therefore backoff to distributions that are conditioned on the nonterminal alone, we cannot make such an independence assumption. If we know the category of a constituent and the lexical category of the head of this constituent, the space of possible expansions of this category is very tightly constrained by the combinatory rules. Once we know that the head of a verb phrase is an intransitive verb $\mathsf{S[dcl]\backslash NP}$, we cannot generate a transitive verb $\mathsf{(S[dcl]\backslash NP)/NP}$ as its child anymore. The jump in performance between the baseline model and the **LexCat** model indicates (not surprisingly) that lexical categories are indeed very good predictors of syntactic structure. However, with 1200 lexical categories in our grammar, it should be obvious that the inclusion of lexical categories can also greatly exacerbate the sparse data problem for the estimation of rule probabilities.

What is required is a way of generalizing to classes of lexical categories such that the additional information is not lost, but more reliable estimates of rule probabilities can be obtained, even for rare lexical categories. For instance, one might want to explore ways of pooling the counts for lexical categories which share the same prefix (where the length of the shared prefix depends on the category of the node to be expanded).

In order to solve the lexical coverage problem, we have explored the very simple solution setting of the frequency threshold for rare words higher. This is relatively indiscriminate, and likely to lead to overly large lexicons. Since the ambiguity rate of a sentence depends to a large extent on the number of lexical categories for the words in the sentence, this can result in much slower parsing. On the other hand, we could use morphological knowledge to generalize from the observed word-category pairs to new lexical entries. For example, assume we have observed the 3rd person singular *buys* with (S[dcl]\NP)/NP. If we know English morphology, we also know the following entries:

(98) a. non-3rd-pers-sg *buy*: (S[dcl]\NP)/NP

b. infinitival *buy*: (S[b]\NP)/NP

c. present participle *buying*: (S[ng]\NP)/NP

d. past participle *bought*: (S[pt]\NP)/NP

e. passive *bought*: S[pass]\NP

A morphological analyzer such as Minnen *et al.* (2001) could be used to generate these word forms. Once these new lexical entries are generated, a technique such as Good-Turing smoothing could be used to estimate their probabilities.

The techniques for smoothing probabilistic lexicons proposed by Eisner (2001) are also highly relevant to this problem, and future work should investigate how they could be adapted to our purposes.

5.11 The impact of tagging errors

All of the experiments described above use the POS tags as given by CCGbank (which are the Treebank tags, with some corrections necessary to acquire correct features on categories).

It is reasonable to assume that this input is of higher quality than can be produced by a POS tagger. We therefore ran the dependency model on a test corpus tagged with the POS tagger of Ratnaparkhi (1996), which is trained on the original Penn Treebank (see **HWDep (+tagger, cutoff:30)** in Table 5.3). [7]

5.12 Analysis

This section takes a closer look at the results, examining the performance of **HWDep** with a cutoff of 30 (using the punctuation rules described above) on the development set.[8] We consider the accuracy of lexical categories and the accuracy of the dependencies in the predicate-argument structure. This allows us to assess the overall performance on local and long-range dependencies. Finally, we examine specific constructions that involve coordination and extraction. These are the constructions where the analyses that CCG can provide are arguably more satisfying than those given by a Treebank parser.

5.12.1 The accuracy of lexical categories

Table 5.5 gives precision and recall scores of the 30 most frequent lexical categories for the dependency model on the development corpus. Not surprisingly, prepositions (in particular the distinction between verb-phrase adjuncts $(\mathsf{(S\backslash NP)\backslash(S\backslash NP)})/\mathsf{NP}$ and complements $\mathsf{PP/NP}$) are hard. The confusion matrix in table 5.6 indicates the most common errors for the most frequent categories for which recall was below 90%.

5.12.2 Recovery of dependencies in predicate-argument structure

Table 5.7 gives the overall performance of **HWDep** on section 23 (Cutoff:30, with POS tagger) as measured by the recovery of word-word dependencies in the predicate argument structure. Non-local dependencies are harder than local dependencies, and unbounded long-range dependencies are harder than locally mediated dependencies. This is not surprising, because in order

[7] In Hockenmaier and Steedman (2002b), we found that with a cutoff of 5, the difference in performance between using a POS tagger and the POS tags of CCGbank was less pronounced, presumably because tagging errors matter less with a lower frequency cutoff.

[8] Unless indicated otherwise, the POS tags of CCGbank were used. However, when we compare our results directly with the performance of other parsers, results of experiments that use POS-tagged input are given in order to allow a fair comparison

Category	#ref	#test	P	R
N	10,498	10,702	96.5	98.4
N/N	7,212	7,190	97.2	96.9
NP[nb]/N	4,077	4,100	98.6	99.1
(NP\NP)/NP	2,123	2,099	89.7	88.7
((S\NP)\(S\NP))/NP	1,044	1,024	72.9	71.5
NP	975	975	98.4	98.4
conj	940	941	98.6	98.7
(S[dcl]\NP)/NP	867	842	89.3	86.7
PP/NP	808	930	69.9	80.5
(S\NP)\(S\NP)	745	736	85.7	84.7
(S[b]\NP)/NP	601	591	90.7	89.2
(S[to]\NP)/(S[b]\NP)	587	587	98.8	98.8
(S[dcl]\NP)/(S[b]\NP)	473	475	99.4	99.8
(NP[nb]/N)\NP	372	380	97.6	99.7
S[pss]\NP	356	345	90.4	87.6
S[adj]\NP	343	339	75.2	74.3
(S[dcl]\NP)/S[dcl]	337	331	97.0	95.3
(S\NP)/(S\NP)	299	296	89.9	89.0
(S[dcl]\NP)/(S[pt]\NP)	296	296	97.6	97.6
(S[ng]\NP)/NP	285	281	87.2	86.0
(NP\NP)/(S[dcl]\NP)	260	256	98.4	96.9
(S/S)/NP	253	251	94.0	93.3
S/S	240	236	92.8	91.3
S[dcl]\NP	232	220	75.5	71.6
(S[dcl]\NP)/(S[pss]\NP)	226	224	94.6	93.8
(N/N)/(N/N)	215	214	81.3	80.9
S[em]/S[dcl]	210	211	95.3	95.7
NP\NP	198	75	44.0	16.7
(S[dcl]\NP)/(S[adj]\NP)	197	195	85.6	84.8
(S[dcl]\NP)/(S[ng]\NP)	186	187	97.3	97.9

Table 5.5: Precision and recall of the 30 most common lexical categories (**HWDep**, sec. 00, cutoff = 30)

Correct	Recall	Proposed	% wrong
(NP\NP)/NP	88.7	((S\NP)\(S\NP))/NP	6.8
		PP/NP	2.5
((S\NP)\(S\NP))/NP	71.5	PP/NP	17.2
		(NP\NP)/NP	8.8
(S[dcl]\NP)/NP	86.7	((S[dcl]\NP)/PP)/NP	5.0
		((S[dcl]\NP)/(S[to]\NP))/NP	1.4
		(S[b]\NP)/NP	1.3
		S[dcl]\NP	1.2
PP/NP	80.5	((S\NP)\(S\NP))/NP	11.9
		(NP\NP)/NP	5.0
(S\NP)\(S\NP)	84.7	S[adj]\NP	3.1
		N	3.0
		(S\NP)/(S\NP)	2.2
		NP\NP	1.3
		((S\NP)\(S\NP))/((S\NP)\(S\NP))	1.1
(S[b]\NP)/NP	89.2	((S[b]\NP)/PP)/NP	5.5
		((S[b]\NP)/(S[to]\NP))/NP	1.2
		S[b]\NP	1.0
S[pss]\NP	87.6	(S[pss]\NP)/PP	6.2
S[adj]\NP	74.3	(S[adj]\NP)/PP	7.3
		N	4.4
		(S\NP)\(S\NP)	3.8
		S[pss]\NP	2.3
		(S[pss]\NP)/PP	1.2
(S\NP)/(S\NP)	89.0	(S\NP)\(S\NP)	6.0
		(S[adj]\NP)/(S[adj]\NP)	1.3
(S[ng]\NP)/NP	86.0	((S[ng]\NP)/PP)/NP	5.3
		((S[ng]\NP)/(S[to]\NP))/NP	2.8
		N/N	2.1
		S[ng]\NP	1.1
S[dcl]\NP	71.6	(S[dcl]\NP)/PP	14.2
		(S[dcl]\NP)/NP	6.0
		(S[dcl]\NP)/(S[adj]\NP)	2.6
		N	1.7
		(S[dcl]\NP)/(S[to]\NP)	1.7

Table 5.6: Confusion matrix for the most common lexical categories with recall scores $< 90\%$. For each category, only errors which account for at least 1% of the total score are given.

	# in ref	# in test	LP	LR	UP	UR
All	48,519	48,763	83.1	83.5	91.1	91.5
All local	44,483	44,607	84.4	84.6	92.0	92.3
All non-local	4,036	4,156	68.7	70.6	80.9	83.6
Locally mediated	3,486	3,573	70.4	72.1	83.2	85.4
Long-range	550	583	58.5	61.3	66.7	71.8

Table 5.7: Recovery of dependencies in predicate-argument structure (sec. 23 **HWDep**, cutoff = 30, with POS tagger). The first row gives overall performance. Row 2 and row 3 give the overall performance on all local and non-local (ie. locally mediated and long-range) dependencies. Rows 4 and 5 show the performance on locally mediated and long-range dependencies.

for a non-local dependency to be correct, at least two attachment decisions have to be correct – and for an unbounded long-range dependency, possibly more.

Table 5.8 gives a detailed analysis of the 30 most frequent dependencies in predicate-argument structure. Table 5.9 gives a detailed analysis of the 30 most frequent locally mediated dependencies, and table 5.10 gives a detailed analysis of the 30 most frequent unbounded long-range dependencies.[9] In the latter two tables, precision is the percentage of long-range dependencies proposed by the parser that matched a long-range or local dependency in the gold standard, and recall is the percentage of long-range dependencies in the gold standard that matched a long-range or local dependency proposed by the parser.

In terms of labelled accuracy, ((S\NP)\(S\NP))/NP2 is among the most difficult relations. This dependency relation indicates the rate of recovery of the correct attachment for VP-modifying prepositional phrases. Similarly hard is ((S\NP)\(S\NP))/NP3, which is the dependency between a VP-modifying preposition and its NP argument. However, here the unlabelled accuracy is much higher. The same is true for PP/NP1, the dependency between the preposition and its NP argument in a complement PP. We have already seen in table 5.6 that these two lexical categories are very likely to be confused. This indicates that the model performs comparatively poorly on distinguishing between PP complements and adjuncts. However, this distinction is often very hard to draw, and it is highly likely that the model also suffers from inconsistencies in the -CLR annotation in the Penn Treebank. Since verbs which subcate-

[9] In VP coordination, the current implementation treats the dependency between the second verb phrase and the subject as long-range, in analogy to the analysis of right node raising.

Dependency	#Ref	#Test	LP	LR	UP	UR
N/N1	7,278	7,277	95.1	95.0	96.5	96.2
NP[nb]/N1	4,138	4,148	96.7	96.9	97.9	97.7
(NP\NP)/NP2	2,389	2,392	84.7	84.9	94.1	95.2
(NP\NP)/NP1	2,195	2,200	79.3	79.5	81.0	80.5
((S\NP)\(S\NP))/NP3	1,157	1,152	69.5	69.2	94.2	95.3
((S\NP)\(S\NP))/NP2	1,064	1,047	64.2	63.2	72.7	78.9
(S[dcl]\NP)/NP2	951	975	80.6	82.7	87.7	93.0
(S[dcl]\NP)/NP1	908	903	83.3	82.8	91.9	93.9
PP/NP1	875	1001	67.9	77.7	95.6	95.5
(S\NP)\(S\NP)2	749	739	83.4	82.2	87.4	88.8
(S[b]\NP)/NP2	645	635	87.4	86.1	93.1	95.5
(S[to]\NP)/(S[b]\NP)2	610	615	98.1	98.9	98.7	99.8
(S[b]\NP)/NP1	534	538	75.3	75.8	81.4	84.8
(S[dcl]\NP)/(S[b]\NP)1	487	488	94.7	94.9	95.3	95.1
(S[dcl]\NP)/(S[b]\NP)2	485	484	99.0	98.8	99.4	98.8
(S[dcl]\NP)/S[dcl]1	387	392	92.1	93.3	94.6	97.2
(NP[nb]/N)\NP1	383	387	95.6	96.6	97.7	96.9
(NP[nb]/N)\NP2	372	387	92.5	96.2	94.6	96.5
(S[dcl]\NP)/S[dcl]2	346	349	93.7	94.5	94.6	96.0
S[pss]\NP1	345	348	77.3	78.0	84.5	89.6
(S\NP)/(S\NP)2	314	310	88.1	86.9	91.0	88.9
(S[dcl]\NP)/(S[pt]\NP)2	302	300	97.3	96.7	99.3	98.0
(S[dcl]\NP)/(S[pt]\NP)1	301	306	93.1	94.7	95.1	96.7
(S[ng]\NP)/NP2	296	294	84.7	84.1	93.2	96.3
(NP\NP)/(S[dcl]\NP)2	280	271	98.5	95.4	99.6	98.2
(NP\NP)/(S[dcl]\NP)1	278	275	74.6	73.7	75.3	75.2
(S/S)/NP2	270	271	91.9	92.2	95.6	98.2
S[adj]\NP1	267	252	71.0	67.0	77.8	77.9
(S/S)/NP1	259	262	89.7	90.7	92.4	92.3
(S[dcl]\NP)/(S[pss]\NP)1	255	251	89.6	88.2	94.4	93.3

Table 5.8: The 30 most frequent dependencies in predicate-argument structure (**HWDep**, sec. 00, cutoff = 30, with POS tagger)

Dependency	#Ref	#Test	LP	LR	UP	UR
(S[b]\NP)/NP1	469	481	77.1	79.3	83.6	88.1
S[pss]\NP1	321	324	78.4	78.8	85.2	89.4
S[adj]\NP1	246	228	73.7	68.3	79.4	79.3
(S[ng]\NP)/NP1	218	218	63.3	63.8	73.4	74.3
(S[pt]\NP)/NP1	130	134	79.9	82.3	94.8	94.6
(S[pss]\NP)/PP1	114	126	63.5	70.2	85.7	86.8
S[b]\NP1	107	90	71.1	59.8	92.2	86.9
(S[dcl]\NP)/NP1	97	97	67.0	66.0	74.2	71.1
(S[b]\NP)/PP1	78	85	67.1	73.1	89.4	91.0
S[ng]\NP1	73	60	70.0	57.5	80.0	86.3
(S[pss]\NP)/NP1	64	64	71.9	71.9	76.6	82.8
(S[adj]\NP)/PP1	60	78	53.9	70.0	79.5	83.3
((S[b]\NP)/PP)/NP1	55	67	49.3	60.0	85.1	92.7
(S[pss]\NP)/(S[to]\NP)1	40	45	71.1	82.5	80.0	90.0
(S[dcl]\NP)/(S[b]\NP)1	38	38	76.3	76.3	76.3	76.3
((S[ng]\NP)/PP)/NP1	38	35	42.9	39.5	77.1	73.7
(S[ng]\NP)/PP1	35	46	45.7	60.0	80.4	82.9
S[pt]\NP1	31	26	76.9	64.5	100.0	100.0
(S[adj]\NP)/(S[to]\NP)1	29	27	88.9	82.8	92.6	86.2
S[dcl]\NP1	28	19	57.9	39.3	79.0	67.9
(S[dcl]\NP)/(S[pt]\NP)1	28	29	72.4	71.4	72.4	71.4
((S[pt]\NP)/PP)/NP1	28	24	62.5	53.6	100.0	92.9
(S[pt]\NP)/PP1	22	23	56.5	59.1	82.6	81.8
(S[ng]\NP)/(S[to]\NP)1	18	25	60.0	83.3	72.0	88.9
(S[dcl]\NP)/PP1	18	20	65.0	72.2	75.0	88.9
(S[pt]\NP)/(S[to]\NP)1	17	16	93.8	88.2	100.0	100.0
(S[dcl]\NP)/(S[pss]\NP)1	17	19	63.2	70.6	68.4	70.6
(S[b]\NP)/(S[to]\NP)1	16	16	93.8	93.8	100.0	100.0
((S[b]\NP)/(S[to]\NP))/NP1	14	18	66.7	85.7	94.4	92.9
(S[dcl]\NP)/S[dcl]1	13	15	80.0	92.3	80.0	92.3

Table 5.9: The 30 most frequent locally mediated dependencies in predicate-argument structure (**HWDep**, sec. 00, cutoff = 30, with POS tagger)

Dependency	#Ref	#Test	LP	LR	UP	UR
(S[b]\NP)/NP1	60	54	59.3	51.7	61.1	58.3
(S[dcl]\NP)/NP1	54	50	72.0	68.5	82.0	87.0
(S[dcl]\NP)/NP2	43	44	68.2	69.8	70.5	79.1
N/N1	32	69	87.0	84.4	87.0	84.4
(S[b]\NP)/NP2	24	25	72.0	75.0	76.0	75.0
S[pss]\NP1	23	22	68.2	65.2	72.7	91.3
(S[dcl]\NP)/S[dcl]2	23	14	85.7	73.9	92.9	91.3
S[adj]\NP1	21	21	52.4	52.4	61.9	61.9
(S[ng]\NP)/NP1	15	17	70.6	80.0	82.4	93.3
(S[dcl]\NP)/(S[pss]\NP)1	13	10	90.0	69.2	90.0	69.2
(S[dcl]\NP)/(S[b]\NP)1	13	15	53.3	61.5	53.3	61.5
(S[ng]\NP)/NP2	12	12	66.7	66.7	66.7	66.7
(S[dcl]\NP)/S[dcl]1	12	11	72.7	66.7	72.7	66.7
(S[dcl]\NP)/(S[to]\NP)1	11	11	72.7	72.7	90.9	72.7
(S[pss]\NP)/(S[to]\NP)1	10	8	25.0	20.0	25.0	20.0
(S[pt]\NP)/NP1	9	4	100.0	44.4	100.0	77.8
(S[pss]\NP)/NP1	9	3	100.0	33.3	100.0	33.3
S[dcl]\NP1	8	6	50.0	37.5	100.0	87.5
(NP\NP)/NP1	8	4	50.0	25.0	50.0	25.0
((S\NP)\(S\NP))/NP2	8	8	25.0	25.0	37.5	25.0
((S[b]\NP)/PP)/NP3	7	2	100.0	28.6	100.0	57.1
S[ng]\NP1	6	4	25.0	16.7	25.0	66.7
(S[pt]\NP)/NP2	6	6	50.0	50.0	50.0	50.0
(S[dcl]\NP)/PP1	6	6	66.7	66.7	100.0	100.0
(S[b]\NP)/PP1	6	8	62.5	83.3	62.5	83.3
((S[dcl]\NP)/NP)/(S[dcl]\NP)2	6	4	75.0	50.0	75.0	50.0
S[b]\NP1	5	6	50.0	60.0	66.7	60.0
(S[pss]\NP)/PP1	5	10	30.0	60.0	80.0	80.0
(S[dcl]\NP)/(S[pt]\NP)1	5	5	100.0	100.0	100.0	100.0
(S[dcl]\NP)/(S[ng]\NP)1	5	5	100.0	100.0	100.0	100.0

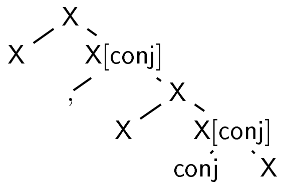
Table 5.10: The 30 most frequent unbounded long-range dependencies in predicate-argument structure (**HWDep**, sec. 00, cutoff = 30, with POS tagger)

gorize for a PP have a different lexical category from those that do not, this problem does not only affect the scores for the dependencies between the preposition, the noun phrase and the verb, but also any dependency between the verb and its other arguments.

5.12.3 Performance on specific constructions

One of the attractions of CCG as a syntactic theory is its analysis of constructions that involve long-distance dependencies arising through coordination and extraction. Here we take a closer look at the performance of the parser on such constructions.

Performance on coordinate constructions In order to assess the model's performance on coordinate constructions, we consider first the evaluation of the $\langle P, H, S\rangle$-dependencies defined in terms of the local tree. Recall that in our grammar, coordination is binarized in the following way:



The head of $\mathsf{X}[\mathsf{conj}]$ is the head of its X daughter. Therefore, the dependency defined by the $\langle P = \mathsf{X}, H = \mathsf{X}, S = \mathsf{X}[\mathsf{conj}]$ gives the accuracy of identifying the heads of both conjuncts and their relation. Table 5.11 gives the accuracy of the most frequent dependencies between conjuncts (as defined in terms of the labelled $\langle P, H, S\rangle$ evaluation).

Note that performance on NP and noun coordination is very poor. We conjecture that the poor performance on noun coordination might (at least to some extent) stem from the flat noun structure in the original Treebank, whereas the low accuracy on NP coordination seems to indicate a different problem. The coordination of multiple conjuncts is one kind of construction where grammars such as ours, but also phrase-structure grammars like the one underlying the Penn Treebank, reach bracketing completeness. This means that the grammar licenses any possible binary bracketing (*[NP and [NP and NP]]* and *[[NP and NP] and NP]*). This is not only true for coordination, but also for appositives (*NP, NP*).[10]

In these cases the number of possible bracketings is exponential in the number of constituents. Therefore, we would expect a particularly poor performance on long sentences. Sec-

[10] As discussed in section 3.6.3, the grammar in CCGbank does not distinguish between NP appositives and NP coordination. Therefore, the figures for $\mathsf{NP} - \mathsf{NP} - \mathsf{NP}[\mathsf{conj}]$ also include appositives.

tion 00 of the Penn Treebank contains what seems to be an article about fines received for violations of industry rules. Five sentences (labelled FRAG) are lists of people, their addresses and their fines. Consider the following example:

(99) *Andrew Derel Adams, Killeen, Texas, fined $15,000; John Francis Angier Jr., Reddington Shores, Fla., $15,000; Mark Anthony, Arlington Heights, Ill., $10,000 and 30-day suspension; William Stirlen, Arlington Heights, Ill., $7,500 and 30-day suspension; Fred W. Bonnell, Boulder, Colo., $2,500 and six-month suspension; Michael J. Boorse, Horsham, Pa.; David Chiodo, Dallas, $5,000, barred as a principal; Camille Chafic Cotran, London, $25,000; John William Curry, fined $5,000, ordered to disgorge $30,000, one-year suspension.*

These sentences are not only very long, they also exhibit exactly the property that almost any binary bracketing is licensed by the grammar. Incidentally, a large proportion of the occurrences of $\mathsf{NP\backslash NP[conj]}$ in the reference corpus are also in these five sentences. The discussion in section 5.8.1 stressed the need for non-local structural information, and it is possible that appropriate features might alleviate this problem. For example, the punctuation rules described above greatly reduce the search space in the above example; however, there are still enough attachment decisions to be made where the dependency model cannot distinguish between the alternative attachments.

Table 5.11 demonstrates a comparatively high performance on right node raising constructions (coordinations of $\mathsf{(S[.]\backslash NP)/NP}$, with $\mathsf{[.]}$ any verbal feature). On the other hand, inspection of the output of the parser reveals that neither of the two instances of argument cluster coordination was analyzed correctly. Both cases involve transitive verbs such as *spend*, where the direct object is followed by a PP. In both cases, the PP following the first object was analyzed as modifying the object, and the two objects were coordinated.

Performance on extraction This section examines the performance on constructions that involve non-local dependencies arising through extraction, such as relative clauses, wh-questions and tough adjectives.

Subject relative clauses engender the most frequent non-local dependencies that arise through extraction. In order to recover the long-range dependency between the head of the NP that is modified by the relative clause and the verbs of the relative clause, the relative clause has to be recognized and attached to the correct noun phrase.

The most common category for subject relative pronouns, $\mathsf{(NP\backslash NP)/(S[dcl]\backslash NP)}$, has

$\tau = \langle P,H,S \rangle$	# Ref	LP	LR
NP – NP – NP[conj]	762	50.7	63.1
N – N – N[conj]	181	52.0	49.2
S[dcl] – S[dcl] – S[dcl][conj]	163	71.8	68.7
S[dcl]\NP – S[dcl]\NP – S[dcl]\NP[conj]	111	78.6	79.3
NP\NP – NP\NP – NP\NP[conj]	44	31.2	11.4
S[b]\NP – S[b]\NP – S[b]\NP[conj]	34	76.3	85.3
N/N – N/N – N/N[conj]	31	34.8	77.42
S[adj]\NP – S[adj]\NP – S[adj]\NP[conj]	24	59.1	54.2
S[pss]\NP – S[pss]\NP – S[pss]\NP[conj]	23	75.0	65.2
S[ng]\NP – S[ng]\NP – S[ng]\NP[conj]	20	60.0	75.0
(S\NP)\(S\NP) – (S\NP)\(S\NP) – (S\NP)\(S\NP)[conj]	17	31.2	29.4
S[to]\NP – S[to]\NP – S[to]\NP[conj]	11	100	81.8
(S[dcl]\NP)/NP – (S[dcl]\NP)/NP – (S[dcl]\NP)/NP[conj]	11	90.0	81.8
(S[ng]\NP)/NP – (S[ng]\NP)/NP – (S[ng]\NP)/NP[conj]	9	85.7	66.7
S[pt]\NP – S[pt]\NP – S[pt]\NP[conj]	8	75.0	37.5
PP – PP – PP[conj]	7	50.0	57.1
(S[b]\NP)/NP – (S[b]\NP)/NP – (S[b]\NP)/NP[conj]	5	100.0	100.0

Table 5.11: The most frequent surface dependencies between conjuncts (sec. 00)

been recovered with precision and recall of 98.4% (252/256) and 96.9% (252/260). Five occurrences of this category in the reference corpus (accounting for 1.9% of the recall score) were misanalyzed as (NP\NP)/(S[b]\NP), the category for subject relative pronouns where the relative clause is in subjunctive mood. This indicates that subject relative clauses are easily detected by a CCG parser. The high accuracy of this lexical category cannot be explained by a lack of lexical ambiguity, since the most frequent relative pronouns *that*, *which* and *who*, which account for 99.5% of the occurrences of (NP\NP)/(S[dcl]\NP) in the training data, have 55, 45 and 14 lexical categories each. In the predicate-argument structure, the dependency between relative pronouns with category (NP\NP)/(S[dcl]\NP) and the heads of the S[dcl]\NP argument has been recovered with LP = 98.5% and LR = 95.4% (UP=99.6%, UR=98.2%).

Whether the relative clause is attached to the correct noun phrase is indicated by the accuracy with which the dependency relations between the relative pronoun (NP\NP)/(S[dcl]\NP) and its first argument have been recovered. Here, a much lower performance is obtained with LP=74.6%, LR=73.7% (UP=75.3%, UR=75.2%). Note that a relative clause can have multiple attachement sites if the preceding noun phrase is complex; this is therefore a much harder problem than identifying the verb in a subject relative clause. For example, if the preceding noun phrase is modified by an preposition, such as *the unit of New York-based Loews Corp. that makes Kent cigarettes*, the relative clause can attach either to the argument of the preposition (*Loews Corp.*) or to the entire noun phrase (modifying *unit*). If the preceding noun phrase is modified by an appositive, such as *Fannie Mae's chairman, David O. Maxwell, who visits Tokyo at least once a year*, the NP argument of *who* has two heads (*chairman* and *Maxwell*), since appositives are represented like coordination in our grammar. Therefore, in order to get 100% accuracy, both attachments have to be resolved correctly. Note that this particular case is an artefact of the grammar, since semantically, *chairman* and *Maxwell* refer to the same individual. Since our current probability model (just like those used by other Treebank parsers such as Collins (1999) or Charniak (2000)), only models local dependencies between words, it does not capture the dependency between the verb of the relative clause and the head of the NP it modifies. Therefore, one solution which might improve performance on this particular construction would be a model that also captures long-range dependencies. The model of Clark *et al.* (2002) and the model presented in the following chapter are of this kind. Alternatively, it is conceivable that appropriate structural features might help disambiguation to some extent. For instance, *that* introduces restrictive relative clauses, and is therefore much more likely to modify the definite NP *the unit* than the indefinite singular NP *Loews Corp*.

Subject extraction from embedded sentences (*who I think buys shares*) requires the special lexical category ((S[dcl]\NP)/NP)/(S[dcl]\NP) for verbs like *think*. On this category, the model achieves a precision of 100% (5/5) and recall of 83.3% (5/6). The case the parser misanalyzed is due to lexical coverage: the verb *agree* occurs in our lexicon, but not with this category.

Free subject relative pronouns, NP/(S[dcl]\NP) (*what matters to me*) obtain 100% precision and 70.0% recall (7/10). The three cases where NP/(S[dcl]\NP) was misanalyzed as embedded questions (S[qem]/(S[dcl]\NP)).
S[qem]/(S[dcl]\NP) itself has 100.0% recall (5/5), and 62.5% precision (8/5, accounting for the three errors on NP/(S[dcl]\NP)). On the dependency between the question word and the head of the S[dcl]\NP, 62.5% LP (100.0% UP) and 100.0% LR were obtained.

There are eight instances of **subject wh-question words** (*Who buys shares?*) with category S[wq]/(S[dcl]\NP) ; all of these as well as the dependencies between the wh-word and the head of the S[dcl]\NP were recovered with 100.0% precision and recall.

In **control verbs** (such as ((S[dcl]\NP)/(S[to]\NP$_i$))/NP$_i$) the subject of their complement verb phrase is co-indexed with another argument of the verb. The most common lexical categories of control verbs, ((S[dcl]\NP)/(S[to]\NP))/NP, ((S[b]\NP)/(S[to]\NP))/NP and ((S[ng]\NP)/(S[to]\NP))/NP, have precision and recall scores of 70.8%/97.1% (34/48 and 34/35), 65.0%/81.3% (13/20 and 13/16), and 52.9%/81.8% (9/17 and 9/11).

Object extraction is an unbounded long-range dependency, and is therefore much harder to recover correctly. The detailed evaluation of long-range dependencies only gives figures for each individual relation, not for each type of construction. It is therefore difficult to give an overall score for the accuracy of specific kinds of construction that involve object extraction, such as object relative clauses, since object extraction results in different long-range dependencies, according to the lexical category of the verb, and not all non-local dependencies of objects arise through extraction. An inspection of all unbounded long-range dependencies of objects (non-subject NP arguments) of verbal categories (any category of the form (S[.]\NP)/$, where [.] is not [adj]) yields a combined LP and LR score of 65.0% (67/103) and 67.0% (67/100). However, although this figure includes all instances of object extraction, it also comprises other unbounded dependencies such as right node raising. Although these are less stringent measures, we can again use other indicators, such as the accuracy of lexical categories, to examine the performance of the parser on specific constructions.

The most common lexical category for **object relative pronouns**, (NP\NP)/(S[dcl]/NP), has precision of 85.0% (17/20) and recall of 81.0% (17/21). It was three times misanalyzed as a simple complementizer, S[em]/S[dcl]. The dependency between the relative pronoun and the head verb of the relative clause was recovered with LP = 84.2%, LR = 76.2% (UP = 94.7%, UR = 95.2%). However, since object extraction is an unbounded long-range dependency, this does not necessarily indicate that the extraction site was correctly recovered. On the dependency between (NP\NP)/(S[dcl]/NP) and its NP argument, we obtain LP = 66.7% (21/14) and LR = 58.3% (21/24) (UP = 76.2, UR = 58.3). Again, this is a hard problem, for the reasons explained above.

Free object relatives, NP/(S[dcl]/NP), have a recall of 94.1% (16/17), and precision of 100%. One occurrence of NP/(S[dcl]/NP) was analyzed as an embedded object question (S[qem]/(S[dcl]/NP)). On the dependency between the head of the relative clause and the relative pronoun, 100.0% LP and 94.1% LR (100.0% UR) were obtained.

Object extraction appears more frequently as a **reduced relative** (*the man John saw*), and there are no lexical categories indicating this extraction. Reduced relative clauses are captured by type-changing rules such as S[dcl]/NP ⇒ NP\NP. This rule was applied 56 times in the gold standard, and 61 times by the parser. 35 times a reduced relative in the gold standard was matched exactly by a reduced relative in the output of the parser (LP = 57.4%, LR = 62.5%). The start of a reduced relative in the gold standard was matched 47 times by a reduced relative in the parser output (P = 77.5%, R = 83.9%).

The lexical category for ***tough*-adjectives**, ((S[adj]\NP)/((S[to]\NP)/NP)), was recovered with a precision of 66.7% (2/3) and recall of 100% (2/2).

5.13 Comparison with alternative approaches

Here we compare our parser with two alternative approaches. One, Clark *et al.* (2002), is another statistical parser for CCG, also trained on CCGbank. This allows us to make a direct comparison between different models for the same grammar formalism. The other comparison is with Treebank parsers, such as Collins (1999) and Johnson (2002b), who presents an algorithm which takes the output of Charniak (2000)'s parser and recovers the empty nodes that are present in the original Treebank annotation to encode non-local dependencies.

5.13.1 Comparison with Clark *et al.* (2002)

Clark *et al.* (2002) present another statistical CCG parser trained on CCGbank, which is based on a conditional (rather than generative) model of the derived dependency structure, including non-surface dependencies. This section describes the model used by Clark *et al.* (2002) and compares the experimental results.

The probability model is similar to Collins (1996), in that it is conditional, and in that it expresses the probability of a parse tree τ as the product of the probability of some syntactic structure which partially describes the parse tree – in Collins' case a sequence of non-recursive base NPs, in Clark *et al.* (2002) the sequence of lexical categories assigned to the words in the sentence – and the probability of set of dependency relations D between the words in the sentence that is defined by the parse tree:

$$P(\tau|S) = P(C|\tau)P(D|\tau,C)$$

The probability of the category sequence, $P(C|\tau)$ is computed using a Maximum Entropy model, similar to the model underlying the POS tagger of Ratnaparkhi (1996). In this model, the lexical category for each word is chosen based on a set of features expressing the identity of the word in question, its POS tag, and the surrounding ⟨word, POS tag⟩ pairs. Since lexical categories in CCG are essentially unsaturated subcategorization frames, knowing the lexical categories for each word constrains the space of possible parse trees considerably. However, the lexical categories in a CCG derivation constrain each other, and this information is lost by only considering the local context of each word. Therefore, the most likely category sequence for a sentence according to this model might not yield a legal derivation. But Clark (2002) shows that, for a model estimated from and tested on CCGbank, a lattice of the most probable category sequences consisting of on average 3.8 categories per word contains the correct lexical category for 98.4% of the words.

The probability of the set of dependency relations D is computed in a manner similar to Collins (1996). Here, dependency relations are defined in terms of argument slots of lexical categories as described in section 5.5. These dependency relations correspond to the predicate-argument structure, and also include non-local dependencies, since Clark *et al.* (2002) use a co-indexation mechanism to encode which dependencies are projected by the lexicon.

However, like Collins (1996), the individual dependency relations are treated as independent. There is no global constraint on what constitutes a set of dependency relations that is licensed by a grammar (ie. that can be represented by a derivation for this sentence). This means that non-zero probability mass is assigned to such unlicensed dependency structures, and therefore, this model is inconsistent, or deficient, for the same reasons as Collins (1996).

The Clark *et al.* parser is not a normal-form parser. Instead, it uses a subsumption check on the generated dependency structures to eliminate spurious ambiguities, and the probabilites of the already generated structures to guide a beam search.

The following table (reprinted from Hockenmaier and Steedman (2002b)) compares our parser with the parser of Clark *et al.* (2002) according to the evaluation of surface and deep dependencies given in Clark *et al.* (2002). Since the only available results for the parser of Clark *et al.* (2002) are on an earlier version of CCGbank, we compare it with an older version of our parser which is trained on the same version of the corpus, published in Hockenmaier and Steedman (2002b). In this version, our parser did not yet produce dependencies in the predicate-argument structure. Therefore, we used Clark *et al.*'s parser to generate these dependencies from the output of our parser (see Clark *et al.* (2002)).[11]

	LP	LR	UP	UR
Clark	81.9%	81.8%	89.1%	90.1%
Hockenmaier	83.7%	84.2%	90.5%	91.1%

Our oracle experiment with the baseline model indicates that by knowing the correct lexical categories for a word, very high performance can be achieved even with a very simple model of derivational structure. The supertagger used by Clark *et al.* is not quite such an oracle since it does not always predict the correct category for a word, and since in the multi-tagging setting used for the parsing experiment, multiple categories are proposed for each word. The multi-tagging setting is necessary for parsing, since there is no guarantee that the obtained category sequence corresponds to a CCG derivation if the supertagger proposes only one category per word. In a single-category setting, only 91% of the sentences in section 23 can be parsed (Clark, 2002). It is difficult to analyze the relative impact on performance of the two components of the Clark *et al.* parser. The supertagger serves the purpose of constraining the search space before parsing, and of providing the probabilities of the sequence of lexical categories, whereas the search of the parser is purely guided by the model of dependency structures.

[11] Due to the smaller grammar and lexicon of Clark *et al.*, our parser can only be evaluated on slightly over 94% of the sentences in section 23, whereas the figures for Clark *et al.* (2002) are on 97%. The set of dependencies in the Clark *et al.* parser also differs in minor aspects to the dependencies reported in this dissertation. For example, the Clark *et al.* parser does not capture dependencies between adjuncts of adjuncts and the adjuncts themselves.

Clark *et al.* (2002) also give a detailed evaluation of individual dependency relations on section 00. Although this is on an earlier version of CCGbank, a comparison of these figures indicates the relative performance of the two parsers.[12] For example, on the most common dependency, between a prenominal modifier $\mathsf{N/N}$ and its argument, Clark *et al.* achieve LP = 92.9% and LR = 95.1%, compared to LP = 94.5% and LR = 94.7% in our current experiments. On the dependency between a VP-modifying preposition and the verb, Clark *et al.* obtain LP = 54.8% and LR = 69.4%, resulting in an F-score of 61.2%. This compares with LP = 63.6% and LR = 62.5%, or an F-score of 63.0%. On the dependency between a complement preposition $\mathsf{PP/NP}$ and its argument, Clark *et al.*'s LP = 70.9%, LR = 67.2% and F-score = 69.0% compares with our LP = 67.5%, LR = 77.0% and F-score of 71.9%. We have argued above that non-local dependencies could be helpful in deciding the correct attachment sites for relative clauses. On the dependency between subject relative pronouns ($\mathsf{(NP\backslash NP)/(S[dcl]\backslash NP)}$) and the NP they modify, Clark *et al.* give LP = 73.8% and LR = 69.2%, compared with LP = 74.2% and LR = 73.4%. For the attachment of object relative pronouns ($\mathsf{(NP\backslash NP)/(S[dcl]/NP)}$), Clark *et al.* obtain LR = 66.7% and LP = 45.5% (our results: LP = 65.0% and LR = 54.2%).

5.13.2 Comparison with Treebank parsers

It is difficult to draw a direct comparison between our results and those of standard Treebank parsers, since the underlying grammars are so different. As described in chapter 3, some inconsistencies and errors in the Penn Treebank had to be corrected before translation to CCG could take place. Although the models used by Treebank parsers such as Collins (1999) and Charniak (2000) are more robust to noise, is is to be expected that they would achieve higher accuracy if they were trained and tested on similarly cleaned up data.

In terms of overall performance, the only meaningful comparison with Treebank parsers is on the recovery of unlabelled dependencies, $\langle\rangle$, since this measure simply awards one point for each word in a sentence, and is therefore less dependent on the exact representation of syntactic structure. Collins (1999), the only Treebank parser for which this evaluation is reported, achieves 90.9% recall for unlabelled dependencies on section 00. On the same section of CCGbank, **HWDep** (cutoff:30, +POS tagger) achieves 91.3% accuracy on $\langle\rangle$ (88.4% for $\langle S\rangle$, and 84.3% for $\langle P,H,S\rangle$).

Standard Treebank parsers do not aim to recover the empty nodes which in the original Penn Treebank represent non-local dependencies. The only exception is Model 3 of Collins

[12] Here, we give results obtained with POS-tagged input

(1999), which recovers *T*-traces with 93.8% precision and 90.1% recall (on section 23), where a recovered trace is correct if it is an argument of the correct head word, is in the correct position relative to the head word (preceding or following) and is dominated by the correct nonterminal label. On subject extraction alone, this model obtains 96.3% precision and 98.8% recall (F-score: 97.5). Although Collins' algorithm contains implicit co-indexation through a gap feature, he does not attempt to actually recover the long-range dependencies that are expressed by these traces. His evaluation indicates merely the detection of long-range dependencies, not the actual accuracy with which they have been resolved. In CCG, we can measure the accuracy with which subject extraction is detected in terms of the accuracy of lexical categories that trigger subject extraction. The most common category for subject extraction, (NP\NP)/(S[dcl]\NP) accounts for 292 of the 354 instances of subject extraction in section 23. It has been recoved with precision and recall of 97.6% and 97.3%. Other subject extraction categories such as ((NP\NP)/(S[dcl]\NP))/N (for wh-determiners, 20 occurrences), and ((NP\NP)/(S[dcl]\NP))\(NP/NP) (for pied-piping with subject extraction, 5 occurrences), have been recovered with 100% accuracy.

But, in contrast to our parser, which can represent a much more complete set of non-local dependencies, Collins only considers *T*-traces, which capture long-range dependencies arising through extraction. However, other kinds of non-local dependencies, such as right node raising or control, are not represented in the output of Collins' parser.

A different comparison is with the trace-recovery algorithm of Johnson (2002b), which inserts null elements into the output of the parser of Charniak (2000) and co-indexes these null elements with other nodes in the tree. In the Penn Treebank, a trace in a relative clause is co-indexed with the relative pronoun, not with the head noun of the noun phrase that is modified by the relative clause. The accuracy of the latter is the long-range dependency that is measured in our predicate-argument evaluation. As discussed above, this is a much harder task, since it also requires to find the correct attachment for the relative clause. Johnson reports precision and recall scores of 85% and 77% on the detection and recovery of the correct antecedents of *T* traces that are co-indexed with WHNPs, using the output of Charniak's parser on section 23. We cannot compare this directly with any single dependency relation in the predicate-argument structure. However, subject relative clauses are by far the most common construction in which such *T* traces are used in the Penn Treebank, and in this case, the dependency relation between the relative pronoun and the head of the relative clause corresponds very closely to Johnson's antecedent recovery mechanism. Almost all occurrences of subject relative pronouns have the category (NP\NP)/(S[dcl]\NP). For the dependency between this category and the verb, we obtain labelled precision and recall of 97.7% and 94.8% (on section 23, using POS-tagged input). Johnson's figure also includes other cases, such as object extraction. This is less straightforward for us to evaluate with a single measure. However, with 260 occurrences in section 00, subject relative clauses are by far the most common dependency relation expressed by *T* traces that are co-indexed with WHNPs; therefore we conjecture that our overall performance on recovering the information expressed by *T* traces is higher than Johnson's. This is not surprising, since in CCG this information is directly encoded in the categories. Therefore, there is no need for a post-processing stage, which is likely to introduce additional noise.

5.14 Conclusion

This chapter has defined a number of generative probability models of CCG derivations, and has demonstrated that with the same smoothing technique and beam search, the best performing model achieves with 91.3% a similar performance on the recovery of unlabelled word-word dependencies that are defined by the local derivation tree as Collins (1999), but with a simpler model that uses fewer conditioning variables. We argue that simple models can perform so well on CCG because of the amount of non-local structural information encoded in CCG categories.

In addition to local dependencies, our parser returns the word-word dependencies in the predicate-argument structure. Here it recovers labelled dependencies with precision and recall of 83.1% and 83.5% and unlabelled dependencies with precision and recall of 91.1% and 91.5%.

In terms of labelled accuracy, our parser achieves seemingly lower performance than Treebank parsers. However, in order to correctly predict a CCG derivation, the parser needs to distinguish between complements and adjuncts and identify instances of extraction and other long-range dependencies. None of this information is required of a Penn Treebank parser in order to perform well according to the standard evaluation metrics. Therefore, it is easier for such parsers to obtain high labelled scores. The null elements that represent long-range dependencies in the Penn Treebank are not reproduced by standard Treebank parsers, and post-processing algorithms such as that of Johnson (2002b) are necessary to reconstruct this information. This introduces additional noise, especially when the parse trees themselves contain errors. Therefore, a direct comparison of the labelled scores of our parser with that of standard Treebank parsers is not appropriate.

It is to be expected that a higher performance could be obtained for CCG if additional features representing non-local, structural properties of the tree could be exploited, even if we are currently not able to benefit from these features. It is very likely that a re-ranking approach similar to Collins (2000) would yield improvements comparable to those found for standard Penn Treebank parsers, in particular if such methods are found to be able to benefit from lexical and structural features simultaneously. The infrastructure which enables such re-ranking experiments to be performed has been provided by the parsers presented in this chapter, and future work ought to address this question.

Another problem that was revealed by our experimental results is that of lexical coverage: a CCG lexicon extracted from the training corpus is incomplete, in that it does not contain all entries for all the words in the training corpus. This coverage problem is reflected in lower performance on sentences for which at least one required lexical category is missing from the lexicon. Adjusting the frequency threshold that defines rare words does alleviate this problem to some extent, but not entirely. More research is required to determine a more principled manner of increasing lexical coverage without causing too much lexical overgeneration.

We are only aware of one alternative approach to statistical parsing with CCG, namely that of Clark *et al.* (2002). However, our best model outperforms this parser in terms of coverage and accuracy. Our currently best results are obtained with a model of bilexical dependencies; but, unlike the approach of Clark *et al.* (2002), not all dependencies captured by this model are linguistically motivated. The next chapter presents another generative model for CCG derivations which captures directly the dependencies represented in the semantic interpretation, including bounded and unbounded long-range dependencies, and does not suffer from the inconsistency problems of Clark *et al.*'s model.

Postscript In Gildea and Hockenmaier (2003), we present a system for automatically identifying PropBank-style semantic roles based on the predicate-argument structure output of the statistical parser presented in this chapter, and compare it against a system that performs the same task, using the output of the Collins (1999) parser. With F-scores of 66.4 (Collins) and 66.8 (CCG), the overall performance of both systems is almost identical. However, on core-arguments, the CCG-based system obtains an F-score of 74.8, and outperforms the system that uses the Collins parser, obtaining an F-score of 72.6. We argue that the better performance of the CCG-based system is due to the recovery of non-local dependencies that are hard to recover from the skeletal parse trees returned by the Collins parser.

Chapter 6

Generative models of predicate-argument structure

The models in the previous chapter treat CCG derivations like phrase structure trees. From a linguistic point of view, such models are ultimately not very satisfying, since, one the one hand, they cannot capture certain bilexical dependencies in the underlying predicate-argument structure, and, on the other hand, they model a number of dependencies that do not correspond to any relation in the semantic interpretation. This problem occurs whenever composition and type-raising are required. Although in English this is restricted to constructions such as relative clauses and right node raising, the dependency model of the previous chapter would systematically capture the wrong dependencies if a different kind of canonical derivation structure was adopted. But if we were to parse languages like Dutch, this problem would be much more prevalent, and we could not expect the kind of model developed in the previous chapter to perform as well. Even for English, it is conceivable that for certain applications such as language modelling, incremental derivations might be more suitable than the derivation structure assumed in CCGbank.

Therefore, this chapter develops a different kind of generative model which exploits CCG's transparent syntax-semantics interface to capture the word-word dependencies defined by the predicate-argument structure rather than by the surface derivation tree. This new model overcomes the linguistic shortcomings of the previous dependency model since it is capable of expressing all and only the dependencies of the predicate-argument structure. This includes long-range dependencies which are beyond the scope of previous generative models. In our current parser implementation, this kind of model requires a very aggressive beam search strategy. Furthermore, the presence of unbounded long-range dependencies precludes us from incorporating lexicalized structural probabilities. Therefore, the overall performance of this model is lower than that of the dependency model of Chapter 5. However, we investigate to what extent this kind of model benefits from capturing different kinds of non-local dependencies, and demonstrate that parsing performance improves significantly when long-range dependencies are not just recovered in the derivation, but also expressed in the model. The experimental results in this chapter are also published as Hockenmaier (2003).[1]

6.1 Introduction

The main motivation for the use of dependency models for parsing is the assumption that a choice between alternative parses for the same sentence can be made according to the plausibility of the relationships between the words in the sentence. What these relationships are is determined by the linguistic representation and by the statistical model.

In the previous chapter, we have seen two kinds of dependency models for CCG. The parser of Clark *et al.* (2002) is based on a conditional model which captures those word-word dependencies that are implicit in the derived predicate-argument structure. It yields very good empirical results; however, from a theoretical point of view, it is unsatisfying, since it is not a consistent probability model. The dependency model of the previous chapter outperforms the Clark *et al.* (2002) parser, and belongs to a class of models for which consistency guarantees can be made. However, this model treats CCG derivations like phrase structure trees, and CCG categories like atomic symbols without internal structure. Such models can only capture the kinds of dependencies expressible by a lexicalized probabilistic context-free grammar, in which the head word of each non-head child of a local tree is dependent on the head word of the head child. Although the normal-form derivations used in CCGbank are as close to the context-free case as possible, this assumption is clearly not warranted anymore when composition and type-raising are required. In this chapter, we first consider some examples where the previous dependency model fails to capture linguistically meaningful relationships between words. This begs the question whether it is possible to define a sound, but linguistically plausible model for CCG which captures the word-word dependencies that are expressed not by the syntactic

[1] The “undirectional” evaluation of dependencies in the predicate-argument structure reported in Hockenmaier (2003) corresponds to the “unlabelled” evaluation in this dissertation. The “unlabelled” evaluation in Hockenmaier (2003) requires the functor-argument relationship of two words to agree with the gold standard. Here, we only take into account whether there is a relation between two words or not. See section 5.5 for further discussion.

derivation, but by the semantic interpretation; that is, a statistical model which is not orthogonal to, but captures directly, what is expressed by the linguistic representation.

Such a model is presented in the second part of this chapter. This model is, to the best of my knowledge, the first generative model to capture all local and long-range word-word dependencies. If it is true that the plausibility of relationships between words can be used to select the correct parse for a sentence, and if it is the case that the relationships that matter for this task are the ones that are akin to those in the predicate-argument structure, then this kind of model ought to ultimately outperform a model which captures different relationships or which captures the same relationships in different manners, depending on the context. Even if the full advantages of such a model might not be revealed in parsing English with the canonical derivation structure of CCGbank, we conjecture that such a model might be especially beneficial for parsing languages with freer word order (where function application alone covers a much smaller fragment of the language), or for other applications such as language modelling.

The underlying predicate-argument structure depends only to a degree on the grammar formalism; therefore it is possible that parsers that are based on other grammar formalisms could equally benefit from such a model. However, by using CCG the parser has direct and immediate access to the predicate-argument structure. This is not the case for less expressive grammars, such as those on which standard Penn Treebank parsers are based.

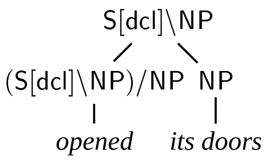
The bilexical dependencies that are captured by the model presented here are defined in terms of argument slots of the lexical categories of words, rather than in terms of local subtrees of the derivation. Such dependencies differ from the dependencies captured by other generative models in two main respects: first, it is not necessarily the case that the dependency holds between the head word of one constituent and the head word of another. Instead, the dependency can be projected from any word within the span of the head constituent, as can happen whenever two functors compose. This necessitates a change in the order in which trees are expanded by the generative model. Second, a word can now be dependent on more than one other word, since coordination and extraction lead to additional non-local dependencies. In order to maintain a consistent probability model, each word is only generated once, and conditioned on all previously generated dependency relations that hold for it. Therefore, this model does not suffer from the problems of some generative models highlighted by Abney (1997), even though the dependencies it captures have a structure that corresponds to a DAG rather than a tree. Thhe probabilities of such multiple dependency relations are difficult to estimate reliably. I will discuss the solution used in the current implementation as well as three alternatives.

First, the shortcomings of the previous model are illustrated, and used to motivate the new model. Then the implementation of the model is described, and experimental results are reported and evaluated.

6.2 Problems with the previous model

In chapter 5, it was shown that a dependency model which treats CCG normal-form derivation trees like phrase-structure trees yields state-of-the-art performance in recovery of word-word dependencies. However, as a model of word-word dependencies, it has two main shortcomings: Since word-word dependencies are captured in terms of local trees, this model does not use the same probability distributions for the same semantic dependencies if they have been arrived at through different derivations. For instance, the dependency probability between a subject and the head verb is modelled by separate distributions depending on whether subject and verb are combined through function application or through type-raising and composition. From a linguistic point of view, this is not desirable, as the relationship between subject and verb is identical, regardless of the derivation. Also, this artificial distinction exacerbates the sparse data problem for dependency counts. Furthermore, if a predicate combines with its argument through composition, the head of the new constituent is the predicate, and therefore any remaining arguments of its argument appear as arguments of the predicate. As will be illustrated below, such spurious dependencies are unlikely to express meaningful relationships. It is unclear whether these extraneous dependencies harm the performance of the parser; but it is implausible that they improve it.

In the dependency model of the previous chapter, we have assumed that binary trees have one head child (with category H and head word w_H) and one non-head child (with category S and head word w_S), and that the model represented dependencies between w_S and w_H by conditioning w_S on w_H. Consider the following tree:



Here, $H = (\mathsf{S[dcl]\backslash NP})/\mathsf{NP}$, and its head word $w_H = \textit{opened}$. $S = \mathsf{NP}$ and $w_S = \textit{doors}$.

Recall that the dependency model of the previous chapter conditions w_S not only on w_H, but also on its own lexical category c_S, on the local tree τ in which the non-head constituent itself is generated (represented in terms of the categories of the parent node, the head child and the non-head child $\langle P,H,S\rangle$), and on the lexical category of w_H, c_H. We will refer to the head word w_X of a constituent X and its lexical category c_X as the lexical head h_X of X.

$$P(w_S \mid c_S, \tau = \langle P,H,S\rangle, h = \langle c_H, w_H\rangle)$$

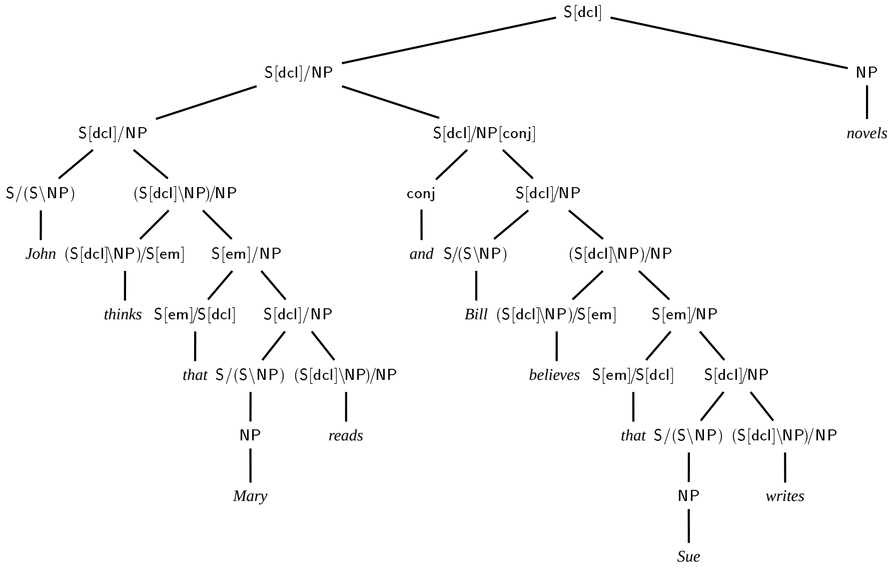
Therefore, in the above tree, the head word *doors* is conditioned on its lexical category (N), on the local tree (parent $\mathsf{S[dcl]\backslash NP}$, head $\mathsf{(S[dcl]\backslash NP)/NP}$ and non-head NP), as well as on *opened*, the head word of the head daughter, and its lexical category:

$$\begin{aligned} P(\textit{doors} \quad \mid \quad & c_S = \mathsf{N}, \\ & \tau = \langle \mathsf{S[dcl]\backslash NP}, \mathsf{(S[dcl]\backslash NP)/NP}, \mathsf{NP}\rangle, \\ & h_H = \langle \mathsf{(S[dcl]\backslash NP)/NP}, \textit{opened}\rangle) \end{aligned}$$

Now, consider the following example:

(100) *John thinks that Mary reads, and Bill believes that Sue writes novels*

This is a right node raising construction, and hence even the normal-form derivation requires type-raising and composition:

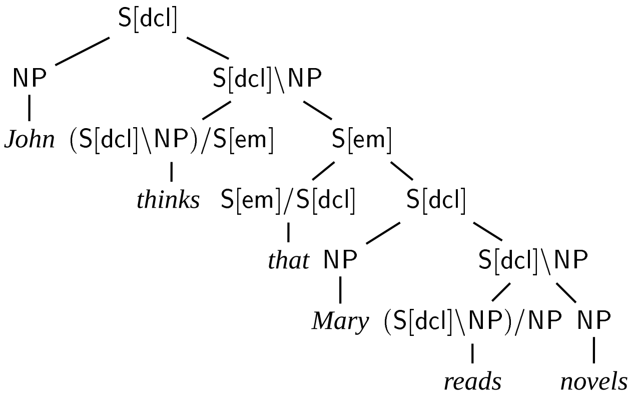


This derivation illustrates both shortcomings of the old dependency model.

First, the dependency between *John* and *thinks* (and similarly, between *Bill* and *believes*) is expressed by the following probability:

$$\begin{aligned} P(\mathit{John} \quad | \quad & c_S = \mathsf{NP}, \\ & \tau = \langle \mathsf{S[dcl]/NP}, \mathsf{(S[dcl]\backslash NP)/NP}, \mathsf{S/(S\backslash NP)} \rangle, \\ & h_H = \langle \mathsf{(S[dcl]\backslash NP)/S[em]}, \mathit{thinks} \rangle) \end{aligned}$$

Compare this with the derivation of the following sentence:



Although in both cases *John* is the subject of *thinks*, the dependency between the two words in the second sentence is modelled by a different distribution, namely:

$$\begin{aligned} P(\mathit{John} \quad | \quad & c_S = \mathsf{NP}, \\ & \tau = \langle \mathsf{S[dcl]}, \mathsf{S[dcl]\backslash NP}, \mathsf{NP} \rangle, \\ & h_H = \langle \mathsf{(S[dcl]\backslash NP)/S[em]}, \mathit{thinks} \rangle) \end{aligned}$$

Within the realm of normal-form derivations, this problem manifests itself for every construction which requires type-raising and composition – that is, all cases involving long-range dependencies caused by extraction and coordination.

The second problem is also exemplified by the above derivation. The embedded sentence is an argument of the complementizer, and the model assumes a dependency between *reads* (or *writes*) and *that*. The complementizer itself is the head of the argument of *thinks*. Since this model treats coordination like a case of modification, whereby one conjunct is treated as head and the other as modifier, *believes* is dependent on *thinks*:

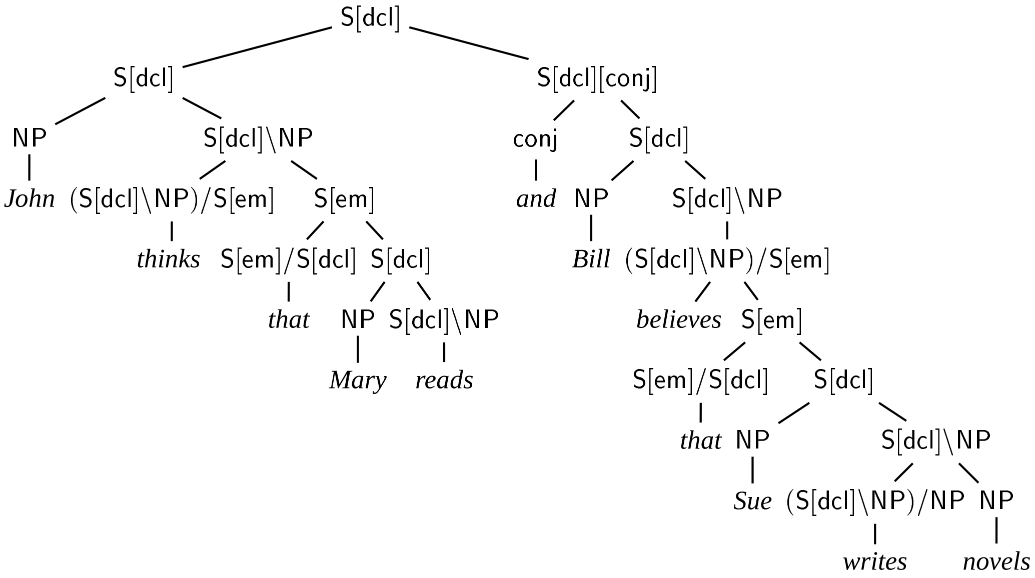
$$\begin{aligned} P(\mathit{believes} \quad | \quad & c_S = \mathsf{(S[dcl]\backslash NP)/S[em]}, \\ & \tau = \langle \mathsf{(S[dcl]\backslash NP)/S[em]}, \mathsf{(S[dcl]\backslash NP)/S[em]}, \mathsf{(S[dcl]\backslash NP)/S[em][conj]} \rangle, \\ & h_H = \langle \mathsf{(S[dcl]\backslash NP)/S[em]}, \mathit{thinks} \rangle) \end{aligned}$$

Thus, *thinks* is assumed to be the head of the entire S[dcl]/NP. Hence, *novels* is conditioned on *thinks*, rather than on *reads* and *writes*:

$$\begin{aligned} P(\mathit{novels} \quad | \quad & c_S = \mathsf{NP}, \\ & \tau = \langle \mathsf{S[dcl]}, \mathsf{S[dcl]/NP}, \mathsf{NP} \rangle, \\ & h_H = \langle (\mathsf{S[dcl]\backslash NP})/\mathsf{S[em]}, \mathit{thinks} \rangle) \end{aligned}$$

But *novels* is not an argument of *thinks*, and this dependency is unlikely to provide useful information.

However, there is an analysis of the above sentence, in which *novels* is the argument of *writes*, and is expressed as such by the model:



If *"writes novels"* occurs in the training corpus, it is quite likely that this parse will be preferred over the other one.

Now, contrast the above example with the following sentence:

(101) *John thinks that Mary reads, and Bill believes that Sue watches movies.*

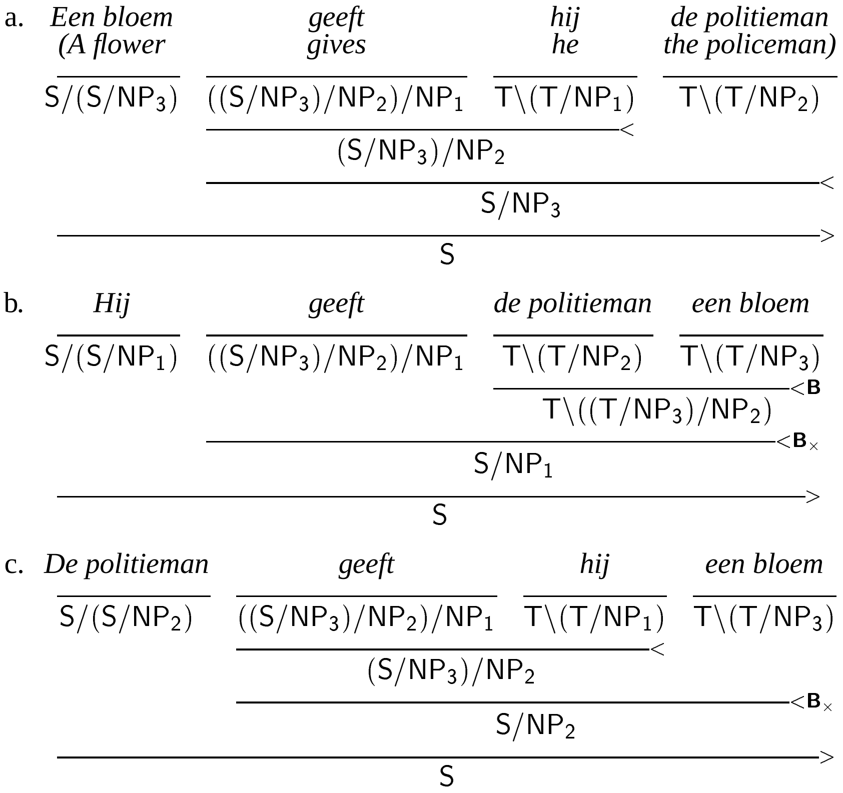
Movies is a very unlikely object of *reads*, but since this relationship is not expressed in the model, this information cannot be used to disambiguate in this case.

Before we define a probability model which overcomes these shortcomings of the previous model as a model of word-word dependencies by modelling the word-word dependencies that are determined by the predicate-argument structure represented by the derivation, we consider briefly the case of another language, Dutch.

6.3 Modelling word-word dependencies in Dutch

Dutch has a much freer word order than English. The analyses given in Steedman (2000) assume that this can be accounted for by an extended use of composition. Consider the following examples of sentences with a ditransitive verb:

(102)



A probability model of the kind developed in the previous chapter estimated from a corpus containing these three sentences would not be able to capture the correct dependencies. Unless we assume that case is represented as a feature on the NP categories, the dependency model could not distinguish between the dependency relations of *bloem* and *geeft* in the first sentence, *Hij* and *geeft* in the second sentence and *politieman* and *geft* in the third sentence. In the second sentence, either the dependency between *politieman* and *geeft* or between *bloem* and *geeft* could not be captured at all.

The cross-serial dependencies of Dutch are one of the syntactic constructions that led people to believe that more than context-free power is required for natural language analysis. Here is an example together with the CCG derivation from Steedman (2000):

(103)

$$
\begin{array}{cccccc}
\textit{dat} & \textit{ik} & \textit{Cecilia} & \textit{de nijlpaarden} & \textit{zag} & \textit{voeren}\\
\textit{(that} & \textit{I} & \textit{Cecilia} & \textit{the hippopotamuses} & \textit{saw} & \textit{feed)}\\
\hline
 & \mathsf{NP_1} & \mathsf{NP_2} & \mathsf{NP_3} & \mathsf{((S\backslash NP_1)\backslash NP_2)/VP} & \mathsf{VP\backslash NP_3}\\
\cline{5-6}
 & & & & \multicolumn{2}{c}{\mathsf{((S\backslash NP_1)\backslash NP_2)\backslash NP_3} \quad {>}\mathbf{B}_\times}\\
\cline{4-6}
 & & & \multicolumn{3}{c}{\mathsf{(S\backslash NP_1)\backslash NP_2} \quad <}\\
\cline{3-6}
 & & \multicolumn{4}{c}{\mathsf{S\backslash NP_1} \quad <}\\
\cline{2-6}
 & \multicolumn{5}{c}{\mathsf{S} \quad <}
\end{array}
$$

Again, the model of chapter 5 would systematically model the wrong dependencies in this case, since it would assume that all noun phrases are arguments of the same verb.

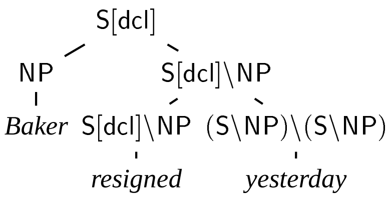
In this analysis, the categories $\mathsf{S\backslash ...\backslash NP_i}$ correspond to a stack, and in order to account for cross-serial dependencies of unlimited depth, there are no limits on the depth of the stack. However, the models proposed in this dissertation are limited to a finite number of category types. This means that if we were to train a model of Dutch, we could only obtain analyses for cross-serial dependencies up to the depth witnessed in the training corpus. This is not a problem, since we want to use this model in an application such as parsing (and assume that the training corpus is a representative sample of the language, so that no cross-serial dependencies of greater depth occur). Probabilistic language models can also be used in applications such as machine translation or generation to rank automatically generated candidate sentences. In that case, we certainly do not want a model which accepts sentences that are incomprehensible to humans. From a linguistic point of view, this is also not a problem if we either consider a probabilistic model of grammar a model of performance, rather than competence.

6.4 Modelling dependencies in the predicate-argument structure

This section explains how local and non-local dependencies can be captured in a generative model. How these probabilities are estimated in the current implementation is discussed in section 6.7.

6.4.1 Modelling local dependencies

Before explaining how the model captures non-local dependencies, we define the probabilities for purely local dependencies without coordination. By excluding non-local dependencies and coordination, at most one dependency relation holds for each word. Consider the following sentence:



This derivation expresses the following dependency relations:

(104) a. $\langle \mathsf{N}, \mathit{Baker} \rangle = \mathit{Arg}(1, \langle \mathsf{S[dcl]\backslash NP}, \mathit{resigned} \rangle)$

b. $\langle \mathsf{S[dcl]\backslash NP}, \mathit{resigned} \rangle = \mathit{Arg}(2, \langle \mathsf{(S\backslash NP)\backslash(S\backslash NP)}), \mathit{yesterday} \rangle$

We assume again that heads are generated before their modifiers or arguments, and that word-word dependencies are expressed by conditioning modifiers or arguments on heads. Therefore, these dependencies can be captured in the following manner:

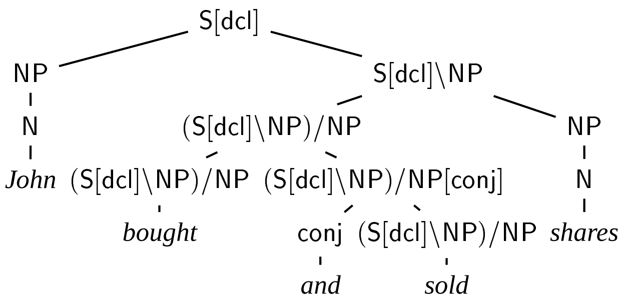
(105) a. $P(w = \mathit{Baker} \mid c = \mathsf{N}, \langle c, w \rangle = \mathit{Arg}(1, \langle \mathsf{S[dcl]\backslash NP}, \mathit{resigned} \rangle))$

b. $P(w = \mathit{yesterday} \mid c = \mathsf{(S\backslash NP)\backslash(S\backslash NP)},\ \langle \mathsf{S[dcl]\backslash NP}, \mathit{resigned} \rangle = \mathit{Arg}(2, \langle c, w \rangle))$

Baker is an argument of *resigned*; however, in the adjunct case, the lexical head of the functor is conditioned on its argument.

6.4.2 Modelling non-local dependencies

As described in section 2.6, the predicate-argument structure that corresponds to a derivation contains not only local, but also long-range dependencies that are projected from the lexicon or through some rules such as the coordination of functor categories. For example, in the following derivation, *John* is the subject of *bought* and of *sold*, and *shares* is the object of *bought* and of *sold*:

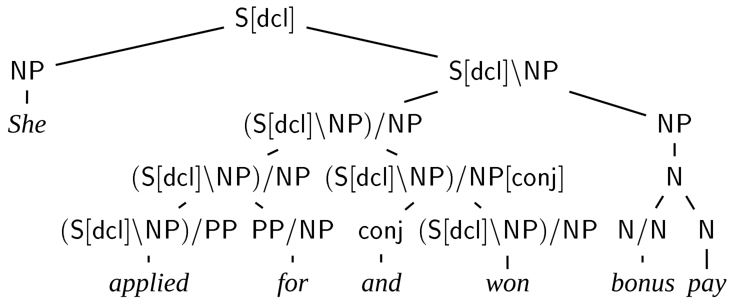


We want to express these multiple dependencies, *John* and *shares* have to be conditioned on *bought* and on *sold*:

$$\begin{aligned}
P(John \quad \mid \quad & c = \mathsf{N}, \\
& \langle c,w\rangle = Arg(1,\langle(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}, bought\rangle), \\
& \langle c,w\rangle = Arg(1,\langle(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}, sold\rangle)) \\
P(w = shares \quad \mid \quad & c = \mathsf{N}, \\
& \langle c,w\rangle = Arg(2,\langle(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}, bought\rangle), \\
& \langle c,w\rangle = Arg(2,\langle(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}, sold\rangle))
\end{aligned}$$

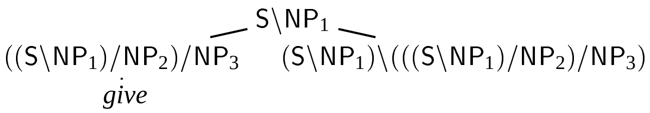
In terms of the predicate-argument structure, *bought* and *sold* are both lexical heads of this sentence. Since neither fills an argument slot of the other, we assume that they are generated independently. This is different from the dependency model in the previous chapter, which conditions the head word of the second and subsequent conjuncts on the head word of the first conjunct. Similarly, in a sentence such as *Miller and Baker resigned*, the current model assumes that the two heads of the subject noun phrase are conditioned on the verb, but not on each other (as is the case in the previous dependency model). Note that in order to capture all dependencies in the above example, *bought* and *sold* have to be generated before any of their arguments. This is also different from the model in the previous chapter, which disregards the dependencies between *sold* and its arguments.

However, it is not always sufficient to simply generate all lexical heads of a constituent at its maximal projection to capture all dependencies involved in right node raising constructions. In the following example, *bonus pay* is an argument of *for*, not of *applied*:

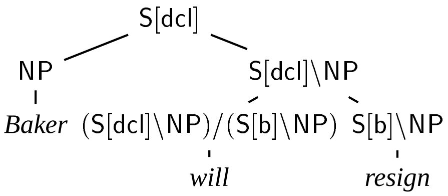


Therefore, head constituents need to be fully expanded before sisters can be generated. This is another difference to the model of the previous chapter, which assumes that word-word dependencies can be expressed at the level of local trees.

Argument-cluster coordination constructions such as *give a dog a bone and a policeman a flower* are another example where the dependencies in the predicate-argument structure cannot be expressed at the level of the local trees that combine the individual arguments. Instead, these dependencies are projected down through the category of the argument cluster:

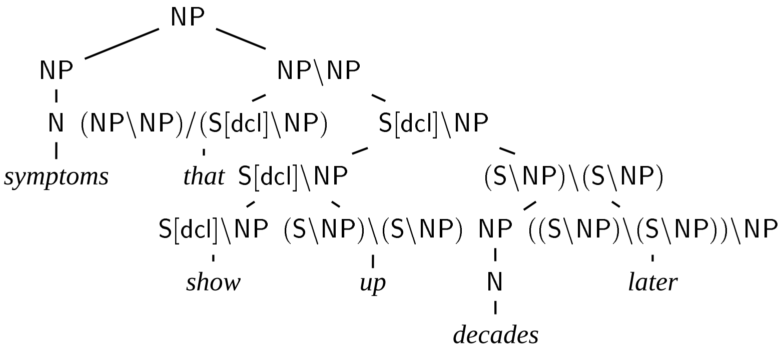


Lexical categories that project non-local dependencies include cases such as relative clauses, control verbs, auxiliaries, modals and raising verbs. For instance, in the following sentence, *Baker* is the subject of *resign*:



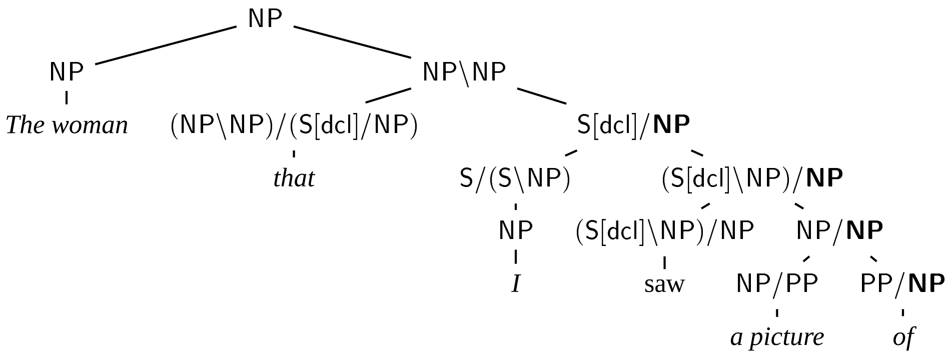
Again, in order to capture this dependency, we assume that the entire verb phrase is generated before the subject.

In relative clauses, there is a dependency between the verbs in the relative clause and the head of the noun phrase that is modified by the relative clause:



Since the entire relative clause is an adjunct, it is generated after the noun phrase *symptoms*. Therefore, we cannot capture the dependency between *symptoms* and *show* in the same manner as in an ordinary sentence. Instead, *show* needs to be conditioned on the fact that its object is *symptoms*. This is similar to the way in which head words of adjuncts such as *yesterday* are generated. In addition to this dependency, we also assume that there is a dependency between *that* and *show*.

The dependency model of the previous chapter assumes that head words can be generated at the maximal projection of a constituent. However, if we want to capture unbounded long-range dependencies such as object extraction, this is no longer true. Consider the following example:



Here, there is a S[dcl]/NP with lexical head (S[dcl]\NP)/NP; however, its NP argument is not the object of the transitive verb.

This problem can be solved by generating words at the leaf nodes instead of at the maximal projection of constituents. After expanding the (S[dcl]\NP)/NP node to (S[dcl]\NP)/NP and NP/NP, the NP that is co-indexed with *woman* cannot be unified with the object of *s*aw anymore.

These examples have shown that two changes to the generative process are necessary if word-word dependencies in the predicate-argument structure are to be captured. First, head constituents have to be fully expanded before non-head constituents are generated. Second, words have to be generated at the leaves of the tree, not at the maximal projection of constituents.

Not all words have functor categories or fill argument slots of other functors. For instance, punctuation marks, conjunctions, and the heads of entire sentences are not conditioned on any other words. Therefore, they are only conditioned on their lexical categories. Therefore, this model contains the following three kinds of word probabilities:

(106) a. Argument probabilities: $P(w \mid c, \langle c, w\rangle = Arg(i, \langle c', w'\rangle))$

The probability of generating word w, given that its lexical category is c and that this word-category pair fills the ith argument slot of $\langle c', w'\rangle$

b. Functor probabilities: $P(w \mid c, \langle c_h, w_h\rangle = Arg(i, \langle c, w\rangle))$

The probability of generating word w, given that its lexical category is c and that the ith argument slot of its lexical category is filled by $\langle c_h, w_h\rangle$

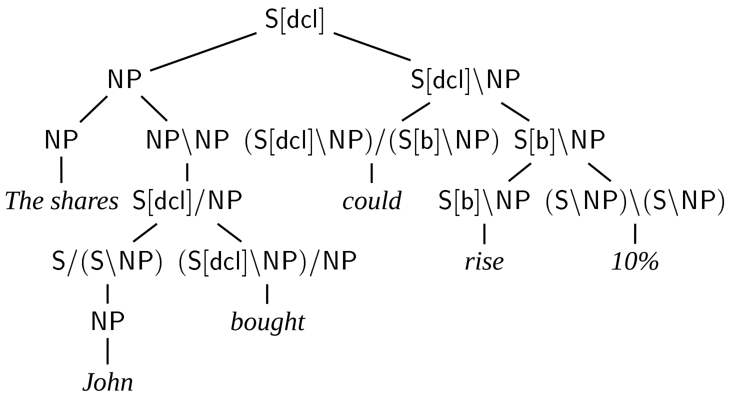
c. Other word probabilities: $P(w \mid c)$.
If a word does not fill any dependency relation, it is only conditioned on its lexical category.

6.5 The structural probabilities

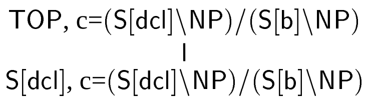
Like in the models described in chapter 5, we assume an underlying process which generates CCG derivation trees starting from the root node. Each node in a derivation tree has a category, a list of lexical heads and a (possibly empty) list of dependency relations to be fulfilled by its lexical heads. As discussed in the previous section, head words cannot in general be generated at the maximal projection if unbounded long-range dependencies are to be captured. Note that this is not the case for lexical categories. Therefore we assume that a node's lexical head category is generated at its maximal projection, whereas head words are generated at the leaf nodes. Therefore, we assume the same structural probabilities as in the **LexCat** model of the previous chapter.

6.6 An example

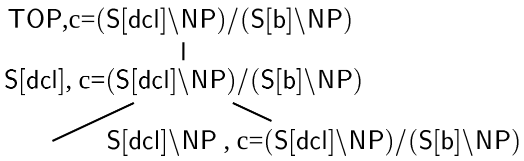
I will use the following example to explain how this model generates multiple dependencies arising through extraction and auxiliary verbs:



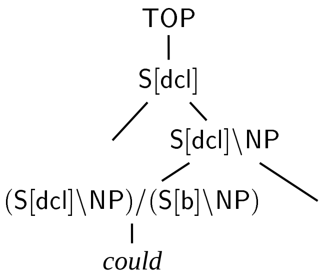
We start at the root node and generate its lexical head category (S[dcl]\NP)/(S[b]\NP). After generating its expansion (unary), the head category of its child, S[dcl], is generated:



Next, a binary expansion with head daughter on the right is chosen, and the category of the head daughter is generated:


TOP,c=(S[dcl]\NP)/(S[b]\NP)
S[dcl], c=(S[dcl]\NP)/(S[b]\NP)
S[dcl]\NP , c=(S[dcl]\NP)/(S[b]\NP)

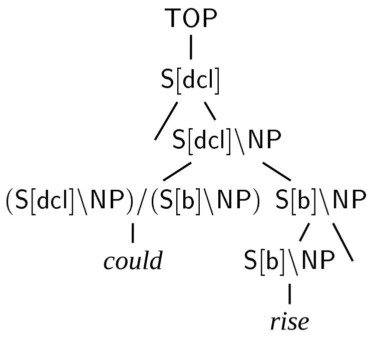
In the models of chapter 5, the non-head daughter of S[dcl] would be generated now. However, the new model proceeds differently. Here, the head daughter is expanded first (in the following, lexical heads are omitted when clear):



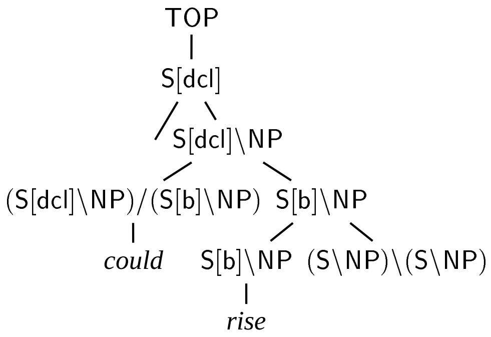
Now the non-head daughter of S[dcl]\NP, S[b]\NP, is generated. After choosing the expansion of S[b]\NP (binary, with head left), its head daughter is generated. It is a leaf node, so a word is generated. This word fills the second argument of ⟨(S[dcl]\NP)/(S[b]\NP), *could*⟩. Therefore, the probability of *rise* is as follows:

$$P(w = \textit{rise} \mid c = \mathsf{S[b]\backslash NP}, \langle c, w\rangle = \textit{arg}_2(\langle(\mathsf{S[dcl]\backslash NP})/(\mathsf{S[b]\backslash NP}), \textit{could}\rangle))$$

This results in the following tree:



In the next step, the non-head daughter of the non-terminal S[b]\NP is generated:



Now, $(\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})$ is expanded further. Since it is an adjunct, its head word, *%* is generated with the following functor probability:

$$P(w = \% \mid c = (\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP}), \langle \mathsf{S[b]}\backslash\mathsf{NP}, \mathit{rise}\rangle = \mathit{Arg}(2, \langle c, w\rangle))$$

We concentrate on the following step, the generation of the subject NP. As explained in section 2.6.6, auxiliary and modal verbs project a dependency between the subject and the main verb. Therefore the head of the subject NP $\langle \mathsf{N}, \mathit{shares}\rangle$ fills two argument slots, and is generated with the following probability:

$$P(w = \mathit{shares} \mid c = \mathsf{N}, \langle c, w\rangle = \mathit{Arg}(1, \langle (\mathsf{S[dcl]}\backslash\mathsf{NP})/(\mathsf{S[b]}\backslash\mathsf{NP}), \mathit{could}\rangle), \langle c, w\rangle = \mathit{Arg}(1, \langle \mathsf{S[b]}\backslash\mathsf{NP}, \mathit{rise}\rangle))$$

Section 6.7.2 explains how this probability can be estimated.

After generating the determiner, the zero relative clause is expanded. Note that the direct object of its head, $\langle (\mathsf{S[dcl]}\backslash\mathsf{NP})/\mathsf{NP}, \mathit{bought}\rangle$, is already instantiated with *shares*. Therefore, *bought* is conditioned on this dependency:

$$P(w = \mathit{bought} \mid c = (\mathsf{S[dcl]}\backslash\mathsf{NP})/\mathsf{NP}, \langle \mathsf{N}, \mathit{shares}\rangle = \mathit{Arg}(2, \langle c, w\rangle))$$

Section 6.7.1 shows how this probability can be estimated, and how it differs from the probability with which the head of a relative clause is generated in the dependency model of the previous chapter.

Finally, the subject of the relative clause is generated. The probability of its head word, *John* is as follows:

$$P(w = \mathit{John} \mid c = \mathsf{NP}, \langle c, w\rangle = \mathit{Arg}(1, \langle (\mathsf{S[dcl]}\backslash\mathsf{NP})/\mathsf{NP}, \mathit{bought}\rangle))$$

In contrast to the model of the previous chapter, this dependency is modelled with the same distribution that would be used in an ordinary main clause.

6.7 Parameter estimation

This sections explains how the probabilities of multiple dependencies can be estimated, and how the probabilities of functor or argument head words are estimated.

6.7.1 Estimating functor and argument probabilities

This model generates words in three different manners—as arguments of functors that are already generated, as functors which have already one (or more) arguments instantiated, or independent of the surrounding context. The last case is simple (and identical to the models in the previous chapter), as this probability can be estimated directly, by counting the number of times *c* is the lexical category of *w* in the training corpus, and dividing this by the number of times *c* occurs as a lexical category in the training corpus:

$$P(w \mid c) = \frac{C(w,c)}{C(c)}$$

In order to estimate the probability of an argument *w*, we count the number of times it occurs with lexical category *c* and is the *i*th argument of the lexical functor $\langle c', w' \rangle$ in question, divided by the number of times the *i*th argument of $\langle c', w' \rangle$ is instantiated with a constituent whose lexical head category is *c*:

$$P(w \mid c, Arg(i, \langle c', w' \rangle)) = \frac{C(\langle c, w \rangle = Arg(i, \langle c', w' \rangle))}{\sum_{w''} C(\langle c, w'' \rangle = Arg(i, \langle c', w' \rangle))}$$

The probability of a functor *w*, given that its *i*th argument is instantiated by a constituent whose lexical head is $\langle c', w' \rangle$ can be estimated in a similar manner:

$$P(w \mid c, \langle c', w' \rangle = Arg(i, \langle c, w \rangle)) = \frac{C(\langle c', w' \rangle = Arg(i, \langle c, w \rangle))}{\sum_{w''} C(\langle c', w' \rangle = Arg(i, \langle c, w'' \rangle)))}$$

Here we count the number of times the *i*th argument of $\langle c, w \rangle$ is instantiated with $\langle c', w' \rangle$, and divide this by the number of times that $\langle c', w' \rangle$ is the *i*th argument of any lexical head with category *c*. For instance, in order to compute the probability of *yesterday* modifying *bought* as in the previous section, we count the number of times the transitive verb *bought* was modified by the adverb *yesterday* and divide this by the number of times *bought* was modified by any adverb of the same category. We have seen that functor probabilities are not only necessary for adjuncts, but also for certain types of non-local dependencies such as the relation between the noun modified by a relative clause and the verb in the relative clause. In the case of zero or reduced relative clauses, some of these dependencies are also captured by the dependency model presented in the previous chapter. However, in that model only counts from the same type of construction could be used, whereas in the new model, the functor probability for a verb in a zero or reduced relative clause can be estimated from all occurrences of the head noun. In particular, all instances of the the noun and verb occurring together in the training data (with the same predicate-argument relation between them, but not necessarily with the same surface configuration) are taken into account by the new model.

Both functor and argument probabilities are interpolated with the word probabilities $P(w|c)$ using the same interpolation technique as described in section 5.8.

6.7.2 Conditioning events on multiple heads

In the presence of non-local dependencies and coordination, the new model requires the conditioning of certain events on multiple heads. Since it is unlikely that such probabilities can be estimated reliably from data, they have to be approximated in some manner. We present four different methods for doing this – standard linear interpolation, a backed-off scheme similar to Collins and Brooks (1995), an approach based on products of experts (Hinton, 1999), as well as a recent proposal by Schmid (2002). However, only the first approach, simple linear interpolation is implemented in the current parser.

Linear interpolation If we assume that all dependencies dep_i that hold for a word are equally likely, we can approximate $P(w \mid c, dep_1, ..., dep_n)$ as the average of the individual dependency probabilities:

$$P(w \mid c, dep_1, ..., dep_n) \approx \frac{1}{n} \sum_{i=1}^{n} P(w \mid c, dep_i)$$

This approximation is used in the current implementation. It has the advantage that it is easy to compute, but might not give a good estimate, since it averages over all individual distributions. Note that we could also use different weights for the individual distributions. These could either be estimated on held-out data, or using the following technique proposed by Collins and Brooks (1995).

Collins and Brook's estimation technique Collins and Brooks (1995) consider the problem of PP-attachment disambiguation (Hindle and Rooth, 1993). Given a sequence `V N1 P N2`, such as *join, board, as, director*, the task is to decide whether a PP consisting of the preposition `P` and a noun phrase headed by the noun `N2` attaches to the verb `V` or to the immediately preceding noun phrase headed by N1. Collins and Brooks propose to estimate the attachment probability $P(A|\mathtt{V},\mathtt{N1},\mathtt{P},\mathtt{N2})$ (where A can take on two values 0 (verb attachment) or 1 (noun attachment)) with a backed-off model. Backed-off estimation was originally proposed by Katz (1987) for language modelling. The underlying idea is that relative frequency estimates are less reliable for events with a low frequency in the training data. Therefore, if the relative frequency $f(X_1,X_2)$ is lower than a given threshold, backed-off estimation replaces the relative frequency estimate $\frac{f(Y,X_1,X_2)}{f(X_1,X_2)}$ of the conditional probability $P(Y \mid X_1,X_2)$ with an estimate of $P(Y \mid X_1)$. Collins and Brooks apply this estimation technique to the problem of PP-attachment. Since $P(A|\mathtt{V},\mathtt{N1},\mathtt{P},\mathtt{N2})$ has only two possible outcomes, they give a formulation in terms of a binary classification task, where noun attachment is chosen if $P(1|\mathtt{V},\mathtt{N1},\mathtt{P},\mathtt{N2}) > 1$.

1. If $f(\mathtt{V},\mathtt{N1},\mathtt{P},\mathtt{N2}) > 0$:
$$P(1|\mathtt{V},\mathtt{N1},\mathtt{P},\mathtt{N2}) = \frac{f(1,\mathtt{V},\mathtt{N1},\mathtt{P},\mathtt{N2})}{f(\mathtt{V},\mathtt{N1},\mathtt{P},\mathtt{N2})}$$

2. Else if $f(\mathtt{V},\mathtt{N1},\mathtt{P}) + f(\mathtt{V},\mathtt{P},\mathtt{N2}) + f(\mathtt{N1},\mathtt{P},\mathtt{N2}) > 0$:
$$P(1|\mathtt{V},\mathtt{N1},\mathtt{P},\mathtt{N}_2) = \frac{f(1,\mathtt{V},\mathtt{N1},\mathtt{P}) + f(1,\mathtt{V},\mathtt{P},\mathtt{N2}) + f(1,\mathtt{N1},\mathtt{P},\mathtt{N2})}{f(\mathtt{V},\mathtt{N1},\mathtt{P}) + f(\mathtt{V},\mathtt{P},\mathtt{N2}) + f(\mathtt{N1},\mathtt{P},\mathtt{N2})}$$

3. Else if $f(\mathtt{V},\mathtt{P}) + f(\mathtt{N1},\mathtt{P}) + f(\mathtt{P},\mathtt{N2}) > 0$:
$$P(1|\mathtt{V},\mathtt{N1},\mathtt{P},\mathtt{N}_2) = \frac{f(1,\mathtt{V},\mathtt{P}) + f(1,\mathtt{N1},\mathtt{P}) + f(1,\mathtt{P},\mathtt{N2})}{f(\mathtt{V},\mathtt{P}) + f(\mathtt{N1},\mathtt{P}) + f(\mathtt{P},\mathtt{N2})}$$

4. Else if $f(\mathtt{P}) > 1$:
$$P(1|\mathtt{V},\mathtt{N1},\mathtt{P},\mathtt{N}_2) = \frac{f(1,\mathtt{P})}{f(\mathtt{P})}$$

5. Else $P(1|\mathtt{V},\mathtt{N1},\mathtt{P},\mathtt{N}_2) = 1$ (the default attachment)

Of interest to us is the manner in which the individual estimates used in the backing-off scheme are defined. Consider the third case. Here, $P(1|\mathtt{V},\mathtt{N1},\mathtt{P},\mathtt{N}_2)$ is defined in terms of the counts $f(1,\mathtt{V},\mathtt{P})$, $f(1,\mathtt{N1},\mathtt{P})$, $f(1,\mathtt{P},\mathtt{N2})$, $f(\mathtt{V},\mathtt{P})$, $f(\mathtt{N1},\mathtt{P})$ and $f(\mathtt{P},\mathtt{N2})$. These are exactly the counts that would be used if $P(1|\mathtt{V},\mathtt{N1},\mathtt{P},\mathtt{N}_2)$ was estimated as the average of the relative frequency estimates $\hat{P}(1|\mathtt{V},\mathtt{P})$, $\hat{P}(1|\mathtt{N1},\mathtt{P})$ and $\hat{P}(1|\mathtt{P},\mathtt{N2})$. However, Collins and Brooks note that on their task, this method outperforms an averaging estimate which assigns the same weight to each of the individual estimates. By contrast, the method of Colins and Brooks amounts to a linear interpolation where the smoothing weights are chosen so that estimates which are based on more occurrences in the training data (and are therefore more reliable) are given more importance.

Our problem is not a binary classification task, and is therefore closer to the original formulation of backed-off estimation for language modelling. Instead of estimating the probability of w given its lexical category c and a set of dependencies $dep_1,...,dep_n$ as the average of the smoothed estimates of the individual dependency probabilities, we could estimate the conditional probability of all dependencies in a manner similar to Collins and Brooks, eg.:

1. If $\sum_i f(c, dep_i) > 0$:

$$\tilde{P}_{BackOff}(w \mid c, dep_1,...,dep_n) = \frac{f(w,c,dep_1),..., f(w,c,dep_n)}{f(c,dep_1), f(c,dep_n)}$$

2. Else:

$$\tilde{P}_{BackOff}(w \mid c, dep_1,...,dep_n) = \frac{f(w,c)}{f(c)} = \hat{P}(w|c)$$

In the first case, we could either use $\tilde{P}_{BackOff}(w \mid c, dep_1,...,dep_n)$ directly, or use linear interpolation to combine it with $\hat{P}(w|c)$.

Since Collins and Brooks state that their technique performed better than simple averaging for their task, we would like to investigate whether an adaptation of this technique could also improve the performance of our parser.

Products of Experts It is possible to combine multiple probabilistic classifiers using the following product rule:

$$P(w \mid c_1,...,c_n) \approx \frac{\prod_{i=1}^{n} P(w \mid c_i)}{\sum_{w'} \prod_{i=1}^{n} P(w' \mid c_i)}$$

This product rule underlies the Products of Experts models proposed by Hinton (1999) (which focuses on the unsupervised training of such models). In contrast to combining classifiers by linear interpolation, this approach does not flatten out the individual distributions. By multiplying the individual model probabilities together, the probability mass of the combined model is forced into the region which is assigned non-zero probability by all models – whereas averaging over all models spreads the probability mass over all regions which are assigned non-zero probability by any of the individual models.

We could do the same, and combine the probabilities of the individual dependency probabilities for a word in the following way:

$$P(w \mid dep_1,...,dep_n) \approx \frac{\prod_{i=1}^{n} P(w \mid dep_i)}{\sum_{w'} \prod_{i=1}^{n} P(w' \mid dep_i)}$$

However, while this modelling technique is in principle more attractive than linear interpolation, the computation of the normalization factor is likely to be very slow, since it requires a summation over all words in the vocabulary for all possible combinations of individual dependencies. Furthermore, as the number of dependencies on which a word is generated is not fixed, a product of experts would have to be generated on the fly every time a word is conditioned on multiple dependencies. Therefore, this approach has not yet been integrated into our parser.

Using mutual information and clustering Schmid (2002) is relevant for our current purposes in providing alternative ways of calculating the probability of a word which is the argument of more than one word and of calculating the probability of a word given its arguments.

First, Schmid shows that the probability $P(w \mid C, g)$ of a word w given a lexical category C (which could also be a dependency relation) and another word h (which he calls the governor can be expressed as the product of $P(w \mid C)$ and the exponential of the pointwise mutual information between w and h ($EMI(w,h)^2$):

$$\begin{aligned} P(w \mid g, C) &= \frac{P(w, g \mid C)}{P(g \mid C)} \\ &= \frac{P(w \mid C)P(w, g \mid C)}{P(w \mid C)P(g \mid C)} \\ &= P(w \mid C)EMI(w, g \mid c) \end{aligned}$$

He demonstrates that this can be extended to express the probability of a word w given two lexical governors g_1 and g_2 in the following way:

$$\begin{aligned} P(w \mid g_1, g_2) &= P(w)\frac{P(w,g_1)}{P(w)P(g_1)}\frac{P(w,g_2)}{P(w)P(g_2)}\frac{P(w,g_1,g_2)P(w)}{P(w,g_1)P(w,g_2)}\frac{P(g_1)P(g_2)}{P(g_1,g_2)} \\ &= P(w)EMI(w,g_1)EMI(w,g_2)\frac{EMI(g1,g2 \mid w)}{EMI(g_1,g_2)} \end{aligned}$$

Note that this requires the computation of the (exponential of) the mutual information between the two head words g_1 and g_2 given w, the word to be predicted, which might be hard to estimate in practice.

[2] The pointwise conditional mutual information $I(x;y \mid z)$ between the values x and y of two random variables given the value z of a third random variable is defined as follows:

$$I(x;y \mid z) = \log \frac{P(x,y \mid z)}{P(x \mid z)P(y \mid z)}$$

However, Schmid's approach is of limited use for our current purposes. Since he is only concerned with control verbs, Schmid assumes that a word can depend on at most two governors, and suggests that this technique could not easily be generalized to words which depend on more than two other words. But the non-local dependencies we are interested in encompass a much wider class than control verbs, and we cannot assume that a word is only dependent on at most two other words. Furthermore, since our dependencies are formulated in terms of relations in the predicate-argument structure, we would have to compute the mutual information not only between pairs of words, but between pairs of lexical heads, ie. word-category pairs, and argument slots.

In order to compute the probability of a word w given its category C and arguments $a_1 ... a_n$, Schmid advocates the use of a clustering model, so that the relationship between w and each of its arguments is expressed in terms of a relationship between different clusters rather than between the words themselves. He proposes to compute the probability $P(w \mid a_1, ... a_n, C)$ in the following manner:

$$\begin{aligned} P(w \mid a_1 ... a_n) &= \frac{P(w, a_1, ... a_n)}{P(a_1, ... a_n)} \\ &= \frac{\sum_c P(c)P(w \mid c)P(C \mid c) \prod_i \sum_r P(r \mid c, i, C)P(a_i \mid r)}{\sum_c P(c)P(C \mid c) \prod_i \sum_r P(r \mid c, i, C)P(a_i \mid r)} \end{aligned}$$

where c are word classes and r "argument classes" (such as WordNet classes). Like the model presented here, Schmid captures dependencies between words and arguments in terms of lexical categories and argument slots, so (although he does not mention categorial grammar), this could easily be applied to CCG. Unfortunately, no experimental results are provided to show how either technique performs in practice, therefore it is difficult to assess their empirical value.

Furthermore, Schmid does not provide a uniform account of word-word dependencies which could be generalized from the limited types of constructions he considers to the dependencies captured in the model proposed in this chapter. For these reasons, it is unclear at the time of writing whether his approach would be amenable to the present framework.

6.8 Dynamic programming and beam search

This section describes how this model is integrated into our CKY chart parser. As explained in the previous chapter, dynamic programming and effective beam search strategies are essential to guarantee efficient parsing with highly ambiguous wide-coverage grammars. Both use the inside probability of constituents. In lexicalized models such as those presented in the previous chapter, where each constituent has exactly one lexical head, and where this lexical head can only depend on the lexical head of one other constituent, the inside probability of a constituent is the probability that a node with the label and lexical head of this constituent expands to the tree below this node. The probability of generating a node with this label and lexical head is given by the outside probability of the constituent.

In the model defined here, the lexical head of a constituent can depend on more than one other word. As explained in section 6.4.2, there are instances where the categorial functor is conditioned on its arguments – the example given above showed that verbs in relative clauses are conditioned on the lexical head of the noun which is modified by the relative clause. Therefore, the inside probability of a constituent cannot include the probability of any lexical head whose argument slots are not all filled.

This means that the equivalence relation defined by the probability model needs to take into account not only the head of the constituent itself, but also all other lexical heads within this constituent which have at least one unfilled argument slot. As a consequence, dynamic programming becomes less effective.

There is a related problem for the beam search: in the new model, the inside probabilities of constituents within the same cell cannot be directly compared anymore. Instead, the number of unfilled lexical heads needs to be taken into account. If a lexical head $\langle w,c\rangle$ is unfilled, the evaluation of the probability of w is delayed. This is a problem for the beam search strategy (which would be even more important, given that dynamic programming works less well).

The fact that constituents can have more than one lexical head causes similar problems for dynamic programming and the beam search.

Also, note that the new model requires more space per constituent than the old models since the data structures for the category objects need to be created for each proposed constituent. Since the probability models of the previous chapter treat categories as atomic symbols, this is not necessary – the predicate-argument structure of the best parse can be built after it has been selected.

In order to be able to parse efficiently with the new model, we use the following approximations for dynamic programming and the beam search:

Two constituents with the same span and the same category are considered equivalent if they delay the evaluation of the probabilities of the same words and if they have the same number of lexical heads, and if the first two elements of their lists of lexical heads are identical (the same words and lexical categories). Note that this is an approximation to true equivalence, since we do not check the entire list of lexical heads. Furthermore, if a cell contains more than 100 constituents, we iteratively narrow the beam (by halving it in size) until the beam search has no further effect or the cell contains less than 100 constituents. This is a very aggressive strategy, and it is not clear how much it affects parsing accuracy. However, more lenient strategies required too much space for the chart to be held in memory. A better way of dealing with the space requirements of this new model would be to implement a packed shared parse forest, but we leave this to future work.

6.9 An experiment

The experimental setup is similar to the previous chapter. We use sections 02-21 of CCGbank for training, section 00 for development, and section 23 for testing.[3] However, since parsing with this model is less efficient, only sentences $\leq$ 40 tokens only are used to test the model. A frequency cutoff of $\leq$ 20 was used to determine rare words, which were dealt with in the same manner as in the previous chapter.

In order to investigate the impact of capturing different kinds of non-local dependencies, four different models were investigated: The baseline model is the **LexCat** model of the previous chapter (tested on the same sentences and with the same frequency cutoff), since the structural probabilities of the new model are like those of this model. This model was tested with the original beam search strategy reported in the previous chapter. **Local** only takes local dependencies into account. **LeftArgs** only takes non-local dependencies that are projected through left arguments ($\backslash \mathsf{X}$) into account. This includes for instance long-range dependencies projected by subjects, subject and object control verbs, subject extraction, left-node raising. Un English, all of these are bounded. **All** takes all bounded and unbounded dependencies into account, in particular it extends **LeftArgs** by capturing also right node raising and object extraction. **Local**, **LeftArgs** and **All** are all tested with the aggressive beam strategy described above.

In all cases, the CCG derivation includes all dependencies; however, it is possible under the models that exclude certain kinds of dependencies that a word is conditioned on no dependencies. In these cases, the word is generated with $P(w|c)$.

[3] In all experiments reported in this chapter, the input is POS-tagged using the tagger of Ratnaparkhi (1996)

	LexCat	Local	LeftArgs	All
Lexical categories:	88.2	89.9	90.1	90.1
Parseval				
Labelled Precision:	76.3	78.4	78.5	78.5
Labelled Recall:	75.9	78.5	79.0	78.7
Bracketing Precision:	82.0	83.4	83.6	83.2
Bracketing Recall:	81.6	83.6	83.8	83.4
Surface dependencies				
$\langle P, H, S \rangle$	78.7	81.0	81.5	81.1
$\langle S \rangle$	83.9	85.4	85.6	85.2
$\langle \rangle$	87.5	88.5	88.8	88.4
Unlabelled 0 wrong	34.4	38.1	38.9	39.0
Unlabelled ≤ 2 wrong:	65.1	68.1	69.5	68.2
Predicate-argument structure (all)				
Labelled Precision:	77.3	80.8	81.6	81.5
Labelled Recall:	78.2	80.6	81.5	81.4
Unlabelled Precision:	88.0	89.7	90.2	90.1
Unlabelled Recall:	89.1	89.5	90.1	90.0
Local dependencies in pred.-arg. structure				
Labelled Precision:	78.9	82.5	83.0	82.9
Labelled Recall:	79.5	82.3	82.7	82.6
Unlabelled Precision:	89.2	91.1	91.3	91.2
Unlabelled Recall:	89.8	90.8	91.0	90.8
All non-local dependencies in pred.-arg. structure				
Labelled Precision:	60.8	62.6	67.1	66.3
Labelled Recall:	64.4	63.0	68.5	68.8
Unlabelled Precision:	75.8	74.9	79.4	78.6
Unlabelled Recall:	80.9	75.5	81.0	81.4
Locally mediated dependencies in pred.-arg. structure				
Labelled Precision:	63.9	64.8	69.0	69.2
Labelled Recall:	65.9	64.1	70.2	70.0
Unlabelled Precision:	80.1	77.7	81.8	81.8
Unlabelled Recall:	82.9	77.2	83.1	83.1
Long-range dependencies in pred.-arg. structure				
Labelled Precision:	46.0	50.4	55.6	52.4
Labelled Recall:	54.7	55.8	58.7	61.2
Unlabelled Precision:	55.5	59.4	65.1	62.3
Unlabelled Recall:	67.7	64.7	67.6	70.5

Table 6.1: Performance of the models (sec. 23, $\leq$ 40 words, with POS tagger).

Table 6.1 gives the performance of all four models on section 23 (POS-tagged input) in terms of the accuracy of lexical categories, Parseval scores, the recovery of word-word dependencies defined in terms of local trees and in terms of the recovery of word-word dependencies in the predicate-argument structure. Here, results are further broken up into the recovery of local, all non-local, locally mediated and unbounded long-range dependencies (see section 2.6.7 for a detailed description of how dependencies are classified in our system).

LexCat does not capture any word-word dependencies. Its performance on the recovery of predicate-argument structure can be improved by 3% by capturing only local word-word dependencies (**Local**). Note that this excludes certain kinds of dependencies that were captured by the dependency model of the previous chapter. For instance, the dependency between the head of a noun phrase and the head of a reduced relative clause (*the shares bought by John*) is captured by the previous dependency model, since *shares* and *bought* are both heads of the local trees that are combined to form the complex noun phrase. However, in the previous dependency model the probability of this dependency can only be estimated from occurrences of the same construction, since dependency relations are defined in terms of local trees and not in terms of the underlying predicate-argument structure. By including non-local dependencies on left arguments (such as subjects) (**LeftArgs**), a further improvement of 0.7% on the recovery of predicate-argument structure is obtained. This model captures the dependency between *shares* and *bought*. In contrast to the dependency model of the previous chapter, it can use all instances of *shares* as the subject of a passive verb in the training data to estimate this probability. Therefore, even if *shares* and *bought* do not co-occur in this particular construction in the training data, the event that is modelled by the new dependency model might not be unseen, since it could have occurred in another syntactic context.

Our results indicate that in order to perform well on non-local dependencies, they have to be included in the model, since **Local**, the model that captures only local dependencies performs worse on non-local dependencies than **LexCat**, the model that captures no word-word dependencies. However, with more than 5% difference on labelled precision and recall on non-local dependencies, the model which captures non-local dependencies on left arguments performs significantly better on recovering non-local dependencies than **Local**. Note that the greatest difference in performance between the models which do capture non-local dependencies and the models which do not is on non-local dependencies. This indicates that, at least in the kind of model considered in this chapter, it is very important to model not just local, but also long-range dependencies.

It is not clear why **All**, the model that includes all dependencies performs slightly worse than the model which includes only long-range dependencies on subjects. This indicates that more work is required to investigate the benefit and cost of capturing different kinds of non-local dependencies for parsing.

6.10 Discussion

This chapter has defined generative probability models of the word-word dependencies as determined by the predicate-argument structure that corresponds to a derivation. In this model, multiple dependencies between a word and its surrounding context can be captured, including those dependencies that arise through control, raising, extraction, right node raising and other constructions. In CCG, all of these dependencies are expressed as relations between lexical functor categories and their arguments. This localization of bounded and unbounded dependencies makes the definition of a generative model of these dependencies straightforward. For languages such as Dutch, we have argued that a model of this kind is required to capture even the correct local dependencies, since the dependency model of the previous chapter would systematically capture semantically incorrect dependencies.

The experiments presented in this chapter demonstrate that the performance of a simple baseline model can be improved significantly if non-local word-word dependencies are also captured, even though the current implementation uses a very simple method of estimating the probabilities of multiple dependencies. We have discussed some alternatives in this chapter which might yield better results. Future work ought to address the question whether these models can be run with a less aggressive beam search strategy, or whether a different parsing algorithm is more suitable. An alternative approach might be to use the new model to re-rank the output of a parser that uses a simpler model to propose candidate analyses.

Generative models such as those developed in this dissertation can not only be used for statistical parsing, but also for other applications such as language modelling. Whereas statistical parsing deals with predicting the structure of a given string, language modelling is concerned with predicting the next word given a sequence of previous words, and models which capture grammatical knowledge such as Charniak (2001), Roark (2001) or Wang and Harper (2002) are beginning to perform at least as well, if not better, as simple trigram models. If the theory of what constitutes an allowable structure (ie. the grammar) is already very constraining, it might not be necessary to model everything that is represented by these structures when deciding which structure to prefer, and non-lexical features might also provide good indicators for the plausibility of the proposed structures. However, if the task is to predict the string itself, a model which generates the words of that string according to local and long-range semantic dependencies between those words might outperform a model which can only take some of these dependencies into account. We therefore believe that language modelling is a task where we might be able to reap the full benefits of the model presented in this chapter.

Chapter 7

Conclusion

This chapter summarizes the main contributions made by this dissertation and suggests some directions for further research.

The main thesis presented here is that generative models over normal-form derivations are an appropriate kind of model for statistical parsing with Combinatory Categorial Grammar (CCG). This thesis is supported by experimental evidence which shows that state-of-the-art performance as measured by recovery of word-word dependencies can indeed be achieved by a simple generative model of bilexical dependencies.

Chapter 3 demonstrates how the Penn Treebank (Marcus *et al.*, 1993), the largest available manually annotated corpus for English, can be translated to a corpus of CCG normal-form derivations, using the information represented by the null elements of the Penn Treebank annotation to yield semantically adequate analyses of long-range dependencies arising through extraction and coordination. Only a tiny fragment of this corpus, the sentences involving gapping, cannot be processed, since the standard analysis of gapping in CCG is difficult to represent in a treebank.

Chapter 5 defines a number of generative probability models over CCG derivation trees and presents experimental results which show that a simple model which includes lexicalized structural probabilities as well as bilexical dependencies achieves similar performance on our corpus (in terms of recovery of unlabelled word-word dependencies) as the best model of Collins (1999) on the original Treebank. Since CCG derivation trees encode long-range as well as local dependencies, the output of a highly accurate wide-coverage CCG parser is ultimately more useful than the linguistically impoverished output of most current Penn Treebank parsers. For example, our parser returns not only the syntactic derivation, but also the dependency relations that constitute the predicate-argument structure. This enables us to evaluate its performance on local and on long-range dependencies. Although some attempts have been made to recover such information from ordinary Treebank parsers, a CCG parser requires neither additional features in its probability model nor a postprocessing phase to represent these dependencies.

Chapter 6 demonstrates that the model in chapter 5 captures the wrong word-word dependencies whenever composition is required, and presents an alternative generative probability model of word-word dependencies that does not suffer from this problem. This new model captures all and only the bilexical dependencies that are represented in the predicate-argument structure. These include non-local dependencies such as those involved in control, raising, extraction and coordinate constructions. In CCG, all of these dependencies are expressed as relations between lexical functor categories and their arguments. This localization of bounded and unbounded dependencies makes the definition of a generative model of these dependencies straightforward. There are a number of obstacles which currently prevent us from reaping the full benefits of such a model: the present implementation uses a very naive way of estimating the probabilities of multiple dependencies, and we also have not yet explored how an effective beam could be implemented with such a model, or whether an alternative parsing algorithm might be more suitable. However, our experimental results demonstrate that this model does perform better than a baseline model that does not capture any dependencies, even though this baseline model is not faced with the same difficulties as the dependency model. We furthermore show that a model that models at least certain kinds of long-range dependencies performs better than one which only captures local dependencies. With our current implementation, the performance of this model is lower than that of the dependency model reported in chapter 5. However, even if it is the case that a model of dependencies in the predicate-argument structure is not necessary for English, we argue that for languages with freer word order, the simple models of chapter 5 are inappropriate, and a model such as developed in chapter 6 is necessary to capture the true word-word dependencies.

Future research. In many ways, the research reported in this dissertation is only a first step. Some of the analyses in CCGbank suffer from the fact that linguistically relevant information (such as the correct structure of complex noun phrases, or a consistent distinction between complements and adjuncts) could not be gathered automatically from the Penn Treebank. Ideally, this ought to be corrected, either in the original Treebank, or in CCGbank. Conversely, some of the preprocessing of the Treebank that was necessary to obtain consistent and linguistically plausible CCG analyses might also be beneficial to the performance of standard Treebank parsers.

Although the probability models presented in Chapter 5 perform very well, further improvements are conceivable. For example, we have not yet been able to integrate structural features (such as distance measures) into a model of word-word dependencies, even though we believe that non-lexical features are also important for parse selection. We have also shown that the lexicon extracted from the training corpus is incomplete, and future research ought to address the question of how to generalize from the observed lexical entries to unobserved, but linguistically plausible, entries. This problem is likely to require more sophisticated estimation or smoothing techniques than those used in this dissertation, as well as additional linguistic knowledge. The focus of this dissertation has been on the development of generative probability models. The advantage of such models lies in their simplicity. However, log-linear models are more expressive. They might provide better smoothing and allow for an integration of additional features. But it is not clear yet whether such models (as in the recent proposal of Clark and Curran (2003)) will in practice outperform the models proposed here.

In the current implementation, the probability models presented in Chapter 6 suffer from an overly aggressive beam search strategy. Alternatively, such models could be used to re-rank the output of a parser which uses a simpler model to generate candidate analyses. We have argued that unlike models of surface dependencies, models of this kind are especially suited for languages with freer word order, such as Dutch. Another, as yet untested, application for these models is language modelling. Grammar-based language models have recently been shown to be very successful. However, current techniques cannot capture the non-local dependencies that are expressed in our model, and we would like to investigate the impact that the recovery of these dependencies and their inclusion in the model have.

Appendix A

Translating the Penn Treebank to CCG

This appendix lists the necessary modifications to the Penn Treebank that are not described in chapter 3 as well as the head-finding rules and the heuristics for determining complements that were used in the program described in chapter 3.

A.1 Head-finding rules

The head-finding rules are adapted from the rules developed by Collins (1999) and Magerman (1994). They are given for each non-terminal label, and are to be read as follows: going from left to right (←), or from right to left (→), the first constituent with label XP is head. Otherwise, search for the next constituent in the list. A ← or → next to the nonterminal label indicates the search direction for all head finding rules for this nonterminal label. If none of the constituents listed is found, the leftmost (←) or rightmost (→) constituent is chosen. In general, constituents which carry a functional tag are not identified as heads, unless this is indicated by an "all" next to the rule.

ADJP (←) SO, (←) ADJP, (←) DOLLAR, (→) VBN, (→) VBG, (→) JJ, (→) JJR, (→) JJS, (→) NN, (→) NNS, (→) NNP, (←) QP, (←) DT, (→) CD, (→) RB, (→) ADVP, (→) RBR, (←) IN, (←) VBD, (←) VBP, (←) VB, (←) NP, (←) FW, (←) RBS, (←) SBAR, (←) PDT

ADVP If the rightmost daughter is RBR, JJ or JJR, it is head. Otherwise: (→) RB, ADVP, JJ, RBR, RBS, IN, JJR, JJS, FW, TO, CD, JJR, JJ, IN, WHADJP (all), NP, JJS, NN, RP

NAC ←: DT, NN, NNS, NNP, NNPS, NP, NAC, EX, DOLLAR, CD, QP, PRB, VBG, JJ, JJS, JJR, ADJP, FW

S-ADV ←: MD, VP, .*-PRD

S ←: MD, VBP, VBD, VBZ, TO, VP, "-PRD", ADJP, S, SBAR, SINV, UCP, INTJ, NP

SINV If the first daughter is ADVP with head word *"so"*, *"So"*, or *"SO"*, it is head. Otherwise ←: VBZ, VBD, VBP, VB, MD, VP, S, SINV, ADJP< NP

SQ ←: VBZ, VBD, VBP, VB, MD, VP, SQ, SBARQ

SBAR If there are two daughters and the first one is WHNP with trace, and the second daughter is a S with a to-infinitival VP, the S is head. Otherwise, ←:
IN, WHNP, WHPP, WHADVP, WHADJP, WDT, RB, MD, DT, X, PP, PRN, UCP, VBD, VB, S, SQ, SINV, SBAR, FRAG

SBARQ ←: WHNP, WHADVP, WHADJP, WHPP, WP (FRAGS as SBARQ), SBARQ

PP If the first daughter is PRN, it is the head, otherwise →: IN, TO, VBG, VBN, VB, RP, PP, FW Else the leftmost child is head.

VP ←: TO, VBD, VBN, MD, VBZ, VB, VBG, VBP, VP, JJ, S, SYM, NN, NNS, NP

QP ←: ATP, QP, ASTAG, NP, NOUN, NN, NNS, RBR, UP, NP, DOLLAR, SYM, CD Else, the rightmost child is head.

WHNP ←: WDT, WP, WPS, WHADJP, WHADVP, WHNP, IN, WRB

WHADJP ←: WRB, WHADVP, WP, RB

WHADVP →: WRB, IN

WHPP →: WHNP

CONJP →: CC, JJ, IN, RB, CONJP

UCP The leftmost child is head

FRAG If the first child is CC or CONJP, and the second child is not SBAR, SBARQ, RB, take the first child. otherwise, ←:
WP, WHADVP, RB, SBAR,IN, S, NP, ADVP, PP

PRN If the first child is ":" or LRB, and the last child is not ":", then the first child is head, unless the second child is SBAR, PP, PRN, IN, or if there are only two children.

If the first child is PRN, it is is head. Otherwise, ←:
PRNS, S, NP, VP, PP, SBAR, PP, UCP, ADJP, ADVP, RB PRNS is the label that is inserted for S, or SINV under PRN to trigger the unary type-changing rule.

INTJ The leftmost child is head

X ←: DT, otherwise take the leftmost child

RRC →: VP, NP, ADVP, ADJP, PP

PRT (→): RP

LST (→): LS, COLON

NP, NPnonBare or NX:

- Within a NP or NPnonBare: going from left to right, find the first NP that does not have an adjunct tag.
- Otherwise: if the first child is a determiner or quantifier, the first child is head. Determiner or quantifier labels are:
  CD, DT, PDR, POSSDT, QP, WDT, WPD, WRB, WP, JJ (if not immediately followed by a NPnonBare), POSS
- Otherwise: if the last child is POS, it is head.
- Otherwise: Going from right to left, the first constituent that has a noun label is head. Noun labels are the following:
  NN, NNP, NNPS, NNS, NX (not co-indexed with a *RNR*-trace), POS, JJR, VBG, VB, VBZ, VBN, JJS, QP
- Otherwise, going from left to right, search for a NP or WHNP (this is for NPs with NP adjuncts).
- Otherwise, search for a $ or ADJP from right to left.
- Otherwise, search for a CD from left to right.
- Otherwise, search from the right for a JJ, RB, QP, DT
- Otherwise, search from the left for a ADVP, FW, INTJ, POSS
- Otherwise, take the leftmost node.

In parallel coordinations (coordinate constructions of the form (XP XP CC XP)), go from left to right, and identify the first constituent with the same label as the parent as head.

Head-finding rules for additional nonterminals The translation procedure adds a number of additional nonterminals and POS tags in order to deal with specific constructions. These serve the purpose of recognizing specific constructions. For example, ATP is the constituent that contains the words *at least/most*; SO is the POS tag assigned to *so* in constructions of the form *so ADJ that*. DOLLAR is the POS tag assigned to the string DOLLARS which replaces monetary expressions. The constituent XNP contains a NP and a comma, and is required to trigger the binary type-changing rules that account for extraposition.

PRNS (the label that is inserted for S, or SINV under PRN to trigger the unary type-changing rule) ←: VP, VB

ATP (for *"(at least/most) X"*) (←) IN

INP (*"(in order) to do something"* (→): IN

pipedPipeNP The leftmost child is head

THANP (the constituent that is inserted in QPs such as *more than three times*, and that includes *than* and everything that follows it): The rightmost child is head

POSSDT (The constituent that includes the possessive *"'s"* and the preceding noun phrase): The rightmost child (ie. the possessive *"'s"*) is head

SOADJ (←): ADJP Otherwise: (→): VBN, VBG, JJ, JJR, JJS, NN, NNS Otherwise: (←) JJ, JJR. Else, the leftmost child is head.

DTP (a constituent consisting of a predeterminer and a determiner) (→): DT

XNP rightmost (all)

A.2 Complement-adjunct distinction

If the node has a complement tag (-SBJ, -CLR, -DTV, -TPC, -PRD), it is a complement, unless it is an ADVP-CLR which expands to a null element, or it also has an adjunct tag in addition to a -CLR tag. PP-TPC nodes are only complements under SINV, or if they carry a -PRD tag. Nodes with -PRD tag that are children of UCPs are not complements. UCPs themselves are treated as if their label was the label of their first child. Nodes with adjunct tags (-ADV, -VOC, -BNF, -DIR, -LOC, -MNR, -TMP (with the exception of PP-TMP nodes under ADJP) and -PRP) cannot be complements. NP-TPC nodes that are not co-indexed (eg. NP-TPC-1) are also treated like adjuncts, since there is a resumptive pronoun or other element.

For each constituent type, this is a list of the constituents that are recognized as complements when they appear as children of this constituent:

ADJP: PP unless it is left to the head or headed by IN *"than"*, S, SBAR (not if the complementizer is *"than"*, *"as"*, *"so"*, *"which"*), NP, SOADJ

ADVP: PP, unless it is left to the head or headed by IN with head word *"than"*, NP, SOADJ, SBAR (not *"than"*, *"as"*, *"so"*, *"which"*, *"before"* or WHADVP, and not if there is an intervening punctuation mark)

NP: NOUN, NPnonBare

NX: NOUN, NPnonBare

NAC: NOUN, NPnonBare

NOUN: SBAR

S: S with numerical index, but not co-indexed with a *RNR*-trace; VP if the head of S is MD

SINV: NP, S (and not a parallel conjunction), SBARQ, VP

SQ: VP, NP, WP, S

SBAR: NN (*"in order to..."*), S, SQ, VP, SINV, SBARQ (FRAG is changed to SBARQ)

SBARQ: SQ, WHNP, SINV, S, SBARQ, NP, VP

PP: NP

VP: NP, VP (unless a parallel conjunction), SBARQ, S, SQ, ADJP, PPs with *T*-trace

S under VP an S under a VP parent is a complement if it is not preceded by a comma and another S.

SBAR under VP An SBAR is a complement if it is:

- not preceded by a comma
- preceded by a comma, and a relative clause with *"which"*
- preceded by a comma, and if there is no other intervening complement between the comma and the verb (this is wrong for intransitive verbs).
- preceded by a comma, and if there is another comma somewhere in the yield between the verb and the immediately preceding comma.

WHNP: NOUN

WHADJP: JJ, ADVP, ADJP

WHADVP: JJ, ADJP, ADVP (unless is the first child), RB (unless it is the first child or headed by "*not*")

WHPP: IN, TO, piedPipeNP

UCP: treat like it had the same label as its first child

FRAG: NP, PP (if left from head)

PRN: RRB, COLON (if the head is LRB or COLON)

X: ADJP, JJR

Complement-adjunct distinction for additional constituents Additional constituents that are inserted during translation have their own complement-adjunct rules.

POSSDT: NP, NOUN

SOADJ: PP, unless it is left to the head or headed by IN with head word "*than*", NP, SOADJ

ATP: JJS, RBS, DT

A.3 Correcting tagging errors

The following heuristics attempt to correct some systematic tagging errors. Only tagging errors that led to errors in category assignment were attempted to correct.

Under ADJP: if a NN, VBN, or VBG appears after a conjunction, it is a second head in a coordinate construction – change it to JJ

***"whether"* as CC; any CC as first child of SBAR:** change to IN (preposition). Subordinating conjunctions in CCG are like modifiers, not like coordinating conjunctions.

Infinitives tagged as VBP: change to VB, eg. within inverted questions

Infinitives within *"do/would"* questions: if the first child of an SQ is any form of "*do*" or "*would*", and the first child of the VP underneath the SQ is not a VB, MD, VBZ or -NONE-, it is changed to VB.

***"and"* is always CC**

If the last child of an NP is JJ, it is changed to NN. This tagging error leads to errors with the head-finding rules, hence it is changed.

VBP mistagged as VB under SQ, if the first or second child is VB, it is changed to VBP.

Mistagged NNPs: occasionally words in capital letters are tagged NNP even though they are not, eg. *"Does"*, *"Should"*. Under ADVP, if the adverb has only one child, and it is tagged as NNP, change this to RB. This is a tagging error, and we do not want unknown NNPs to have adverbial categories.

Unrecognized prepositions and subordinating conjunctions: if the first child of a PP is tagged NN, it is changed to IN.

Possessive *'s* as VBZ: if the possessive *"'s"* under POSSDT is labelled VBZ, it is changed to POS

VBZ as possessive "*'s*": under a VP, possessive *"'s"* is labelled VBZ.

Simple past tense (VBD) mistagged as past participles (VBN) If a VBN is the head of a VP and its grandparent is a S, we assume it should not be a past participle but a simple past tense form (VBD). Under SINV, if the first child is VBN, it is changed to VBD.

Bare infinitives (VB) mistagged as VBP, VBD or VBN: in to-infinitives – if they are children of a VP which itself is the child of a VP whose head is TO.

-ing forms tagged as NN, NNP, JJ, RB If an "ing"-form appears within a VP, and is tagged as NN, NNP, JJ or RB, it is changed to VBG.

-ed forms tagged as NN, NNP, JJ If a form eding in "ed" appears within a VP and is tagged as NN, NNP or JJ, it is changed to past tense VBD if the parent of the VP is S, and to past participle VBN if the parent of the VP is a VP or if the VP is passive. (This only captures mistagged regular verbs.)

Other forms tagged as NN, NNP or JJ Other verbs are also somtimes mistagged as NN, NNP or JJ. They are changed to VBZ if the parent of the VP is a sentence, yielding a declarative sentence, to VB if the parent of the VP is a VP headed by a do-form, and to VBN otherwise.

RB as first child of VP: changed to VB/VBG/VBN] (depending on ending) if there is no other head in the VP.

VBZ mistagged as NNS underneath VP if an NNS is the first or second child of a VP, or the third child and immediately preceded by a conjunction, it is a VBZ.

VB mistagged as VBP: if a VBP is a child of a VP that is a child of an SINV, the VBP is changed to VB.

VBN mistagged as VBD: if a VBD is a child of a VP that is a child of an SINV, the VBD is changed to VBN.

VB mistagged as NN: if a NN is a child of a VP that is a child of an SINV, the NN is changed to VB.

***"that"* underneath a NP is DT** (not IN).

RB as head of VPs embedded in TO-VPs (there should be other cases too, but this one is easiest to detect). Are changed to VB

NNS appearing as first or second child in a VP Are changed to VBZ (3rd person singular verb)

A.4 Changes to constituents

This section summarizes the changes to the trees before binarization. There are different kinds of changes: some have to do with inserting additional structure that is not there in the Treebank analysis (such as for QPs or the additional noun level within NPs); others are necessary for binarization to work correctly (for instance, in a lot of cases, modifiers within coordinate constructions appear at the same level as the conjuncts rather than either within one of the conjuncts or taking scope over the entire coordinate constructions), some are required because the CCG analysis of that particular construction differs from the Treebank account (eg. small clauses), and others (fewer) aim to correct errors in the Treebank that would otherwise lead to incorrect categories.

A.4.1 Preprocessing NPs

Most changes to NPs have to do with the insertion of a noun level into base (non-recursive) NPs, the reanalysis of possessive *"'s"*, and the placement of modifiers in coordinate constructions.

Reanalysis of NP structure

- The possessive *‘s* and *‘* are analyzed as functors from `NP`s to determiners: a new constituent (labelled `POSSDT`) is inserted, which consists of the `NP` and the possessive *‘s*.

- A “non-bare” `NP` is inserted in an `NP` after the first child if the first child is a `PDT` (*all, such, both, half* etc.), a `WP` (*what else*, *what other means*), or a `JJ` with head word *such*), or if the first child is a `DT`, and the new token to be added is a `NP`:

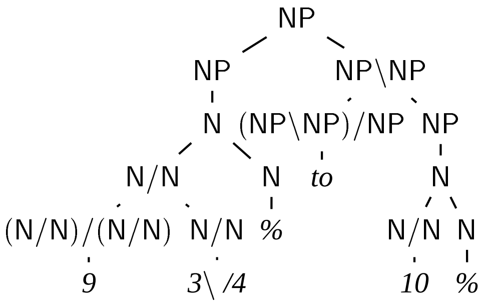
```
(NP (DT both)
    (NP (DT the) (NN quake))
    (CC and)
    (NP (NNP Hugo)))

(NP (DT all)
    (NP (DT the) (NN agency) (POS ’s))
    (NNS policies))
```

- In coordinate construction, `S-NOM` should be recognized as conjunct, and a unary projection `NP` → `S-NOM` is inserted.

- In dates (`NP-TMP`) of the form *“Feb. 18”*, the month takes the number as complement.

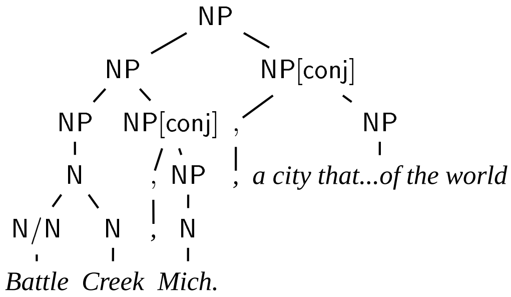
- If `NP` → `NP TO NP`, we obtain the same analysis as for `QP`s

```
 (NP (NP (QP (CD 8) (CD 1\/2))
         (NN %))
     (TO to)
     (NP (QP (CD 8) (CD 3\/8))
         (NN %)))
```



- If there are two adjacent NP children, the first of which has a comma as last child, the comma is lifted up the the top level in order to obtain the appositive analysis for the second NP:

```
(NP (NP (NP Battle Creek)
        (, ,)
        (NP (NNP Mich.))
        (, ,))
    (NP a city that calls itself the breakfast capital of the world))
```



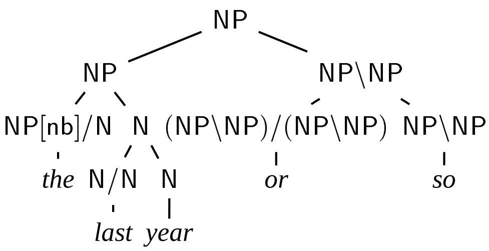
- If there are any clausal postmodifiers within a base NP, insert another NP consisting of the base NP only. This avoids the postmodifier becoming a modifier of the noun. For example, if the last child of an NP is a QP, the QP is a postmodifier (*or more, or so, out of X*)

```
(NP (DT the)
    (JJ last)
    (NN year)
    (QP (CC or)
        (RB so)))
```

This tree is changed in the following way before the noun level is introduced:

```
(NP (NP (DT the)
        (JJ last)
        (NN year))
    (QP (CC or)
        (RB so)))
```

Here is the CCG reanalysis:



We do not make this change if the QP is a purely numerical expression such as the following:

```
(NP (DT the)
    (VBG remaining)
    (QP (CD 2.2)
        (CD million)))

(NP (NN age)
    (QP (CD 59)
        (CD 1\/2)))
```

Similarly, there are cases where appositions are not recognized because there is an adjunct such as a PP, SBAR, PRN after the first NP:

```
(NP (NP (DT the) (NNP Center))
    (PP (IN for) (NP (NNP Security) (NNP Policy)))
    (, ,)
    (NP (DT a) (JJ conservative) (NNP Washington) (NN think-tank))
```

In these cases, we insert a NP which includes the NP and its adjunct.

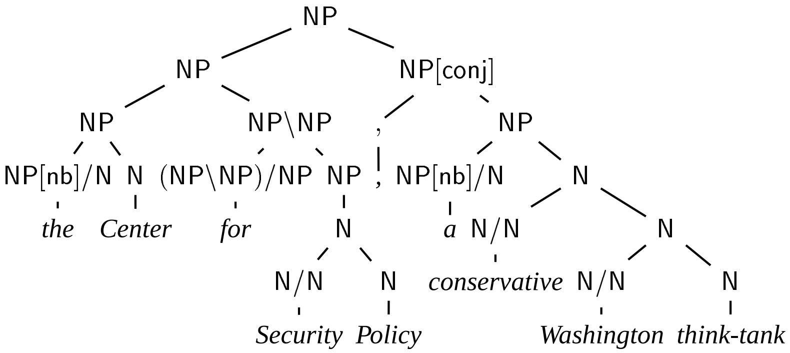
Also, if there is a RB followed by a NP within a NP, an NP which includes the RB and NP, is inserted.

In coordinate NPs, if there is a postmodifier adjacent to one of the conjunct NPs (except the last one). attach it to this NP. Postmodifiers are PP, PRN, SBAR, RRC, ADJP. These postmodifiers must not be coindexed with a *RNR* trace.

```
(NP (NP (DT the)
        (NNP Center))
    (PP (IN for)
        (NP (NNP Security)
            (NNP Policy)))
    (, ,)
    (NP (DT a)
        (JJ conservative)
        (NNP Washington)
        (NN think-tank)))
```

This becomes

```
(NP (NP (NP (DT the)
            (NNP Center))
        (PP (IN for)
            (NP (NNP Security)
                (NNP Policy))))
    (, ,)
    (NP (DT a)
        (JJ conservative)
        (NNP Washington)
        (NN think-tank)))
```



A similar preprocessing step deals with RB and ADVP immediately preceding a conjunct NP.

- If the last child of a NP with more than two children is a QP, it is usually an adjunct to the whole NP such as *or more*, *or so*, *out of X*, and we insert a separate NP level that includes everything up to the QP:

```
(NP (DT the)
    (JJ last)
    (NN year)
    (QP (CC or) (RB so)))
```

```
(NP (CD one)
    (JJ new)
    (NN ringer)
    (QP (IN out) (IN of) (CD 10)))
```

The new structure is then:

```
(NP (NP (DT the) (JJ last) (NN year))
    (QP (CC or) (RB so)))
```

```
(NP (NP (CD one) (JJ new) (NN ringer))
    (QP (IN out) (IN of) (CD 10)))
```

We do not make this change if the `QP` is a purely numerical expression such as the following:

```
(NP (DT a)
    (JJ fixed)
    (QP (CD 107) (CD 3\/4)))
(NP (DT the)
    (VBG remaining)
    (QP (CD 2.2) (CD million)))
```

Similarly, there are cases where appositions are not recognized because there is an adjunct such as a `PP, SBAR, PRN` after the first NP:

```
(NP (NP (DT the) (NNP Center))
    (PP (IN for) (NP (NNP Security) (NNP Policy)))
    (, ,)
    (NP (DT a) (JJ conservative) (NNP Washington) (NN think-tank))
```

In these cases, we insert a `NP` which includes the `NP` and its adjunct.

If there is a `RB` followed by a `NP` within a `NP`, an `NP` which includes the `RB` and `NP`, is inserted.

- Furthermore, `*T*` traces within VPs containing a “passive PP”, that is a PP which contains a passive trace, are all changed to `*`

These changes mean that only NPs appear under NPs if there is a conjunction, or an adjunct such as a relative clause or PP at the same level. There are still some cases of NPs under nouns, eg:

```
(NP (DT the)
    (NP (NNP Oct.) (CD 19) (, ,) (CD 1987))
    (NN market)
    (NN collapse))
```

These are treated as adjuncts.

Multiple NPs and modifiers Similarly, if there are appositives and other modifiers such as relative clauses (SBAR), reduced relative clauses (RRC, VP) or PPs modifying the same NP, additional structure has to be imposed to obtain the correct analysis. There are three types of cases:

- `NP → NP, NP, MODIFIER (,)`
  The second NP is an appositive, and the modifier modifies the first NP. This becomes
  `NP → (NP, NP) , MODIFIER (,)`
- `NP → NP , NP MODIFIER`
  The modifier modifies the second NP. This becomes
  `NP → NP , (NP MODIFIER)`
- `NP → NP, MODIFIER , NP`
  Since appositives are analyzed like coordinate lists, merge the first NP and the modifier to an NP:
  `NP → (NP, MODIFIER) , NP`

Insertion of noun level: There are a few cases where annotators inserted an NP as a noun level, but these are extremely rare, and we leave them as they are:

```
(NP (DT a)
    (NP (NP (JJ small)
            (, ,)
            (JJ tight)
            (JJ facial)
            (NN shot))
```

```
        (PP (IN of)
            (NP (NP (NNP David)
                    (NNP Dinkins))
                (, ,)
                (NP (NP (JJ Democratic)
                        (NN candidate))
                    (PP (IN for)
                        (NP (NP (NN mayor))
                            (PP (IN of)
                                (NP (NNP New)
                                    (NNP York)
                                    (NNP City))))))))))
(NP (DT the)
    (NP (JJ social)
        (NNS studies)
        (NN section)))
(NP (DT a)
    (NP (NNS cold-cuts)
        (NN buffet)))
```

NPnonBare: this is inserted in any NP whose first child is a PDT.

NPs with determiners/quantifiers: Insert a noun into a NP, NAC or NX after DT, POSS, POSSDT, QP, WDT, WRB, WP$ if they are not followed by a conjunction, a punctuation mark or a NPnonBare, or appear within a parallel conjunction ((XP XP CC XP) or parallel list ((XP XP , XP).

Inserting nouns into mass nouns, proper nouns, bare plurals: If a noun phrase does not have a determiner, and is not a personal or demonstrative pronoun, we insert a noun level immediately under the NP, so that the entire NP is also a noun. This applies to bare plurals, mass nouns and proper nouns.

Possessives: If a POSSDT (see possessives above) is followed by an NP, change the NP to a noun.

See below for the treatment of pied piping.

A.4.2 Preprocessing VPs

- Sometimes, VP coordinations are not bracketed properly, so that the main verb of the first VP appears at the same level as the VP conjuncts, eg:

  ```
  (VP (VBZ is)
      (VP restricted to his home much of the day)
      (CC and)
      (VP isn't allowed to work as a journalist))
  ```

  In these cases, a VP is inserted which contains the verb and the following VP:

  ```
  (VP (VP (VBZ is)
          (VP restricted to his home much of the day))
      (CC and)
      (VP isn't allowed to work as a journalist))
  ```

  This can also happen with the verb of the second conjunct:

  ```
  (VP (VP are very short-term)
      (CC and)
      (VBP are)
      (VP going to create high turnover))
  ```

  This is changed in a similar way.

- Another type of bracketing error in coordinate VPs attaches the main verb of the first conjunct outside the entire conjunction, eg:

  ```
  (VP (VBZ has)
      (VP (VP expanded its sale force to about 20 people from about 15)
          (CC and)
          (VP (VBZ hopes)
              (S to expand its sales ...))))
  ```

  This error can be recognized by the fact that the verb in the second conjunct is tagged VBZ (other cases include VBP). This is changed to the correct bracketing:

  ```
  (VP (VP (VP (VBZ has)
              (VP expanded its sale force to about 20 people from about 15)
          (CC and)
          (VP (VBZ hopes)
              (S to expand its sales ...))))
  ```

- Sometimes, NPs in VPs aren’t bracketed properly:

```
(VP (VB buy)
    (ADJP (RB closely)
          (VBN held))
    (NNS concerns))
```

- PP conjunctions within VP are sometimes not bracketed properly:

```
(VP (VB switch)
    (NP his retirement accounts)
    (PP-DIR out of three stock funds)
    (CC and)
    (PP-DIR into a money market fund))
```

In order to recognize the PP coordination, an extra PP is inserted:

```
(VP (VB switch)
    (NP his retirement accounts)
    (PP (PP-DIR out of three stock funds)
        (CC and)
        (PP-DIR into a money market fund))
```

- Ellipsis null elements (*?*) are cut out.

- If the last two children of a VP are ”, NP” and there is another VP at the same level, the NP is out of construction:

```
(VP (VP (VBD fell)
        (NP-EXT  0.1 %)
        (PP-TMP in September))
    (, ,)
    (NP (NP the first decline)
        (PP-TMP since February 1987)))
```

We introduce a separate level XNP that includes the comma and the NP, which then triggers a binary type changing rule $(\mathrm{S}\backslash\mathrm{NP})\backslash(\mathrm{S}\backslash\mathrm{NP}) \rightarrow , \mathrm{NP}$

```
(VP (VP (VBD fell)
        (NP-EXT  0.1 %)
```

```
             (PP-TMP in September))
         (XNP (, ,)
              (NP (NP the first decline)
                  (PP-TMP since February 1987)))
```

Sometimes the comma appears as last child of the VP child. This is also changed to the XNP structure.

- VP coordination/lists:

  ```
  VP --> VP, VP, ADV VP, VP
  ```

  is rebracketed as

  ```
  VP --> VP, VP, (ADV/RB VP), VP
  ```

  Similarly,

  ```
  VP --> ADVP/RB VP, VP...
  ```

  is changed to

  ```
  VP --> (ADVP/RB VP), VP...
  ```

  even though there is a genuine ambiguity.

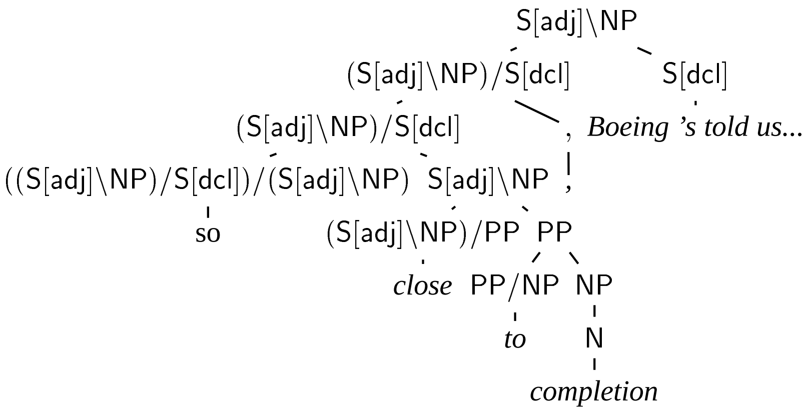
A.4.3 Preprocessing ADJPs

"so" + adjective; "as"+ adjective

- *"so"* ADJP...: If there is a PP and an SBAR, we want the PP to modify the adjective, and the SBAR to be an argument of the *"so"*:

  ```
  (ADJP-PRD (ADJP (RB so)
                  (RB close))
            (PP (TO to)
                (NP (NN completion)))
            (, ,)
            (SBAR (-NONE- 0)
                  (S (NP-SBJ (NNP Boeing))
                     (VP (VBZ ’s)
  ```

```
(VP (VBN told)
    (NP (PRP us))
    (SBAR (-NONE- 0)
          (S (NP-SBJ (EX there))
             (VP (MD wo)
                 (RB n't)
                 (VP (VB be)
                     (NP-PRD (DT a)
                             (NN problem)))))))))))
```



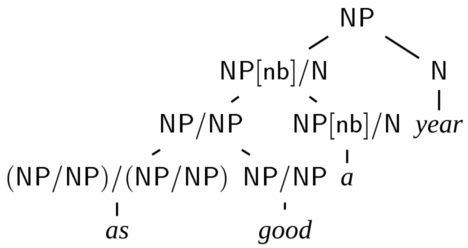
- Within a noun phrase, an adjective phrase with *"so"* or *"as"* can modify a determiner, eg *"so personal an issue"*, *"as good a year"* (but not *"so many people"*). The Treebank gives this construction a flat analysis:

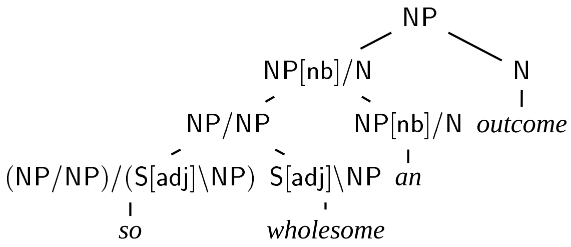
```
(NP (ADJP (RB so)
          (JJ personal))
    (DT an)
    (NN issue))
```

  However, since we distinguish determiners and nouns, we analyze this construction as follows:

```
(NP (DTP (ADJP (RB so)
               (JJ personal))
         (DT an))
    (Noun (NN issue)))
```

  Here the ADJP modifies the determiner:





- If an ADJP expands to a conjunction of ADJPs, the following heuristics were implemented to deal with adjuncts of these conjuncts that appear at the same level as the coordination itself: If a PRN child immediately precedes an ADJP, it is re-analyzed as a child of its ADJP sibling. Anything that appears after the last ADJP is re-analyzed as a child of this ADJP.

A.4.4 Preprocessing ADVPs

Incorrect complement tags: ADVP under NP are not complements; PRD tags are eliminated.

Adverbial noun phrases that contain a determiner (DT) followed by a nominal tag (NN) that are labelled as ADVPs are preprocessed in the same manner as NPs. This guarantees that nouns are inserted.

ADVPs that consist of three children where the second child is "to" are preprocessed like QPs.

***"more than"*:** *than* takes *more* as complement.

A.4.5 Preprocessing Ss

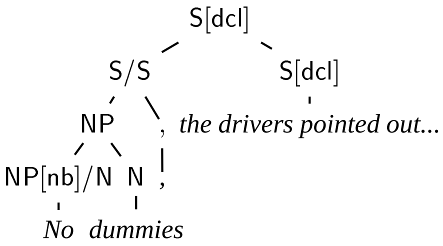
- The binary type-changing rule for sentences is triggered if the first element of a sentence is a S-ADV small clause with empty NP-SBJ and NP-PRD:

```
(S (S-ADV (NP-SBJ (-NONE- *-1))
          (NP-PRD No dummies))
   (, ,)
   (NP-SBJ-1 the drivers)
   (VP pointed out they still had space ...).
```

This is changed to:

```
(S (XNP (S-ADV (NP-SBJ (-NONE- *-1))
          (NP-PRD No dummies))
   (, ,))
   (NP-SBJ-1 the drivers)
   (VP pointed out ...)).
```

This yields the following CCG analysis:



- `NP-SBJ` with surrounding parentheses at the `S` level: move the parentheses underneath the `NP-SBJ`:

```
(S (-LRB- -LRB-)
   (NP-SBJ (DT the)
           (NN departure))
   (-RRB- -RRB-)
   (VP (MD will)
       (VP (VB be)
           (ADJP-PRD (RB fairly)
                     (JJ irrelevant))
           (PP (IN for)
               (NP (DT the)
                   (NN company))))))
```

This is changed to:

```
(S (NP-SBJ (-LRB- -LRB-)
           (NP-SBJ (DT the)
                   (NN departure))
```

```
                (-RRB- -RRB-))
        (VP (MD will)
            (VP (VB be)
                (ADJP-PRD (RB fairly)
                          (JJ irrelevant))
                (PP (IN for)
                    (NP (DT the)
                        (NN company))))))
```

- S-TPC with a null subject should be S-ADV.
- Under an S, constituents such as ADVP, PP, TO that appear adjacent to a VP should appear under the VP, not at S-level. If the VP is surrounded by commas, the modifier and the commas should all appear under the VP.
- Conjunctions: like in the VP and NP case, premodifiers of the second or further conjunct that appear as sisters of the conjunct are moved underneath the conjunct.
- There is a similar binary type-changing rule for NP adjuncts that appear at the end of a sentence and are preceded by commas.
- under S, PRNS, SBAR, SINV, VP:
- If there is a child IN immediately followed by a S, both of them should go under a SBAR-ADV:

  ```
  (S (S The governor could n't make it)
     (, ,)
     (IN so)
     (S the lieutenant governor welcomed the special guests)
     (. .))
  ```

- If there are two children marked with the SBJ tag, only the second one is the real subject.
- ”(S S ’...’ CC C)” to ”(S (S ’...’) CC ...)”: true for ”...”, quotes, –, -

A.4.6 Preprocessing QPs

The Treebank assumes a flat internal structure for QPs (“quantifier phrases”).

Coordination in QPs If there is a conjunction within the QP, and the conjunction is not the first child, we assume that everything to the left and everything to the right of the conjunction is a QP. If there is a preposition (IN) before the conjunction (*between...and...*), then the first conjunct is only what comes between the preposition and the conjunction. Examples:

```
(QP (DT all) (CC but) ($ $) (CD 40,000))
(QP (IN between) (CD 1) (NN %) (CC and) (CD 3) (NN %))
```

Prepositions in QPs If there is one of the following prepositions within the QP, then everything that comes after the preposition is rebracketed as a NP, and a PP is inserted which consists of the preposition and the NP: *of*, *out*, *as*, *in*, *than*, *to*. Examples:

```
(QP (RBR more) (IN than) (CD 30))
(QP (RB Only) (CD five) (IN of) (DT the) (CD 40))
(QP (JJ as) (RB much) (IN as) (CD 15))
(QP (RB Only) (CD one) (IN in) (CD four))
```

These NPs are further preprocessed to insert nouns etc. The prepositions *to* and *in* are analyzed like conjunctions which take the left and right noun phrases as arguments. Their category is the category that is assigned to the QP.

***"as much as..."*:** The first *"as"* subcategorizes for the *"much"* and the following preposition.

***"well over..."*, etc.** If the first child of a QP is an adverb (RB), and the second is a preposition (IN, but not *from*), then the adverb modifies the preposition. Example:

```
(QP (RB well) (IN over) (CD 9))
```

***"nearly a third"*, etc:** If the first child of a QP is an adverb (RB), and the second is a preposition (IN, but not *from*), then the adverb modifies the entire QP. Examples:

```
(QP (RB nearly) (DT a) (JJ third))
```

A.4.7 Preprocessing RRCs

RRC is a label assigned to reduced relative clauses. However, a search with tgrep finds only 55 RRCs in the entire Treebank, since most reduced relative clauses are annotated differently. For instance, reduced relatives consisting of past or present participles should be annotated VP (Bies *et al.*, 1995, p. 230), eg.:

```
(NP (NP government figures)
   (VP (VBD released)
        (NP (-NONE- *) )
        (NP-TMP (NNP Wednesday) ))))))
```

Some instances of RRC expanding to VP can be found:

```
(NP (NP (NNS workers))
    (RRC (VP (VBN exposed)
             (NP (-NONE- *))
             (PP-CLR to it)
             (ADVP-TMP more than 30 years ago))))
```

These are re-labelled as VP.

There are also RRCs with a ADJP child:

```
 (NP (NP (NNS stores))
    (RRC (ADJP open)
         (NP-TMP more than one year)
```

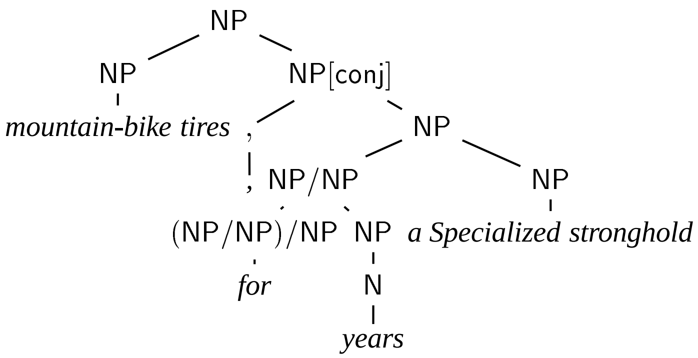
These are also re-labelled as ADJP, given that the same construction can appear elsewhere as ADJP:

```
(NP (NP (NNS funds))
    (ADJP (JJ open)
          (PP only to institutions)))
```

Appositives consisting of a sentential modifiers and a noun phrases, such as the following examples, are also annotated as RRC:

```
(NP (NP mountain-bike tires)
    (, ,)
    (RRC (PP-TMP for years)
         (NP (DT a Specialized stronghold))))
```

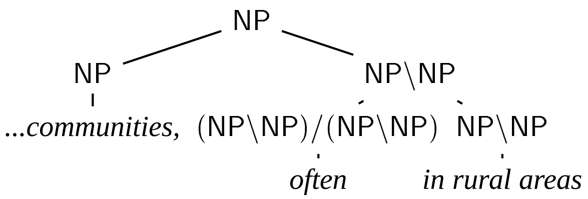
Given that in our grammar, appositives are analyzed like NP-coordination, we simply relabel these instances of `RRC` as `NP`; the sentential modifier is then treated like a premodifier of the second noun phrase:



Note that this leads to overgeneration in the grammar, since only appositive noun phrases can be modified in this manner.

Otherwise, if a `RRC` contains a `PP` child, it is re-labelled as `PP`. These are constructions consisting of a `PP` and a modifier, eg:

```
(NP (NP an estimated 92 communities))
    (, ,)
    (RRC (ADVP-TMP often)
         (PP-LOC (IN in)
                 (NP rural areas)))
```



A.4.8 Preprocessing `SINV`s

Elliptical inversion As described in section 3.5.7, we analyze elliptical inversion after certain function words like *so, than, as* by letting the function word subcategorize for an inverted, elliptical sentence. *Than* and *as* are analyzed as specifiers of `SBAR`:

```
(SBAR-ADV (IN as)
          (SINV (VBZ does)
                (NP-SBJ President Bush)
                (VP (-NONE- *?*))))
```

so appears as within an SINV:

```
(SINV (ADVP-TPC-1 (RB so))
      (VP (VBZ does)
          (ADVP (-NONE- *T*-1)))
      (NP-SBJ (NP almost everyone else))
              (PP-LOC in the book))
```

This analysis is modified so that *so* is also a specifier of an SBAR.

Verb fronting: As described in section 3.5.7, we analyze the subject NP as an object of the verb; therefore we replace the VP-trace with the NP-SBJ:

```
(SINV (VP-TPC-1 (VBG Following)
                (NP the feminist and population-control lead))
      (VP (VBZ has)
          (VP (VBN been)
              (VP (-NONE- *T*-1))))
      (NP-SBJ a generally bovine press)
      (. .))
```

This example becomes:

```
(SINV (VP-TPC-1 (VBG Following)
                (NP the feminist and population-control lead))
      (VP (VBZ has)
          (VP (VBN been)
              (NP a generally bovine press)))
      (. .))
```

A.4.9 Preprocessing SBARs

***“(Six months) before/after S”*:** The temporal NP receives a complement tag and is therefore analyzed as an argument of the subordinating conjunction.

Free relatives: Free relatives are analyzed as SBAR-NOM and receive category NP.

***so that, as though, as if* etc:** If an SBAR has two adjacent subordinating conjunctions as children (tagged as IN) next to a sentence S, another SBAR is inserted which consists of the second subordinating conjunction and the S.

***what about, what if*:** If an SBAR has a WP child that is immediately followed by IN, a new constituent is inserted which constists of the IN and its right sister. If the right sister is an NP (*what about...*), this new constituent is a PP-complement, if it is a S (*what if...*), the new constituent is a SBAR-complement.

A.4.10 Preprocessing UCPs

UCPs in small clauses: if a UCP-PRD appears within a small clause (S) where the subject NP is empty, cut out the empty subject and the S-node.

A.4.11 Preprocessing PPs

PPs under WHNPs: PPs under WHNP modify the noun under the first WHNP or NP, not the WHNP itself, eg:

```
(WHNP-1 (WHNP (WP what)
              (NN kind))
        (PP of initiative))))
```

This is rebracketed so that the PP modifies the noun *kind*. The following example is rebracketed so that the PP modifies the noun *opposition*.

```
(WHNP-3 (NP (WP$ whose)
            (NN opposition))
        (PP to foreign airline investment))))
```

PPs under NP do not carry the PRD tag.

If a PP has a WHNP child, it is relabelled as WHPP.

***"because of", "instead of"*:** a PP with complement tag is inserted, which consists of the preposition *of* and the adjacent NP.

Within a PP, if a relative clause (an SBAR with WHNP in specifier position) appears which is adjacent to an NP, a new NP is inserted which consists of the NP and the relative clause.

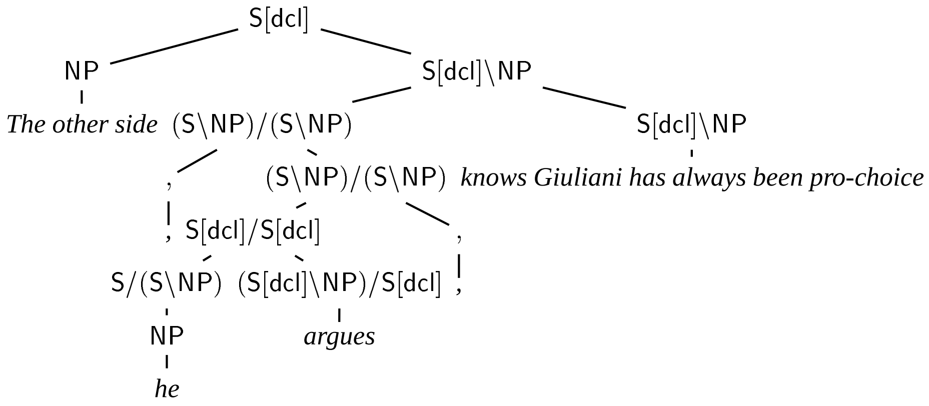
A.4.12 Preprocessing PRNs

Trees rooted in PRN: If a complete tree is rooted in PRN, its label is changed to the label of its first child, or to it second child if the first child is an opening parenthesis.

Sentences as PRNs: In direct speech, parentheticals indicate the speaker, eg.:

```
(PRN (, ,)
     (S (NP-SBJ (PRP he))
        (VP (VBZ argues)
            (SBAR (-NONE- 0)
                  (S (-NONE- *T*-1)))))
     (, ,))
```

This construction is analyzed with a unary type-changing rule:



If a colon is followed by a PRN, insert the colon into the parenthetical, and reanalyze the parenthetical.

PRNs that appear under coordinate constructions and that are adjacent to constituents that are conjuncts (have the same label as the parent node) are re-analyzed as children of this conjunct.

A.4.13 Preprocessing FRAGs

NP-SBJ traces: NP-SBJ null elements under FRAG are eliminated.

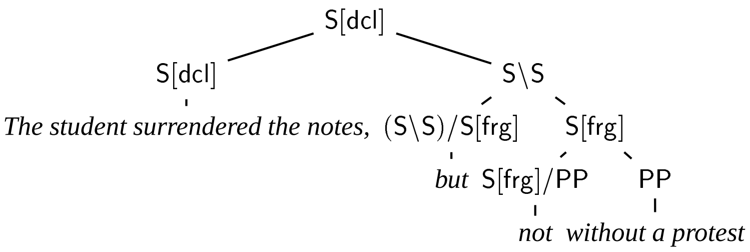
***"not"* is head:** in a FRAG, an adverb (RB) with head word *"not"* is head, unless it is preceded by WRB, WHADVP, NP:

```
(FRAG (RB Certainly) (RB not) (NP the lawyers) (. .))
(FRAG (CC And) (WHADVP why) (RB not) (. ?))
```

FRAGs that contain a wh-word are SBAR (if the FRAG is a child of a VP), or SBARQ.

If a conjunction (CC) is immediately followed by a FRAG, the conjunction is re-analyzed so that it takes the fragment as complement:

```
(TOP (S (S (NP-SBJ The student)
           (VP surrendered the notes))
        (, ,)
        (CC but)
        (FRAG (RB not)
              (PP without a protest))
        (. .)))
```



A.4.14 Preprocess Xs

If the first child is NP, NN, NNP or SYM, then the entire constituent is relabelled as NP. If the first child is IN, TO, PP, the constituent is relabelled as PP. If the first child is DT and it is the only child, or if the last child is a NN or NNS, relabel the constituent as NP. Otherwise, if the parent of the node is SBAR and this node is its first child, relabel the X as S. Else, relabel it as FRAG.

A.4.15 Other changes

”at least/most X” If *at* is immediately adjacent to a superlative adjective or adverb (eg.*least* or*most*), it subcategorizes for this adjective, which receives category S[asup]\NP.

Parentheses, dashes: for any constituent, if it has a sequence of three children of the form ''LRB/dash XP RRB/dash'', and it is not the case the XP and its parent are both NPs, let the parentheses or dashes be children of the XP.

As explained in section 3.7.4, we analyze parentheticals that consist of a dash followed by a constituent that is not normally an adjunct so that the dash and the adjacent constituent are an adjunct where the dash is "head" and the adjacent constituent its "argument". We also follow this analysis for dashes and colons under S and VP.

Eliminating *?* null elements (ellipsis): *?* null elements are ignored by the translation algorithm. If a constituent has only one child which is a *?* null element, it is eliminated. However, sometimes they are heads of constituents that have other, non-null elements, eg.:

```
(VP (VBP do)
    (VP (-NONE- *?*)
        (PP-LOC in South Carolina)))
```

In this case, the non-null elements (here the PP-LOC) are moved up one level in the tree, and the constituent which now contains only the null element is eliminated:

```
(VP (VBP do)
    (PP-LOC in South Carolina))
```

Bibliography

Abney, S. (1997). Stochastic attribute-value grammars. *Computational Linguistics*, **23**(4), 597–618.

Ajdukiewicz, K. (1935). Die syntaktische Konnexität. In S. McCall, editor, *Polish Logic 1920-1939*, pages 207–231. Oxford University Press. Translated from *Studia Philosophica*, 1, 1-27.

Aone, C. and Wittenburg, K. (1990). Zero morphemes in Unification-based Combinatory Categorial Grammar. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 188–193, Pittsburgh, PA.

Baker, J. K. (1979). Trainable grammars for speech recognition. In D. H. Klatt and J. J. Wolf, editors, *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550.

Baldridge, J. (2002). *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Baldrige, J. and Kruijff, G.-J. (2002). Coupling CCG and hybrid logic dependency semantics. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Language*, **29**, 47–58.

Bar-Hillel, Y. (1964). *Language and Information*. Addison-Wesley, Reading, MA.

Bar-Hillel, Y., Gaifman, C., and Shamir, E. (1960). On categorial and phrase structure grammars. In Y. Bar-Hillel, editor, *Language and Information*, pages 99–115. Addison-Wesley, Reading, MA. 1964.

Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, **22**(1), 39–71.

Bierner, G. (2001). *Alternative phrases: theoretical analysis and practical applications*. Ph.D. thesis, Division of Informatics, University of Edinburgh.

Bies, A., Ferguson, M., Katz, K., MacIntyre, R., Tredinnick, V., Kim, G., Marcinkiewicz, M. A., and Schasberger, B. (1995). *Bracketing Guidelines for Treebank II Style Penn Treebank Project*. University of Pennsylvania.

Black, E., Abney, S., Flickinger, D., Grishman, C. G. R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., and Strzalkowski, T. (1991). A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 306–311, San Mateo, CA.

Blaheta, D. and Charniak, E. (2000). Assigning function tags to parsed text. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 234–240, Seattle, WA.

Bod, R. (1998). *Beyond Grammar: An Experience-Based Theory of Language*. CSLI Lecture Notes. CSLI Publications, Stanford, CA.

Bod, R. (2001). What is the minimal set of fragments that achieves maximal parse accuracy? In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics and the 10th Conference of the European Chapter*, pages 66–73, Toulouse, France.

Booth, T. L. and Thompson, R. A. (1973). Applying probability measures to abstract languages. *IEEE Transactions on Computers*, **22**(5), 442–449.

Bresnan, J., editor (1982). *The mental representation of grammatical relations*. MIT Press, Cambridge, MA.

Briscoe, E. (2000). Grammatical acquisition: Inductive bias and coevolution of language and the language acquisition device. *Language*, **76**, 245–296.

Cahill, A., McCarthy, M., van Genabith, J., and Way, A. (2002). Automatic annotation of the Penn Treebank with LFG F-structure information. In *LREC 2002 Workshop on Linguistic Knowledge Acquisition and Representation - Bootstrapping Annotated Language Data*, pages 8–15, Las Palmas, Spain.

Caraballo, S. A. and Charniak, E. (1998). New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, **24**(2), 275–298.

Carpenter, B. (1991). The generative power of Categorial Grammars and Head-driven Phrase Structure Grammars with lexical rules. *Computational Linguistics*, **17**(3), 301–314.

Carpenter, B. (1992). Categorial grammars, lexical rules, and the English predicative. In R. Levine, editor, *Formal Grammar: Theory and Implementation*, chapter 3. Oxford University Press.

Carroll, J., Briscoe, E., and Sanfilippo, A. (1998). Parser evaluation: a survey and a new proposal. In *Proceedings of the First International Conference on Language Resources and Evaluation (LREC)*, pages 447–454, Granada, Spain.

Carroll, J., Minnen, G., and Briscoe, E. (1999). Corpus annotation for parser evaluation. In *Proceedings of the EACL-99 Workshop on Linguistically Interpreted Corpora (LINC-99)*, pages 35–41, Bergen, Norway.

Charniak, E. (1996). Tree-bank grammars. Technical Report CS-96-02, Department of Computer Science, Brown University.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 14th National Conference of the American Association for Artificial Intelligence*, pages 598–603, Providence, RI.

Charniak, E. (1999). A Maximum-Entropy-inspired parser. Technical Report CS-99-12, Department of Computer Science, Brown University.

Charniak, E. (2000). A Maximum-Entropy-inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139, Seattle, WA.

Charniak, E. (2001). Immediate-head parsing for language models. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 116–123, Philadelphia, PA.

Chen, J. and Vijay-Shanker, K. (2000). Automated extraction of TAGs from the Penn Treebank. In *Proceedings of the 6th International Workshop on Parsing Technologies*, Trento, Italy.

Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, San Francisco, CA.

Chi, Z. and Geman, S. (1998). Estimation of probabilistic context-free grammars. *Computational Linguistics*, **24**(2), 299–305.

Chiang, D. (2000). Statistical parsing with an automatically-extracted Tree Adjoining Grammar. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 456–463, Hong Kong.

Clark, S. (2002). Supertagging for Combinatory Categorial Grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6)*, pages 19–24, Venice, Italy.

Clark, S. and Curran, J. R. (2003). Log-linear models for wide-coverage CCG parsing. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, Sapporo, Japan.

Clark, S. and Hockenmaier, J. (2002). Evaluating a wide-coverage CCG parser. In *Proceedings of the LREC Beyond PARSEVAL workshop*, page 2002, Las Palmas, Spain.

Clark, S., Hockenmaier, J., and Steedman, M. (2002). Building deep dependency structures using a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 327–334, Philadelphia, PA.

Collins, M. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191, Santa Cruz, CA.

Collins, M. (1997). Three generative lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Computer and Information Science, University of Pennsylvania.

Collins, M. (2000). Discriminative reranking for natural language parsing. In *Proceedings of the 17th International Conference on Machine Learning*, Stanford, CA.

Collins, M. and Brooks, J. (1995). Prepositional attachment through a backed-off model. In D. Yarovsky and K. Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 27–38, Somerset, NJ.

Collins, M. and Duffy, N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 263–270, Philadelphia, PA.

Curry, H. B. and Feys, R. (1958). *Combinatory Logic*, volume I. North-Holland, Amsterdam.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *First International Workshop on Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer Verlag, New York.

Doran, C. and Srinivas, B. (1994). Bootstrapping a wide-coverage CCG from FB-LTAG. In *Proceedings of the Third International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+3)*.

Doran, C. and Srinivas, B. (2000). Developing a wide-coverage CCG system. In A. Abeillè and O. Rambow, editors, *Tree Adjoining Grammars. Formalisms, Linguistic Analysis and Processing*, pages 405–426. CSLI Publications.

Eisner, J. (1996). Efficient normal-form parsing for Combinatory Categorial Grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 79–86, Santa Cruz, CA.

Eisner, J. (2001). *Smoothing a probabilistic lexicon via syntactic transformations*. Ph.D. thesis, Computer and Information Science, University of Pennsylvania.

Gazdar, G., Klein, E., Pullum, G. K., and Sag, I. A. (1985). *Generalised Phrase Structure Grammar*. Blackwell, Oxford.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, **4**, 1–58.

Gildea, D. (2001). Corpus Variation and Parser Performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202, Pittsburgh, PA.

Gildea, D. and Hockenmaier, J. (2003). Identifying semantic roles using Combinatory Categorial Grammar. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, Sapporo, Japan.

Goodman, J. (1997a). Global thresholding and multiple-pass parsing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Providence, RI.

Goodman, J. (1997b). Probabilistic feature grammars. In *Proceedings of the International Workshop on Parsing Technologies*, Cambridge, MA.

Harris, T. (1963). *The Theory of Branching Processes*. Springer, Berlin, Germany.

Henderson, J. C. and Brill, E. (1999). Exploiting diversity in natural language processing: Combining parsers. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 187–194, College Park, MD.

Henderson, J. C. and Brill, E. (2000). Bagging and boosting a Treebank parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 34–41, Seattle, WA.

Hepple, M. and Morrill, G. (1989). Parsing and derivational equivalence. In *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, pages 10–18, Manchester, UK.

Hindle, D. and Rooth, M. (1993). Structural ambiguity and lexical relations. *Computational Linguistics*, **19**(1), 103–120.

Hinton, G. E. (1999). Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN)*, volume I, pages 1–6, Edinburgh, UK.

Hockenmaier, J. (2001). Statistical parsing for CCG with simple generative models. In *Proceedings of Student Research Workshop, 39th Annual Meeting of the Association for Computational Linguistics and 10th Meeting of the European Chapter*, pages 7–12, Toulouse, France.

Hockenmaier, J. (2003). Parsing with generative models of predicate-argument structure. In *Proceedings of the 41st Annual Meeting of the ACL*, Sapporo, Japan.

Hockenmaier, J. and Steedman, M. (2002a). Acquiring compact lexicalized grammars from a cleaner Treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*, pages 1974–1981, Las Palmas, Spain.

Hockenmaier, J. and Steedman, M. (2002b). Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 335–342, Philadelphia, PA.

Hockenmaier, J., Bierner, G., and Baldridge, J. (2000). Providing Robustness for a CCG System. In *Proceedings of ESSLLI'2000 Workshop on Linguistic Theory and Grammar Implementation*, pages 97–112, Birmingham, UK.

Hockenmaier, J., Bierner, G., and Baldridge, J. (2002). Extending the coverage of a CCG system. *Research in Logic and Computation*, **1**(4).

Hwa, R. (1999). Supervised grammar induction using training data with limited constituent information. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 73–79, College Park, MD.

Jelinek, F. (1997). *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, MA.

Johnson, M. (1998). PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, **24**(4), 613–632.

Johnson, M. (2002a). The DOP estimation method is biased and inconsistent. *Computational Linguistics*, **28**(1), 71–76.

Johnson, M. (2002b). A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 136–143, Philadelphia, PA.

Johnson, M., Geman, S., Canon, S., Chi, Z., and Riezler, S. (1999). Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 535–541, College Park, MD.

Joshi, A. and Srinivas, B. (1994). Disambiguation of super parts of speech (or supertags): Almost parsing. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, pages 154–160, Kyoto University, Japan.

Joshi, A., Levy, L., and Takahashi, M. (1975). Tree-adjunct grammars. *Journal of Computer Systems Science*, **10**, 136–163.

Karttunen, L. (1989). Radical lexicalism. In M. Baltin and A. Kroch, editors, *Alternative Conceptions of Phrase Structure*. Chicago University Press, Chicago.

Kasami, T. (1965). An efficient recognition and syntax algorithm for context-free languages. Scientific Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford MA.

Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, **ASSP-35**(3), 400–401.

Kingsbury, P. and Palmer, M. (2002). From Treebank to PropBank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*, Las Palmas, Spain.

Kingsbury, P., Palmer, M., and Marcus, M. (2002). Adding semantic annotation to the Penn Treebank. In *Proceedings of the Human Language Technology Conference*, San Diego, CA.

Kinyon, A. and Prolo, C. (2002). Identifying verb arguments and their syntactic function in the Penn Treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*, pages 1982–1987, Las Palmas, Spain.

Klein, D. and Manning, C. D. (2001). Parsing with Treebank grammars: Empirical bounds, theoretical models, and the structure of the Penn Treebank. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics and the 10th Conference of the European Chapter*, pages 330–337, Toulouse, France.

Komagata, N. (1999). *Information Structure in Texts: A Computational Analysis of Contextual Appropriateness in English and Japanese*. Ph.D. thesis, Computer and Information Science, University of Pennsylvania.

Lambek, J. (1958). The mathematics of sentence structure. *American Mathematical Monthly*, **65**, 154–170.

Lin, D. (1998). Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*, Granada, Spain.

Magerman, D. M. (1994). *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, Department of Computer Science, Stanford University.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

Marcus, M., Kim, G., Marcinkiewicz, M., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Human Language Technology Workshop.*

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, **19**, 313–330.

Minnen, G., Carroll, J., and Pearce, D. (2001). Applied morphological processing of English. *Natural Language Engineering*, **7**(3), 207–223.

Mitchell, T. M. (1997). *Machine Learning*. McGraw Hill.

Nunberg, G. (1990). *The linguistics of punctuation*. Number 18 in CSLI Lecture Notes. CSLI Publications.

Osborne, M. and Briscoe, T. (1998). Learning Stochastic Categorial Grammars. In T. M. Ellison, editor, *Proceedings of CoNLL97: Computational Natural Language Learning*, pages 80–87, Somerset, NJ.

Pareschi, R. and Steedman, M. (1987). A lazy way to chart parse with categorial grammars. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 81–88, Stanford, CA.

Pereira, F. and Schabes, Y. (1992). Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Newark, DE.

Pollard, C. and Sag, I. (1994). *Head Driven Phrase Structure Grammar*. CSLI/Chicago University Press, Chicago, IL.

Ratnaparkhi, A. (1996). A maximum entropy part-of-speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, Philadelphia, PA.

Ratnaparkhi, A. (1998). *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, Computer and Information Science, University of Pennsylvania.

Resnik, P. (1992). Probabilistic Tree-Adjoining Grammar as a framework for statistical natural language processing. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, volume 2, pages 418–424, Nantes, France.

Riezler, S., King, T. H., Kaplan, R. M., Crouch, R., Maxwell, J. T., and Johnson, M. (2002). Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 271–278, Philadelphia, PA.

Roark, B. (2001). Probabilistic top-down parsing and language modeling. *Computational Linguistics*, **27**(2), 249–276.

Ross, J. R. (1967). *Constraints on Variables in Syntax*. Ph.D. thesis, MIT. Published as “Infinite Syntax!”, Ablex, Norton, NJ. 1986.

Sarkar, A. (1998). Conditions on consistency of probabilistic Tree Adjoining Grammars. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING)*, pages 1164–1170, Montreal, Canada.

Schabes, Y. (1992). Stochastic Lexicalized Tree-Adjoining Grammars. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, volume 2, pages 426–432, Nantes, France.

Schmid, H. (2002). Lexicalization of probabilistic grammars. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, Taipei, Taiwan.

Steedman, M. (1987). Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory*, **5**, 403–439.

Steedman, M. (1996). *Surface Structure and Interpretation*. MIT Press, Cambridge, MA. Linguistic Inquiry Monograph, 30.

Steedman, M. (2000). *The Syntactic Process*. MIT Press, Cambridge, MA.

Uszkoreit, H. (1986). Categorial Unification Grammars. In *Proceedings of the 11th International Conference on Computational Linguistics (COLING)*, pages 187–194, Bonn, Germany.

Van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths, London, second edition.

Vijay-Shanker, K. and Weir, D. (1990). Polynomial time parsing of Combinatory Categorial Grammars. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Pittsburgh, PA.

Vijay-Shanker, K. and Weir, D. (1993). Parsing some constrained grammar formalisms. *Computational Linguistics*, **19**, 591–636.

Vijay-Shanker, K. and Weir, D. (1994). The equivalence of four extensions of context-free grammar. *Mathematical Systems Theory*, **27**, 511–546.

Villavicencio, A. (1997). *Building a wide-coverage Combinatory Categorial Grammar*. Master's thesis, Computer Laboratory, University of Cambridge.

Villavicencio, A. (2002). *The Acquisition of a Unification-Based Generalised Categorial Grammar*. Ph.D. thesis, Computer Laboratory, University of Cambridge.

Wang, W. and Harper, M. P. (2002). The SuperARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*.

Watkinson, S. and Manandhar, S. (2001). Translating Treebank Annotation for Evaluation. In *Workshop on Evaluation for Language and Dialogue Systems, ACL/EACL*, pages 21–28, Toulouse, France.

Wittenburg, K. (1987). Predictive combinators: a method for efficient processing of Combinatory Grammars. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 73–80, Stanford, CA.

Wittenburg, K. B. (1986). *Natural Language Parsing with Combinatory Categorial Grammar in a Graph-Unification Based Formalism*. Ph.D. thesis, University of Texas at Austin.

Wood, M. M. (1993). *Categorial Grammar*. Linguistic Theory Guides. Routledge, London.

Xia, F. (1999). Extracting Tree Adjoining Grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*.

Xia, F., Palmer, M., and Joshi, A. (2000). A uniform method of grammar extraction and its applications. In *Proceedings of the 2000 Conference on Empirical Methods in Natural Language Processing*, pages 53–62, Hong Kong.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time $O(n^3)$. *Information and Control*, **10**(2), 189–208.

Zeevat, H., Klein, E., and Calder, J. (1987). An introduction to Unification Categorial Grammar. In N. Haddock, E. Klein, and G. Morrill, editors, *Edinburgh Working Papers in Cognitive Science*, volume 1, pages 195–222. University of Edinburgh.