Alternative Phrases

Theoretical Analysis and Practical Application

Gann Bierner

Doctor of Philosophy
The University of Edinburgh
2001

Abstract

All right, but apart from the sanitation, the medicine, education, wine, public order, irrigation, roads, the fresh-water system and public health, what have the Romans ever done for us?

(Monty Python, *The Life of Brian*)

Alternative phrases identify selected elements from a set and subject them to particular scrutiny with respect to the sentence's predicate. For instance, in the above example, sanitation, medicine, etc. are all identified as elements in the set of things "the Romans have done for us" that should not be included in the response to the question. They are *alternative* responses to the desired ones. Alternative phrases come in a variety of constructions and perform a variety of tasks: excluding elements (*apart from*), expressing preference for particular elements (*especially*), and simply identifying representative examples (*such as*).

Not a great deal of work has been done on alternative phrases in general. Hearst (1992) used a pattern-matching analysis of certain alternative phrases to learn hyponyms from unannotated corpora. Also, a few examples from a subset of alternative phrases, called *exceptive phrases*, have been studied, most recently, by von Fintel (1993) and Hoeksema (1995). But not all constructions are amenable to pattern-matching techniques, and the work on exceptive phrases focuses on some very specific semantic points. The focus of this thesis is to present a general program for analyzing a wide variety of alternative phrases including their presuppositional and anaphoric properties.

I perform my analyses in Combinatory Categorial Grammar, a lexicalized formalism. The semantic aspects of the analysis benefit greatly from the concept of alternative sets, sets of propositions that differ in one or more argument (Karttunen and Peters, 1979; Rooth, 1985, 1992; Prevost and Steedman, 1994; Steedman, 2000a). In addition, elegant solutions are made possible by separating the semantics into assertion and presupposition (Stalnaker, 1974; Karttunen and Peters, 1979; Stone and Doran, 1997; Stone and Webber, 1998; Webber *et al.*, 1999b)— with each performing quite different tasks.

My second goal is to demonstrate the practicality and importance of this

analysis to real systems. Although it is relevant to many practical applications, I will focus primarily on natural language information retrieval (NLIR) as a case study. In such a domain, queries like *Where can I find other web browsers than Netscape for download?* and *Where can I find shoes made by Buffalino, such as the Bushwackers?* are often observed. I review several techniques for NLIR and demonstrate that implementations of those techniques perform poorly on such queries. I show that understanding alternative phrases can enable simple techniques which greatly improve precision.

To bridge the gap between these goals, I present Grok, a modular natural language system. Several general NLP issues necessary to support my linguistic analysis are discussed: anaphora resolution, processing of presuppositions, interface to knowledge representation, and the creation of a wide-coverage lexicon. Special attention is paid to the lexicon, which is a combination of a hand-built and an acquired lexicon.

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

Gann Bierner

Acknowledgements

When I arrived in Philadelphia to begin graduate school, I was extremely depressed. I had just finished four wonderful years at Rice University, and now I was in an empty room, in a large house, in an unfamiliar city. What am I doing here? I could be earning big bucks in industry right now!

So I did what I always do when things are looking a bit grim. I called my parents. They reminded me that new places are always difficult until you begin to meet friendly, helpful people. They were right, of course, and I soon found that the University of Pennsylvania (and later, the University of Edinburgh) had friendly, helpful people everywhere I looked. Without them, my graduate school career and this dissertation would never have been possible.

One hears a lot of horror stories about advisors. I was extraordinarily lucky that *my* advisors, Bonnie Webber and Mark Steedman, could not have been more different. They have been unfailing guides through the strange, overgrown jungle that is graduate school, and their comments and suggestions for this thesis were absolutely invaluable. They are also excellent cooks.

Many, many others have contributed to this thesis directly and indirectly. Some are Mark Core, Heidi Harley, Frank Keller, Geert-Jan Kruijff, Ivana Kruijff-Korbayova, Alex Lascarides, Johanna Moore, Tom Morton, Ambrose Nankivell, Jane Neumann, Kelly Smith, and Matthew Stone. I would particularly like to thank Marc Moens and Ewan Klein, my thesis proposal committee, and Jason Baldridge and Julia Hockenmaier, my office mates with whom I spent more hours at the white board than I can count. Also, the administrative help of Mike Felker at the University of Pennsylvania and Betty Hughes at the University of Edinburgh has been invaluable.

I would also like to thank Edwin Cooper, Bryan Sivak, and Emily Cooper from Electric Knowledge for their significant expenditure of time, effort, and data on behalf of my practical application. I would also like to thank the folks at SourceForge, the hosts of Grok. They have done a fantastic job.

My family has really seen me through these four years. My parents, Mark Bierner, Cassandra James, and Judy and Eric Soslau, have been encouraging,

loving, and supportive throughout my life. Thank you!

Most of all I would like to thank my wife, Constance Cooper Bierner. Her love and companionship brighten every day of my life. In addition to her emotional support, I, and this thesis, have benefited from her sharp mind, excellent proofreading skills, and awesome brownies.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

I begin with some data discovered in a corpus of queries submitted to the The Electric Monk (Monk 1999), the successor of the On Point natural language search system described in Cooper (1997), by users of its public portal. The examples in (1) show a query followed by the Monk's response, and then a follow-up query.

(1) a. *What is the drinking age in Afghanistan?*
(search results)
What is the drinking age in ***other*** *countries?*

b. *Where can I find web browsers for download?*
(search results)
Where can I find ***other*** *web browsers* ***than*** *netscape for download?*

c. *Where can I find a list of all the shoe manufacturers in the world?*
(search results)
Where can I find shoes made by Buffalino, ***such as*** *the Bushwackers?*

d. *Where are online auctions indexed?*
(search results)
Are there ***other*** *auction search engines* ***besides*** *BidFind?*

In each case, particular words are used to restrict the results: e.g. **such** (**as**), **other** (**than**), and **besides**. I will call these words, and others like them, *alternative markers*. Alternative markers along with their syntactic argument (e.g. *other countries*), I will call *alternative phrases*. Through their presuppositions, alternative markers provide a rich source of knowledge about the world. For example, the utterances in (1) imply that Afghanistan is a country, Netscape is a web browser, Bushwackers are shoes, and BidFind is an auction search engine. Anaphoric reference can sometimes be critical for these inferences. In (1a), for instance, *other countries* anaphorically depends on *Afghanistan* in the previous query.

1.1 Linguistic Analysis

The semantics of alternative phrases has not been treated extensively in the literature, although there has been work on some specific examples. Karttunen and Peters (1979) and Rooth (1985, 1992), for example, have done work on the focus particles **even** and **only**, which are related. More recently, von Fintel (1993) and Hoeksema (1995) have done in-depth semantic analyses of the exceptive markers **but** and **except** (**for**). However, there are many more alternative markers to consider, and furthermore, there are aspects of the analyses that deserve further attention.

The primary purpose of this dissertation is to present a computational analysis of a wide range of alternative phrases. By computational, I mean that my main concern is representing meaning in a way that is amenable to processing in a computational system. In Chapter 3 and Chapter 4, I present a syntactic/semantic analysis in Combinatory Categorial Grammar (CCG). I show that an elegant account can be achieved through a separation of semantics into assertion and presupposition discussed in Stalnaker (1974); Karttunen and Peters (1979) and with respect to lexicalized grammars in Stone and Doran (1997); Stone and Webber (1998); Webber *et al.* (1999b). The role of the assertion will be to carve out the appropriate set represented by the alternative phrase. The presuppositions, on the other hand, account for information communicated about the alternative marker's referent (e.g. *Afghanistan*, in (1a)). I also account for the anaphoric effects noted above. In Chapter 2, I discuss several relevant constructions so that the analyses and examples of alternative phrases can be viewed in a larger context.

1.2 A Robust System

The analyses of alternative phrases presented in Chapter 3 and Chapter 4 stand on their own, but they were originally motivated by their use in queries such as those in (1). For those analyses to be relevant to natural language information retrieval (NLIR) as well as other systems, it is necessary to show that they can be used in a real-world environment. Some linguistic analyses, while accounting for the pertinent data, may not be suitable for use in real systems. An analysis in GB theory or minimalism, for example, may not be amenable to efficient parsing.

Therefore, in Chapter 5, I present Grok, a large scale NLP system in which much of the alternative phrase analysis has been implemented. I discuss several issues including robust grammar development, parsing, using preprocessing to

handle non-uniform input and to simplify parsing, and semantic interpretation. This chapter is a bridge between the analysis in Chapters 2–4 and the practical application in Chapter 6 but is also a contribution in itself.

1.3 Practical Applications

A principled approach to alternative phrases is applicable to a wide variety of practical applications. Natural language search engines, as in (1), are only one such example. Further examples include coreference, named entity recognition, hyponym acquisition, database querying, and information extraction.

In Chapter 6, I demonstrate that alternative phrases are not just applicable to practical applications but also that a thoughtful treatment of them leads to improved performance. I discuss in depth the case study of natural language information retrieval. First, I show that alternative phrases are common enough in dialogue to warrant attention. I then demonstrate that the linguistic analysis along with the Grok system can be used as a front end to an NLIR system to improve the precision of queries containing alternative phrases.

1.4 Future Work and Conclusions

In addition to recognition, *generation* of alternative phrases is an important display of the practicality and flexibility of the analyses. The benefit of generating with these words is greater textual economy achieved by using the presuppositions of these words to satisfy communicative goals. However, alternative phrases present challenges for generation. Chapter 7 discusses these issues and proposes a way forward to developing a generator that can support these words.

Also in this chapter, I briefly discuss more ways alternative phrases can be useful in the NLIR setting. For one, I discuss the possibility of using the generation of alternative phrases to propose alternate queries to a user when the results of a previous query are unsatisfactory. I also propose using alternative phrases to acquire knowledge from queries, show how this can help NLIR, and discuss the inherent problems.

Chapter 2

A CCG English Grammar Fragment

2.1 Introduction

Chapter 3 will present an in-depth analysis of alternative phrases. However, the purpose of this dissertation is not to simply provide this analysis in isolation, but also to show how it fits into a larger context. Thus, this chapter briefly sketches germane aspects of an English grammar, but the analyses are not meant to be taken as theoretical results themselves. (For an in-depth look at a categorial grammar for English that is generally consistent with this chapter, see Carpenter 1989, 1992.) I particularly emphasize aspects of the grammar used in examples in the following chapters in order to make their derivations less mysterious.

I also take this opportunity to discuss my semantic and syntactic framework and introduce some notational conventions.

2.2 Background

2.2.1 Semantics

In this dissertation, I will use the typed lambda calculus to represent my semantics. The type system consists of the basic types e, entity, and t, truth-value. These are then combined to form function types, such as $\langle e, t\rangle$, which map entities to truth-values. In addition, I also require that the semantics be defined over the standard logical connectives ($\wedge$, $\vee$, and $\neg$) as well as existential and universal quantification ($\exists$ and $\forall$). This is simply the first-order extensional part of Montague semantics (Montague, 1970).

It should be understood that the representation of the semantics is distinct from its interpretation. Expressions of type $\langle e, t\rangle$ denote *characteristic functions* of sets whose elements are the entities for which the function returns true. Sets

have type $\langle e, t\rangle$ which should be interpreted as a function that returns true if the element is in the set and false if it is not. I attempt to separate the representation and interpretation of semantic forms as much as possible and will explicitly point out moments when I am discussing interpretation.

I discuss the semantic types for particular lexical items later on in this chapter.

2.2.2 CCG

My semantic analysis is tied to a syntactic analysis using Combinatory Categorial Grammar (CCG) (Steedman 1996, 2000b). CCG is a lexicalized grammar that encodes both the syntactic and semantic properties of a word in the lexicon. For instance, a transitive verb such as **likes** might have the lexical entry in (2). Note that for simplicity, I will sometimes abbreviate these lexical entries to: **likes** $\vdash$ $\mathsf{S\backslash NP/NP}$: $\lambda x\lambda y.\mathtt{like}(y, x)$.

A word on formatting. In lexical entries, derivations, and even in the text, I will display the orthography of a lexical entry like **this** and syntactic categories like $\mathsf{S\backslash NP}$. Semantics is displayed in *this* style except for constants which are displayed `thus`.

(2) **likes** $\vdash \begin{cases} syn: & \mathsf{S\backslash NP/NP} \\ sem: & \lambda x\lambda y.\mathtt{like}(y, x) \end{cases}$

In this notation for categories, results are always found to the left of the slash. (In some notations, this is not the case (Lambek, 1958; van Benthem, 1986, 1991).) The symbol / refers to a rightward-looking category and \ to a leftward-looking category. The | symbol refers to a category that finds its argument on either its left or right. (2) states that the syntactic category of **likes** is a functor that requires its argument, a noun phrase, on its right. The corresponding semantic argument is simultaneously collected and bound to the outer variable x. A new functor is returned whose syntactic argument, another noun phrase, must be on its left. The corresponding semantic argument is bound to y. The result is a sentence whose semantics is $\mathtt{like}(y, x)$ with x and y bound as described above.

In basic categorial grammar, CG, categories combine using the forward and backward function application rules:

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

b. *Backward Application* (<):
$\mathsf{Y} : a \quad \mathsf{X\backslash Y} : f \quad \Rightarrow \quad \mathsf{X} : f(a)$

The derivation in (4) shows how these rules combine the lexical items in *John likes Mary* into single syntactic and semantic categories. Here, > indicates forward application and <, backward application.

(4)
$$\dfrac{\dfrac{\textbf{John}}{\mathsf{NP} : \texttt{john}} \quad \dfrac{\dfrac{\textbf{likes}}{\mathsf{S}\backslash\mathsf{NP}/\mathsf{NP} : \lambda x\lambda y.\texttt{like}(y,x)} \quad \dfrac{\textbf{Mary}}{\mathsf{NP} : \texttt{mary}}}{\mathsf{S}\backslash\mathsf{NP} : \lambda y.\texttt{like}(y,\texttt{mary})}>}{\mathsf{S} : \texttt{like}(\texttt{john},\texttt{mary})}<$$

CCG adds to the basic CG base three classes of rules, each of which corresponds to one of the simplest combinators of Curry and Feys (1958). The three combinators incorporated into CCG are composition, type-raising, and substitution, abbreviated as **B**, **T**, and **S** respectively. Only composition and type-raising are relevant for this dissertation.

(5) Rules corresponding to the composition combinator **B**.

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

b. *Forward Crossing Composition* (>**B**$_\times$):
$\mathsf{X}/\mathsf{Y} : f \quad \mathsf{Y}\backslash\mathsf{Z} : g \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X}\backslash\mathsf{Z} : \lambda x.f(g(x))$

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

d. *Backward Crossing Composition* (<**B**$_\times$):
$\mathsf{Y}/\mathsf{Z} : g \quad \mathsf{X}\backslash\mathsf{Y} : f \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X}/\mathsf{Z} : \lambda x.f(g(x))$

(6) Rules corresponding to the type-raising combinator **T**.[1]

a. *Forward Type-raising* (>**T**):
$\mathsf{X} : a \quad \Rightarrow_{\mathbf{T}} \quad \mathrm{T}/(\mathrm{T}\backslash\mathsf{X}) : \lambda p.p(a)$

b. *Backward Type-raising* (<**T**):
$\mathsf{X} : a \quad \Rightarrow_{\mathbf{T}} \quad \mathrm{T}\backslash(\mathrm{T}/\mathsf{X}) : \lambda p.p(a)$

These rules can be restricted for particular languages. For instance, forward crossing composition is not allowed in English—a fact that will prove useful for my analyses in Chapter 4. It is also important to note that the introduction of type-raising and composition allow different, but semantically equivalent, derivations of the same string. Example (7) shows an alternative derivation to (4).

[1] T in these rules is a variable over categories.

(7)
$$\dfrac{\dfrac{\dfrac{\textbf{John}}{\mathsf{NP} : \texttt{john}} \quad \dfrac{\textbf{likes}}{\mathsf{S}\backslash\mathsf{NP}/\mathsf{NP} : \lambda x\lambda y.\texttt{like}(y,x)} \quad \dfrac{\textbf{Mary}}{\mathsf{NP} : \texttt{mary}}}{\mathsf{S}/(\mathsf{S}\backslash\mathsf{NP}) : \lambda p.p(\texttt{john})}\,{>}\mathbf{T}}{\mathsf{S}/\mathsf{NP} : \lambda x.\texttt{like}(\texttt{john},x)}\,{>}\mathbf{B}}{\mathsf{S} : \texttt{like}(\texttt{john},\texttt{mary})}\,{>}$$

The semantics presented in this dissertation uses the lambda calculus. However, I add that this is implemented in my system (Section 5.8) with a unification-based representation that mirrors the syntax (Steedman, 1996, p.14). Consequently, the derivation in (4) can also be shown as below. This will become significant in the discussions of practical application (Chapter 6) and generation (Section 7.3) where working with λ-expressions is difficult.

(8)
$$\dfrac{\dfrac{\textbf{John}}{\mathsf{NP}:\texttt{john}} \quad \dfrac{\dfrac{\textbf{likes}}{(\mathsf{S}:\texttt{like}(y,x)\backslash\mathsf{NP}:y)/\mathsf{NP}:x} \quad \dfrac{\textbf{Mary}}{\mathsf{NP}:\texttt{mary}}}{\mathsf{S}:\texttt{like}(y,\texttt{mary})\backslash\mathsf{NP}:y}\,{>}}{\mathsf{S}:\texttt{like}(\texttt{john},\texttt{mary})}\,{<}$$

One reason for choosing a lexicalized system such as CCG is the close coupling of the syntax and semantics of an analysis. This simplifies the problem of presenting a semantic theory while at the same time showing how it can be used to robustly parse and understand real data. Section 5.3.2 provides more discussion regarding using CCG in robust systems. I believe that the ideas presented in this thesis can also be implemented in other lexicalized grammar formalisms.

2.3 Nouns, Noun Phrases, and Determiners

CG grammars generally distinguish common nouns (N) from noun phrases (NP), corresponding to the semantic distinction between $\langle e,t\rangle$ and e. In essence, common nouns are therefore interpreted as properties and noun phrases as entities.

Unfortunately, this distinction leads to several incongruities in the rest of the grammar. For instance, all plural nouns must have both NP and N lexical entries to accommodate the examples in (9a). It is a good theoretical and engineering principle to minimize the number of syntactic categories (the Principle of Head Categorial Uniqueness—Steedman 2000b, p.33), so this is undesirable.

(9) a. $[\text{Dogs}]_{\mathsf{NP}}$ ate my homework.
$[\text{Five}]_{\mathsf{NP/N}}$ $[\text{dogs}]_{\mathsf{N}}$ ate my homework.
$[\text{All}]_{\mathsf{NP/N}}$ $[\text{dogs}]_{\mathsf{N}}$ ate my homework.

b. $[\text{All}]_{\mathsf{NP/NP}}$ $[\text{five}]_{\mathsf{NP/N}}$ $[\text{dogs}]_{\mathsf{N}}$ ate my homework.
$[\text{All}]_{\mathsf{NP/N}}$ $[\text{five}]_{\mathsf{N/N}}$ $[\text{dogs}]_{\mathsf{N}}$ ate my homework.

These examples presuppose that determiners have the category NP/N, but for examples like (9b), we see that things are more complex. Because multiple determiners can be strung together, determiners must either have NP/NP or N/N as an additional category. Again, having these additional categories is undesirable. (Note that determiners do not freely combine—see Keenan (1996) for an overview—but accounting for these phenomena is well beyond the scope of this thesis.)

These examples indicate that, syntactically at least, the distinction between NP and N does not lend itself to elegant analyses. Instead, I make the distinction between bare and non-bare noun phrases, where the non-bare NPs are those that have combined with determiners. Therefore all common nouns, singular and plural, are given the category $\mathsf{NP}_{\mathsf{bare:+}}$ and determiners are given $\mathsf{NP}_{\mathsf{bare:-}}/\mathsf{NP}$, where NP is underspecified for the bare feature. This allows the successful parsing of the above examples with far fewer lexical entries. It also simplifies the analyses in Chapter 3. Of course, unless verbal categories are restricted, sentences like *Dog ate my homework* will be allowed. This is the same as in the XTAG project (XTAG-group, 1999), which has a similar analysis for common nouns. Their solution is to allow these sentences with the restriction that singular, bare nouns must be interpreted as mass nouns. My general scheme for interpreting NPs is discussed in Section 5.7.3.

This analysis also allows a simpler analysis of relative clauses. The standard categorial analysis gives subject and object relative pronouns the category $(\mathsf{N}\backslash\mathsf{N})/(\mathsf{S}|\mathsf{NP})$ for relative clauses like *the dogs that ate my homework*. Bare plurals, on the other hand, such as *dogs that eat homework*, require the relative pronoun to be given the category $(\mathsf{NP}\backslash\mathsf{NP})/(\mathsf{S}|\mathsf{NP})$. We now give relative pronouns the category $(\mathsf{NP}\backslash\mathsf{NP})/(\mathsf{S}|\mathsf{NP})$, which is sufficient for both types relative clause.

Semantically, we must still account for the loss of the distinction between properties and entities. I claim that common nouns are still analyzed as functions of type $\langle e, t\rangle$, and thus a typical lexical entry would be (10). I also analyze proper nouns as functions of type $\langle e, t\rangle$ that return true for the entity denoted by the proper noun, and false otherwise. Note that I will often simplify the expression $\lambda x.\mathtt{p}(x)$ to `p` to save space. Thus $\lambda x.\mathtt{dog}(x)$ could be written as `dog`, as is the case for `john` and `mary` in (4).

(10) **dog** $\vdash \mathsf{NP}_{\mathsf{bare:+}} : \lambda x.\mathtt{dog}(x)$

We will see that the distinction between N and NP appeals to the fact that, as mentioned earlier, expressions of type $\langle e, t\rangle$ denote characteristic functions of a set whose elements are the entities for which the function returns true. Also, following Landman (1989) and Link (1983), a plural noun is closely associated with the power-set of the elements for which its function returns true.

Determiners are particularly difficult to analyze and are the subject of much study in order to correctly account for scoping ambiguities. Recent research has suggested that, actually, most determiners should not be analyzed as true quantifiers. Indefinite NPs, for instance, can sometimes just be analyzed as arbitrary objects, and delayed or immediate instantiation of this arbitrary object accounts for narrow and wide scope readings. (See Park 1995, 1996 and Steedman 1999 for details.) Furthermore, work by Koller and Niehren (2000) has shown that it is possible to resolve scope ambiguity after parsing by instantiating an underspecified semantic form. The simple analysis given below is consistent with these underspecified forms, so such techniques could be used. Therefore, scope ambiguity will not be discussed further since it is not required for this dissertation. The examples below show simple analyses for some common determiners (many more possible features are discussed in XTAG-group 1999).

(11) **the** $\vdash \mathsf{NP}_{\mathsf{num:}\boxed{1},\mathsf{def:}+,\mathsf{bare:}-}/\mathsf{NP}_{\mathsf{num:}\boxed{1}}$
two $\vdash \mathsf{NP}_{\mathsf{num:plural},\mathsf{def:}-,\mathsf{bare:}-}/\mathsf{NP}_{\mathsf{num:plural}}$
a $\vdash \mathsf{NP}_{\mathsf{num:singular},\mathsf{def:}-,\mathsf{bare:}-}/\mathsf{NP}_{\mathsf{num:singular}}$
some $\vdash \mathsf{NP}_{\mathsf{num:}\boxed{1},\mathsf{def:}-,\mathsf{bare:}-}/\mathsf{NP}_{\mathsf{num:}\boxed{1}}$

I will treat the basic semantics of determiners as the identity function with type $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$. A determiner, then, does not have a semantic effect until interpretation. At that point, the interpretation of an $\mathsf{NP}_{\mathsf{bare:}-}$ depends on the features that have been added by the determiner.

For example, an indefinite feature (`def:-`) will cause a new entity (a group, if the number is plural) to be created with the semantic property of the NP. Thus, *a dog* yields the semantic form $\mathtt{dog}_{bare:-,def:-}$. This will be interpreted as an anonymous entity which has the property of being a dog.

A definite feature will cause the interpretation to be an element or elements selected from a property's associated set. **The**, for example, will return the element of the set with maximal cardinality (there must be only one such element) restricted to the domain of discourse. Thus, if the set of dogs in the discourse is simply Fido, as in (12a), then *the dog* returns `fido` since it is the subset of maximum cardinality. It is important to remember that `fido` has type $\langle e, t\rangle$ and thus denotes the characteristic function of a set containing the element Fido. As

mentioned earlier, plurals are interpreted as the power-set of the singular. Thus the maximum subset of a plural is the original set of the singular. In (12b), then *The dogs* is interpreted as the full set of dogs. *Every dog*, as another example, simply returns the set denoted by *dog*. This leads to the same result as *the dogs* but contains no uniqueness requirement.

(12) a. *dog*: $\{\texttt{fido}\}$
the dog: $max(\{\texttt{fido}\}) = \texttt{fido}$

b. *dog*: $\{\texttt{fido}, \texttt{spot}, \texttt{phred}\}$
dogs: $\{\{\texttt{f}, \texttt{s}, \texttt{p}\}, \{\texttt{f}, \texttt{s}\}, \{\texttt{s}, \texttt{p}\}, \{\texttt{f}, \texttt{p}\}, \texttt{f}, \texttt{s}, \texttt{p}\}$
the dogs: $max(\{\{\texttt{f}, \texttt{s}, \texttt{p}\}, \{\texttt{f}, \texttt{s}\}, \{\texttt{s}, \texttt{p}\}, \{\texttt{f}, \texttt{p}\}, \texttt{f}, \texttt{s}, \texttt{p}\}) = \{\texttt{f}, \texttt{s}, \texttt{p}\}$

c. *dog*: $\{\texttt{fido}, \texttt{spot}, \texttt{phred}\}$
every dog: $\{\texttt{fido}, \texttt{spot}, \texttt{phred}\}$

2.4 Pronouns and Reflexives

Pronouns and reflexives are simply represented by variables. It is then the job of a resolution module, described in Section 5.7.2, to determine the antecedent. Both a binding theory and the syntactic features are used in restricting the possible antecedents.

Some examples are given below.

(13) **he** $\vdash \mathsf{NP}_{\mathsf{per:3,num:singular,gen:male,refType:pron}} : x$
himself $\vdash \mathsf{NP}_{\mathsf{per:3,num:singular,gen:male,refType:refl}} : x$
they $\vdash \mathsf{NP}_{\mathsf{per:3,num:plural,refType:pron}} : x$

2.5 A Note on Features

To this point, I have explicitly listed feature structures as comma separated lists of name/value pairs. It now becomes convenient to abbreviate this in the following manner. All binary features, such as bare : +, will now be expressed without the colon separator: e.g. bare+. For other features, unless there is an ambiguity, the name will be omitted. For example, I will write num : plural simply as plural.

2.6 Small Clauses and the Copula

Since in CCG we do not postulate empty categories, the burden of small clause constructions like in (14) must fall on either the NP or its modifier. We choose to have the modifiers, adjectives and prepositional phrases, do the job, giving the analyses in (15). The feature pred is used to indicate small clause predication. This comes from the "predicate raising" analysis of the inverse copular construction discussed in Heycock (1994). Although small clauses may not seem relevant to this dissertation, they are used in the analysis of queries in Section 2.7.

(14) a. I consider John smart.

b. I consider John the culprit.

c. I consider John in big trouble.

d. John is smart.

e. John is the culprit.

f. John is in big trouble.

(15) **is** $\vdash \mathsf{S}\backslash\mathsf{NP}/(\mathsf{S}_{\mathsf{pred}}\backslash\mathsf{NP}) : \lambda p\lambda x.p(x)$
consider $\vdash \mathsf{S}\backslash\mathsf{NP}/(\mathsf{S}_{\mathsf{pred}}\backslash\mathsf{NP})/\mathsf{NP} : \lambda x\lambda p\lambda y.\mathtt{consider}(y, p(x))$
smart $\vdash \mathsf{S}_{\mathsf{pred}}\backslash\mathsf{NP} : \lambda x.\mathtt{smart}(x)$
the $\vdash (\mathsf{S}_{\mathsf{pred}}\backslash\mathsf{NP})/\mathsf{NP} : \lambda x\lambda y.(x = y)$
in $\vdash (\mathsf{S}_{\mathsf{pred}}\backslash\mathsf{NP})/\mathsf{NP} : \lambda x\lambda y.\mathtt{in}(y, x)$

There are a few things to note about these lexical entries. First, **consider** does not take an $\mathsf{S}_{\mathsf{pred}}$ as an argument, but rather the subject and predicate separately. The purpose of this is to correctly allow extraction, as described in Section 2.7.

(16) a. The culprit is John.

b. * I consider the culprit John.

c. I consider the culprit to be John.

The inverse copular data in (16) raise additional syntactic problems that are irrelevant to the discussion in this thesis. In particular, while (16b) and (16c) are handled by the current analysis, (16a) is not. But I believe that the analysis to this point is consistent with the analysis in Heycock (1994), who treats this issue in depth.

2.7 Queries

Chapter 6 of this dissertation focuses on queries in the context of natural language information retrieval systems. Consequently, it is important to have an analysis for queries. I capture the data in (17).

(17) a. What/who likes spam?
b. What/who does Mary like?
c. What/which food does Mary like?
d. What/which person likes spam?
e. Does Mary like spam?
f. Can/could/shall/should/... Mary eat spam?
g. Who can/could/shall/should/... eat spam?
h. Why/when/how does Mary eat spam?
i. Is Mary insane/the leader/in the house?
j. How/where/who is Mary?

The analysis accounting for this data, (18), uses *mode* features to distinguish between sentence types. The mode can be indicative (ind), interrogative (int), predicative (pred), or non-inflected (base). The questions in (17) are correctly accounted for by this analysis[2] (e.g. (19)) and also reject the sentences in (20).

(18)

is	$\vdash$	$\mathsf{S}_{\mathsf{int}}/(\mathsf{S}_{\mathsf{pred}}\backslash\mathsf{NP})/\mathsf{NP}$	$\lambda x\lambda p.p(x)$
why/where/when/how	$\vdash$	$\mathsf{S}_{\mathsf{int}}/\mathsf{S}_{\mathsf{int}}$	$\lambda x.\mathtt{why/.../how}(x)$
where/when/how	$\vdash$	$\mathsf{S}_{\mathsf{int}}/(\mathsf{S}/(\mathsf{S}_{\mathsf{pred}}\backslash\mathsf{NP}))$	$\lambda p.p$
does	$\vdash$	$\mathsf{S}_{\mathsf{int}}/\mathsf{S}_{\mathsf{base}}$	$\lambda x.x$
can/will/...	$\vdash$	$\mathsf{S}_{\mathsf{int}}/\mathsf{S}_{\mathsf{base}}$	$\lambda x.\mathtt{can/will}(x)$
		$(\mathsf{S}_{\mathsf{ind}}\backslash\mathsf{NP})/(\mathsf{S}_{\mathsf{base}}\backslash\mathsf{NP})$	$\lambda p\lambda x.\mathtt{can/will}(p(x))$
what/who	$\vdash$	$\mathsf{S}_{\mathsf{int}}/(\mathsf{S}_{\mathsf{ind}}\backslash\mathsf{NP})$	$\lambda x.x$
		$\mathsf{S}_{\mathsf{int}}/(\mathsf{S}_{\mathsf{int}}/\mathsf{NP})$	$\lambda x.x$
what/which	$\vdash$	$\mathsf{S}_{\mathsf{int}}/(\mathsf{S}_{\mathsf{ind}}\backslash\mathsf{NP})/\mathsf{NP}$	$\lambda p\lambda q\lambda x.p(x)\wedge q(x)$
		$\mathsf{S}_{\mathsf{int}}/(\mathsf{S}_{\mathsf{int}}/\mathsf{NP})/\mathsf{NP}$	$\lambda p\lambda q\lambda x.p(x)\wedge q(x)$

(19) a.

$$\dfrac{\dfrac{\textbf{Who}}{\mathsf{S}_{\mathsf{int}}/(\mathsf{S}_{\mathsf{ind}}\backslash\mathsf{NP}) : \lambda x.x} \quad \dfrac{\textbf{likes spam}?}{\mathsf{S}_{\mathsf{ind}}\backslash\mathsf{NP} : \lambda x.\mathtt{like}(x,\mathtt{spam})}}{S_{int} : \lambda x.\mathtt{like}(x,\mathtt{spam})}\,{>}$$

[2] I do not consider echo questions like *Mary likes what?* and embedded questions like *I know what Mary likes.* Also, as this thesis does not deal with negation, I do not handle negative questions like *Doesn't Mary like spam?*

b.

What	**food**	**does**	**Mary like**?
$\mathsf{S_{int}/(S_{int}/NP)/NP}$: $\lambda p.\lambda q.\lambda x.p(x) \wedge q(x)$	NP : $\texttt{food}$	$\mathsf{S_{int}/S_{base}}$: $\lambda x.x$	$\mathsf{S_{base}/NP}$: $\lambda x.\texttt{like}(\texttt{mary}, x)$
	$\mathsf{S_{int}/(S_{int}/NP)}$: $\lambda q \lambda x.\texttt{food}(x) \wedge q(x)$ (>)		$\mathsf{S_{int}/NP}$: $\lambda x.\texttt{like}(\texttt{mary}, x)$ (>**B**)
$\mathsf{S_{int}}$: $\lambda x.\texttt{food}(x) \wedge \texttt{like}(\texttt{mary}, x)$ (>)			

c.

Does	**Mary like spam**?
$\mathsf{S_{int}/S_{base}}$: $\lambda x.x$	$\mathsf{S_{base}}$: $\texttt{like}(\texttt{mary}, \texttt{spam})$
$\mathsf{S_{int}}$: $\texttt{like}(\texttt{mary}, \texttt{spam})$ (>)	

d.

Who	**can**	**eat spam**?
$\mathsf{S_{int}/(S_{ind}\backslash NP)}$: $\lambda x.x$	$\mathsf{(S_{ind}\backslash NP)/(S_{base}\backslash NP)}$: $\lambda p \lambda x.\texttt{can}(p(x))$	$\mathsf{S_{base}\backslash NP}$: $\lambda y.\texttt{eat}(y, \texttt{spam})$
	$\mathsf{S_{ind}\backslash NP}$: $\lambda y.\texttt{can}(\texttt{eat}(y, \texttt{spam}))$ (>)	
$\mathsf{S_{int}}$: $\lambda y.\texttt{can}(\texttt{eat}(y, \texttt{spam}))$ (>)		

One interesting aspect of these derivations is that the analysis for **does**[3] is the same for (19b) and (19c) since it can combine through both forward application and forward composition. However, because forward crossing composition (> **Bx**) does not exist in English, (20a) fails. If such strings are to be grammatical, as in *John doesn't like spam, Mary doesn't like spam... Who DOES like spam?*, **does** can be given the same category as **can** and **will**. This category is the standard non-inverted auxiliary category, allowing sentences like *Mary does/can/... eat spam*, so it was not created solely for this purpose.

(20) a. * Who [does]$_{\mathsf{S/S}}$ [like spam]$_{\mathsf{S_{base}\backslash NP}}$?

b. * [Who]$_{\mathsf{S_{int}/(S_{int}\backslash NP)}}$ [like spam]$_{\mathsf{S_{base}\backslash NP}}$?

c. * [What]$_{\mathsf{S_{int}/(S_{int}/NP)}}$ [Mary like]$_{\mathsf{S_{base}/NP}}$?

d. * What [does]$_{\mathsf{S_{int}/S_{base}}}$ [Mary likes]$_{\mathsf{S_{ind}/NP}}$?

Note that the small clause analysis in the previous section is used in queries containing the copula, (21). A special category is required for **how**, **where**, and **when**. Clearly, the semantics of this category would have to specify what sort of property was required (method, location, time), but I have not done that here.

[3]Note that another analysis of the English auxiliary gives the category $\mathsf{S/(S\backslash NP)/NP}$ (Carpenter, 1992). This analysis works perfectly well for the examples given in this thesis, but I will use the $\mathsf{S/S}$ analysis for simplicity.

(21) a.

How	**is**	**Mary**?
$\mathsf{S_{int}/(S/(S_{pred}\backslash NP))}$: $\lambda p.p$	$\mathsf{S_{int}/(S_{pred}\backslash NP)/NP}$: $\lambda x\lambda p.p(x)$	NP : $\texttt{mary}$
	$\mathsf{S_{int}/(S_{pred}\backslash NP)}$: $\lambda p.p(\texttt{mary})$ (>)	
$\mathsf{S_{int}}$: $\lambda p.p(\texttt{mary})$ (>)		

b.

Is	**Mary**	**insane**?
$\mathsf{S_{int}/(S_{pred}\backslash NP)/NP}$: $\lambda x\lambda p.p(x)$	NP : $\texttt{mary}$	$\mathsf{S_{pred}\backslash NP}$: $\lambda x.\texttt{insane}(x)$
$\mathsf{S_{int}/(S_{pred}\backslash NP)}$: $\lambda p.p(\texttt{mary})$ (>)		
$\mathsf{S_{int}}$: $\texttt{insane}(\texttt{mary})$ (>)		

Also, in Section 2.6, **consider** was given the category $\mathsf{S\backslash NP/(S_{pred}\backslash NP)/NP}$ rather than $\mathsf{S\backslash NP/S_{pred}}$ to allow extraction. With the latter category, (22) would not be accepted because English does not have forward crossing composition, but the sentence is allowed in (23) with the other category.

(22) *What does [Mary consider]$_{\mathsf{S/S_{pred}}}$ [edible]$_{\mathsf{S_{pred}\backslash NP}}$?

(23)

What	**does**	**Mary**	**consider**	**edible**?
$\mathsf{S/(S/NP)}$: $\lambda x.x$	$\mathsf{S/S}$: $\lambda x.x$	$\mathsf{S/(S\backslash NP)}$: $\lambda p.p(\texttt{mary})$	$\mathsf{S\backslash NP/(S_{pred}\backslash NP)/NP}$: $\lambda x\lambda p\lambda y.\texttt{consider}(y, p(x))$	$\mathsf{S\backslash NP_{pred}}$: $\lambda x.\texttt{ed}(x)$
				$\mathsf{T\backslash T/(S\backslash NP_{pred})}$: $\lambda p.p(\lambda x.\texttt{ed}(x))$ (<T)
			$\mathsf{(S\backslash NP)/NP}$: $\lambda x\lambda y.\texttt{consider}(y, \texttt{ed}(x))$ (<B)	
		$\mathsf{S/NP}$: $\lambda x.\texttt{consider}(\texttt{mary}, \texttt{ed}(x))$ (>B)		
	$\mathsf{S/NP}$: $\lambda x.\texttt{consider}(\texttt{mary}, \texttt{ed}(x))$ (>B)			
S : $\lambda x.\texttt{consider}(\texttt{mary}, \texttt{ed}(x))$ (>)				

2.8 The Full English Lexicon

For more information on the English lexicon in Grok (Section 5.8) see:

http://grok.sourceforge.net/

Chapter 3

Connected Alternative Phrases

Luke... There is... another...Sky...Sky...walker.
(Yoda, *Return of the Jedi*)

3.1 Introduction

I begin my discussion of alternative phrases by introducing the concept of an *alternative set*. An alternative set is a set of propositions which differ with respect to how one or more arguments are filled. For example, the alternative set $\{\texttt{like}(\texttt{mary}, \texttt{jen}), \texttt{like}(\texttt{mary}, \texttt{bob}), ...\}$, summarized as $\lambda x.\texttt{like}(\texttt{mary}, x)$, represents the entities that Mary likes.

An early discussion of these structures is provided in Karttunen and Peters (1979) where an analysis is given for the focus particle **even**. They noted, as had been well studied in the literature, that **even** contributes meaning to a sentence but does not affect its truth value. That is, the sentence in (24)[1] conveys the primary meaning in (25) but also the meanings in (26). They argue that (26) is due to conventional implicature rather than presupposition as had been suggested in previous literature.

(24) Even Bill likes Mary.

(25) Bill likes Mary.

(26) a. Other people besides Bill like Mary.

b. Of the people under consideration, Bill is least likely to like Mary.

They provide the "rough" representation of (26a) in (27) where a is defined as the *focus* of the sentence and *...x...* is the *scope*. In the example above, the focus

[0]This chapter is an extended version of Bierner (2000) which, in turn, is an extended version of Bierner (1999).

[1]These examples are taken from Karttunen and Peters (1979).

is `bill` and the scope is $\texttt{like}(x, \texttt{mary})$. This representation states that there are more values for x than Bill. Another way of saying this is that the alternative set defined by $\lambda x.\texttt{like}(x, \texttt{mary})$ contains $\texttt{like}(\texttt{bill}, \texttt{mary})$ and the size of the set is greater than one.

(27) There are other x under consideration besides a such that ...x...

Rooth (1985, 1992) uses alternative sets to develop a detailed account of focus, particularly with the focus particle **only**. **Only** involves the restriction of an alternative set to a single element. For example, in (28), out of the alternative set described by $\lambda x.\texttt{like}(\texttt{mary}, x)$, only one element, $\texttt{like}(\texttt{mary}, \texttt{bob})$, is true. For an overview of focus using the related "structured meanings" approach, see von Stechow (1991).

(28) Mary likes only Bob

Alternative sets are of more than theoretical interest: Prevost and Steedman (1994) and Steedman (2000a) have shown that alternative sets form the semantic basis for contrast in intonation, and this can be exploited in speech generation (Bierner, 1998). Steedman (2000a) defines the concepts of *theme* (roughly the question the utterance addresses) and *rheme* (roughly the answer it provides) in terms of alternative sets: a theme presupposes an alternative set which is restricted by a rheme. Thus in (29b)[2], the theme, *Marcel proved* evokes an alternative set such as (30). The rheme, *completeness*, restricts that alternative set to the single item $\texttt{prove}(\texttt{marcel}, \texttt{completeness})$. Contrastive stress is then given to words in the rheme that contribute to the restriction of the alternative set—in this case simply **completeness**.

(29) a. What result did Marcel prove?

b. (Marcel proved) (COMPLETENESS)

(30) $$\left\{ \begin{array}{l} \texttt{prove}(\texttt{marcel}, \texttt{decidability}) \\ \texttt{prove}(\texttt{marcel}, \texttt{soundness}) \\ \texttt{prove}(\texttt{marcel}, \texttt{completeness}) \end{array} \right\}$$

Alternative set semantics is also useful in describing a large class of frequently occurring words, such as **besides**, **such** (**as**), and **other** (**than**), which I will call *alternative markers*. For example, **besides**, in the question *Who does Mary like besides Bob?*, appeals to the alternative set $\lambda x.\texttt{like}(\texttt{mary}, x)$ and considers all elements except for $\texttt{like}(\texttt{mary}, \texttt{bob})$.

[2]These examples are taken from Steedman (2000a).

As I discussed in Chapter 1, these words are relevant to practical applications because of their frequency and their linguistic properties. In natural language information retrieval (NLIR), for example, these words are used to restrict (and possibly order) the set of results. In addition, they imply information about the world that, in some systems, can be useful in answering that and future queries. The examples in (1) from the The Electric Monk that I discussed in Chapter 1, repeated below, demonstrate this.

(31) a. *What is the drinking age in Afghanistan?*
(search results)
What is the drinking age in ***other*** *countries?*

b. *Where can I find web browsers for download?*
(search results)
Where can I find ***other*** *web browsers* ***than*** *netscape for download?*

c. *Where can I find a list of all the shoe manufacturers in the world?*
(search results)
Where can I find shoes made by Buffalino, ***such as*** *the Bushwackers?*

d. *Where are online auctions indexed?*
(search results)
Are there ***other*** *auction search engines* ***besides*** *BidFind?*

For the Monk to answer the second query in (31a), it must identify Afghanistan as a country and exclude it from the current search. Similar reasoning in (31b) should conclude that Netscape is a web browser to be excluded from the search. For these examples, it is important to note that incorrectly including Afghanistan and Netscape in the results can overwhelm the results to the extent that no other answers are returned to the user. This is a serious failure in a retrieval system and is discussed in depth in Section 6.6.

In (31c), one must conclude that Bushwackers are shoes made by Buffalino which can be non-exclusively included in the search. Finally, in (31d), BidFind must be an auction search engine and excluded from the search.

In the above examples, the alternative markers are used in what I will call *connected alternative phrases* (modeled after similar terminology in Hoeksema 1995). These phrases are closely bound to the NP to which they refer. I will also consider *free alternative phrases* which behave more like parentheticals: e.g. (32).

(32) Other than Fido, every dog likes going for walks.

As I discuss in Chapter 6, alternative markers are practically relevant beyond the scope of NLIR, but these examples adequately demonstrate the properties of

alternative markers listed in (33). In terms of (31a), for example, these properties correspond to **other**'s anaphoric dependence to Afghanistan, the implication that Afghanistan is a country, and that Afghanistan should be excluded from the countries under consideration. I will focus on the first two properties but will also discuss the third.

(33) Properties of alternative markers:

a. They depend upon anaphora.

b. They presuppose and imply facts about the world.

c. They include or exclude alternatives from a larger set.

The rest of this chapter begins with a summary of previous work and my approach to the problem. I continue with a detailed discussion of connected alternative phrases in this chapter and free alternative phrases in the next.

3.2 Previous Work

Hearst (1992) demonstrates cases where pattern matching can be used to extract knowledge from some constructions with alternative phrases and gives the patterns in (34) as examples.

(34) a. **such** NP **as** {NP,}* {**or**|**and**} NP

b. NP {, NP}* {,} {**or**|**and**} **other** NP

c. NP {,} **including** {NP,}* {**or**|**and**} NP

d. NP {,} **especially** {NP,}* {**or**|**and**} NP

This technique may be adequate for her purpose, the acquisition of hyponyms from large corpora, because the goal is to take advantage of easily available information, not handle a wide variation of linguistic phenomena. Although the number of constructions is limited, that is perhaps made up for by the speed and robustness of pattern-matching techniques.

Pattern matching is not adequate for my purposes. For one thing, by itself it simply cannot handle anaphoric examples like (31a). Also, even for its primary purpose of extracting knowledge, it is likely to produce poor results if applied to free alternative phrases because of their attachment ambiguity. In addition, while the above patterns for **such**, **including**, and **especially** are completely reliable, (34b) can produce incorrect results. (35) shows that discourse and world knowledge must be taken into account.

(35) John's pet dog, Fido, was sick.
So John and the other dogs went for a walk.

In contrast, von Fintel (1993) and Hoeksema (1995) give in-depth semantic analyses. While they treat both connected and free constructions, they focus entirely on *exceptive phrases* (ways of referring to exceptions) illustrated by the lexical items **but** and **except** (**for**). The thrust of their analyses is directed towards how these words interact with determiners to determine the final set of entities. Since in queries, alternative phrases less frequently interact with determiners in this way, this interaction with determiners does not appear to be a significant problem in the NLIR application that I am concerned with. I am much more interested in the anaphoric and inferential aspects of these words, issues generally skirted by Hoeksema and von Fintel. I will discuss the relevant aspects of their work, with respect to free alternative phrases, in more depth in Chapter 4.

3.3 My Approach

In this thesis, I present an alternative, formal approach to alternative phrases that is wider in scope than the alternatives reviewed in Section 3.2 (although less detailed in some respects than von Fintel and Hoeksema's work).

Previously, in Chapter 2, I only presented assertional semantics. For my analysis of alternative phrases, I will also use presupposition. I take the pragmatic view of presuppositions explored by Lewis (1979) and Stalnaker (1974) which, stated loosely, sees them as propositions that must be true for an utterance to make sense. (For an overview of presupposition, see Beaver 1997.) I separate lexical semantics into assertion and presupposition as in Stalnaker (1974) and Karttunen and Peters (1979). The idea is also used in Webber *et al.* (1999b) to capture anaphoric (non-structural) links between discourse connectives and material derivable from previous discourse, and in Stone and Doran (1997) and Stone and Webber (1998) for natural language generation. This reveals a simple and elegant analysis.

The assertion is computed during the derivation as shown in Section 2.2.2. Evaluating presuppositions is a more complex issue and is described in Section 5.7.1. For now, I simply show the resulting presuppositions after the derivation. I will write lexical entries in the following form, where the semantic parameters scope both the assertion and presupposition:

$$\textbf{lexical item} \vdash \begin{cases} syn: & \text{syntactic category} \\ sem: & \lambda... \begin{cases} assert: & proposition \\ presup: & proposition* \end{cases} \end{cases}$$

I discussed earlier how alternative sets play an important role in understanding alternative phrases. Alternative sets are sets of propositions, but, in a restricted view, they can also be viewed as the pair of a property and a set. The set, called the *ground*, contains the differing members of the alternative set, and the property is the result of abstracting over those members. The property and set resemble the sets and their characteristic functions described in Section 2.3. My analyses refer to alternative sets through the relation $alts(p, q)$ which is defined as follows, where p and q are expressions of type $\langle e, t\rangle$:

$$(36) \quad alts(p,q) \iff \exists A \in \textit{alt-sets} \;\; s.t. \;\; \forall x.(p(x) \vee q(x)) \rightarrow x \in ground(A)$$

Intuitively, this relation specifies that the two sets of entities denoted by p and q can be found together in the ground of at least one alternative set in the knowledge base. The description component of the alternative set (i.e. the property) need not be known.

It is important to note that although I will talk about unifying these structures, they are relations—not just logical forms. Therefore, although this is glossed over in most of this thesis, the relation's properties should be respected, namely that it is symmetric and reflexive, but not transitive. This is implemented in Grok (Section 5.8) so that, for example, if a symmetric relation fails to unify, the unification is tried again with the arguments reversed.

The alternative phrases I analyze fall into two classes: those that assemble a set from elements and those that excise a set from a larger set (as in exceptive phrases). In either case, one particular set of elements is of interest, the *figure*. With assembly words, the figure is either admitted into the set, called the *ground*, or combined with a *complement* to form a set. With excision, the figure is explicitly excluded from the ground. The figure may derive from structurally-related constituents, or it may be given anaphorically.

The following sections show how the use of presupposition along with the *alts* relation can allow us to analyze both connected and free alternative phrases to capture their anaphoric and inferential properties.

3.4 Connected Alternative Phrases

3.4.1 Besides

Besides is an excision word whose figure and ground are given structurally, as opposed to anaphorically. (37), derived from (31d), demonstrates this by its presupposition that the figure, BidFind, is an auction search engine. I provide the analysis in (38), which shows the ground and the figure being taken as structural arguments that are then used in both the assertional and presuppositional parts of the semantics. In other constructions involving **besides**, the ground can derive from any argument or adjunct and also has an assembly use, as in the sentence *Besides John, Mary likes spam*. See Chapter 4 for further discussion.

(37) Are there auction search engines besides BidFind?

(38) **besides** $\vdash$
$$\begin{cases} syn: & \mathsf{NP\backslash NP/NP} \\ sem: & \lambda f \lambda g \begin{cases} assert: & \lambda x.g(x) \wedge \neg f(x) \\ presup: & \forall x.f(x) \rightarrow g(x) \\ & alts(f, \lambda x.g(x) \wedge \neg f(x)) \end{cases} \end{cases}$$

In this analysis, the assertion returns all entities of the ground except for the figure. The analysis asserts nothing of the figure regarding clause-level predication, as it may be true of the figure (39b) or it may not (39a).

(39) a. Fido is vicious and I hate him, but I like dogs besides Fido.
b. Fido may be my favorite, but I like dogs besides Fido.

In either case the speaker is committed to the fact that the figure (`fido`) has the ground property (`dogs`). This is expressed as the first presupposition in the analysis. (Since figures can be realized pronominally, the presupposition refers to the referent of the figure rather than its form. Section 5.7.1 discusses the consequences of this for implementation.) I choose to express this as a universal quantification for reasons of type consistency. The simplest way to interpret this expression is as set membership. That is, the elements of f are a subset of the elements of g, as in *dogs besides Fido*. This is implicit in the analysis of Hoeksema (1995). However, with certain instantiations the relation might not be set membership, as in *dogs besides poodles* where the relationship is *subtype*. This issue is at the heart of the problem of interpreting *generics* (see Carlson and Pelletier 1995), and Section 5.7.3 discusses heuristics for disambiguating the interpretation of these expressions.

The second presupposition states that the figure is an alternative to the other entities under consideration. This is a more general case of the first presupposition which explicitly refers to the property of the alternative set containing the figure and ground: i.e. `dog`. The *alts* relation leaves the alternative set, and thus the property, unspecified. I will show in Section 3.4.2 that this plays an important role.

The analysis is illustrated in the derivation in (40).

(40)

auction search engines	**besides**	**BidFind**
NP : `ase`	(NP\NP)/NP : $\lambda f \lambda g \lambda x.g(x) \wedge \neg f(x)$	NP : `bf`
	NP\NP : $\lambda g \lambda x.g(x) \wedge \neg \texttt{bf}(x)$ (>)	
NP : $\lambda x.\texttt{ase}(x) \wedge \neg \texttt{bf}(x)$ (<)		

presupposition set: $\begin{cases} \forall x.\texttt{bf}(x) \rightarrow \texttt{ase}(x) \\ alts(\texttt{bf}, \lambda x.\texttt{ase}(x) \wedge \neg \texttt{bf}(x)) \end{cases}$

Here, the most pertinent presupposition of **besides** communicates that BidFind is an auction search engine. If this knowledge is not already available to a system, it may be accommodated in the sense of Lewis (1979). But this is a limited form of accommodation in that we assume that these presuppositions can be resolved with entities already available from the discourse or a very limited set of the common ground (Section 3.6.3). I do not postulate new entities or keep around partial, or underspecified, representations for later instantiation.

The semantics of the NP *auction search engines besides BidFind* uses the assertion of **besides** to yield $\lambda x.\texttt{ase}(x) \wedge \neg \texttt{bf}(x)$—i.e. entities that are auction search engines and not Bidfind.

This analysis and those following identify $\forall x.f(x) \rightarrow g(x)$ as a presupposition. I show now that this is supported by several standard tests for presupposition (as opposed to assertion and implicature) as discussed in van der Sandt (1992), Beaver (1995) and Lagerwerf (1998).

Embedding Tests

The idea behind embedding tests is that the embedded context changes the assertion of the sentence without affecting a presupposition. For example, in the classic negation test, a presupposition should survive a negated context. For example, both (41a) and (41b) communicate (41c), which is the presupposition of **stop**.

(41) a. Jones stopped beating his grandmother.

b. Jones did not stop beating his grandmother.

c. Jones was beating his grandmother.

Similarly, with **besides**, both (42a) and (42b) presuppose (42c).

(42) a. I like dogs besides Fido.

b. I don't like dogs besides Fido.

c. Fido is a dog.

Presuppositions should also project through modal contexts and questions. Indeed, this is the case.

(43) a. It is possible that I like dogs besides Fido.

b. Do you like dogs besides Fido?

c. Fido is a dog.

Discourse Tests

The remaining tests recognize that presuppositions are affected by their context. If the context contains the information expressed by the presupposition, then the presupposition is satisfied. If the context contains information contradicting the presupposition, the presupposition is rejected, and the discourse is considered unacceptable. If the context does not entail the information expressed by the presupposition, then the presupposition is accommodated—as in the basic examples for **besides** I have presented so far.

(44) a. Fido is a dog. I like dogs besides Fido.

b. #Fido is a cat. I like dogs besides Fido.

c. Fido is a dog.

In (44), we see that this test succeeds for **besides**. The second sentence of both (44a) and (44b) contains the presupposition in (44c). The first example, where the context is consistent with the presupposition, is acceptable, while the second is not (on the assumption that the set of dogs excludes cats).

A final test is how a presupposition behaves within a conditional. In most conditionals, the presupposition is made available to the rest of the discourse. However, if the antecedent of the conditional entails the information communicated by the presupposition, then that information is not available to the rest of the discourse.

For example, Lagerwerf (1998) presents the following examples. In the first sentence of (45), **regrets** triggers the presupposition that the father is dead. The second sentence, in which the father is alive, is infelicitous. In (46), on the other hand, the presupposition does not project through the conditional and thus the second sentence is acceptable even though it implies the father is alive.

(45) If he was crying, then he regrets killing his father.
#But if he was happy, then his father is alive and kicking.

(46) If he killed his father, then he regrets killing his father.
But if his father is alive, he will be glad his father managed to survive.

Similarly, in (47), the presupposition that Fido is a dog survives the conditional, causing the second sentence, which implies that Fido is a cat, to be unacceptable. In contrast, in (48), the presupposition does not project past the conditional. Thus, implication in the second sentence that Fido is a cat is acceptable.

(47) If John has a big backyard, he owns dogs besides Fido.
#But if John's backyard is small, all his pets are cats.

(48) If Fido is a dog, John owns dogs besides Fido.
But if Fido is a cat, all his pets are cats.

Given that these standard tests are satisfied, I conclude that $\forall x.f(x) \rightarrow g(x)$, in the analysis of **besides** in (38), is a presupposition.

3.4.2 Such

Such, as in (31c) (repeated in (49) in a simplified form), differs from **besides** in two ways. First, it is an assembly word: Bushwackers, the figure, is required to be included in the resulting set. Secondly, the figure can derive anaphorically as well as structurally. Example (50) supports the second claim by showing that **such** can even appear in a different sentence than its figure, clearly indicating it is a discourse anaphor. The same analysis I propose for this will also account for (49′), where **such** and **as** are separated within a phrase. It would be possible, though less compact, to have a separate lexical entry to handle this example structurally.

(49) Where can I find shoes such as the Bushwackers?

(50) Bushwackers are very comfortable.
Where can I find such shoes?

(49′) Where can I find such shoes as the Bushwackers?

The examples given above use the restrictive sense of **such**. That is, it restricts the interpretation of the noun phrase by some property or properties of the figure. For example, we might only want shoes that are comfortable—like the Bushwackers. **Such** can also be used in a non-restrictive way, as in (51), where it simply provides an example of the set, but does not restrict it.

(51) Where can I find comfortable shoes, such as the Bushwackers?

The syntactic distribution of NP taking **such** can be seen in (52), showing, as discussed above, that **such** and the ground NP can be in any order. Here, parentheses indicate optionality.

(52) a. dogs such as Fido
b. * dogs such
c. a dog such as Fido
d. * a dog such
e. such dogs (as Fido)
f. such a dog (as Fido)
g. any such dog
h. any such dog as Fido

It is tempting, then, to give **such** the syntactic category NP|NP, the non-directional slash indicating that the argument can be taken on either side. However, if the ground NP appears first, the **as** modifier must appear, as indicated in (52b) and (52d). The non-directional category would incorrectly allow these examples. Therefore, I propose a separate analysis for each order, (53).

(53) a. **such** $\vdash \begin{cases} syn: & \mathsf{NP}_{\mathsf{eq}+,\mathsf{comp}-}/\mathsf{NP}_{\mathsf{kind}+} \\ sem: & \lambda g \begin{cases} assert: & \lambda x.g(x) \vee f(x) \\ presup: & \forall x.f(x) \rightarrow g(x) \\ & alts(f, \lambda x.g(x) \vee f(x)) \end{cases} \end{cases}$

b. **such as** $\vdash \begin{cases} syn: & \mathsf{NP}_{\mathsf{eq}+,\mathsf{comp}-} \backslash \mathsf{NP}_{\mathsf{kind}+}/\mathsf{NP} \\ sem: & \lambda g \lambda f \begin{cases} assert: & \lambda x.g(x) \vee f(x) \\ presup: & \forall x.f(x) \rightarrow g(x) \\ & alts(f, \lambda x.g(x) \vee f(x)) \end{cases} \end{cases}$

Since one cannot say *such the dog*, *such every dog*, or even *such any dog* the NP argument must be further restricted to be interpretable as a kind. For notational simplicity, I indicate this as a feature (kind+) on the syntax.

I treat **such** in (53) as having two presuppositions: (1) the figure, f, has the properties associated with the ground, g, and (2) the figure and assertion are alternatives. These are the same as for **besides**. The assertional semantics provided here is for non-restrictive **such**. For restrictive **such**, the set returned by the assertion should actually be $\lambda x.g(x) \vee f(x)$ restricted by salient properties of f, such as `comfortable` in (50). If no properties are available, an effective default might be to return the most specific subsumer of f with respect to g. I believe that this finer-grained semantics is only useful for practical systems given a large knowledge base. In the particular case of NLIR, we would also require an understanding of the retrieved documents that goes beyond simple indexing of words and phrases. This exists for certain specific domains, but not for general web-based retrieval. I discuss this issue more in Chapter 6.

The semantic analysis of **such** allows the simple syntactic/semantic derivation of *such shoes* given in (54).

(54)
$$\dfrac{\dfrac{\textbf{such}}{\mathsf{NP/NP} : \lambda g \lambda x.g(x) \vee f(x)} \quad \dfrac{\textbf{shoes}}{\mathsf{NP} : \texttt{shoe}}}{\mathsf{NP} : \lambda x.\texttt{shoe}(x) \vee f(x)}\,{>}$$

$$\text{presupposition set:} \quad \begin{cases} \forall x.f(x) \rightarrow \texttt{shoe}(x) \\ alts(f, \lambda x.\texttt{shoe}(x) \vee f(x)) \end{cases}$$

At this point, the semantics is dependent on the free variable f, the figure, which represents an anaphorically presupposed expression of type $\langle e, t \rangle$ (which could be a set). This is reflected by the fact that, in isolation, *such shoes* does not make sense. Although such anaphoric reference is difficult to resolve, in some constructions we can identify the figure without bringing full resolution techniques to bear. Some of these constructions are those that contain the word **as**. My analysis of NP-taking **as** is given in (56), from which we can perform the derivation in (57). At the same time, it rejects the NPs in (55). The eq feature, used to mark equative phrases, is responsible for restricting the first example. Lexical NPs are labeled with eq− and as requires eq+.

In a general sense, the syntactic analysis for **as** (and **than** in the next section), is consistent with the transformational-style account given in McCawley (1988, ch. 20) in that it views it as an adjunct of the NP rather than a complement.

(55) a. * [shoes]$_{\mathsf{NP_{eq-}}}$ [as the Bushwackers]$_{\mathsf{NP\backslash NP_{eq+}}}$

b. * [shoes]$_{\mathsf{NP}}$ [such]$_{\mathsf{NP/NP\ or\ NP\backslash NP/NP/as}}$

(56) **as** $\vdash \begin{cases} syn: & \mathsf{NP\backslash NP_{eq+,comp-}/NP} \\ sem: & \lambda x\lambda y \begin{cases} assert: & y \\ presup: & alts(x,y) \end{cases} \end{cases}$

(57)

such	**shoes**	**as**	**the Bushwackers**
$\mathsf{NP_{eq+}/NP}: \lambda g\lambda x.g(x) \vee f(x)$	$\mathsf{NP_{eq-}}: \mathtt{shoe}$	$(\mathsf{NP\backslash NP_{eq+}})/\mathsf{NP}: \lambda x\lambda y.y$	$\mathsf{NP}: \mathtt{b}$
	$\mathsf{NP_{eq+}}: \lambda x.\mathtt{shoe}(x) \vee f(x)$ (<)	$\mathsf{NP\backslash NP_{eq+}}: \lambda y.y$ (>)	
$\mathsf{NP}: \lambda x.\mathtt{shoe}(x) \vee f(x)$ (<)			

presupposition set: $\begin{cases} \text{a.} & \forall x.f(x) \rightarrow \mathtt{shoe}(x) \\ \text{b.} & alts(f, \lambda x.\mathtt{shoe}(x) \vee f(x)) \\ \text{c.} & alts(\mathtt{b}, \lambda x.\mathtt{shoe}(x) \vee f(x)) \end{cases}$

The presupposition set is the union of the presuppositions of **such** and **as**, as bound during the derivation. The remaining variable, f, can be determined solely from the presupposition set of (57) using the old AI planning heuristic "use existing objects" (Sacerdoti, 1977) to avoid inventing new objects when others are already available. In particular, we can unify (57b) and (57c), discovering that f, the figure, is $\mathtt{b}$, or Bushwackers. This then instantiates (57a), yielding $\forall x.\mathtt{b}(x) \rightarrow \mathtt{shoe}(x)$: i.e. Bushwackers are shoes. Unifying logical forms to instantiate variables in this way follows the "interpretation as abduction" paradigm (Hobbs *et al.*, 1988, 1993), where this merging is performed to exploit redundancy for "getting a minimal, and hence a best, interpretation." This is, in essence, what I am trying to do.

That Bushwackers are a type of shoe may or may not already be present in the discourse. If it is not, as before we can accommodate the fact.

3.4.3 Other

The semantic analysis in (59) defines **other**, like **besides**, as an excision word that excludes the figure from the ground[3]. However, as with **such**, it differs from **besides** in that the figure can be derived anaphorically rather than being contained structurally. Also as with **such**, data shows us that the NP argument can

[3] **Other** can appear in other syntactic constructions which I review in Chapter 4.

appear on either side of **other** but that NPs such as *dogs other* are ungrammatical (58). Thus, for the same reason, I propose two lexical entries.

As mentioned in Section 2.3, I exclude N as a basic category and instead, distinguish between bare and non-bare NPs. Non-bare NPs cannot be rightward arguments of **other** (58f), so I place a bare+ restriction on these NPs (62).

(58) a. dogs other than Fido
b. * dogs other
c. a dog other than Fido
d. * a dog other
e. other dogs (than Fido)
f. * other a dog (than Fido)
g. any other dog (than Fido)

(59) a. **other** $\vdash \begin{cases} syn: & \mathsf{NP}_{\mathsf{comp+,eq-}}/\mathsf{NP}_{\mathsf{bare+}} \\ sem: & \lambda g \begin{cases} assert: & \lambda x.g(x) \wedge \neg f(x) \\ presup: & \forall x.f(x) \rightarrow g(x) \\ & alts(f, \lambda x.g(x) \wedge \neg f(x)) \end{cases} \end{cases}$

b. **other than** $\vdash \begin{cases} syn: & \mathsf{NP}_{\mathsf{comp+,eq-}}\backslash\mathsf{NP}/\mathsf{NP} \\ sem: & \lambda g \lambda f \begin{cases} assert: & \lambda x.g(x) \wedge \neg f(x) \\ presup: & \forall x.f(x) \rightarrow g(x) \\ & alts(f, \lambda x.g(x) \wedge \neg f(x)) \end{cases} \end{cases}$

In derivation (60), the analysis in (59a) interprets *other countries* as the set of countries not including the figure. As with **such**, the figure must be available from elsewhere in the sentence, from the discourse, or from the common ground (Section 3.6.3).

(60)
$$\dfrac{\dfrac{\textbf{other}}{\mathsf{NP/NP} : \lambda g \lambda x.g(x) \wedge \neg f(x)} \quad \dfrac{\textbf{countries}}{\mathsf{NP} : \mathtt{country}}}{\mathsf{NP} : \lambda x.\mathtt{country}(x) \wedge \neg f(x)}\,{>}$$

presupposition set: $\begin{cases} \forall x.f(x) \rightarrow \mathtt{country}(x) \\ alts(f, \lambda x.\mathtt{country}(x) \wedge \neg f(x)) \end{cases}$

While the full problem of identifying a presupposed figure is not yet solved (cf. Section 3.6), we can decisively identify the figure for **other** in constructions

containing **than**. The word **than** has a very similar analysis to **as** except for its eq and comp features. These avoid over-generating sentences such as (63) while performing the derivation in (64) (from the example in (31b)). This use of features to avoid over-generation follows the XTAG analysis for comparatives in XTAG-group (1999) to block phrases like *more brighter than dark*, *as patient than Bill*, and *more patient as Bill*.

(61) **than** $\vdash \begin{cases} syn: & \mathsf{NP}\backslash\mathsf{NP}_{\mathsf{eq}-,\mathsf{comp}+}/\mathsf{NP} \\ sem: & \lambda x\lambda y \begin{cases} assert: & y \\ presup: & alts(x,y) \end{cases} \end{cases}$

(62) * [other]$_{\mathsf{NP}/\mathsf{NP}_{\mathsf{bare}+}}$ [a dog]$_{\mathsf{NP}_{\mathsf{bare}-}}$

(63) a. * [other web browsers]$_{\mathsf{NP}_{\mathsf{comp}+,\mathsf{eq}-}}$ [as Netscape]$_{\mathsf{NP}\backslash\mathsf{NP}_{\mathsf{eq}+}}$

b. * [such web browsers]$_{\mathsf{NP}_{\mathsf{comp}-,\mathsf{eq}+}}$ [than Netscape]$_{\mathsf{NP}\backslash\mathsf{NP}_{\mathsf{comp}+}}$

(64)

other	**web browsers**	**than**	**Netscape**
$\mathsf{NP}_{\mathsf{comp}+}/\mathsf{NP}$: $\lambda g\lambda x.g(x) \wedge \neg f(x)$	$\mathsf{NP}_{\mathsf{comp}-}$: `browser`	$(\mathsf{NP}\backslash\mathsf{NP}_{\mathsf{comp}+})/\mathsf{NP}$: $\lambda x\lambda y.y$	NP : `netscape`
	$\mathsf{NP}_{\mathsf{comp}+}$: $\lambda x.\texttt{browser}(x) \wedge \neg f(x)$ (>)		$\mathsf{NP}\backslash\mathsf{NP}_{\mathsf{comp}+}$: $\lambda y.y$ (>)
	NP : $\lambda x.\texttt{browser}(x) \wedge \neg f(x)$ (<)		

presupposition set: $\begin{cases} \forall x.f(x) \rightarrow \texttt{browser}(x) \\ alts(f, \lambda x.\texttt{browser}(x) \wedge \neg f(x)) \\ alts(\texttt{netscape}, \lambda x.\texttt{browser}(x) \wedge \neg f(x)) \end{cases}$

As in Section 3.4.2, we unify the last two presuppositions and thus determine that the figure is Netscape. Our first presupposition is instantiated to indicate that Netscape is a browser, which we can accommodate if not already known. In addition, the assertional semantics denotes browsers that are not Netscape.

A remaining problem involves **other** NPs with post-modifiers which can convey either old material (for identifying what is to be excluded) or new material (to be predicated of the new entity). In speech, intonation can disambiguate: for instance, in *penguins and other birds that CAN fly*, `bird` but not `flying` is predicated of `penguin`, while in *robins and other birds that can fly*, both are predicated of `robin`. The same is true for prepositional phrases as in *crows and other birds with RED wings* and *cardinals and other birds with red wings*. For most practical applications, however, disambiguation remains a problem.

In my analysis, this phenomenon is reflected by an attachment ambiguity. If the post-modifier attaches to the ground alone, then the presupposition of **other**

requires that the post-modifier apply to the figure (65). If it attaches to the entire alternative phrase, then the post-modifier has no effect on the figure (66).

(65)

robins	...	**other**	**birds**	**that can fly**
NP : `robin`		NP/NP : $\lambda g\lambda x.g(x) \wedge \neg f(x)$	NP : `bird`	NP\NP : $\lambda p\lambda x.p(x) \wedge \texttt{fly}(x)$
			NP : $\lambda x.\texttt{bird}(x) \wedge \texttt{fly}(x)$ <	
		NP : $\lambda x.\texttt{bird}(x) \wedge \texttt{fly}(x) \wedge \neg f(x)$ >		
		NP : $\lambda x.\texttt{bird}(x) \wedge \texttt{fly}(x) \wedge \neg \texttt{robin}(x)$		

presupposition: $\forall x.\texttt{robin}(x) \rightarrow \texttt{bird}(x) \wedge \texttt{fly}(x)$

(66)

penguins	...	**other birds**	**that CAN fly**
NP : `penguin`		NP : $\lambda x.\texttt{bird}(x) \wedge \neg f(x)$	NP\NP : $\lambda p\lambda x.p(x) \wedge \texttt{fly}(x)$
		NP : $\lambda x.\texttt{bird}(x) \wedge \neg f(x) \wedge \texttt{fly}(x)$ <	
		NP : $\lambda x.\texttt{bird}(x) \wedge \neg \texttt{penguin}(x) \wedge \texttt{fly}(x)$	

presupposition: $\forall x.\texttt{penguin}(x) \rightarrow \texttt{bird}(x)$

If this is true, a situation where the modifier can only attach to *birds*, rather than *other birds*, should yield the reading where the figure has the modifier's property. Thus, using a different syntactic form of **other** (the semantics works out the same—see Section 4.3), we can produce the sentences in (67). Despite conflicting world knowledge, (67a) communicates that penguins fly. Even with the same intonation as in our previous example, (67b) also communicates that penguins fly—as predicted. World knowledge seems to not have an effect since the result is the same with both **penguins** and **robins**. Because this conflicts with the contrastive stress on **can**, this sentence is perceived as odd. (68) contains a short discourse that makes such a sentence palatable by providing another referent for the contrastive stress. The meaning of the sentence is that John does not like penguins and does not like robins, but he does like flying birds that are not robins.

(67) a. Other than penguins, John likes every bird that can fly.

b. #Other than penguins, John likes every bird that CAN fly.
#Other than robins, John likes every bird that CAN fly.

(68) John doesn't care for penguins because they can't fly.
But other than robins, he likes every bird that CAN fly.

3.4.4 Comparatives

Comparatives such as **better**, **faster**, and **longer** behave very similarly to **other**. Representing the semantics of comparatives, and adjectives in general, is a com-

plex topic which I will not address here (see Beesley 1983 and Klein 1991 for an overview). However, I will briefly discuss cases where they have presuppositions which one can take advantage of in a practical system.

Syntactically and presuppositionally, I treat comparatives similarly to **other** though, in some instances, they lack one of its two presuppositions. Consider the examples in (69).

(69) a. Fido walks. Bigger dogs run.
b. Bigger dogs than Fido run.
c. Dogs bigger than Fido run.

While (69a) and (69b) presuppose that Fido is a dog, this is not the case for (69c). (Consider the sentence *Dogs bigger than a bread-box are common.*) This observation was made at least as early as Chomsky (1965, p.180, 234). I take the observation as additional evidence that there should be separate analyses for taking arguments on the left and the right.

I discuss comparatives further in Section 6.4.3.

3.5 Scoping Alternative Phrases

Anaphoric alternative markers can scope each other. (70) gives two examples from the British National Corpus (BNC).

(70) a. Like other such instincts, however, envy is both preservative and destructive.
b. After four days and nights of play, the final table had seen him bust out such other local legends as Clyde "Slippery" Coleman and the dread Dave Crunkleton.

In (70a), where **other** has wide scope, the figures for both **other** and **such** are the same. That is, the sentence refers to instincts that are not envy but are similar to envy. It would be nice if it fell out of my theory that this is the case. However, this is not quite as simple as when we unified the presupposition of **other** with that of **than**. As (71) shows, the *alts* presuppositions of **other** and **such** for this example are not the same. Note that I represent the semantics of restrictive **such** as $g \sim f$, which means, roughly, elements of g that are like f.

(71) such: $alts(f_1, \texttt{instinct} \sim f_1)$
$\forall x.f_1(x) \rightarrow \texttt{instinct}(x)$
other: $alts(f_2, (\texttt{instinct} \sim f_1) \wedge \neg f_2)$
$\forall x.f_2(x) \rightarrow (\texttt{instinct} \sim f_1)(x)$

Because the presuppositions are different, they can not be unified, and the figures cannot be collapsed. However, it is unlikely that the "use existing objects" heuristic is restricted to simple unification of existing logical forms. Rather, the idea is to collapse objects when that is the simplest consistent explanation, so some simple reasoning is not out of place.

(72) $alts(f, f)$

(73) $alts(f, x) \rightarrow alts(f, x \vee f)$

(74) $(\texttt{instinct} \sim f_1) \equiv ((\texttt{instinct} \sim f_1) \vee f_2)$

I reintroduce the reflexive property of the *alts* relation mentioned in Section 3.3, (72). This axiom has the immediate consequent in (73). This simple bit of reasoning results in a slight variation of **other**'s presupposition, shown in (75). The only piece of knowledge that we need is that f_2 is included in $(\texttt{instinct} \sim f_1)$. This knowledge is given as the second presupposition of **other** in (71) and allows the equivalence in (74).

(75) $alts(f_2, (\texttt{instinct} \sim f_1) \wedge \neg f_2)$

$\rightarrow$	$alts(f_2, ((\texttt{instinct} \sim f_1) \wedge \neg f_2) \vee f_2)$	(73)
$\equiv$	$alts(f_2, ((\texttt{instinct} \sim f_1) \vee f_2) \wedge (\neg f_2 \vee f_2))$	*distributivity*
$\equiv$	$alts(f_2, (\texttt{instinct} \sim f_1) \vee f_2)$	*complementarity*
$\equiv$	$alts(f_2, \texttt{instinct} \sim f_1)$	(74)

The resulting relation can now unify with the *alts* presupposition of **such** in (71), collapsing the figures such that $f_1 = f_2$. Looking ahead, Section 3.6.1 describes how **like** produces the presupposition in (76). This can unify with the original presupposition of **other** in (71), identifying f_2 as `envy`, and thus f_1 as well.

(76) $alts(\texttt{envy}, (\texttt{instinct} \sim f_1) \wedge \neg f_2)$

Note that there is no aspect of (75) that is specific to **such**. Thus, for any presupposition $alts(f, x \wedge \neg f)$ of **other**, it will also be true that $alts(f, x)$. Given that all alternative phrases produce a presupposition of the form $alts(f', x')$, this analysis predicts that **other** will prefer to share the figure of any alternative phrase that it scopes. This is demonstrated in (77) where the ships being referred to by *other smaller ships* are different from and smaller than the same thing—the fifty warships.

(77) I knew that just beyond the narrow sea separating the two countries there were at least fifty warships ready to attack us, with many other smaller ships.

In contrast, when **such** has wide scope, it is not the case that both figures are likely to be the same. This is the case in (70b) where the local legends being referred to are different from the referent of *him* and similar to Coleman and Crunkleton. And in (78), the men are similar to Certon and Sandrin. It is unclear whether they are younger than Attaingnant or Janequin and Sermisy, but the pertinent point is that *younger* does not refer to Certon and Sandrin.

(78) And beside Janequin and Sermisy, Attaingnant brought out the songs of such younger men as the immensely prolific Pierre Certon and Sandrin, whose "Doulce memoire" was transcribed for lute or keyboard all over Europe from Spain to Poland.

The presuppositions shown in (79) show why this is so. Here, the reflexive property of the *alts* relation does not help us. The two presuppositions remain ununifiable, so the "use existing objects" heuristic of unifying the presuppositions cannot be used.

(79) other: $alts(f_1, \texttt{local-legend} \wedge \neg f_1)$
such: $alts(f_2, (\texttt{local-legend} \wedge \neg f_1) \sim f_2)$

These interactions are very interesting and deserve further investigation with respect to other anaphoric phrases. But that is beyond the scope of this thesis.

3.6 Finding the Figure

Identifying the figure when interpreting words like **such** and **other** (i.e. when the figure is not given structurally) appears comparable to determining the referent of a pronoun or NP. However, there are several common constructions, besides those created with **as** and **than**, that make the presupposed figure easily identifiable.

3.6.1 Exploiting the Presence of Other Alternative Markers

Like and **unlike** are assembly words, but unlike other alternative markers I have discussed, they do not take a ground. Rather, they take a figure and complement and presuppose that they are alternatives. For example, in *Unlike Mary, John likes spam*, we presuppose that Mary and John are alternatives. In this case, no explicit evidence is given for what alternative set they belong to. However, one source of evidence is the appearance of **other** in one of the arguments—e.g. in the first sentence of this paragraph. The analysis of **unlike** in (80) provides the partial derivation in (81). Note that the syntax is a simplification of that

presented in Section 4.4. Aspects of the semantics, such as the status of $\neg p(x)$ as a presupposition, are also discussed in that section.

(80) **unlike** $\vdash \begin{cases} syn: & \mathsf{S/VP/NP/NP} \\ sem: & \lambda x\lambda y\lambda p \begin{cases} assert: & p(y) \\ presup: & \neg p(x) \\ & alts(x,y) \end{cases} \end{cases}$

As in Sections 3.4.2 and 3.4.3, we can "use existing objects" and unify the last two presuppositions, determining that the figure is the set of **like** and **unlike** which, through the first presupposition, are inferred to belong to the set of alternative markers.

(81)

unlike	**other**	**alt markers**,	"**like**" **and** "**unlike**"
$((\mathsf{S/VP})/\mathsf{NP})/\mathsf{NP}$: $\lambda x\lambda y\lambda p.p(y)$	$\mathsf{NP/NP}$: $\lambda g\lambda x.g(x) \wedge \neg f(x)$	NP : $\mathtt{am}$	NP : $\lambda x.x \in \{\text{“like”}, \text{“unlike”}\}$
	$\mathsf{NP} : \lambda x.\mathtt{am}(x) \wedge \neg f(x)$ >		
$(\mathsf{S/VP})/\mathsf{NP} : \lambda y.\lambda p.p(x)$ >			
$\mathsf{S/VP} : \lambda p.p(\lambda x.x \in \{\text{“like”}, \text{“unlike”}\})$ >			

presupposition set: $\begin{cases} \forall x.f(x) \rightarrow \mathtt{am}(x) \\ \neg p(\lambda x.\mathtt{am}(x) \wedge \neg f(x)) \\ alts(f, \lambda x.\mathtt{am}(x) \wedge \neg f(x)) \\ alts(\lambda x.x \in \{\text{“like”}, \text{“unlike”}\}, \lambda x.\mathtt{am}(x) \wedge \neg f(x)) \end{cases}$

It is important to note that this still works perfectly well if the NPs are reversed, as in (82). Remember that *alts* is a symmetric relation and therefore the unification process will try both orders of the arguments. Thus, the interpretation succeeds.

(82)

unlike	"**like**" **and** "**unlike**",	**other**	**alt markers**
$((\mathsf{S/VP})/\mathsf{NP})/\mathsf{NP}$: $\lambda x\lambda y\lambda p.p(y)$	NP : $\lambda x.x \in \{\text{“like”}, \text{“unlike”}\}$	$\mathsf{NP/NP}$: $\lambda g\lambda x.g(x) \wedge \neg f(x)$	NP : $\mathtt{am}$
$\mathsf{S/VP/NP} : \lambda y\lambda p.p(y)$ >		$\mathsf{NP} : \lambda x.\mathtt{am}(x) \wedge \neg f(x)$ >	
$\mathsf{S/VP} : \lambda p.p(\lambda x.\mathtt{am}(x) \wedge \neg f(x))$ >			

presupposition set: $\begin{cases} \forall x.f(x) \rightarrow \mathtt{am}(x) \\ \neg p(\lambda x.x \in \{\text{“like”}, \text{“unlike”}\}) \\ alts(f, \lambda x.\mathtt{am}(x) \wedge \neg f(x)) \\ alts(\lambda x.\mathtt{am}(x) \wedge \neg f(x), \lambda x.x \in \{\text{“like”}, \text{“unlike”}\}) \end{cases}$

Unlike is not the only lexical item that can identify the figure in this way. Free alternative phrases such as **in addition to**, **besides**, and **other than**, when used in an assembly context, do so as well.

3.6.2 List Contexts

List contexts such as (83) also provide a situation in which we can identify the figure. I include a presupposition with **and** (and list-forming commas) that states the coordinated items are alternatives—the same presupposition given for **as** and **than** (Section 3.4.2). The presupposition of **and** is instantiated in (83) to (84a). The presuppositions of **other** are instantiated as in (60), repeated in (84b). This is now equivalent to (64) where we can identify the figure, Afghanistan, though unification.

(83) What is the drinking age in Afghanistan and other countries?

(84) a. $alts(\texttt{afghanistan}, \lambda x.\texttt{country}(x) \wedge \neg f(x))$

b. $\forall x.f(x) \rightarrow \texttt{country}(x)$
$alts(f, \lambda x.\texttt{country}(x) \wedge \neg f(x))$

(85) What is the drinking age in Afghanistan, other countries, and Dallas?

In the case of (85), however, we must ensure that Dallas is not considered a country. Assuming that the grammar collects list items from left to right (easily implemented in CCG or through an incremental parser), *Afghanistan* and *other countries* will be combined first. Because we unify presuppositions as soon as possible (as in Hobbs *et al.* (1988, 1993)), by the time *Dallas* is combined into the list, the figure has already been resolved to Afghanistan in the manner described above. Therefore, Dallas cannot be the figure and is not identified as a country.

(86) John's pet dog, Fido, was sick.
So, John and the other dogs went for a walk.

There are cases where this heuristic seems to fail, such as (86). Clearly, John should not be presupposed to be a dog. Making this mistake is due to not taking into account that, to be considered, possible antecedents must be compatible with the presupposition. Thus, John should not be considered when resolving the figure because we know he is a dog owner, hence human, hence not a dog. This means that the process by which a figure is found through the unification of presuppositions fails, leaving the figure unbound. The problem is now reduced

to normal anaphoric resolution. Restrictions on antecedents and how to handle intersentential reference are discussed in Section 3.6.3.

It is possible that these cases could be handled completely through standard anaphora resolution. However, I prefer to have firmer control so that the first choice for the figure is the set of all previous entities in the list. It is questionable whether existing mechanisms for anaphora resolution would do this rather than just choosing the most recent entity. This sort of reference is present in other cases, such as in *John, Mary, and their friends*, so anaphora resolution algorithms must eventually support it. When this happens, a standard resolution algorithm can replace the procedure discussed in this section.

3.6.3 Intersentential Reference

Finding the figure outside of the sentence is handled through standard discourse anaphora techniques. (Our implementation does this with an analysis based on c-command and salience described in Section 5.7.2.) A presupposition concerning the figure, then, becomes a further restriction placed on an antecedent. If no antecedent is found that is known to meet this restriction, an antecedent consistent with it is chosen and the presupposition accommodated. Thus, in (31a), repeated in (87), when evaluating the second sentence, we look for an antecedent that is a country.

(87) *What is the drinking age in Afghanistan?*
What is the drinking age in ***other*** *countries?*

If Afghanistan is known to be a country, we choose it. Otherwise, if it is consistent with being a country, we choose it and accommodate that fact. Otherwise we try the common ground before failing.

That is, presuppositions can also be licensed by elements of the common ground that come from the speaker's and hearer's shared physical or cultural situation. While in general, completely specifying user's and system's common ground is impossible, we can do quite well in constrained domains like NLIR queries. In particular, the user, the location of the user, and the user's web browser can all be considered part of the common ground. Thus, in the absence of alternative evidence, *other countries* probably excludes the user's current location, while *other browsers* almost certainly excludes the one being used.

3.6.4 Preliminary Evaluation

I have hand-tested the heuristics given in Sections 3.6.1 and 3.6.2 for finding the figure of **other** on three corpora: a subset of the British National Corpus (BNC), a corpus of home maintenance instructions (RD) Digest (1991), and one month of queries from the Monk. The BNC contains a great deal of literature and literary criticism, while the RD is a more constrained "how to" text. The dialogues in the Monk's user interactions are the shortest and most constrained of all.

	Without Default		With Default	
	precision	recall	precision	recall
BNC	100%	10%	47%	47%
RD	100%	43.4%	57.8%	57.8%
Monk	100%	69.6%	78.3%	78.3%

Table 3.1: Accuracy of Figure-finding Heuristics

Table 3.1 gives scores for precision and recall where precision is the number of *figures* correctly identified out of those attempted and recall is the number correct out of all instances of the word **other**, excluding idioms like *on the other hand*. I show two sets of scores. The first uses only the techniques described in Section 3.6. The second includes a default which chooses the most recent sentential subject that is not the *other* phrase itself. This is a simple heuristic for selecting the most salient entity, which is done more intelligently in my implementation (Chapter 5). With the default, precision and recall are the same because the procedure identified a figure for all instances of **other**. Note that the gold standard is not 100% because there are cases in the data sets where not enough of the discourse was available to identify the figure.

These scores simply suggest that the current approach is practical since the heuristics give significantly greater accuracy with more constrained texts such as are found in the queries of NLIR systems like the Monk.

Comparing results with and without the default, in the latter case 30.4% of the time no element is chosen as the figure, causing NPs of the form "other y" to be translated to just simply `y` in the Boolean query. This will cause *false positives*, pages incorrectly containing the figure, to be included in the results of the query.

With the default, the set of instances where false positives are returned is reduced to 21.7%. In these cases, the NP "other y" is translated into `y AND NOT z` (not really—see Section 6.1) where z is an incorrectly identified figure. Here there are two possibilities: z could be of interest to the user, causing important

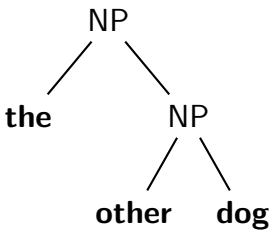
documents to not be returned (*false negatives*). Alternatively, z could be irrelevant, in which case no false negatives occur, and the query is essentially just `y`, causing false positives. Further investigation into the frequency of these situations and the effect they have on users' ability to effectively complete their tasks will help pinpoint the appropriate tradeoffs to make in a practical system, but this is beyond the scope of this dissertation.

3.7 Determiners

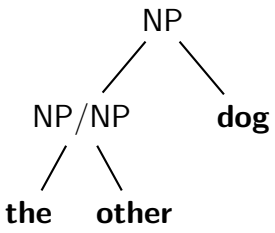
So far in this chapter, I have only shown derivations where the ground is a bare plural. This was meant to simplify the examples and explanation. However, this does hide some important issues that arise with determiners. Consider the sentence in (88).

(88) John likes Fido, but Mary likes *the other dog*.

(89) a.



b.



Syntactically, there is a choice in the structure of *the other dog*. First, as in (89a), *other dog* could combine first to form the noun phrase argument of the determiner. This would be consistent with my current analysis. The other option, (89b), is for the determiner and the alternative marker to combine to form a new determiner.

There are two reasons to favor the second analysis over the first. The first is simply that, in English, the word **another** is already an example of a determiner having combined with an alternative marker. A further reason is that, with determiners, some examples where unification is used to identify the figure no longer work. For example, in the phrase *unlike the other dog, Fido...*, the current analysis would produce the presuppositions in (90). In the second *alts* relation, produced by **unlike**, the second argument has combined with the determiner, while in the first *alts* relation, this is not the case. The two relations therefore cannot unify and the figure is not correctly identified.

(90)

unlike	**the**	**other**	**dog**,	**Fido**
$((\mathsf{S}/\mathsf{VP})/\mathsf{NP})/\mathsf{NP}$: $\lambda x\lambda y\lambda p.p(y)$	NP/NP $\lambda x.x_{the}$	NP/NP : $\lambda g\lambda x.g(x) \wedge \neg f(x)$	$\mathsf{NP} : \mathtt{dog}$	$\mathsf{NP} : \mathtt{fido}$

$$\frac{\dfrac{\dfrac{\dfrac{}{((\mathsf{S}/\mathsf{VP})/\mathsf{NP})/\mathsf{NP} : \lambda x\lambda y\lambda p.p(y)} \quad \dfrac{\dfrac{}{\mathsf{NP}/\mathsf{NP} : \lambda x.x_{the}} \quad \dfrac{\dfrac{}{\mathsf{NP}/\mathsf{NP} : \lambda g\lambda x.g(x) \wedge \neg f(x)} \quad \dfrac{}{\mathsf{NP} : \mathtt{dog}}}{\mathsf{NP} : \lambda x.\mathtt{dog}(x) \wedge \neg f(x)}>}{\mathsf{NP} : [\lambda x.\mathtt{dog}(x) \wedge \neg f(x)]_{the}}>}{(\mathsf{S}/\mathsf{VP})/\mathsf{NP} : \lambda y\lambda p.p(y)}> \quad \dfrac{}{\mathsf{NP} : \mathtt{fido}}}{\mathsf{S}/\mathsf{VP} : \lambda p.p(\mathtt{fido})}>$$

presupposition set: $\begin{cases} \forall x.f(x) \rightarrow \mathtt{dog}(x) \\ alts(f, \lambda x.\mathtt{dog}(x) \wedge \neg f(x)) \\ \neg p([\lambda x.\mathtt{dog}(x) \wedge \neg f(x)]_{the}) \\ alts(\mathtt{fido}, [\lambda x.\mathtt{dog}(x) \wedge \neg f(x)]_{the}) \end{cases}$

The solution is to add lexical entries for alternative markers that explicitly subcategorize for determiners as well as the ground. The analysis is then able to apply the determiner to the ground in the semantics to yield the appropriate presupposition. An example of such an entry is shown in (91). This allows the more compatible analysis of *the other dog* in (92). The *alts* presupposition is now able to unify with $alts(\mathtt{fido}, [\lambda x.\mathtt{dog}(x) \wedge \neg f(x)]_{the})$ and determine that the figure is Fido.

(91) **other** $\vdash \begin{cases} syn: & \mathsf{NP}_{\mathsf{comp}+,\mathsf{eq}-} \backslash (\mathsf{NP}/\mathsf{NP})/\mathsf{NP}_{\mathsf{bare}+} \\ sem: & \lambda g\lambda q \begin{cases} assert: & q(\lambda x.g(x) \wedge \neg f(x)) \\ presup: & \forall x.f(x) \rightarrow g(x) \\ & alts(f, q(\lambda x.g(x) \wedge \neg f(x))) \end{cases} \end{cases}$

(92)

the	**other**	**dog**
$\mathsf{NP}/\mathsf{NP} : \lambda x.x_{the}$	$\mathsf{NP}\backslash(\mathsf{NP}/\mathsf{NP})/\mathsf{NP}$: $\lambda g\lambda q.q(\lambda x.g(x) \wedge \neg f(x))$	$\mathsf{NP} : \mathtt{dog}$

$$\frac{\dfrac{\dfrac{}{\mathsf{NP}/\mathsf{NP} : \lambda x.x_{the}} \quad \dfrac{}{\mathsf{NP}\backslash(\mathsf{NP}/\mathsf{NP})/\mathsf{NP} : \lambda g\lambda q.q(\lambda x.g(x) \wedge \neg f(x))}}{\mathsf{T}/(\mathsf{T}\backslash(\mathsf{NP}/\mathsf{NP})) : \lambda p.p(\lambda x.x_{the})}>\mathbf{T} \quad \dfrac{}{\mathsf{NP} : \mathtt{dog}}}{\mathsf{NP}/\mathsf{NP} : \lambda g.[\lambda x.g(x) \wedge \neg f(x)]_{the}}>\mathbf{B}}{\mathsf{NP} : [\lambda x.\mathtt{dog}(x) \wedge \neg f(x)]_{the}}>$$

presupposition set: $\begin{cases} \forall x.f(x) \rightarrow \mathtt{dog}(x) \\ alts(f, [\lambda x.\mathtt{dog}(x) \wedge \neg f(x)]_{the}) \end{cases}$

Chapter 4

Free Alternative Phrases

> *Yeah, True Love is the greatest thing in the world, except for a nice MLT—mutton, lettuce and tomato sandwich, when the mutton is nice and lean, and the tomato is ripe. They're so perky.*
>
> (Miracle Max, *The Princess Bride*)

As others have pointed out, most recently von Fintel (1993) and Hoeksema (1995), alternative markers can appear in freer contexts than those described in Chapter 3. These papers are particularly interested in the *exceptive marker* **except for** and perform in-depth analyses of its assertional semantics with respect to quantifiers, as in (93).

(93) Except for John, every student attended the meeting.

This dissertation has been primarily concerned with how presupposition and anaphora in alternative phrases imply things about how the figure relates to the world. In the example above, this would be that John is a student and that he did not attend the meeting. Though this is only discussed superficially by von Fintel and Hoeksema, it will remain my main concern. Furthermore, von Fintel and Hoeksema miss an interesting observation. Consider the sentences in (94)[1].

(94) a. Other than John, every student attended the meeting.

b. Other than John, three students attended the meeting.

c. Other than John, few students attended the meeting.

d. **Q:** Who attended the meeting? Was the class president there?

A: Other than John, who is the principal, every STUDENT attended the meeting. So, yes, the class president was there.

[1]Dialects vary in the acceptability of these sentences. For many, **apart from** or **besides** is a more appropriate alternative marker than **other than**.

Classifying these sentences based on whether or not John attended the meeting and whether or not he is a student, we see that there is quite a variety of behavior.

(95)

	is student	is not student
attended meeting	(94b), (94c)	(94b), (94d)
did not attend meeting	(94a)	

Using the determiner **every**, the sentence communicates that John did not attend the meeting and that he is a student. With **few**, the sentence still communicates that John is a student but also that he attended the meeting. With **three**, John attended the meeting but may or may not be a student depending on slight variations in intonation. And, finally, we see that in an example with **every** engineered such that John cannot be a student, the sentence reads that John did attend the meeting.

This chapter will explain this phenomenon. In the process, I will discuss several examples of free alternative phrases with different semantic properties, and I will sketch a CCG account of the syntax for some of the simpler constructions.

4.1 Syntax of Simple Free Alternative Phrases

4.1.1 Introduction

The literature on the semantics of free exceptives, for the most part, glosses over or entirely ignores syntactic issues. I, too, choose to skirt this issue as much as possible, but since I require complete lexical entries, I will provide a sketch of a CCG analysis. In general, the goal of this analysis is not to account for every last bit of data. Its goal is to give alternative phrases access to both the figure and ground NPs. This was crucial for the semantic treatment of connected alternative phrases and will remain so here.

I begin with a cursory look at the syntactic distribution of free alternative phrases. Figure 4.1 shows some example sentences organized by the position of the alternative phrase. Positions are marked by (s)ubject, (v)erb, (o)bject, (wh)-word, and X, which is the alternative phrase.

This table is not complete in that I only consider simple intransitive and transitive verbs and alternative phrases with NP arguments. As Hoeksema notes, alternative phrases can occur in all major sentential adverbial positions—sentence initial, sentence final, and before the verb. Von Fintel points out that sentence-final positions like in (96) seem to be indicating an afterthought or repair and does not treat them.

	Subject
X s v o	Other than the president, every officer attended the meeting.
s X v o	Every officer, other than the president, attended the meeting.
s v X o	* Every officer attended, other than the president, the meeting.
s v o X	Every officer attended the meeting, other than the president.
X wh s v	* Other than the president, what meeting did every officer attend?
X wh v o	Other than the president, who attended the meeting?
wh X s v	* What meeting, other than the president, did every officer attend?
wh X v o	Who, other than the president, attended the meeting?
wh s v X	What meeting did every officer attend, excluding the president?
wh v o X	Who attended the meeting, other than the president?

	Object
X s v o	Other than the president, the meeting included few officers.
s X v o	* The meeting, other than the president, included every officer.
s v X o	The meeting included, other than the president, every officer
s v o X	The meeting included every officer, other than the president.
X wh s v	Other than the president, who did the meeting include?
X wh v o	* Other than the president, what meeting included every officer?
wh X s v	Who, other than the president, did the meeting include?
wh X v o	* What meeting, other than the president, included every officer?
wh s v X	Who did the meeting include, other than the president?
wh v o X	What meeting included every officer, excluding the president?

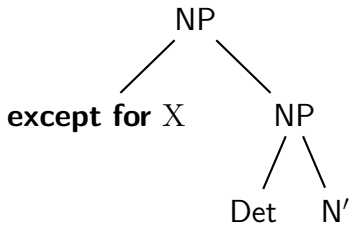
Figure 4.1: Syntactic Distribution of Free Alternative Phrases

(96) The meeting included every officer, other than the president (of course).

Of course, not all free alternative markers can appear in all of these positions. The analysis I provide below will provide an easy way of restricting the syntactic distribution to certain positions.

Von Fintel assumes a syntactic analysis that causes the exceptive phrase to combine with the full NP:

(97)



As von Fintel points out, this leads to problems with semantic analyses that need access to the semantics of the N′, since in this tree, the exceptive phrase connects at the NP level. He appeals to the analysis of Bach and Cooper (1978) which introduces a free variable at the N′ level which can be used by relative clauses and, as it turns out, exceptive phrases. *Every student*, for example, can

have the semantics in (98a) and *Except for John, every student* therefore has the semantics in (98b).

(98) a. $\lambda r.[\![every]\!]([\![student]\!] \cap r)$

b. $[\![every]\!]([\![student]\!] \cap \overline{\{j\}})$

I will also have the alternative phrase syntactically subcategorize for the NP. I reserve judgment on the Cooper variable solution, but I leave it on the table as a possibility. For now, I continue the policy introduced in Section 2.3 of having quantifiers merely contribute semantic features that will be resolved in a later interpretation phase. Thus, the phrases *do* have access to the N′ semantics.

Before describing this analysis, I will briefly discuss the possibility of a sentence-adjunct analysis.

4.1.2 A Sentence-adjunct Analysis

Parentheticals are sometimes treated as sentential adjuncts (McCawley, 1988). Syntactically, this would seem at first to work fairly elegantly. If we give alternative phrases the category S|S, the following derivations succeed.

(99) a. $\dfrac{\dfrac{\textbf{Unlike John},}{\mathsf{S|S}} \quad \dfrac{\textbf{Mary likes spam}.}{\mathsf{S}}}{\mathsf{S}}\,{>}$

b. $\dfrac{\dfrac{\textbf{Mary likes spam}}{\mathsf{S}} \quad \dfrac{\textbf{, unlike John}.}{\mathsf{S|S}}}{\mathsf{S}}\,{<}$

c. $\dfrac{\dfrac{\dfrac{\textbf{Mary}}{\mathsf{NP}}\,{>}\mathbf{T}}{\mathsf{S/(S\backslash NP)}} \quad \dfrac{\textbf{, unlike John},}{\mathsf{S|S}} \quad \dfrac{\textbf{likes spam}.}{\mathsf{S\backslash NP}}}{\dfrac{\mathsf{S/(S\backslash NP)}}{\mathsf{S}}\,{<}\mathbf{B}_{\times}}\,{>}$

However, it is also the case that undesirable sentences, such as (100a), are allowed with the sentential analysis. Furthermore, the coordinate sentence in (100b) shows that this analysis does not allow some grammatical sentences. This is because the coordination forces *unlike John* to combine with *hates beer* and this is not possible without forward crossing composition, which English lacks.

(100) a. $*$**What food** , **other than Mary**, **repels everyone**?

$$\dfrac{\dfrac{\dfrac{\text{S/(S\NP)}}{\text{S/(S\NP)}}\ \text{S|S}}{\text{S}}<\mathbf{B}_{\times}\qquad \dfrac{\text{S\NP}}{\text{S}}>}{}$$

b. **Mary** **loves spam** **but** , **unlike John**, **hates beans**.

$$\dfrac{\text{NP}\quad \dfrac{\text{S\NP}}{\text{S}}>\quad \text{CONJ}\quad \dfrac{\text{S|S}\qquad \text{S\NP}}{\text{S}}*}{}$$

In addition to these syntactic deficiencies, the sentential adjunct analysis does not make particularly useful semantic distinctions. Because this analysis only has access to the sentence as a whole, the ground or complement must be identified anaphorically. This seemed fairly clear-cut in the cases in Chapter 3, but now matters are complicated by the fact that, as we will see in the next few sections, the possible locations for the ground/complement differ for various alternative phrases. If it is possible to capture these distinctions in the syntactic account, it will not be necessary to postulate such specific lexical restrictions on anaphoric reference.

4.1.3 A Different Adjunct Analysis

In my syntactic analysis, alternative phrases are (forward and backward) adjuncts of verb phrases and type-raised NPs. The purpose of this is to have both the figure and ground readily available for use in the semantics. In its most general case, these categories can be described simply as (101) which uses the $-convention described in (Steedman, 2000b, p.42). In essence, S$ represents all categories whose final result is S. In practice, the examples in this thesis all use categories that are instantiations of the more specific forms in (102). At first glance, the category S|(S|NP) might generate concern in that it seems to support non-order-preserving type-raising. However, note that S/(S/NP) is only a valid category for topicalized NPs and wh-words, which are valid targets for alternative phrases. The category S\(S\NP) is also not a concern since it does not appear at all in the English lexicon.

(101) $\text{S\$}|\text{S\$}$

(102) a. $(\text{S}|_{\text{i}}\text{NP})|(\text{S}|_{\text{i}}\text{NP})$

b. $(\text{S}|_{\text{i}}(\text{S}|_{\text{j}}\text{NP}))|(\text{S}|_{\text{i}}(\text{S}|_{\text{j}}\text{NP}))$

Restrictions on the directionality of these categories will yield the restrictions on the syntactic distribution of particular alternative markers. To simplify mat-

ters, I introduce the notational abbreviations in (103). The feature $\mathsf{ind|int}$ means that the argument must be tensed. This will become important shortly.

(103)
$$\begin{array}{lcl} \overrightarrow{\mathsf{NP}} & \Rightarrow & \mathsf{S}/(\mathsf{S}_{\mathsf{ind|int}}\backslash\mathsf{NP}) \\ \overleftarrow{\mathsf{NP}} & \Rightarrow & \mathsf{S}\backslash(\mathsf{S}_{\mathsf{ind|int}}/\mathsf{NP}) \\ \overrightarrow{\overrightarrow{\mathsf{NP}}} & \Rightarrow & \mathsf{S}/(\mathsf{S}_{\mathsf{ind|int}}/\mathsf{NP}) \\ \overleftrightarrow{\mathsf{NP}} & \Rightarrow & \mathsf{S}|(\mathsf{S}_{\mathsf{ind|int}}|\mathsf{NP}) \end{array}$$

Figure 4.2 contains the syntactic categories required for the free alternative phrase in each of the constructions in Figure 4.1.

	subject	*object*
X s v o	$\overrightarrow{\mathsf{NP}}$ / $\overrightarrow{\mathsf{NP}}$	(S/NP)/(S/NP)
s X v o	$\overrightarrow{\mathsf{NP}}$ \ $\overrightarrow{\mathsf{NP}}$ *or* (S\NP)/(S\NP)	*
s v X o	(S\NP)\(S\NP)	$\overleftarrow{\mathsf{NP}}$ / $\overleftarrow{\mathsf{NP}}$ *or* (S/NP)\(S/NP)
s v o X	(S\NP)\(S\NP)	$\overleftarrow{\mathsf{NP}}$ \ $\overleftarrow{\mathsf{NP}}$
X wh s v	*	$\overrightarrow{\overrightarrow{\mathsf{NP}}}$ / $\overrightarrow{\overrightarrow{\mathsf{NP}}}$
X wh v o	$\overrightarrow{\mathsf{NP}}$ / $\overrightarrow{\mathsf{NP}}$	*
wh X s v	*	$\overrightarrow{\overrightarrow{\mathsf{NP}}}$ \ $\overrightarrow{\overrightarrow{\mathsf{NP}}}$
wh X v o	$\overrightarrow{\mathsf{NP}}$ \ $\overrightarrow{\mathsf{NP}}$	*
wh s v X	(S\NP)\(S\NP)	(S/NP)\(S/NP)
wh v o X	(S\NP)\(S\NP)	$\overleftarrow{\mathsf{NP}}$ \ $\overleftarrow{\mathsf{NP}}$

Figure 4.2: Categories for Free Alternative Phrases

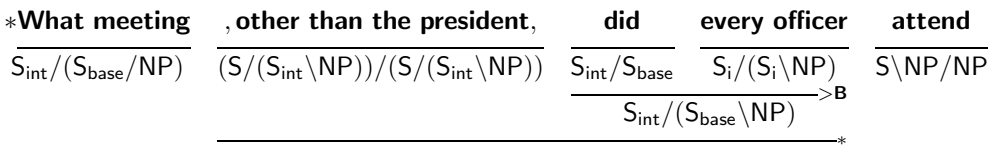
Note that these examples do not cover constructions like *Does/can/must every officer attend the meeting?* As described in Section 2.7, the category for **does**/**can**/**must** is $\mathsf{S}_{\mathsf{int}}/\mathsf{S}_{\mathsf{base}}$. This category may compose into the type-raised subject, reducing the analysis to those for indicative sentences.

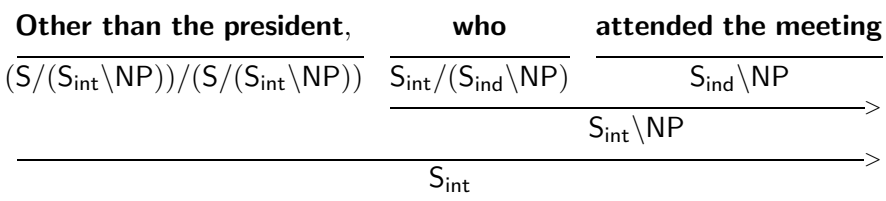
In addition to the sentences in Figure 4.1, these categories allow the coordinate construction in (100b) not allowed by the sentential-adjunct analysis:

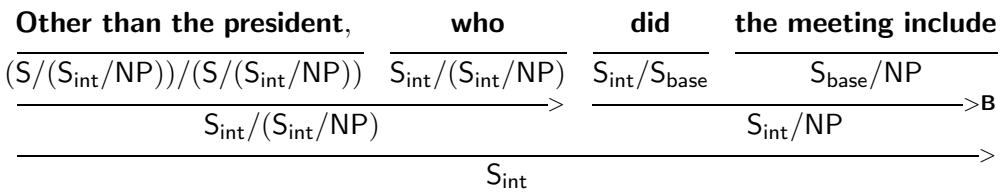
(104)
$$\dfrac{\dfrac{\dfrac{\dfrac{\textbf{Mary}}{\mathsf{NP}} \quad \dfrac{\textbf{loves spam}}{\mathsf{S}\backslash\mathsf{NP}} \quad \dfrac{\textbf{but}}{\mathsf{CONJ}} \quad \dfrac{\textbf{, unlike John,}}{(\mathsf{S}\backslash\mathsf{NP})/(\mathsf{S}\backslash\mathsf{NP})} \quad \dfrac{\textbf{hates beans.}}{\mathsf{S}\backslash\mathsf{NP}}}{\mathsf{S}\backslash\mathsf{NP}}\,{>}}{\mathsf{S}\backslash\mathsf{NP}}\,{<\Phi>}}{\mathsf{S}}\,{<}$$

A few questionable or ungrammatical constructions must be restricted. First, the pre-subject position for alternative phrases is not possible in questions: for

example in **What meeting, other than the president, did every officer attend.* This is restricted by the requirement in (103) that the arguments of the type-raised NPs must be tensed. This is demonstrated in (105). This example explicitly shows in *every officer* that the S categories of a type-raised NP share the same features. Thus, the untensed feature base, from **did**, forms a category that cannot combine with the alternative phrase. However, the related, but grammatical, sentences in (106) and (107) are still allowed.

(105) 

(106) 

(107) 

Also, in no case does it seem grammatical to have a free alternative phrase directly before the verb that refers to the object, (108). This is because the adjunct has no way to combine with the object NP since it is being blocked by the subject NP. Furthermore, the alternative phrase is positioned such that the verb cannot combine with the subject to free things up. This is the same for the interrogative case.

(108) a. * The meeting [, other than the president,]$_{(S/NP)/(S/NP)}$ [included]$_{S\backslash NP/NP}$ every officer.

b. * What meeting [, other than the president,]$_{(S/NP)/(S/NP)}$ [included]$_{S\backslash NP/NP}$ every officer?

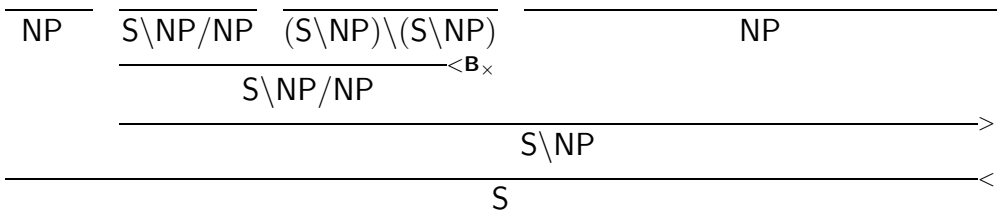
Other restrictions are also the result of the standard CCG rule set. There is simply no way for the alternative phrase to reach past the wh-word:

(109) a. * Other than the president, [what meeting]$_{S/(S/NP)}$ did every officer attend?

b. * Other than the president, [what meeting]$_{S/(S\backslash NP)}$ included every officer?

Finally, I look at the subject-attaching, post-verb position in (110). As with adverbial phrases, this is generally allowed with a heavy object and backwards crossing composition. These sentences are odd, but because heavy objects seem to improve matters, I permit them in my syntactic analysis.

(110) #**Mary** **attended**, **unlike John**, **the extraordinarily long meeting**.



I now briefly discuss remaining syntactic considerations—commas. I will then continue with a discussion of three classes of free alternative phrases. Each class demonstrates a particular subset of the syntactic distribution shown above. The syntactic distributions that I present in these sections are the result of a very small data set of grammaticality judgments and should not be taken as fact. However, part of the elegance of the syntactic analysis is that it is simple to modify the distribution through restrictions on the non-directional slashes.

4.2 What About the Commas?

There remains the interesting question of when to allow the categories presented in this section to combine with the rest of a sentence. These constructions are, in some sense, parentheticals and must be specified as such to be allowed. This can be done through intonation in speech or punctuation (commas, parentheses, hyphens) in written text. How do we restrict alternative phrases to these contexts?

On their own, alternative phrases must be restricted from combining with anything else—that is, they must be "locked". The "key" to the lock is the comma. This can be implemented with features. In my particular grammar, this can be done using mode features on S. Since, in my grammar, the mode feature is never left unspecified on the verb, we can use a conflicting feature on alternative phrases to cause a conflict. This is the lock.

(111) $S\$|S_{ap}\$$

The key to the lock is the comma. Thus, the category for commas marking pre- and post-sentential alternative phrases would be as in (112a), and sentence-medial alternative phrases as in (112b). These categories simply take a locked

alternative phrase and return an unlocked alternative phrase which is now free to combine with the rest of the sentence. But without the comma, a free alternative phrase would yield an ungrammatical sentence. These categories are very similar to those used for intonational boundary tones in Steedman (2000a), which is not surprising since intonation marks free alternative phrases in speech.

(112) a. $(S\$|S\$)|(S\$|S_{ap}\$)$
b. $(S\$|S\$)/,/(S\$|S_{ap}\$)$

This solution is somewhat *ad hoc* in that it requires that the mode feature be specified for all verbs. This might not always be desirable. Furthermore, using this feature as a lock does not correspond with the original purpose of the feature. The alternative, though, is to create a new feature, but then that must be instantiated for every verb just as the mode feature is. For now, I leave the discussion here, but in Section 7.1, I discuss an avenue for further research.

4.3 Other (than)

4.3.1 Syntax

Figure 4.3 contains the syntactic distribution for **other** (**than**) in a free context. Given the discussion in Section 4.1, all that is required is that we restrict the categories in (102) so that they only accommodate this data. The resulting categories are in (113). Although more will be said about the syntax of alternative phrases later in the chapter, I will now continue with a discussion of semantics.

(113) $(S|_{i}NP)|(S|_{i}NP)/NP$
$(S/(S|_{i}NP))|(S/(S|_{i}NP))/NP$

4.3.2 Semantic Observations

The data in (114) and (115) show that sentences behave differently with respect to whether the predicate of the main clause applies to the figure and whether the figure has the property of the ground.

(114) a. Other than Sam, almost every/every/most/all officer(s) attended the meeting.
b. Other than Sam, two/many/some officers attended the meeting.
c. Other than Sam, few/no officers attended the meeting.
Other than Sam, who attended the meeting?

	Subject
X s v o	Other than the president, every officer attended the meeting.
s X v o	Every officer, other than the president, attended the meeting.
s v X o	*
s v o X	Every officer attended the meeting, other than the president.
X wh s v	*
X wh v o	Other than the president, who attended the meeting?
wh X s v	*
wh X v o	Who, other than the president, attended the meeting?
wh s v X	What meeting did every officer attend, other than the president?
wh v o X	Who attended the meeting, other than the president?

	Object
X s v o	Other than the president, the meeting included few officers.
s X v o	*
s v X o	The meeting included, other than the president, every officer
s v o X	The meeting included every officer, other than the president.
X wh s v	Other than the president, who did the meeting include?
X wh v o	*
wh X s v	Who, other than the president, did the meeting include?
wh X v o	*
wh s v X	Who did the meeting include, other than the president?
wh v o X	*

Figure 4.3: Syntactic Distribution of **Other** (**than**)

(115) **Q:** Do you know who is attending the premiere? I'm trying to get autographs from my favorite actresses Mary and Sally.

A: Other than Mary, every actress is attending the premiere.
So Sally is there but Mary is not.

A': Other than John, the director, every ACTRESS is attending the premiere.
So both of them should be there.

The variation seems primarily due to the ground's quantifier. For universals or near-universals, as in (114a), the sentence communicates that Sam is an officer and did not attend the meeting. Other quantifiers, as in (114b), communicate that Sam did attend the meeting but remain neutral about whether Sam is an officer. Finally, some quantifiers, (114c), communicate that Sam attended the meeting and is an officer. (Note that *Other than Sam, officers attend meetings* is ambiguous because **officers** can be interpreted as either the generic or as *some officers*.)

To further complicate matters, (115) shows that the addition of world knowledge can change the meaning. The first answer is equivalent to (114a) except that the information that Mary is an actress is already given. As the answer indicates, Mary, the figure, is not attending the meeting. However, when *Mary* is replaced by *the director* and contrastive stress is placed on *actress*, the reading is that the director is not an actress and is attending the meeting.

As I have mentioned, I am primarily interested in what these sentences tell us about the figure, (116), so I will concentrate on that in the coming discussion. These facts are not a focus of von Fintel and Hoeksema's work. Von Fintel gives a semantics that assumes the negative reading of (116a), and Hoeksema only briefly mentions Gricean Implicature as the means by which (116b) is communicated.

(116) a. $p(f)$ or $\neg p(f)$ (Sam did/didn't attend)

b. $g(f)$ (Sam is an officer)

Thus, given that f is the figure (`sam`), g is the ground (`officer`), and p is the property (`attend`), I propose that the assertional semantics for **other than** is $p(\lambda x.g(x) \wedge \neg f(x))$, the same as the analysis in Section 3.4.3. The assertion makes no judgement to the questions of (116) since the figure is removed from the set under consideration. Information about the figure will be communicated with presupposition. This is all I will say about the assertion, which is compatible with the work of von Fintel and Hoeksema.

4.3.3 Denial of Expectation

The key observation is that Sam is an exception to what the assertion of the sentence might lead one to believe—a denial of expectation. *Most officers attended the meeting*, for example, might lead one to conclude that Sam attended the meeting. The addition of *other than Sam* denies this.

The concept of denial of expectation is used by Lagerwerf (1998) to analyze the causal connective **although** and by Webber *et al.* (1999a) for **however**. For example, in Lagerwerf's example (117), the first clause might communicate an expectation that Greta Garbo married. The second clause denies this.

(117) Although Greta Garbo was considered to be the yardstick of beauty, she never married.

(118) **Defeasible Modus Ponens**
$\forall x(\phi > \psi), \phi(\delta) \mid\!\approx \psi(\delta)$,
but not $\forall x(\phi > \psi), \phi(\delta), \neg\psi(\delta) \mid\!\approx \psi(\delta)$

The mechanism Lagerwerf uses to explain this is *Defeasible Modus Ponens* as described by Asher and Morreau (1995). The $>$ notation is the *defeasibly implies* condition that is paraphrased as, if $\phi > \psi$ and ϕ holds then ψ normally holds. This is the expectation. If ϕ is instantiated as $\phi(\delta)$ then it "defeasibly validates" $\psi(\delta)$. However, this validation is not possible if there is information contradictory to the conclusion. Lagerwerf's analysis for **although** includes the presupposition in (119). Thus, the case of Greta Garbo can be represented as in (120)—an instantiation of defeasible modus ponens. In this way Lagerwerf shows that defeasible modus ponens is exactly the mechanism to describe denial of expectation. Basically, **although** presupposes an expectation which it immediately denies.

(119) *Although p, q* presupposes $p' > \neg q'$
where p' and q' are propositions that are generalizations of p and q.

(120) $\forall x(beau(x) > marry(x)), beau(g), \neg marry(g) \not\approx marry(g)$

Other than, on the other hand, does not presuppose an expectation. Rather, it tests whether one exists. If so, it is denied. For instance, not considering Sam, if every officer attended the meeting, there is an expectation that Sam attended the meeting as well (as long as Sam is an officer). But if two officers attended the meeting, there is no such expectation. These are the sort of expectations that **other than** tests for, as shown formally in the left-hand side of (121) by the expression $p(\lambda x.g(x) \wedge \neg f(x)) > p(f)$. I will refer to this expectation as $\mathcal{E}_{p,g,f}$ in the following discussion.

(121) $\mathcal{E}_{p,g,f} \leftrightarrow \neg p(f)$
where p is a property, g is the ground, and f is the figure.

(122) a. $\mathcal{E}_{p,g,f} \rightarrow \neg p(f)$
b. $\neg\mathcal{E}_{p,g,f} \rightarrow p(f)$

The formulas in (122) follow from the formula in (121). (122a) means that if the expectation exists, then we deny the expectation. From (122b), if the expectation does not exist, then we affirm the truth of the expectation. In a nutshell, **other than** denies the expectation of its assertion in a way very similar to **although**. It is useful to go through the examples step by step:

(123) *Other than Sam, every officer attended the meeting.*
a. assertion: $\texttt{attended}(\lambda x.\texttt{officer}_{every}(x) \wedge \neg\texttt{sam}(x))$
translation: Not considering Sam, every officer attended the meeting.

b. expectation: $\texttt{attended}(\texttt{sam})$
translation: The assertion normally entails that Sam also attended the meeting.

c. $\mathcal{E}_{p,g,f} \rightarrow \neg p(f)$

d. $\mathcal{E}_{p,g,f}$ is true. Therefore, by our formula, $\neg p(f)$, is also true. This contradicts the expectation by saying that Sam did not, in fact, attend the meeting.

(124) *Other than Sam, two officers attended the meeting.*

a. assertion: $\texttt{attended}(\lambda x.\texttt{officer}_{two}(x) \wedge \neg\texttt{sam}(x))$
translation: Not considering Sam, two officers attended the meeting.

b. expectation:
translation: There is no expectation.

c. $\neg\mathcal{E}_{p,g,f} \rightarrow p(f)$.

d. $\mathcal{E}_{p,g,f}$ is false. Therefore, by our formula, $p(f)$, is true. This says that although we had no reason to believe that Sam attended the meeting from the assertion, he did, in fact, attend.

But where do these expectations come from? I will simply express these expectations explicitly. For example, existing in the knowledge base is an expectation that if an entity is a member of a set and a property is true for every member of that set (not considering the entity), then the property is normally true of that entity. This is expressed in (125), although a more general rule might say that if the property is true for more than a certain percentage of the set, then the property is true of the entity. Expectations such as this one, along with defeasible modus ponens, can validate $\mathcal{E}_{p,g,f}$ in the formula in (121). Other determiners, such as **some**, would not validate $\mathcal{E}_{p,g,f}$.

(125) $\forall p, f, g[(\forall x.f(x) \rightarrow g(x)) \wedge p(\lambda x.g_{every}(x) \wedge \neg f(x)) > p(f)]$

This interacts very well with our data. Consider one of the examples in (114a): *Other than Sam, every officer attended the meeting.* The presupposition in (121) is instantiated to:

(126) $\texttt{attend}(\lambda x.\texttt{officer}_{every}(x) \wedge \neg\texttt{sam}(x)) > \texttt{attend}(\texttt{sam}) \leftrightarrow \neg\texttt{attend}(\texttt{sam})$

We now need to test to see whether $\texttt{attend}(\lambda x.\texttt{officer}_{every}(x) \wedge \neg\texttt{sam}(x))$ does defeasibly imply $\texttt{attend}(\texttt{sam})$. The expectation in (125) can be instantiated as in (127).

(127) $(\forall x.\texttt{sam}(x) \rightarrow \texttt{officer}(x)) \wedge \texttt{attend}(\lambda x.\texttt{officer}_{every}(x) \wedge \neg\texttt{sam}(x))$
$> \texttt{attend}(\texttt{sam})$

Thus, by defeasible modus ponens, every officer attending the meeting (not considering Sam) does defeasibly imply that Sam attended, *as long as we believe that Sam is an officer*. Without conflicting information, this can be accommodated, explaining why the sentences in (114a) communicate that Sam is an officer.

However, in certain situations, it is not possible to accomodate that Sam is an officer (or, more generally, that the figure has the property of the ground) because the discourse has contradicted this. This is true in (115A') where the contrastive stress (and the knowledge that John is male) contributes to the belief that the director is not an actress. Therefore, there is no expectation from actresses attending the premiere that the *director* is attending the premiere. Through the same reasoning as in (124), we conclude that the director is attending the premiere.

(114b) is a similar case because I have not specified any expectation saying that if a property is true of two/many/some x, then it is true of an arbitrary entity of type x[2]. Two officers attending does not imply that any particular officer attended. Thus the defeasible implication in (127) is not possible. According to the original presupposition in (121) and the reasoning in (124) we conclude that Sam did attend the meeting.

At this point, I have explained two of the three cases. First, I explained why, for universals such as **every**, the sentence communicates that Sam is an officer and did not attend the meeting. I also explained why, in the case of quantifiers like **two** and **some**, the sentence communicates that Sam attended the meeting, but nothing about Sam being an officer. It remains to show why quantifiers like **few** and **no** behave like **two** and **some** but also specify that, like with **every**, Sam is an officer.

4.3.4 Negative Environments

Negative Quantifiers

Consider (128), where in some cases it is required that the figure be an actress. Denying this, as in (128A'), leads to an unacceptable sentence. For other examples, (128A"), this is not the case.

[2]There is nothing preventing such expectations being specified. **Many**, in particular, is a borderline case.

(128) **Q:** Do you know who is attending the premiere? I'm trying to get autographs from my favorite actresses Mary and Sally.

A: Other than Mary, few/no/less than ten actresses are attending the premiere.
So Mary is there but Sally is probably not.

A': #Other than John, the director, few/no/less than ten ACTRESSES are attending the premiere.

A": Other than John, the director, many/some/three ACTRESSES are attending the premiere.

Examples that require the figure to be an actress share the characteristic that they are environments which allow negative polarity items, i.e. negative polarity triggers (see Ladusaw (1979) for an in-depth description of negative polarity items and their triggers). Some classic negative polarity items include **bother**, **any**, and **anymore**, as in (129).

(129) a. I didn't bother notifying him.
*I bothered notifying him.

b. I didn't write any email.
*I wrote any email.

c. He never writes anymore.
*He writes anymore.

Example (130) shows that these cases also work for the environments shown in our original examples. However, this is more awkward for other quantifiers, (131). This dichotomy is noted in (Ladusaw, 1979, ch. 6).

(130) a. Few/no/less than ten actresses bothered attending the premiere.

b. Few/no/less than ten actresses have any hope of attending the premiere.

c. Few/no/less than ten actresses attend premieres anymore.

(131) a. * Many/some/three actresses bothered attending the premiere.

b. * Many/some/three actresses have any hope of attending the premiere.

c. * Many/some/three actresses attend premieres anymore.

The interesting thing about negative quantifiers is that they are equivalent to positive contexts with the predicate negated (discussed in depth in Horn 1989). This can be viewed as invoking an alternative set of negated propositions. For example, *few actresses attended* invokes the alternatives summarized by (132)—e.g. *most actresses did not attend*.

(132) $\lambda x.\texttt{actress}(x) \wedge \neg\texttt{attend}(x)$

Consequently, the same mechanism that we applied earlier to **every** and **most** also applies to **few** and **no**, with the only difference being that the predicate is negated. Thus, *Other than Sam, few officers attended the meeting* is treated as *Other than Sam, most officers did not attend the meeting*. The expectation, then, is that Sam is one of the alternatives that did not attend the meeting. However, Sam is an exception, and, as long as Sam is considered an officer, the sentence communicates that he *did* attend the meeting. Again, this is the same mechanism used for universals and near-universals.

Wh-questions

In addition to **few**, Ladusaw notes that questions, including wh-questions, are negative polarity triggers, although not in as clear a way as the other triggers I have discussed so far. (He points out that such questions are not always meant to elicit information. The question in (133c), for example, is more expressing disbelief that actresses attend premieres.)

(133) a. ? What actresses bothered attending the premiere?

b. What actresses have any hope of attending the premiere?

c. What actresses attend premieres anymore?

This coincides with the interesting observation that questions with free alternative phrases behave more like **few** than **many** in (128). The question in (134a) strongly suggests that Mary is an actress. Thus, choosing a figure that is clearly not an actress, as in (134a), causes the question to be difficult to interpret. But the effect is not as strong as with the contexts discussed above, just as the degree of "negativeness" is not as extreme.

(134) a. Other than Mary, what actresses are attending the premiere?

b. ? Other than John, the director, what ACTRESSES are attending the premiere?

Since I have just shown that wh-questions have a degree of negativeness, I propose that they behave similarly to other negative polarity triggers like **few** and **no**. That is, for the purposes of this analysis, there there is a preference that the predicate (attending the meeting) not be true for any given officer. The negated question is the reverse. In *what officers did not attend the meeting?*, there is a preference that the predicate (not attending the meeting) *is* true for any given officer. This case, as with **few** and **no**, validates the expectation $\mathcal{E}_{p,g,f}$. Since Sam is an exception to the expectation, the question communicates that he *did* attend the meeting. Furthermore, $\mathcal{E}_{p,g,f}$ is only validated if Sam is an officer, explaining why this is communicated by the question.

This hypothesis must be verified by a great deal more research into the properties of wh-questions. This is beyond the scope of this thesis, and I leave it for future work.

4.3.5 A Further Consideration

In this analysis, I have conflated what some have considered to be different phenomena. I find the distinction dubious, but include it for the sake of completeness. Von Fintel observes that in some examples the removal of the figure from the ground is not necessary for the quantification to be true, while in others it is required. For example, he suggests that for the sentences in (135), most cabinet members liking the proposal can be true without removing Joan from the set of cabinet members. Similarly, few employees accepting the pay cut can be true without removing John. Joan and John, he says, might be considered "notable exceptions" to these statements.

(135) a. Except for Joan, most cabinet members like the proposal.
b. Except for John, few employees accepted the pay cut.

(136) Except for the assistant professors, most faculty members supported the dean.

On the other hand, in the restrictive example (136), the assistant professors *must* be removed from faculty members before it is true that most of them supported the dean. This is more easily observed when one substitutes **when excluding** for **except for**.

As I mentioned, I have conflated these two phenomena in the hopes of providing a unified analysis for determining when and how the information in (116) is communicated. I have skirted the issue by simply stating that the assertion is $p(\lambda x.g(x) \wedge \neg f(x))$, which is consistent with examples like those in (136).

In some cases, it does no harm to have this as the assertion for (135). Consider that if most cabinet members excluding Joan like the proposal, then it is true that most of the complete set of cabinet members like the proposal. However, if few employees when excluding John accepted the pay cut, it may not be the case that few employees *including* John accepted the paycut. I, like von Fintel, will basically ignore this and continue to treat the assertion as restrictive. This is consistent with my general program which is primarily concerned with the presuppositional properties of these words and what they tell us about the figure.

More importantly, I believe that this distinction is very difficult to discern. At first blush, the examples in (135) are the same as that in (136), and, in fact, when one looks at the sentences closely, their actual meanings are somewhat unclear. Although there is clearly a concern with respect to these examples, I leave further investigation to future work.

4.3.6 Lexical Entries

Given these syntactic and semantic observations, I add the following lexical entry for **other than**:

$$(137)\quad \textbf{other than} \vdash \begin{cases} syn: & (\mathsf{S}/\mathsf{NP})|(\mathsf{S}/\mathsf{NP})/\mathsf{NP} \\ sem: & \lambda f\lambda p\lambda g. \begin{cases} assert: & p(\lambda x.g(x) \wedge \neg f(x)) \\ presup: & \mathcal{E}_{p,g,f} \leftrightarrow \neg p(f) \\ & alts(f,g) \end{cases} \end{cases}$$

The lexicon also must contain an entry for the type-raised NP adjunct, as in (139) and (140). Ultimately, the semantics turns out the same, but I will show it for completeness sake. Here, $t^{ID} \equiv t(\lambda x.x)$.

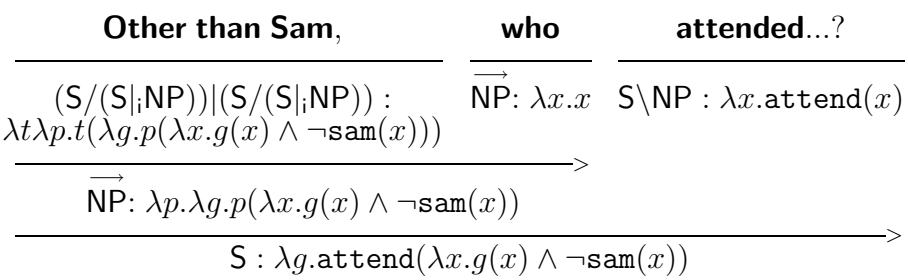
$$(138)\quad \textbf{other than} \vdash \begin{cases} syn: & (\mathsf{S}/(\mathsf{S}|_{\mathsf{i}}\mathsf{NP}))|(\mathsf{S}/(\mathsf{S}|_{\mathsf{i}}\mathsf{NP}))/\mathsf{NP} \\ sem: & \lambda f\lambda t\lambda p. \begin{cases} assert: & t(\lambda g.p(\lambda x.g(x) \wedge \neg f(x))) \\ presup: & \mathcal{E}_{p,t^{ID},f} \leftrightarrow \neg p(f) \\ & alts(f,t^{ID}) \end{cases} \end{cases}$$

As the assertion is a little odd, I show derivations where it is used in both indicative and interogative contexts.

(139)

Other than Sam,	**every officer**	**attended**...
$(\mathsf{S}/(\mathsf{S}\|_{\mathsf{i}}\mathsf{NP}))\|(\mathsf{S}/(\mathsf{S}\|_{\mathsf{i}}\mathsf{NP})) :$ $\lambda t\lambda p.t(\lambda g.p(\lambda x.g(x) \wedge \neg \mathtt{sam}(x)))$	$\overrightarrow{\mathsf{NP}}: \lambda p.p(\mathtt{officer})$	$\mathsf{S}\backslash\mathsf{NP} : \lambda x.\mathtt{attend}(x)$
$\overrightarrow{\mathsf{NP}}: \lambda p.p(\lambda x.\mathtt{officer}(x) \wedge \neg \mathtt{sam}(x))$		
$\mathsf{S} : \mathtt{attend}(\lambda x.\mathtt{officer}(x) \wedge \neg \mathtt{sam}(x))$		

(140)



The result of an interrogative is interpreted as the characteristic function for the answer to the query. Because the assertion in (140)is of type $\langle\langle e,t\rangle,t\rangle$, the result is the set of functions, g, for which $\texttt{attend}(\lambda x.g(x) \wedge \neg\texttt{sam}(x))$ is true. These functions can be sets of entities, kinds, and anything else of type $\langle e,t\rangle$. Thus, when testing whether Bill attended the meeting, for instance, the proposition to test is (141a). This, in turn, is the same as (141b) since the only entity who is Bill but not Sam is Bill.

(141) a. $\texttt{attend}(\lambda x.\texttt{bill}(x) \wedge \neg\texttt{sam}(x))$

b. $\texttt{attend}(\texttt{bill})$

(142) $\texttt{attend}(\lambda x.\texttt{sam}(x) \wedge \neg\texttt{sam}(x))$

When testing Sam, though, the resulting proposition is (142). There are no entities who, at the same time, are and are not Sam. The proposition is false, given that $\texttt{attend}(\emptyset)$ is interpreted as "nobody attended", and Sam is correctly excluded.

4.3.7 More Examples

There are several other free alternative markers that behave in a similar manner to **other than**. **Besides**, for some dialects, exhibits the same characteristics as the examples of **other than** shown so far.

Except for is the interesting case that is treated extensively by von Fintel and Hoeksema. Unlike **other than**, **except for** cannot be used in the contexts shown in (143). Hoeksema presents an analysis that excludes these by requiring **except for** to combine with universal statements.

(143) a. * Except for Sam, the officer attended the meeting.

b. * Except for Sam, some officers attended the meeting.

c. * Except for Sam, many officers attended the meeting.

Furthermore, unlike with **other than**, (116b), the fact that Sam is an officer must be communicated. This can be seen by substituting **except for** for **other than** in

the sentences (115). The resulting sentences are clearly odd. I therefore add $p(f)$ as a presupposition. As mentioned earlier, **excluding** works in a similar way.

Finally, **in addition to** completely does away with the presupposition proposed in (121). The property is always true for the figure, so we use $p(f)$ instead. It is therefore not possible to accommodate that the figure is a member of the ground, but the possibility of that being so is not precluded. Note that in the contexts of (114b), **other than** reduces to this semantics.

4.4 (Un)like

4.4.1 Syntax

	subject	object
X s v o	Unlike the president, the secretary attended the meeting.	*
s X v o	The secretary, unlike the president, attended the meeting.	*
s v X o	*	*
s v o X	The secretary attended the meeting, unlike the president.	*
X wh s v	*	*
X wh v o	*	*
wh X s v	Who, unlike the president, attended the meeting?	*
wh X v o	*	*
wh s v X	*	*
wh v o X	Who attended the meeting, unlike the president?	*

Figure 4.4: Syntactic Distribution of **Unlike**

Unlike the words in the previous section, **unlike**'s positions of attachment are restricted to the subject for the basic examples shown in Figure 4.4. This is easily handled syntactically by restricting all $\mathsf{S}|\mathsf{NP}$ categories to $\mathsf{S}\backslash\mathsf{NP}$.

However, it is also the case that **unlike** can refer to the object of topicalized sentences, as shown in (144).

(144) a. Unlike Bill, John, I will never understand.

b. John, unlike Bill, I will never understand.

c. John, I will never understand, unlike Bill.

In CCG, topicalization is accounted for by assigning the category $\mathsf{S}/(\mathsf{S}/\mathsf{NP})$ to the topicalized noun phrase (Steedman, 1987). The first two sentences, then, can be accepted by giving the alternative phrase the category $(\mathsf{S}/(\mathsf{S}|_{\text{¡}}\mathsf{NP}))|(\mathsf{S}/(\mathsf{S}|_{\text{¡}}\mathsf{NP}))$. Note that this is consistent with the categories for **other than** given in Section 4.3.

Since **other than** can also occur with topicalized sentences, this is a good prediction.

Since the new category is a relaxation of slashes for **unlike**, everything accounted for previously is still accepted. However, this will now allow the ungrammatical question *Unlike the president, who did the meeting include?*. Since all allowed questions can be accounted for with a VP adjunct category, (S\NP)|(S\NP), I just require that the other category only combine with indicative sentences. (145) shows a derivation for the first topicalized sentence in (144).

(145)
$$\dfrac{\dfrac{\dfrac{\textbf{Unlike Bill},}{(S/(S|_iNP))|(S_{ind}/(S|_iNP))} \quad \dfrac{\textbf{John},}{S_{ind}/(S/NP)}}{S/(S/NP)}> \quad \dfrac{\textbf{I will never understand}}{S/NP}}{S}>$$

The third topicalized sentence, (144c), can be allowed with the category (S/NP)\(S/NP). However, this also allows the sentence *John eats, unlike spam, beans*, which we can restrict with a topic feature. By default, noun phrases are given the category NP_{topic-} while topicalized noun phrases are $S/(S/NP_{topic+})$. The new category for **unlike** is then $(S/NP)\backslash(S/NP_{topic+})$.

4.4.2 Semantics

Unlike is semantically similar to **in addition to** in that it does not have the full presupposition in (121). But rather than having the presupposition $p(f)$, it has $\neg p(f)$. The lexical entries for **unlike** are given below, and some derivations are shown in Section 3.6.1.

(146) **unlike** $\vdash$
$$\begin{cases} syn: & (S\backslash NP)|(S\backslash NP)/NP \\ sem: & \lambda f\lambda p\lambda g \begin{cases} assert: & p(g) \\ presup: & \neg p(f) \\ & alts(f,g) \end{cases} \end{cases}$$

(147) **unlike** $\vdash$
$$\begin{cases} syn: & (S/(S|_iNP))|(S_{ind}/(S|_iNP))/NP \\ sem: & \lambda f\lambda t\lambda p \begin{cases} assert: & t(\lambda g.p(g)) \\ presup: & \neg p(f) \\ & alts(f,t^{ID}) \end{cases} \end{cases}$$

One might argue that $\neg p(x)$ should be an assertion rather than a presupposition. This is a tricky issue because the presupposition tests discussed in Section 3.4.1 are

often inappropriate or difficult to apply with complex sentences (Beaver, 1995). I now discuss this in more depth.

4.4.3 Presupposition Tests

The negation test is often difficult to apply in these cases because one must use a somewhat artificial means, such as *it is not the case that*, to establish a negated context around the entire sentence. The un-negated sentence, (148a), communicates (148c), but this is less clear for (148b).

(148) a. Unlike John, Mary hates spam.

b. It is not the case that unlike John, Mary hates spam.

c. John does not hate spam.

That John does not hate spam does seem to project through the question in (149b), but it is not so clear that it projects out of the modal context in (149a).

(149) a. It is possible that unlike John, Mary hates spam.

b. Does Mary, unlike John, hate spam?

c. John does not hate spam.

I now apply the test in which preceding context that supports the presupposition creates an acceptable discourse while context that contradicts the presupposition does not.

(150) a. John is fond of spam. Unlike John, Mary hates spam.

b. #John finds spam repulsive. Unlike John, Mary hates spam.

c. John does not hate spam.

The second sentence of both (150a) and (150b) contain the presupposition in (150c). The first example, where the context is consistent with the presupposition, is acceptable, while the second is not. This test shows that $\neg p(x)$ cannot be an implicature because implicature is defeasible. However, the same effect would be observed if $\neg p(x)$ were an assertion, and thus this test does not help us choose between presupposition and assertion.

The final test is that a presupposition should not project past a conditional whose antecedent communicates the same things as the presupposition. In (151), the presupposition that Mary did not eat the spam survives the conditional, causing the second sentence to be unacceptable since it communicates that she

did eat the spam. In (152), the presupposition that Mary did not eat the spam does not project. Had it, the next sentence would be odd because it implies that there is a possibility that she did eat the spam. But these examples take a great deal of effort to interpret and are not unequivocal.

(151) If he is sick, then, unlike Mary, John ate the spam.
#But if not, Mary must have eaten it all herself.

(152) If Mary did not eat the spam, then, unlike Mary, John ate the spam.
But if she ate the spam, John had the baked beans.

In the end, I can only conclude that the presupposition tests are inconclusive and that it is not certain whether $\neg p(x)$ is a presupposition or an assertion. Fortunately, this distinction does not appear to matter for my purposes, and until better tests are developed for complex phenomena like free alternative phrases, I will continue to treat them as presuppositions.

4.4.4 Some Interesting Constructions

Up till this point, I have only considered data in which the alternative marker subcategorizes for a noun phrase. Furthermore, I have carefully created examples to work out as simply as possible. I now take a brief look at other, more complex examples that I have observed in real data—most of which comes from the British National Corpus (Burnard, 1995). I choose to do this with **unlike**, because **unlike** seems slightly more prolific than other alternative markers, but much of this discussion is applicable to them as well.

Relative Clauses

I have already shown that alternative phrases perform correctly with wh-extraction. However, I have yet to do so for extraction from relative clauses. (153) shows how our existing categories suffice for subject extraction in the phrase *...independent producers which, unlike utilities, aren't regulated....*

(153)

producers	**which**	, **unlike utilities**,	**aren't regulated**
NP : $\texttt{prod}$	NP\NP/(S\NP) : $\lambda p\lambda q\lambda g.p(g) \wedge q(g)$	(S\NP)|(S\NP) : $\lambda p\lambda g.p(g)$	S\NP : $\lambda x.\neg\texttt{reg}(x)$
		S\NP : $\lambda g.\neg\texttt{reg}(g)$ (>)	
	NP\NP : $\lambda q.\lambda g.\neg\texttt{reg}(g) \wedge q(g)$ (>)		
NP : $\lambda g.\neg\texttt{reg}(g) \wedge \texttt{prod}(g)$ (<)			

presupposition set: $\begin{cases} \neg\neg\texttt{reg}(\texttt{util}) \equiv \texttt{reg}(\texttt{util}) \\ alts(\texttt{util}, \texttt{prod}) \end{cases}$

The derivation correctly produces the presupposition that utilities are regulated. The assertion can be read as non-regulated entities that are also independent producers.

The Copula

My analysis for **unlike** is also sufficient to handle the following interesting copular example.

(154) Among the senior Nazis with whom I came into daily contact was Reichsmarshal Hermann Goering, but unlike the three other officers I have previously mentioned, here was a man I detested from the first moment I came across him.

Example (155) is a derivation for a simpler version of the relevant portion of this sentence. I take the semantics of **here** to be a reference to an entity, e (actually, a singleton set containing the entity). A full account of the semantics would also involve information about the location of the entity, but that is not relevant here.

(155)

unlike	**the other officers**,	**here**	**was**	**a man I detested**
$\overrightarrow{\mathsf{NP}} / \overrightarrow{\mathsf{NP}} / \mathsf{NP} :$ $\lambda f \lambda t \lambda p.t(\lambda g.p(g))$	$\mathsf{NP} :$ $\lambda x.\texttt{o}(x) \wedge \neg f(x)$	$\overrightarrow{\mathsf{NP}}:$ $\lambda p.p(e)$	$\mathsf{S}\backslash\mathsf{NP}/(\mathsf{S}\backslash\mathsf{NP}) :$ $\lambda p \lambda x.p(x)$	$\mathsf{S}\backslash\mathsf{NP} :$ $\lambda x.\texttt{m}(x) \wedge \texttt{d}(\texttt{I}, x)$
	$\overrightarrow{\mathsf{NP}} / \overrightarrow{\mathsf{NP}} : \lambda t \lambda p.t(\lambda g.p(g))$		$\mathsf{S}\backslash\mathsf{NP} : \lambda x.\texttt{m}(x) \wedge \texttt{d}(\texttt{I}, x)$	
	$\overrightarrow{\mathsf{NP}} : \lambda p.p(e)$			
	$\mathsf{S} : \texttt{m}(e) \wedge \texttt{d}(\texttt{I}, e)$			

presupposition set: $\begin{cases} \forall x.f(x) \rightarrow \texttt{officer}(x) \\ alts(f, \lambda x.\texttt{officer}(x) \wedge \neg f(x)) \\ \neg(\texttt{man}(\lambda x.\texttt{officer}(x) \wedge \neg f(x)) \wedge \\ \quad \texttt{detest}(\texttt{I}, \lambda x.\texttt{officer}(x) \wedge \neg f(x))) \\ alts(e, \lambda x.\texttt{officer}(x) \wedge \neg f(x)) \end{cases}$

I begin by discussing the presuppositions, of which the first two come from **other** and the last two from **unlike**. As always, we can unify the two *alts* relations to identify the figure—in this case, e. Thus the remaining presuppositions correctly propose that the entity is an officer and that the other officers were not men that the speaker detested.

The assertion states that the the entity being referred to is a man and the speaker detests him.

Prepositional Arguments

Until this point, I have only been concerned with constructions where **unlike** subcategorizes for a noun phrase. However, as the following examples from the Penn Treebank show, it is also possible for **unlike** to subcategorize for prepositional phrases.

(156) a. One reason futures are said to add volatility is that– unlike in stocks– people can speculate in futures with little money down.

b. According to Institutional Brokers Estimate System, Wall Street market strategists see only a 2.4% jump in company profits in 1990 – unlike in 1987, when profits a year out looked good (they did soar 36% in 1988).

In my previous analyses, I have treated **unlike** and other alternative phrases as adjuncts, but not sentential adjuncts. I argued for this on the basis that the possible syntactic positions of the alternative phrases and locations of the complement, while problematic for sentential adjuncts, were well predicted by a more specific analysis.

With prepositional phrase arguments, restrictions are relaxed a bit, but some of the objections still remain. The primary objection is still that there are acceptable coordinate sentences that would not be allowed by a sentential adjunct analysis: e.g. example (157). Even so, there are some things we no longer have to worry about. For example, (158) shows that **unlike** can now appear after the verb, which was not the case in (100a). Furthermore, as in these examples and in (156′), the complement does not have to appear in the sentence—giving a distinctly anaphoric feel to the examples. This contrasts to our previous examples where the possible locations for the complement were more restricted.

(157) Mary entered the tournament and, unlike in previous attempts, made it to the second round.

(158) Mary took a trip to Rio.
There Mary experienced, unlike in her UK vacation, only a few rainy days.

(156′) a. Futures are said to add volatility. One reason is that– unlike in stocks– people can speculate with little money down.

b. According to Institutional Brokers Estimate System, Wall Street market strategists see only a 2.4% jump in company profits– unlike in 1987.

I therefore propose another lexical entry for **unlike** in (159). Syntactically, this category is very permissive, allowing the various constructions I have mentioned. However, it does not attempt to syntactically identify the complement, c, but instead anaphorically presupposes its existence. That complement is presupposed to be an alternative to the figure and, furthermore, in a relationship (specified by the preposition) with a property that is related to the original sentence. In addition, that property does not hold for the figure.

(159) **unlike** $\vdash$
$$\begin{cases} syn: & (\mathsf{S}|_{\mathsf{i}}\mathsf{NP})|(\mathsf{S}|_{\mathsf{i}}\mathsf{NP})/\mathsf{NP}/(\mathsf{PP}/\mathsf{NP}) \\ sem: & \lambda q\lambda f\lambda p \begin{cases} assert: & p \\ presup: & q(c,p') \\ & \neg q(f,p') \\ & alts(f,c) \end{cases} \end{cases}$$

(160) Assertion: $\texttt{in}(\texttt{futures}, \texttt{can}(\texttt{speculate}(\texttt{people})))$
Presuppositions: $\texttt{in}(c,p')$
$\neg\texttt{in}(\texttt{stocks},p')$
$alts(\texttt{stocks},c)$

For example, the assertion and presuppositions for *...unlike in stocks, people can speculate in futures...* are shown in (160). The most obvious values for c and p' are `futures` and `can(speculate(people))` because the assertion, from which p' should be taken, proposes the appropriate relation between them. Furthermore, the resulting *alts* relation between `stocks` and `futures` should be considered acceptable to the KB since they are both forms of investment. The remaining presupposition says that people cannot speculate in stocks (with little money down). It may seem odd that the first presupposition turns out to be the same as the assertion in this example. This is because, here, the complement and p' are explicitly available in the assertion, but this is not always the case.

If the complement and p' are not available in the assertion, as in (156′), this information must be accessed anaphorically from the discourse. At this point, it is important to point out that the logical form $\texttt{in}(c,p')$ is simply notation for what is likely to be a far more complex and interesting knowledge representation. I do not propose that we find c and p' through a process of unification. But as the precise nature of a knowledge representation that will allow a search for the relevant relationships is beyond the scope of this dissertation, I leave that for future work.

It is not always the case that the relationship between p' and the figure and complement is explicitly specified—consider (161). For cases like these, I propose

a lexical entry, (162), which is very similar to the one above except that the relationship is unspecified.

(161) But unlike ordinary prisons, members of most societies conform freely and willingly...

(162) **unlike** $\vdash$
$$\begin{cases} syn: & (\mathsf{S}|_{\mathsf{i}}\mathsf{NP})|(\mathsf{S}|_{\mathsf{i}}\mathsf{NP})/\mathsf{NP} \\ sem: & \lambda f\lambda p \begin{cases} assert: & p \\ presup: & q(c,p') \\ & \neg q(f,p') \\ & alts(f,c) \end{cases} \end{cases}$$

The presuppositions and a possible paraphrase for the assertion for this sentence is shown in (163). Here, in a similar manner as above, the complement can be found to be `societies` and p', `conform(members)`. Thus, it is also presupposed, correctly, that in prisons, the members do not conform. I do not deny that this analysis requires a great deal from knowledge representation and reasoning. Nevertheless, I do not believe this is unreasonable and, as mentioned, propose it for future research.

(163) Assertion: $\texttt{in}(\texttt{societies}, \texttt{conform}(\texttt{members}))$
Presuppositions: $\texttt{q}(c, p')$
$\neg\texttt{q}(\texttt{prisons}, p')$
$alts(\texttt{prisons}, c)$

4.5 Especially

Finally, there are also "specifiers" such as **especially**, **in particular**, **particularly**, and **specifically**, which promote certain entities as being good examples within an alternative set. Figure 4.5 shows several examples.

Semantically, all constructions of this sort communicate at least the fact that the figure has the property of the ground, (116b). In the case of the examples, this is that stocks are risky funds. In addition to this, I add the assertion that the figure is an exemplar of a combination of the ground and the predicate of the sentence. In the case of *John recommends risky funds, especially stocks*, this would mean that stocks are a good example of risky funds that John recommends.

especially $\vdash$
$$\begin{cases} syn: & (\mathsf{S}|_{\mathsf{i}}\mathsf{NP})\backslash(\mathsf{S}|_{\mathsf{i}}\mathsf{NP})/\mathsf{NP} \\ sem: & \lambda f\lambda p\lambda g \begin{cases} assert: & p(g) \wedge exemplar(f, \lambda x.p(x) \wedge g(x)) \\ presup: & \forall x.f(x) \rightarrow g(x) \\ & alts(f,g) \end{cases} \end{cases}$$

	Subject
X s v o	*
s X v o	Risky funds, especially stocks, can devastate the short-term investor.
s v X o	*
s v o X	Risky funds can devastate the short-term investor, especially stocks.
X wh s v	*
X wh v o	*
wh X s v	*
wh X v o	What funds, especially stocks, can devastate short-term investors?
wh s v X	*
wh v o X	What funds can devastate short-term investors, especially stocks?

	Object
X s v o	*
s X v o	*
s v X o	*
s v o X	John recommends risky funds, especially stocks.
X wh s v	*
X wh v o	*
wh X s v	What funds, especially stocks, does John recommend?
wh X v o	*
wh s v X	What funds does John recommend, especially stocks?
wh v o X	*

Figure 4.5: Syntactic Distribution of **Especially**

$$\textbf{especially} \vdash \begin{cases} syn: & \overleftrightarrow{\mathsf{NP}} \setminus \overleftrightarrow{\mathsf{NP}} \\ sem: & \lambda f \lambda t \lambda p \begin{cases} assert: & t(p) \wedge exemplar(x, \lambda x.p(x) \wedge t^{ID}(x)) \\ presup: & \forall x.f(x) \rightarrow t^{ID}(x) \\ & alts(f, t^{ID}) \end{cases} \end{cases}$$

It is interesting to point out that the evaluation of this relation depends on the point of reference. For example, the sentence *John recommends risky funds, especially stocks* is a statement that the hearer is requested to take on in their own knowledge. Thus, with the proposition in (164) the hearer should now believe that stocks are a good example of risky funds that John recommends.

(164) $exemplar(\texttt{stocks}, \lambda x.\texttt{risky\_funds}(x) \wedge \texttt{recommend}(\texttt{john}, x))$

However, in a question like *What risky funds, especially stocks, does John recommend?*, the hearer attributes the exemplar relation to the questioner but does not take it on himself. Thus, **especially** communicates to the hearer what type of answers the questioner would prefer. In the case of this example, the hearer should reply with stocks before anything else.

There is a lot going on here, and I leave further investigation for future work.

4.6 Conclusion

In the last two chapters, I have given an in-depth account of the semantics of alternative phrases using alternative sets, presupposition, and a "use existing objects" heuristic also used in abductive approaches to discourse interpretation.

Although I only treat a subset of alternative phrases, I still go significantly further than previous research. Up till this point, formal semantic analyses have been restricted to a few examples (**but** and **except for**) and are primarily concerned with the assertional semantics. Pattern-matching techniques which attempt to handle more examples are not effective for free alternative phrases and would need to be extended to account for alternative phrases with anaphoric reference.

I have discussed the connected alternative markers **besides**, **such** (**as**), and **other** (**than**) and the free alternative markers **besides**, **other** (**than**), **excluding**, **except for**, **in addition to**, **unlike**, **especially**, and more. This is significantly more breadth than previous semantic approaches. It is true that I do not display the same depth regarding assertional semantics (determining only the referred-to set), but I believe my approach can incorporate the previous work. The major new contribution of my analyses is what is being expressed about the *figure*, the NP argument of the alternative marker. These properties also put these analyses on a deeper theoretic level than the pattern-matching approaches.

Chapter 5

A Robust CCG System

This chapter is the result of joint work with Jason Baldridge and Julia Hockenmaier. More details can be found in Hockenmaier *et al.* (2000) and Hockenmaier (2000).

5.1 Introduction

The computational analysis provided earlier is only "useful" if it can be implemented in a real-world NLP system. Many efforts have been made to produce wide-coverage systems, but they face a particular challenge when required to support the analysis of alternative phrases.

In this chapter, I begin by discussing the requirements for an NLP system that can support the analysis presented in Chapters 2–4 as well as the practical application in Chapter 6. I then briefly discuss several systems and evaluate them with respect to these requirements. Finally, I describe an implemented NLP system, Grok, and show how it addresses the necessary requirements.

5.2 Requirements for an NLP System

There are several requirements for a system to be capable of supporting my analysis of alternative phrases in a real-world application.

When an NLP system describes itself as *wide-coverage* , it can mean one of two things.

Syntactic Coverage First, the term can pertain to the ability of its grammar to recognize a large variety of linguistic constructions—often the focus of hand-built grammars. This is certainly necessary for our system. We require a system flexible enough to accommodate the analyses of connected and free

alternative phrases that are my primary focus. In addition, it must cover as wide a range of constructions as possible to support the natural language information retrieval application in Chapter 6 as well as future applications.

Parsing Coverage We also require the ability to parse significant portions of free text. This notion of wide coverage corresponds much more to the view generally held among statistical NLP researchers. For them, one of the foremost concerns of parsing should be the ability to assign a linguistic analysis (structure, derivation, etc.) to virtually all the data one processes, giving less importance to issues such as grammaticality.

It is not the case that syntactic coverage leads to parsing coverage. To achieve parsing coverage, we require a large lexicon, which is independent of the syntactic analyses. There must also be support for classes of words that are too large to list in a lexicon: e.g. proper nouns. Finally, the system must handle input that has not been tokenized, edited, or cleaned up in any way.

Conversely, wide parsing coverage does not imply good syntactic coverage. Being able to assign structure to a large proportion of text does not mean that the structures are consistent with linguistic analysis or form a coherent basis for compositional semantics.

Semantics and Presupposition The system must be capable of processing the semantics of alternative phrases in a manner sufficient to support the practical application in Chapter 6. This does not require particularly in-depth semantic theory. For instance, I do not need intensionality, possible worlds, temporality, modalities, etc.

However, simple dependency relations are not enough. I require the ability to support the semantics described in Section 2.2.1; basically the first-order extensional part of Montague semantics. A system must also support a basic understanding of relations (such as the *alts* relation) and have some way of representing presuppositions.

Lexicalized Grammar A further requirement is that the system use a highly lexicalized grammar. This reflects the nature of the analyses in Chapter 3 and Chapter 4 but is not, in general, a requirement for a robust natural language understanding system.

Anaphora Resolution, Knowledge Representation, etc. The previous requirements deal primarily with the coverage and expressiveness of a system's

grammar. However, a system that supports alternative phrases must also be able to perform anaphora resolution, and therefore maintain a discourse and salience list. In addition, the *alts* relation used in the presuppositions in Chapter 3 and Chapter 4 requires that alternative sets be maintained in some manner.

Given that the grammar and parsing requirements are met in a system, these more interpretation-oriented requirements can be added in a relatively modular manner. I therefore restrict my discussion of previous work to the previous requirements.

5.3 Previous Work

5.3.1 Statistical Parsing

Very wide parsing coverage is usually achieved through statistical approaches. Charniak (1999) and Collins (1998) are currently the cutting edge of this research with parsers that produce around 90% precision/recall on the Wall Street Journal Corpus. The parsing coverage of these techniques is very impressive, but they do not necessarily provide particularly good syntactic structure. Collins' second model, for instance, labels all verbs within object extracted relative clauses as intransitive (Collins, 1998, p. 176).

In addition, from these statistical methods one can obtain some dependency analysis but, as discussed, this does not meet my semantic needs. Finally, in general these parsers are not based on lexicalized grammars, and it would be difficult to incorporate the lexical semantics presented in this dissertation.

As I shall show, these problems are not intrinsic, and statistical techniques will play an important role in making Grok robust.

5.3.2 Hand-Built Grammars

Hand-built grammars, on the other hand, are usually more concerned with syntactic coverage, i.e. providing adequate syntactic analyses of a wide range of linguistic phenomena. For example, the XTAG project has built a substantial English LTAG lexicon which covers an extensive number of English constructions (XTAG-group, 1999; Joshi and Schabes, 1992). Similarly, LinGO is a large hand-built grammar for HPSG (Copestake, 1999; Pollard and Sag, 1994).

While possessing wide syntactic coverage, by themselves these grammars are

not particularly robust. Many hand-built grammars have rather small lexicons, and the larger ones (like XTAG and LinGO) were time-consuming and expensive to build. And unlike in statistical approaches, hand-built grammars contain no information regarding the preference of one lexical entry over another. Therefore, they can produce many analyses for a sentence, but they cannot effectively choose between them unless they have associated semantic forms that violate some semantic property.

In addition, hand-built grammars are often lacking in semantics. The XTAG lexicon, for example, contains no semantic information. There is no theoretical reason why LTAG should not be capable of handling semantics and even presuppositions, as demonstrated in Joshi and Vijay-Shanker (1999); Webber *et al.* (1999b); Stone and Doran (1997); Palmer and Wu (1995), but so far this has not been incorporated into a wide-coverage grammar. LinGO, on the other hand, does have semantics, but does not yet incorporate lexical presupposition or its projection into clausal semantics (Copestake, 1999). That grammar, then, would have to be extended to account for alternative phrases.

DORIS (Bos, 2000) is a good example of a system with a hand-built grammar, but with rather different emphasis than XTAG and LinGO. DORIS is fully capable of fulfilling all of the semantic requirements, including presupposition. It even has an analysis of **other**, although only in contexts like *Mary took the Volvo and John took the other car.* However, as of now it has very low parsing coverage with a lexicon of only a few thousand words.

5.3.3 Lexicon Acquisition

To address the problem of low parsing coverage and/or accuracy in hand-built grammars researchers have turned their attention to the use of corpora in order to extract lexical syntactic information. Such techniques have been used to create grammars semi-automatically for LFG (Kuhn *et al.*, 1998; van Genabith *et al.*, 1999). Also, Villavicencio (1997) did a semi-automatic translation of the Alvey Natural Language Tools English grammar (Grover *et al.*, 1993) to create a large CCG lexicon. The original Alvey lexicon was, in itself, acquired semi-automatically. Xia (1999) has automatically acquired LTAG lexicons from the University of Pennsylvania Treebank (Marcus *et al.*, 1993a). And IBM's Slot Grammar also uses an acquired lexicon (McCord, 1990).

Since these systems or grammars are based on formal linguistic systems, in general they provide wide syntactic coverage. And since the lexicons were acquired automatically or semi-automatically, they achieve a much higher level of

parsing coverage than simple hand-built grammars. However, they are still insufficient in that they do not have a semantic component capable of handling the analysis for alternative phrases. The Alvey grammar has the most developed semantic component but does not incorporate presupposition. It is also the case that the Alvey grammar uses a form of Generalized Phrase Structure Grammar (GPSG) which, while attributing some semantic information to lexical entries, is not as lexicalized as would be necessary to easily incorporate the proposed semantics for alternative sets. This is reflected in the fact that in Villavicencio's translation to CCG, much of the semantic information appears to have been lost.

5.3.4 Grok

Grok, a modular NLP system written in Java, provides a CKY-style chart parser (Kasami, 1965; Younger, 1967) and maintains a discourse model including alternative sets, a salience list based on *centering* (Grosz *et al.*, 1995), and a dynamic ISA hierarchy. More importantly, Grok supports a wide-coverage CCG lexicon with both assertional and presuppositional semantics. Like the systems in Section 5.3.3, the lexicon is both acquired and based on a formal grammar system. The result is both good syntactic and parsing coverage. The rest of this chapter will show how the lexicon (including lexical entries for alternative markers like those given in Chapter 3) has been created for Grok. I also discuss performing efficient parsing with this grammar and show how the results are interpreted and stored in a discourse model.

5.4 A Lexicon

5.4.1 An Acquired Lexicon

Recent work by Hockenmaier (2000) has shown that a large CCG lexicon can be induced from an annotated corpus such as the Penn Treebank (Marcus *et al.* 1993b). For unseen data withheld from the same source, this lexicon contains the correct category for 97.25% of the words. This is not due to extremely ambiguous lexical entries as the average number of categories per word is 1.7. Not only does this technique bootstrap a wide-coverage lexicon, it also is capable of collecting the frequency of a category for a given word. This allows simple probabilistic parsers to be written as described in Section 5.5. These parsers can be made more general and accurate by incorporating the context-sensitive probabilities produced by super-tagging (Bangalore, 1997) or more advanced probabilistic models (as in

work currently being carried out). But they still have the problem, described in Section 5.3.1, that no semantics (assertional or presuppositional) is associated with the categories. In addition, little morphological information is available from the acquired lexicon.

The following sections describe how these two problems are solved.

5.4.2 A Hand-Built Lexicon for Closed-class Items and Open-class Categories

Structure of the Lexicon

A hand-built lexicon is used to supplement the acquired lexicon described above. This lexicon consists of entries for closed-class words such as prepositions and determiners. In addition, categories are included for open class words, but the words themselves are not listed in the lexicon, being instead supplied by a morphological analyzer. (Grok's morphological information is actually retrieved from a large database provided by the XTAG project.) Given a lexical item, a morphological analyzer provides one or more possibilities for the item's part of speech, stem, and morphological features. These features are then incorporated into the lexicon for every entry of the given stem and part of speech. Thus, a simple lexical entry such as (165a) specifies the syntactic and semantic category for a stem. This, together with the information provided by the morphological analyzer (165b), expands to the categories in (165c).

(165) a. **walk** $\vdash \mathsf{S}\backslash\mathsf{NP} : \lambda x.\mathtt{walk}(x)$

b.
walks:	V	walk	3sg pres
walked:	V	walk	past
	V	walk	pparticiple

c. **walks** $\vdash \mathsf{S}_{\mathsf{3,sing,pres}}\backslash\mathsf{NP} : \lambda x.\mathtt{walk}(x)$
walked $\vdash \mathsf{S}_{\mathsf{past}}\backslash\mathsf{NP} : \lambda x.\mathtt{walk}(x)$
walked $\vdash \mathsf{S}_{\mathsf{ppart}}\backslash\mathsf{NP} : \lambda x.\mathtt{walk}(x)$

Actually, the hand-built lexicon does not have specific lexical entries like (165a), but rather uses the notion of *families* from XTAG to organize the hand-built lexicon into collections of categories. Each family is marked with an associated part of speech corresponding to the POS tags produced by the morphological analyzer. Thus, rather than lexical entries for each open class word, we have open

class families such as (166)[1]. A semantic form for an open class family is a template for the semantics of the lexical items in that family. The variable P is filled by the stem of the lexical item. For example, the semantics of the intransitive word **walks** would be $\lambda x.P(x)$ where P is bound to `walk`: i.e. $\lambda x.\texttt{walk}(x)$.

(166)

a. Intransitive:	V	$\mathsf{S\backslash NP}$	$\lambda x.P(x)$
b. Transitive:	V	$\mathsf{S\backslash NP/NP}$	$\lambda x\lambda y.P(y,x)$
c. Ditransitive:	V	$\mathsf{S\backslash NP/NP/NP}$	$\lambda x\lambda y\lambda z.P(z,y,x)$
		$\mathsf{S\backslash NP/PP/NP}$	$\lambda x\lambda y\lambda z.P(z,x,y)$
d. Sent Comp:	V	$\mathsf{S\backslash NP/S_{ind\|int}}$	$\lambda p\lambda x.P(x,p)$
		$\mathsf{S\backslash S\|NP}$	$\lambda x\lambda p.P(x,p)$
e. Control:	V	$\mathsf{S\backslash NP/(S_{to}\backslash NP)}$	$\lambda p\lambda x.P(x,p)$
f. Noun:	N	$\mathsf{NP_{bare+,comp-}}$	$\lambda x.P(x)$
g. Adjective:	A	$\mathsf{NP_{comp-}/NP_{bare+}}$	$\lambda p\lambda x.P(x) \wedge p(x)$
h. Comparative:	A	$\mathsf{NP_{comp+}/NP_{bare+}}$	$\lambda p\lambda x.P(x,f) \wedge p(x)$
		$\mathsf{NP_{comp+}\backslash NP/NP/}$"than"	$\lambda\_\lambda f\lambda p\lambda x.P(x,f) \wedge p(x)$

Open-class words are assumed to belong to all families with a part of speech suggested by the morphological analyzer. For instance, the morphological entries in (165b) associate **walked** with the `V` part of speech. **Walked** therefore belongs to all families with the `V` part of speech and is given every entry of those families. These are the intransitive, transitive, ditransitive, sentential complement, and control families shown in (166).

Our families include nouns, pronouns, verbs with a variety of subcategorization frames, adjectives, comparatives, adverbs, modals, small clauses, wh-words, prepositions, conjunctions, and more. Many of these are described in Chapter 2. We do not, though, include a family for everything. For instance we do not list the categories that would be needed to construct proper nouns: e.g. "Mr. P. T. Barnum" and "Royal Bank of Scotland". Section 5.7.1 discusses this in more detail.

Over-generation

Used alone, this hand-built lexicon would achieve wide coverage, but at the price of high ambiguity and over-generation. For example, the word **devoured** will be given an intransitive entry even though it can only appear in a transitive context.

[1]This list of families is an example and is not complete. In addition, some families have more entries than are shown, and I have only included some of the more important features on the categories.

However, over-generation is not necessarily a bad thing. Dialects vary in the possible families for particular words. For example, **enjoy** can be used in an intransitive context in some dialects but not in others. By being permissive, the hand-built lexicon has no problem in accommodating these differences.

Research in language acquisition shows that such permissiveness is present in children (Bowerman, 1982). They often generalize a phenomenon and only later apply it to a restricted set of lexical items—e.g. sentences expressing cause and effect relations. Bowerman notes that children seem to generalize constructions like *Harry pulled his socks up* and produce sentences like those in (167). Although these sentences are odd, especially the second, they are understandable.

(167) a. Don't hug me off my chair.

b. I'll jump that down. [about to jump on a bathmat put on top of water in tub]

So starting with an over-generating lexicon is not unreasonable. We would still like to reduce the over-generation for the sake of parsing efficiency and to use the same lexicon for generation (Section 7.3). The next section shows how the hand-built lexicon can be restricted to categories observed in a particular corpus.

5.4.3 Merging the Lexicons

In Grok, various modules, such as the parser, query the lexicon module for the lexical entries of a particular word. That module, in turn, queries the hand-built and acquired lexicons. These two sets of results must then be combined. The ideal case would be for the lexical entries of the hand-built lexicon to mirror those of the acquired lexicon. Then, it would be a simple matter to take the morphological features and the semantics from the hand-built lexicon to supplement the acquired one. Unfortunately, it is common for each lexicon to propose some categories that the other does not. This gives rise to three situations:

The first is the simple case where a syntactic category is proposed both by the hand-built lexicon and the acquired lexicon. In this case, the category is accepted. A small concern is that the categories are probably not exactly the same since the hand-built lexicon generally proposes significantly more morphological information. Thus, we unify the two syntactic categories and include the semantics from the hand-built category and the statistical information from the acquired category. For example, (168) shows compatible categories for **that** proposed in both lexicons. The hand-built syntactic category has more morphological information,

and the acquired lexicon specifies that the extracted argument is the subject[2]. The unified category contains all of this information. The combined lexical entry also contains the semantics from the hand-built entry and the probability from the acquired entry.

(168) a. Hand-Built
that $\vdash (\mathsf{NP}_{\mathsf{bare+}}\backslash\mathsf{NP}_{\mathsf{bare+}})/(\mathsf{S}_{\mathsf{ind}}|\mathsf{NP}) : \lambda p\lambda q\lambda x.p(x) \wedge q(x)$

b. Acquired
that $\vdash (\mathsf{NP}\backslash\mathsf{NP})/(\mathsf{S}_{\mathsf{ind}}\backslash\mathsf{NP}),\ P = .24$

c. Combined
that $\vdash (\mathsf{NP}_{\mathsf{bare+}}\backslash\mathsf{NP}_{\mathsf{bare+}})/(\mathsf{S}_{\mathsf{ind}}\backslash\mathsf{NP}) : \lambda p\lambda q\lambda x.p(x) \wedge q(x),\ P = .24$

The second case is where, for a particular word, a category is specified in the hand-built lexicon but not in the acquired lexicon. It is tempting to disregard such a category as being the result of the lexicon's over-generation: e.g. proposing an intransitive category for **devours**. However, it is possible that the category is valid but could not be acquired from the corpus. For example, the acquisition algorithm is not capable of acquiring categories for multi-word lexical items like **other than** and **such as**, which are given categories in the hand-built lexicon. Fortunately, these categories almost always occur for closed-class items. Thus, we employ the heuristic of taking the category if it is in a closed-class family and disregarding it otherwise. Of course, no statistical information is available for these categories and further research is necessary to determine how best to provide an approximation for this (Section 7.2). For now, we simply assign them an arbitrary, high probability. This prefers hand-built, closed-class entries over acquired ones.

The third and most difficult case is when a word has an acquired category not in the hand-built lexicon. This frequently occurs in large open classes where the word does not occur in our morphological database. For example, the morphological database in Grok does not have an entry for **sneaker** while the acquired lexicon correctly proposes an NP category.

Lacking a corresponding entry in the hand-built lexicon results in impoverished morphology and no semantic information. One might expect that the tight connection of syntax and semantics inherent in CCG would allow one to deduce semantics directly from categories. Were it the case that each syntactic category had exactly one possible semantic category, this would indeed be trivial. But this

[2]If the acquired lexicon also proposes the object extraction category, this would be included as well.

is not the case, as shown by (169). In this simple example, **big** and **dog** are both NP modifiers. However, while **big** compositionally combines with an NP, such as **house**, to denote an entity which is both a house and big, **dog house** does not denote an entity which is both a house and a dog. Instead, a different semantic relation, *rel*, is communicated. I do not attempt to determine what this relation is (it is different for **dog house**, **dog sled**, **dog slobber**, etc.) since this is a difficult, well-studied problem that is not germane to this thesis. Rather, the point is that adjectives and nouns have a common syntactic category in this lexicon but different semantics. This is one example of a general problem.

(169) a. **big** $\vdash$ NP/NP : $\lambda p \lambda y.p(y) \wedge \texttt{big}(y)$

b. **dog** $\vdash$ NP/NP : $\lambda p \lambda y.p(y) \wedge rel(\texttt{dog}, y)$

It seems obvious in this case that these examples can be distinguished by the fact that **dog**, in the Treebank, is annotated with the noun POS tag (NN) while **big** is annotated as an adjective (JJ). This is true, but there are many possible features that can affect these choices—syntactic arity, syntactic features, morphology, etc. Therefore, rather than attempting to hand-code heuristics to determine semantic categories from syntactic categories, I have used a machine learning approach.

5.4.4 Learning Semantics for Acquired Categories

I started with randomly selected entries from the acquired lexicon while maintaining the distribution of word/category pairs observed in the corpus. I excluded numbers, punctuation, and proper names as these are handled in Grok through other means (see Section 5.6).

For each of these lexical entries, I produced training data consisting of the following features: the orthography of the word, its suffix, the number of syntactic arguments, various features of the acquired category, and whether the syntactic category is a verb, verb modifier, noun phrase, or noun phrase modifier. Also included is the correct semantic template for the lexical entry. These semantic forms include those for basic verbs, adjectives, pronouns, determiners, noun-noun constructions (e.g. **dog house**), prepositions, and alternative phrases. There are 21 semantic templates in all. These semantic forms contain several ambiguities like that in (169), but even so, it is still a rather impoverished set. It is sufficient, though, to fulfill the goals in the practical applications in Chapter 6. Also, there is no information about the semantics for categories that do not appear in the hand-built lexicon. But, as discussed later this section, these are not frequent.

I tested both the decision tree and maximum entropy approaches to machine learning on this task. I use Ripper (Cohen, 1995) to induce a decision tree. Maximum entropy is a powerful statistical method for estimating decisions by combining diverse pieces of information. Ratnaparkhi (1998) explores the maximum entropy framework with respect to many linguistic problems and demonstrates how tools using it can achieve high accuracy and domain independence.

I began with 10,000 entries from the acquired lexicon. Of these, 8866 were contained in the hand-built lexicon and thus had associated semantics with which to train the models. 90% of the data points were used for training and 10% for testing.

	training	testing
Baseline	86.4%	73.2%
Ripper	97.2%	97.3%
Maxent	97.8%	97.1%

Table 5.1: Evaluation of Inferring Semantics

As shown in Table 5.1, both the decision tree and the maximum entropy approaches perform in the ninety-seventh percentile at predicting the correct semantic form given the features described above. I compare these results to a baseline which simply chooses the most frequent semantic form for a word and guesses the overall most frequent semantic form if the word is unknown. As one can see, this performs significantly worse than both machine learning techniques.

To interpret the usefulness of these results, I present a few more statistics, illustrated in Figure 5.1. For 95.8% of the tokens in the training corpus for the acquired lexicon, the syntactic category was available in the hand-built lexicon. For 88.7% of the tokens, the exact entry was found in the hand-built lexicon. These are data on which the models were trained but not on which they will be used.

These statistics mean that for 7.1% of the tokens, the hand-built lexicon had the appropriate category but was unable to associate that category with the lexical item. These are the cases for which we believe our models will provide good performance for guessing semantic templates. It is possible these particular lexical items are fundamentally different from the other 88.7%, which is why their entries were not found in the hand-built lexicon. However, our morphological analyzer is simply a large database and does not contain a number of open class words (e.g. **sneaker**, as described above). Thus, it is exactly for these tokens that the approaches evaluated in Table 5.1 are meant. We believe, then, that for this

7.1%, we can predict the correct semantic form around 97% of the time.

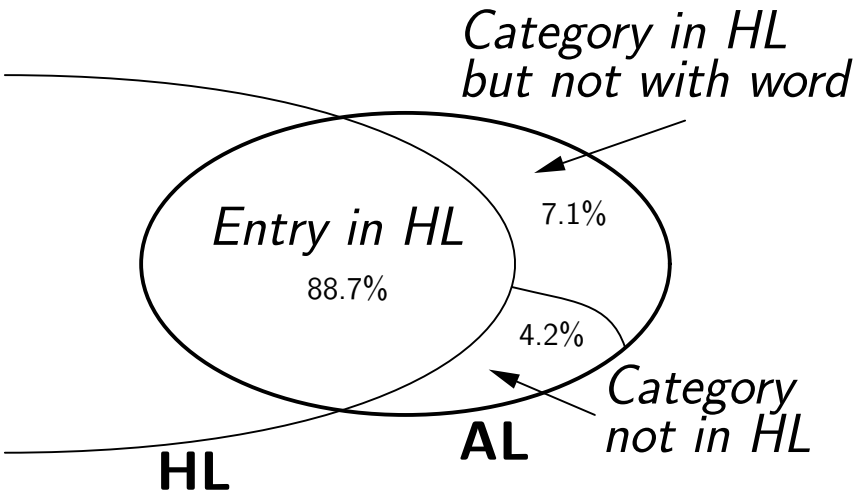


Figure 5.1: Supplying Semantics to the Acquired Lexicon

The remaining 4% of the tokens are those for which the hand-built lexicon does not even contain the syntactic category. Since the training data only consists of entries contained in the hand-built lexicon, we have no reason to believe that the statistical models will perform well in predicting their semantic forms. It would be possible to test this by hand-building a test set from this class, but I have not done so for two reasons. First, many of these categories are the result of noise in the Treebank and can be discounted. Second, the lexical entries produced when creating a test set could be easily integrated into the grammar and used to retrain the statistical models. In fact, this process was used to improve the hand-built grammar.

5.5 Efficient Parsing

The acquired lexicon specifies a set of possible categories for each word, and also includes unigram probabilities (of the form $P(category \mid word)$ or $P(word \mid category)$). The task of the parser is then to select the appropriate categories from this lexicon. As a baseline evaluation, I have used a lexicon which includes probabilities for a word having a particular category, and I have incorporated the following first-best search strategy into Grok's chart parser: Each edge in the chart is given a "measure" which is the average of the probabilities of its composite categories. Edges are sorted based on the measure and processed in

descending order. Termination occurs when there are no more edges or a sentence is derived for the entire string.

This parser processed a list of 41 hand-created sample sentences (meant to exercise the system on various linguistic constructions) in 32.83 seconds compared to 160.73 seconds when there is no preference between edges. This is an 80% reduction in run time. While speed alone is not a rigorous performance criterion, this can be viewed as confirmation that the simple unigram probabilities are already good predictors of categories during parsing. The accuracy of the parser has not been tested since its construction was meant to satisfy an engineering rather than a theoretical goal, but it is unlikely that the accuracy is worse than the basic chart parser where there are not even the simple unigram probabilities to help choose among parses.

A long-term goal is more realistic statistical parsing with CCG, and in order to do this, we need a statistical model, such as Collins (1998), which includes lexical head-dependency probabilities. Hockenmaier (2000) is an important first step towards this goal, and current research is promising. An extremely naive statistical model has already performed at 79% accuracy on the labeling task for unseen Treebank text.

5.6 Preprocessing

I have discussed a robust way of generating a CCG lexicon which does not depend on using extensive human labor. Nonetheless, even the Wall Street Journal text tends to be quite messy from the perspective of a CCG parser. For instance, the following passage from the WSJ contains a number of problems:

(170) Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29. Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.

The most obvious problems are tokenization and sentence detection. Is "Vinken," one token or two? Where is the sentence boundary in "...Nov.29. Mr. Vinken..."? Other issues include subsets of the unknown word problem: e.g. recognizing dates and named entities and dealing with abbreviated terms and numerical units in the text.

These are problems faced by all natural language understanding systems, and a great deal of research has gone into preprocessing techniques to deal with them. We have implemented a number of preprocessing modules within an open architecture (see Section 5.8) in the hopes that the open standards will facilitate the

reuse of code. In Section 5.8, I discuss how Grok is already being used in the NLP community. In the rest of the section, I briefly discuss the implementation of some of these components. I also point out how certain preprocessing techniques can, in particular, help parsers of lexicalized grammars.

We have created a pipeline of preprocessing components which use XML documents in the input/output specifications. XML (World Wide Web Consortium, 1997) provides an elegant means for structuring texts. We have built the components themselves by using maximum entropy probability models.

At present, we have built a tokenizer and sentence detector which use maximum entropy models with features based on those presented in Ratnaparkhi (1998) and Reynar and Ratnaparkhi (1997), respectively. Other tasks such as paragraph detection or dealing with figures and tables have not been implemented as yet, but the existing components have been designed to work with XML documents in a manner which permits additions such as these with minimal effort.

With tokenized and sentence detected data in hand, we are much closer to the goal of feeding the text to the CCG parser. However, there are still many things we can do to ease the parser's work and reduce the burden on the lexicon. One such task is named entity recognition. We feel that names should not be derived through standard CCG derivations for several reasons. First, the acquired lexicon is limited to those names it observes in the data. Furthermore, for a name such as **John**, the acquired lexicon produces two categories: NP for **John** alone and NP/NP as in **John Smith**. This is particularly detrimental since a large factor in the efficiency of CCG parsing is dependent on lexical ambiguity and there tend to be a lot of names in text. Finally, names lack recursive structure, are not compositional, and fit a somewhat standard format, so finding a CCG derivation for them will not tell us much. We thus choose to put the burden of deriving names on another component. Then, when a sentence is given to the parser, it receives names as a chunk whose components are invisible. A side benefit of this is that we can use the results of name detection to know whether or not we can decapitalize sentence initial words so that we do not need dual lexical entries for non-names occurring at the beginning of sentences.

Our maximum entropy software, based on Ratnaparkhi (1997), has allowed us to implement a simple though high-performing named entity recognizer in less than a day. The results for evaluation on 20,468 training sentences and 4552 unseen testing sentences are given in Table 5.2. Named entity recognition is a well-studied task, and it should be a simple matter to substitute a more developed approach such as Mikheev (1999), which achieves about 98.5% accuracy on unseen

data and was used in the winning entry in MUC-7 (Mikheev *et al.*, 1998).

	training	testing
precision	96.1%	95.9%
recall	96.9%	94.8%

Table 5.2: Evaluation of Name Detection

This approach can be extended to other phenomena such as dates, appositives, and compound nouns. Dates can be handled similarly to names. Both nominal appositives (such as *the Dutch publishing group*) and adjectival appositives (such as *61 years old*) have a very restricted syntactic positioning and are clearly delimited by commas, so we expect another maximum-entropy component would do well in detecting them. However, unlike names, appositives have a compositional meaning and can have recursive structure. Even so, at the top they are still of the category NP or NP/NP. With appositives detected, we could give the parser the appositive's lexical items with the goal NP for nominal appositives or NP/NP for adjectival appositives in order to compute its meaning before proceeding with the rest of the sentence.

Cutting up the text in the manner described above can yield significant gains for a CCG parser. The text in (170) provides an excellent example of the benefits. If we were to perform name, date, and appositive detection on that text, a standard chart parser would visit only 25% of the cells it would using the unprocessed tokens. Furthermore, since the complexity for lexicalized grammars such as CCG is also dependent on the number of categories per word, the reduction in the number of categories for dates, numbers, and names provided by dealing with them in this way will translate into yet more efficiency gains, especially since the maximum entropy components themselves are extremely efficient.

5.7 Representing and Working with Knowledge

The anaphoric aspects of my analysis for alternative phrases requires that Grok must have some way of storing, accessing, and manipulating the semantics generated by the parser. This is, of course, an enormous task. What I will show with the Grok system is that it is not necessary to pay particularly close attention to every aspect of such a system. At the same time, *something* must be done about *everything*.

Modularity is the key. Grok uses the philosophy that complex behavior can result from the interaction of simple pieces. In a well-engineered system, if more

attention must be paid to a particular task, a more complex module can be substituted for a simpler one and the system will continue to function as before but with the improved functionality of the new module. This is handled through Java *interfaces*—requirements a module must implement. For example, the interface for a tokenizer is shown in (171b).

(171) a.
```
public interface Pipelink {
    public void process(NLPDocument doc);
    public Set requires();
}
```

b.
```
public interface Tokenizer extends Pipelink {
    public String[] tokenize(String s);
}
```

This interface specifies that a tokenizer must have a function `tokenize` that takes a string of text and returns an array of strings which are the tokens. The modifier `extends Pipelink` means that a Tokenizer must also implement the `Pipelink` interface which allows it to be used in a pipeline. This requires two methods. One takes an XML document and processes it (in this case, by identifying tokens) and the second returns a set of modules that must be present earlier in the pipeline. A tokenizer, for instance, might require that a sentence detector be earlier in the pipeline. Grok's pipeline implementation ensures that such requirements are satisfied. How these interfaces are organized and implemented is described in Section 5.8.

The rest of this section describes aspects of the Grok knowledge system that are directly relevant to the interpretation of alternative phrases.

5.7.1 Presupposition

The dissertation so far has shown how the rules of CCG direct the evaluation of syntax and assertional semantics in a derivation, but not how presuppositions are evaluated.

I have indicated that presuppositions are stored lexically and are scoped by the same parameters as the assertional semantics. During parsing, as adjacent strings are combined via CCG rules, the corresponding presuppositions are combined as well. This is a compositional, monotonic approach that does not address the projection problem (Karttunen and Peters, 1979). Since I only consider presupposition to the extent necessary to drive my analysis, I do not address this well-studied problem here.

Packaging presuppositions within the derivation allows Grok to entertain multiple interpretations simultaneously if it uses a parsing technique that builds many derivations in parallel. At this time, Grok does not do incremental consistency checking, but rather waits until a particular parse is chosen. This maintains the monotonicity of the KB but does not exploit the disambiguating effect of presuppositions and, in turn, the potential reduction of parsing time. Incremental interpretation also requires incremental anaphora resolution since presupposed figures can be realized by pronouns. To this end, I have implemented the incremental centering model given in Strube (1998). An evaluation of the utility of incremental processing is ultimately desirable, but I leave this for future work.

5.7.2 Resolution

The anaphoric nature of some alternative phrases makes resolution an important issue. The two important aspects of resolution are restrictions and salience.

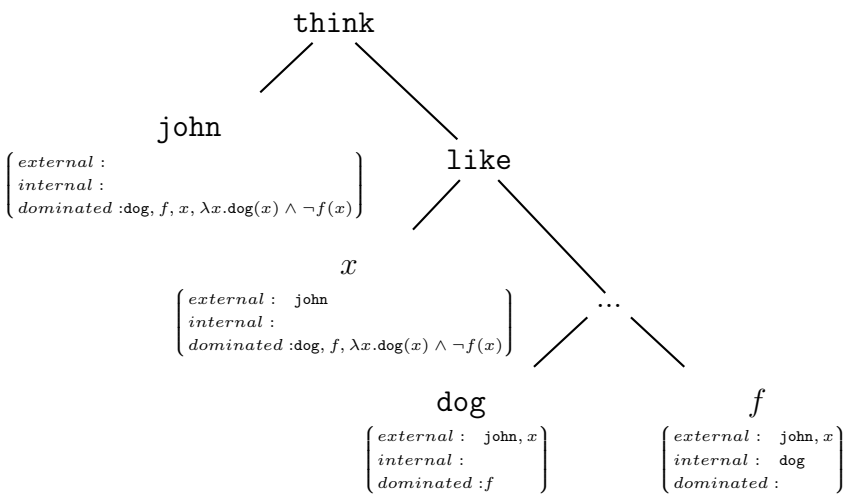
Salience is currently computed based on the ideas of centering theory (Grosz *et al.*, 1995). This produces an ordered list of possible antecedents. Resolution consists of choosing the first antecedent that is consistent with the restrictions. A simple example of restrictions are syntactic features such as gender and number.

Another type of restriction is the dominance restriction. Given a logical form, we compute three things for each portion, x, of the logical form: external dominators of x, local dominators of x, and things dominated by x. Local dominators are within the same predicate-argument structure and external dominators are outside. So, the sentence *John thinks he likes other dogs* produces the logical form in (172a) and the dominance relations in (172b). The dominance relation described here is c-command as it is described in Steedman (1996, p.19). It is basically the traditional view of c-command except that we compute on semantic, not syntactic, structures—that is, it should more properly be called f-command (Bach and Partee, 1980).

Steedman (1996) summarizes how these relations are used to restrict anaphora resolution in a theory similar to that of Chomsky (1981). Reflexives, for instance, must be bound to a local dominator, Condition A. Other anaphors are restricted by Condition B which states that they cannot bind to local dominators. All anaphors are restricted by Condition C which states that they cannot be bound by anything they dominate. Finally, referring expressions such as definite NPs cannot resolve to any dominator or anything it dominates.

(172) a. $\texttt{think}(\texttt{john}, \texttt{like}(x, \lambda x.\texttt{dog}(x) \wedge \neg f(x))$

b.



The consequence of these restrictions in the above example is that the pronoun *he* cannot resolve to dog, $\lambda x.\texttt{dog}(x) \wedge \neg f(x)$, or f. Thus, it must resolve to john or something else earlier in the discourse. f, the figure of *other dogs*, cannot resolve to dog but can resolve to anything else, including john and the referent of x.

As noted in Section 3.6.3, figures have further restrictions placed on them. *John thinks other dogs like Mary* presupposes that the figure of *other dogs* is a dog. This restriction is attached to f so that it can be used in the resolution process. Thus, when attempting to bind f to john, a check is made to see if John is a dog. If so, John is chosen. If John is not to be a dog, he is rejected. If John is consistent with being a dog and there are no salient dogs available in the discourse, John is chosen and the fact that he is a dog is accommodated. Grok does not handle cataphora, so Mary is not considered as a possible referent of f.

5.7.3 Interpretation and Hierarchical Relations

Grok maintains alternative sets as nodes in a single-rooted inheritance hierarchy with two types of relations, member and subtype. The distinction between these two relations corresponds to a general ambiguity in the interpretation of NPs in general. This is particularly relevant to the presupposition of many alternative markers that the elements of the figure have the property of the ground.

For example, I have taken names, e.g. **Netscape** and **Bushwackers**, to denote individuals. Inclusion in an alt-set therefore corresponds to being a member of its corresponding node in the hierarchy: e.g. from *browsers other than Netscape*, netscape becomes a member of browser. Bare plural NPs, e.g. **browsers**, can denote kinds. Inclusion in an alt-set here corresponds to being a subtype. For example, in *applications other than browsers*, browser must be inserted as a

subtype of the node corresponding to `application`.

As discussed in Carlson and Pelletier (1995), there is a general ambiguity in English as to whether singular NPs (definite or indefinite) should be interpreted specifically (173) or generically (174).

(173) a. The lion frightened my sister.

b. A lion walks into a bar and says to the bartender, ...

(174) a. The lion lives in Africa.

b. A lion is a powerful beast.

This same ambiguity leads to a problem for determining whether a singular NP should be represented as a member or a subtype. I have not tried to solve the problem in full generality. Rather, I use the following heuristic:

For the proposition $\forall x.f(x) \rightarrow g(x)$, g is a node of the hierarchy (a *kind*). For f, bare plurals are interpreted as kinds. Definite singulars are interpreted as individuals if an antecedent can be found in the discourse or common ground, and kinds otherwise. As above, if f denotes a kind, then f is made a subtype of g. If f denotes individuals, they become members of g.

For example, in *applications other than browsers*, *browsers* is interpreted as a kind which is therefore made a subtype of `application`. In *an animal other than the lion*, if a particular lion is in the discourse or common ground, that entity becomes a member of `animal`. Otherwise, *the lion* is interpreted as a kind and becomes a subtype of `animal`. Indefinite singulars are highly ambiguous; with alternative phrases they are interpreted as kinds, as in *an application other than a browser*.

Another issue regarding the ISA hierarchy is that a complete KB is far too large to load into memory when running Grok. The solution is to have an independent server to supply this information on demand. When Grok requires information about browsers, for instance, it sends the request to the server and receives the entry for `browser` as well as all its paths to the root. This caches relevant information in the local KB but does not make it discourse-relevant since the discourse model is a separate data structure. I assume these hierarchies will be highly branching but relatively shallow. This is supported by the hypernym structure of WordNet (Miller, 1990)—whose maximum depth is sixteen nodes. Such a system has been implemented in Grok although a taxonomy has not yet been incorporated.

5.8 The Grok System

As stated earlier, modularity is very important in building a large, complex system such as Grok. This not only makes for easier development, but it also allows many developers to work on a system. To facilitate this, we have started an OpenNLP API, a specification for NLP resources, so that people can write modules to be transparently used in the larger system. Adding the probabilistic parser described in Section 5.4.3 will be handled in this way. Some of the OpenNLP API is as follows, and more interfaces can be easily added. What I have been calling Grok is a particular implementation of the specifications in the OpenNLP API.

```
AccommodatePolicy    // Determines what should be accommodated
Category             // Syntactic and semantic forms
    Denoter          // Semantic Form
       Abstraction   // Equivalent of a lambda extraction
       FC            // A thing in the world---like an entity of event
       Kind          // A generic
       Variable      // A variable
    Synner           // Syntactic Form
Derivation           // Represents derivation information from a parse
DominanceHandler     // maintains dominance information for Denoters
Feature              // Feature information
Generator            // Translates syntax and semantics to natural lang
KB                   // Knowledge base
Lexicon              // A storage location for lexical items
Constituent          // Orthography plus syntactic/semantic info
Mouth                // An object that presents a string to a user
    Synthesizer      // Performs text to speech
Parser               // Parses a string
Rules                // Describes how lexical items combine
SalienceList         // Maintains salience information
Hierarchy            // Any sort of hierarchy- ISA, Part-Whole, ...
Pipelink             // An element in a pipeline
    Tokenizer        // Text tokenization
    POSTagger        // Part of speech tagging
    NameFinder       // Name identification
    SentenceDetector // End of sentence detection
```

5.8.1 Comparison to GATE

This philosophy of a modular NLP system to support the reuse of components is similar to that of GATE, another modular NLP system. (Cunningham *et al.* (1997) has a nice overview of GATE as well as other large natural language systems.) There are a few main differences between GATE and OpenNLP. First, GATE is a software system that manages modules, tracks data, etc. This involves installing

GATE before being able to use a system with GATE components. OpenNLP, on the other hand, is strictly an API. That is, there is no *implementation* involved, simply interfaces requiring components to provide a certain amount of support, and thus installation of OpenNLP is simply the downloading of a file—*opennlp.jar*.

Having a central system handle information produced by modules also inherently causes GATE to be biased towards certain aspects of NLP, namely text analysis:

> *For modules like taggers, parsers, discourse analyzers (i.e. just about anything that performs an analysis task) the GATE integration model provides a convenient and powerful abstraction layer based on storing information in association with the text under analysis. For resources like lexicons or corpora, no such layer exists. Similarly, for modules that do generation-side tasks, since there is no text under analysis, the utility of a text-based model is limited.*
>
> Cunningham *et al.* (1999)

Although a new version of GATE is under development to solve this problem, the OpenNLP architecture is not subject to the problem since it is simply an API. The other side of the coin is that GATE, as a full system, comes with much more support and is a mature system. OpenNLP is very new and requires a great deal more work. This brings me to the second main difference between the two systems—that OpenNLP is a free, open-source project while GATE is not. This means that anyone may enhance OpenNLP in any way as long the enhancements are made available for all. Given sufficient interest in the community, this allows very rapid and sophisticated development as has been shown by such successful projects as Linux, Apache, and Perl. At the time of this writing, three months after its first release, several people have written to say that they are using Grok for a variety of projects including topic identification in web pages, representing intonation, and writing a Turkish grammar. Grok is also being considered for use in an intelligent tutoring system, an information extraction system, and the dialogue system of the RIALIST (RIALIST, 2000) group at NASA. Grok has been downloaded 509 times, it's homepage has had 20,760 page views, and it is in the top 7% of the most active projects on SourceForge, a centralized host for more than 7000 open-source projects.

5.8.2 The Grok *Architecture*

Grok is a particular implementation of the OpenNLP API. The Grok architecture is still in flux, but Figure 5.2 represents many of the current modules and how

they communicate with each other. Function calls always follow links down the page and return values follow them up the page. The information being passed between modules is represented as labels on the links. For *type1/type2*, *type1* is the parameter and *type2* is the return type. If there is a single label, then it is both the parameter and the return type unless the arrow is uni-directional in which case there is no return type. Note that for simplicity, whenever possible I label modules by the interface they implement. However, the structure of Grok is in no way enforced by OpenNLP. OpenNLP is merely an API—a collection of conventions about how to write modules. Grok puts these modules together in a structured form.

The following discussion is a step-by-step look at Grok's major components. I focus on the parts of Grok relevant to this thesis and, throughout, I refer the reader to the sections of the thesis where these issues were discussed.

The Grok architecture is divided into four primary sections which communicate via a few external modules. The `Agent` is the primary entry point into the system. Most user interfaces to Grok create an agent, send it natural language input, and retrieve syntactic and semantic information.

Preprocessing

When the agent receives natural language input, it first sends it to the preprocessor. The preprocessor creates an XML document called an `NLPDocument` and sends it to each element, or `Pipelink`, in a pipeline. The `Pipelink` interface is shown and described on page 88. To summarize, each `Pipelink` takes an XML document and adds information to it. For instance, the `Tokenizer` would take every sentence in the document and separate them into tokens. A `Pipelink` must also specify what modules must be present earlier in the preprocessing pipeline. For instance, Grok's tokenizer requires that the document be sentence-detected first.

The output in (175b) shows the state of an NLP document for the sentence in (175a) after being modified by both the sentence detector and the tokenizer. The name finder, in (175c), adds the information that *Mary Jones* is a name. As mentioned in Section 5.4.2, this is used to relieve pressure from the lexicon where sparse data problems make it unlikely that an exhaustive list of names can be listed. In addition, parsing efficiency is improved since names are combined before the parsing process—e.g. the category NP/NP is not needed for first names to combine with last names.

(175) a. Mary Jones saw the other dog.

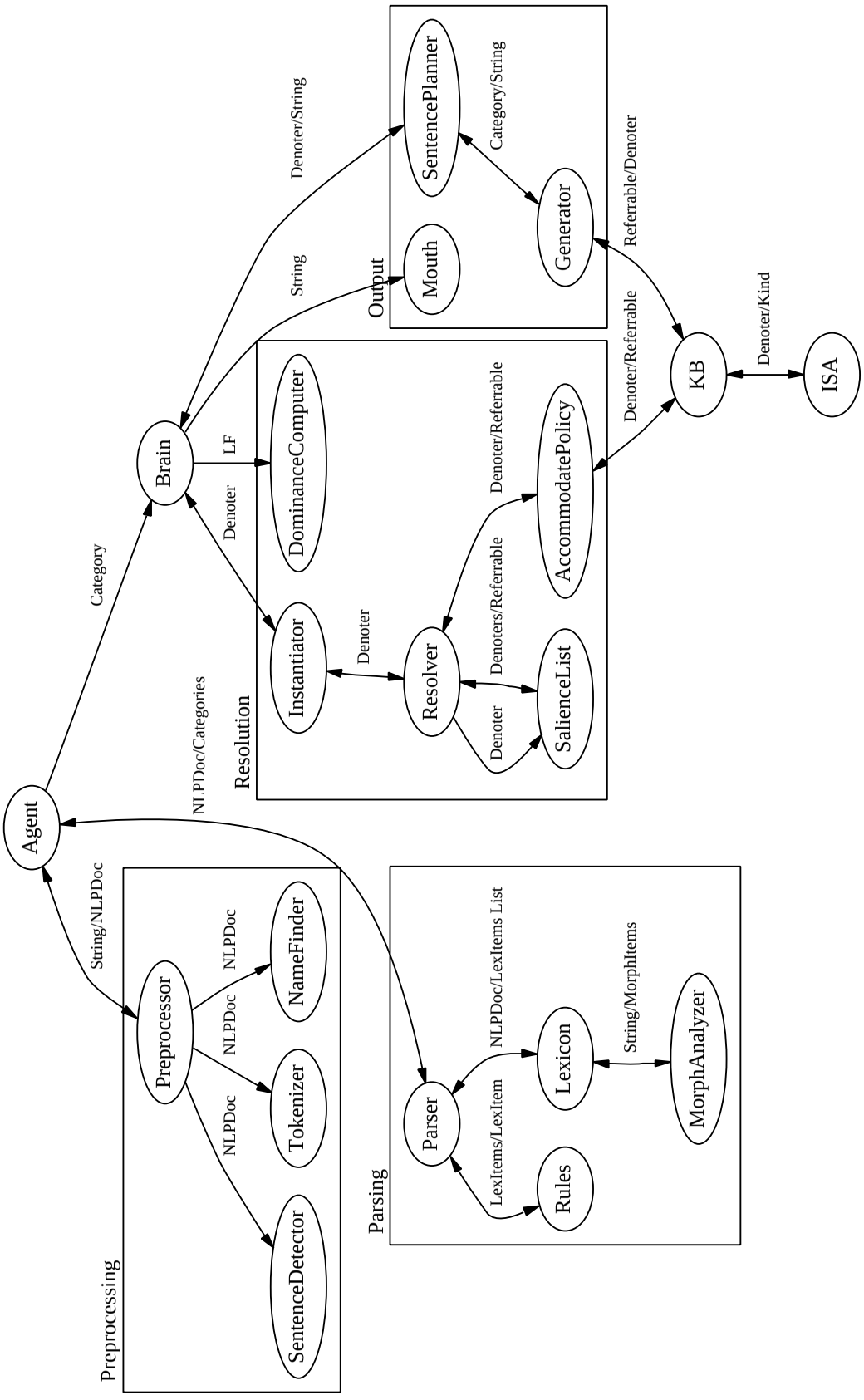


Figure 5.2: Overview of the Grok Architecture

b.
```
<NLPDOC>
  <TEXT>
    <PAR>
      <SENT>
        <TOK> <LEX>Mary</LEX> </TOK>
        <TOK> <LEX>Jones</LEX> </TOK>
        <TOK> <LEX>saw</LEX> </TOK>
        <TOK> <LEX>the</LEX> </TOK>
        <TOK> <LEX>other</LEX> </TOK>
        <TOK> <LEX>dog</LEX> </TOK>
        <TOK> <LEX>.</LEX> </TOK>
      </SENT>
    </PAR>
  </TEXT>
</NLPDOC>
```

c.
```
<NLPDOC>
  <TEXT>
    <PAR>
      <SENT>
        <NAME>
          <TOK> <LEX>Mary</LEX> </TOK>
          <TOK> <LEX>Jones</LEX> </TOK>
        </NAME>
        <TOK> <LEX>saw</LEX> </TOK>
        <TOK> <LEX>the</LEX> </TOK>
        <TOK> <LEX>other</LEX> </TOK>
        <TOK> <LEX>dog</LEX> </TOK>
        <TOK> <LEX>.</LEX> </TOK>
      </SENT>
    </PAR>
  </TEXT>
</NLPDOC>
```

For more information on the importance of preprocessing in NLP systems, see Section 5.6. The section also discusses maximum entropy, which is used to implement all of Grok's preprocessing components.

Parsing

The agent takes the NLP document returned by the preprocessor and passes it on to the parsing component. Several parsers are implemented in Grok including a standard CKY chart parser as well as the unigram probabilistic parser described in Section 5.5.

Grok parsers assume a lexical grammar and make use of two modules, `Rules` and `Lexicon`. First, the parser sends the NLP document to the lexicon which returns a list of possible lexical entries for each token in the document. Lexical

entries are created as described in Section 5.4. The parser then uses the `Rules` module to construct constituents out of the lexical entries. For the CCG implementation, the rules consist of combinators such as application, composition, and type-raising. However, an implementation of another formalism can certainly be done. For example, a TAG implementation would encode adjunction and substitution.

The parser then returns the results that satisfy certain restrictions. This is usually that the resulting syntactic category unify with $\mathsf{S}_{\mathsf{ind|int}}$. In the case of the document in (175c), the result is the bundled category of syntax, assertion, and presupposition shown in (176).

(176)
$$\begin{cases} \text{syntax:} & \mathsf{S}_{\mathsf{ind}} \\ \text{assertion:} & \texttt{saw}(\texttt{Mary Jones}, \lambda x.\texttt{dog}(x) \wedge \neg f(x)) \\ \text{presup:} & \forall x.f(x) \rightarrow \texttt{dog}(x) \end{cases}$$

Resolution

The agent now passes the results from the parser to the "brain". The brain contains all system knowledge, including discourse and world knowledge. It begins processing a parse by having a module compute the dominance relations of the semantic portion of the parse. This process, described in Section 5.7.2, places restrictions on what logical forms can refer to. The brain then uses the `Instantiator` and `Resolver` modules[3] to instantiate logical forms to referables, e.g. events, kinds, or entities.

The salience list is used to find the most salient entity that satisfies the restrictions of the logical form. Presuppositions are resolved before the assertion, so for (176), $\forall x.f(x) \rightarrow \texttt{dog}(x)$ would be resolved first. There are two items to resolve in this logical form, f and `dog`. The first thing the resolver does is to add the restriction that f be consistent with the property $\lambda x.\texttt{dog}(x)$ as described in Section 3.6.2. Assuming that the salience list has the entities in (177), the first entity consistent with f is John_1. (Referables are indexed by a unique number. The preceding descriptive text is purely for readability.)

(177) [John_1, Mary_2]

As discussed in Section 5.7.3, `dog` is interpreted as a kind. However, our discourse does not contain the kind `dog`. At this point the resolver must decide whether it should create, or accommodate, such a kind. The `AccommmodatePolicy`

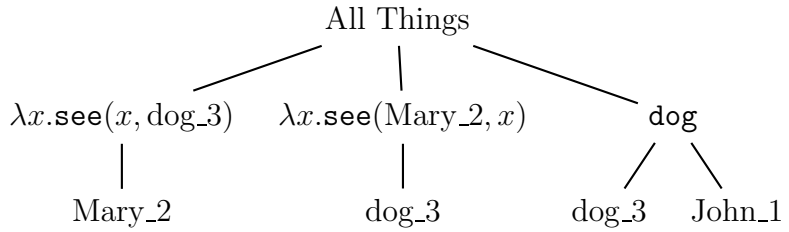
[3]These modules are so closely linked that they will most likely become a single module in a future version of Grok.

module contains information about what sorts of referables may be accommodated. In our implementation, kinds can be accommodated but, for example, antecedents for pronouns cannot.

When evaluating the assertion, f and `dog` have already been resolved. `Mary Jones` is resolved to the entity Mary_2 since it is the first consistent entity on the salience list. The only thing remaining is the entity referred to by $\lambda x.\texttt{dog}(x) \wedge \neg f(x)$) (i.e. a dog that is not John). Such an entity does not exist on the salience list so, again, the `AccommodatePolicy` module is consulted. The accommodation is allowed and so a new dog is created. A new event is then created for the entire proposition, shown in (178a). The resulting ISA hierarchy now includes the new information in (178b).

(178) a. $\texttt{see}(\text{Mary\_2}, \text{dog\_3})\text{\_4}$

b.



Output

These components control how information is transmitted back to a user of Grok. The brain might desire to communicate with the user for a number of reasons. First, its input might have been a query, in which case the brain would communicate the answer. The brain might also communicate a message when there is a problem. For instance, if the sentence *She likes spam* is given in isolation, Grok communicates `I do not know who you are referring to.`

The brain communicates with the outside world through `Mouth` modules. The mouths implemented in Grok are a text field in the user interface and a connection to the Festival speech synthesis system (Black and Taylor, 1997). Thus Grok can produce both text and speech output.

This segment of Grok is not as relevant to this thesis as the others, but the `Generator` module is used in Algorithms 1 and 2 in Chapter 6.

Categories

In my discussion of Grok's architecture, I have mentioned several data structures: categories, referables, kinds, events, variables, etc. These are sub-types of the `Category` interface.

Figure 5.3 shows the hierarchical structure of these categories. These are the syntactic and semantic data structures passed throughout much of the system. Here, interfaces are contained in boxes and classes, in ovals. Dotted lines indicate the implementation of an interface while solid lines indicate the subclass relation. I include this diagram as an example of how Grok and OpenNLP interrelate. Here, all interfaces (rectangles) are module specifications in OpenNLP and all classes (ovals) are implementations of those interfaces in Grok. This particular implementation encodes CCG categories, but this is hidden from many modules. For instance, the Grok parser only relies on OpenNLP interfaces and not on any particular implementation of them. Therefore, it would readily accept an implementation of TAG, for instance, instead of CCG without any change.

More Information

For a more detailed discussion of the modules and what they do, see the Grok and OpenNLP homepages:

http://grok.sourceforge.net
http://opennlp.sourceforge.net

5.9 Conclusion

This chapter has discussed a number of aspects of producing a robust, wide-coverage system that will support the analysis of alternative phrases presented in Chapter 3. This is both a contribution to the community and a bridge between the analyses and their relevance to practical applications discussed in Chapter 6.

First and foremost, I discussed ongoing research in developing a large English lexicon as a combination of an acquired lexicon and a lexically incomplete, hand-built one. The hand-built lexicon is meant to provide precise semantics and syntax for closed-class sets of words and semantics for open class words acquired through techniques described in Hockenmaier (2000) and Hockenmaier *et al.* (2000). I then described how this wide-coverage lexicon is embedded in a larger system that supports parsing, generation, and knowledge representation.

Preliminary experimentation with the system supports the belief that a well-designed, modular system can produce satisfactory results even when the modules themselves are sometimes fast and simple solutions for complex problems. Furthermore, with a well-designed system, these modules can be easily and transparently replaced with more interesting modules to produce, hopefully, better results.

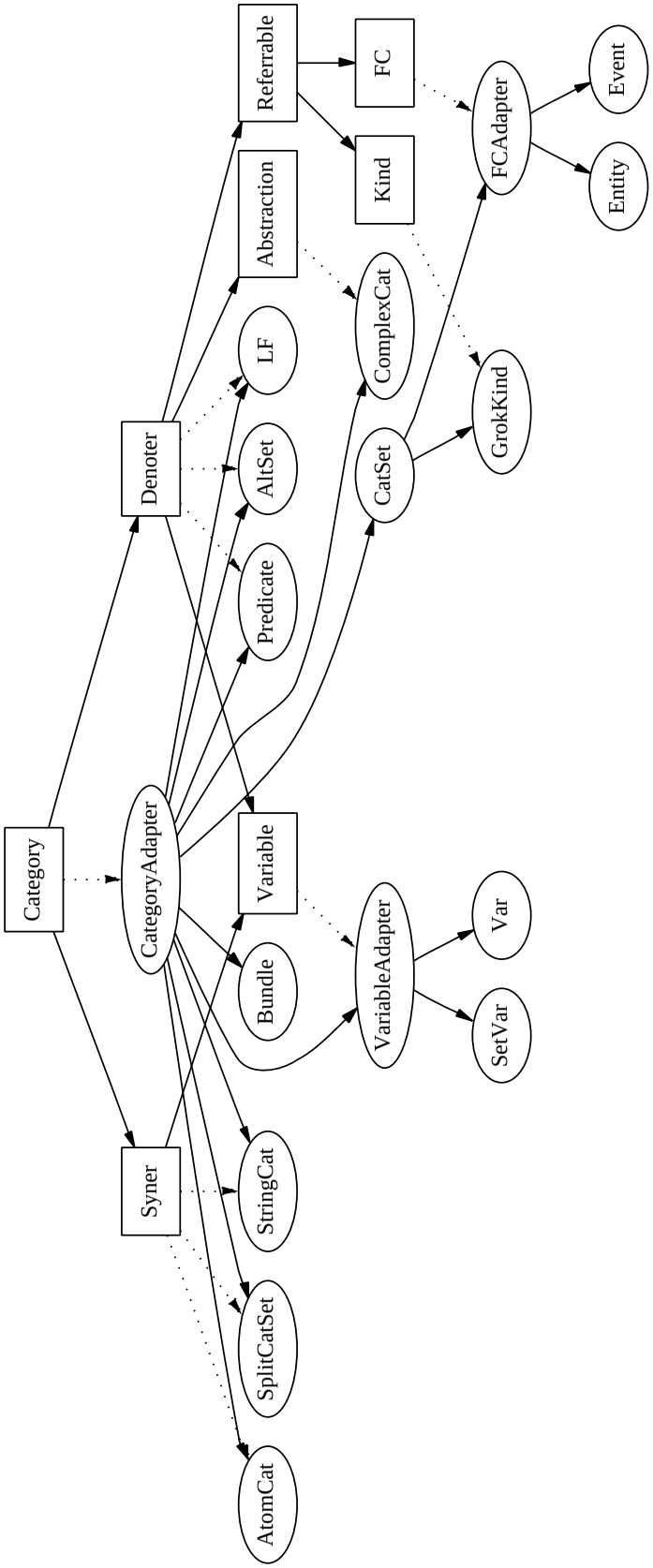


Figure 5.3: Grok's Category Hierarchy

Importantly, at this time, the system successfully supports most of the work presented in Chapter 3. I do not handle certain aspects of free alternative phrases (common-sense reasoning), but the implementation is more than sufficient for the discussion of practical application in Chapter 6.

Chapter 6

Practical Application

In Chapter 3 and Chapter 4, I presented an in-depth analysis of a class of frequently-occurring words I call alternative markers. In Chapter 5, I showed that this analysis can be implemented within a large system. In this chapter I argue that there are practical benefits to this, that there are applications in which understanding alternative phrases is valuable.

Any application where type information about referables is useful would benefit from the work in this dissertation. For instance, Hearst (1992) showed that these constructions are useful for finding hyponyms, and thus for automatic thesaurus construction and similar tasks. Also, named entity classification would benefit in cases where alternative phrases disambiguate people, companies, countries, etc. A related task is information extraction, where data is collected from a document and used to fill a template. Alternative phrases can indicate what slot in the template an entity should fill. Coreference also needs to take alternative phrases into account to avoid, for example, the phrase *other X* coreferring with the original X: e.g. *other dogs* probably does not corefer to the golden retrievers mentioned in the previous sentence. Furthermore, knowledge of alternative phrases helps identify coreference in cases like (179) where *the other substance* corefers with *spam*.

(179) Spam and beef are quite different—beef comes from cows while the origin of the other substance is unknown.

One need not use the full extent of the analysis presented in this thesis for these tasks. However, this thesis points the way to how one might implement simpler instantiations. For example, while pattern recognition can be effective to a certain point, it becomes insufficient by itself for examples containing anaphora and examples with free alternative phrases where there are many possibilities for the ground or complement. Furthermore, sophisticated noun phrase detection is

necessary for pattern matching techniques to be at all effective, as evidenced by the patterns given by Hearst in (34).

Another important application, one that inspired this thesis, is information retrieval systems. With the large and rapidly increasing amount of information available in databases and the World Wide Web, easy and accurate information retrieval is (and has been for many years) a heavily researched area. Much of this research has been directed towards the back end of information retrieval systems—that is, given a query in a convenient format (such as Boolean algebra), finding the answer or a set of relevant documents. Now that millions of people have regular access to these information sources, researchers are considering the problem that most users have never seen a Boolean expression. How, then, can a system allow an average user to create effective queries that can be used by these back end systems? One answer is to allow users to use what they know best—natural language.

Many systems exist which accept natural language queries, and the techniques they employ work to varying degrees of effectiveness for single, non-discourse embedded queries. However, they do not take into account alternative phrases, which people use to narrow the search space or more precisely identify the information they want when normally formulating one-shot or follow-up questions.

Applying this research to NLIR amounts to increasing the amount of information in a query recognized as relevant and to use that information correctly. `"web browsers" AND netscape`, for example, is not an appropriate search request for *What are some web browsers besides Netscape* and yet that is what many systems produce. With the approach presented in this dissertation a more reasonable query can be formed.

This chapter begins with a discussion of NLIR systems and shows that they fail to account for alternative phrases. I then show how this dissertation can benefit NLIR systems, taking The Electric Monk as an example.

6.1 Why Natural Language?

The purpose of applying my analysis of alternative phrases to natural language information retrieval is to show the importance of this class of words and to prove that my analysis can support real world applications. However, it is important to consider why natural language is important to IR in the first place. Why are keyword-based Boolean queries not enough?

The first, simple, reason is that not all users are willing or able to learn how

to express Boolean queries. Small children are a good example, as indicated by the following hearsay evidence from a user of The Electric Monk:

> *As an early elementary school teacher I love your site. It makes it so easy for young students. No keywords!! which I've had trouble trying to explain. I'm passing your address on to other teachers and parents.*
> Monk (1999)

Search engines often attempt to simplify the use of Boolean queries by implicitly conjoining the set of keywords entered by the user. However, conjunction is not always sufficient, and the user must learn how to express concepts like negation. This is complicated by the fact that there is no external force requiring IR systems to use consistent notation for Boolean expressions. For instance, negation is often expressed with the minus, -, symbol but can also be expressed with the reserved word `NOT`. Documentation describing the notation is often extensive. Natural language, on the other hand, is, well, natural. Lewis and Sparck-Jones (1996) point this out as an important aspect of the text retrieval agenda. Referring to "Boolean logic or other user-befuddling query syntax"[1], they claim that "users should not be required to express their needs in a heavily controlled and highly artificial language."

While I believe it is valid, the above argument is mostly an expression of the intuition many people have about NLIR. A more precise argument is that the anaphoric nature of natural language allows it to be much more terse than Boolean expressions. The simple use of pronouns (as in Section 6.3) is one example. A more interesting example is the anaphoric reference of **other** in example (1a), above. Questions are not asked in isolation, and being able to refer to previous discourse can greatly simplify queries.

Finally and most important, Boolean algebra simply cannot express some perfectly normal queries. For instance, how might one express *...other web browsers than Netscape...*? One possibility is `''web browser'' AND NOT Netscape`. This, however, will exclude all pages with the string **Netscape**. Since relevant pages probably list a number of web browsers, including Netscape, this is undesirable. An better approximation can probably be achieved using the `NEAR` operator provided by many document IR systems, but this is still not really the intended meaning of the original sentence. I discuss this issue in relation to the Monk in Section 6.4.

[1]The analysis in this paper, incidentally, would infer from this that Boolean logic is user-befuddling query syntax.

Another example, a slightly modified form of (1c), is *Where can I find shoes such as Bushwackers?* (discussed in Section 3.4.2). This requests shoes that have some set of properties in common with Bushwackers (that they be comfortable, for instance). In this particular case, a Boolean query could include the relevant keyword, **comfortable**, but the purpose of this construction is to express this information through example, possibly meaning that the user cannot, or does not care to, express the properties explicitly.

It is true that document retrieval systems, in general, do not support an operational semantics for expressing these queries any better than Boolean algebra. This is because they do not know enough about the meaning of the documents. However, other retrieval systems support a more precise operational semantics. It makes sense, then, to abstract away from the operational semantics of the retrieval system. A user should at least be able to express the full meaning of their query in a front-end system (natural language being the obvious choice), and then that query can be translated into the most precise form that the back-end retrieval system can handle. For many queries, this will already be better than what can be expressed in Boolean algebra. Furthermore, as particular retrieval systems improve their operational semantics, the results of these queries can be improved in a way that is transparent to the user.

6.2 Previous Work

There are many different levels of sophistication in automatically understanding natural language queries. These approaches include key word extraction, machine learning, pattern recognition, and full parsing. In this section, I briefly review and describe an implementation of each of these techniques.

To give a basic idea of how these systems perform on the queries that this dissertation has been interested in, I show the results of these systems for the query, *What are some states besides Texas?*. The domain of this query was dictated by my test of Geoquery, a natural language front end to a geographical data base. Note that it is intrinsically unfair to judge these systems based on their response to this question, as some are designed to perform well for different types of questions and have varying degrees of coverage. The test is merely to give a general sense of how they behave.

6.2.1 Keyword Extraction

Keyword extraction is by far the simplest of the techniques. It makes no attempt to understand the query. Instead it removes all function words from the query and uses the conjunction of the remaining words as a Boolean query. Of course, the words we have been interested in (**other** (**than**), **such** (**as**), **besides**, etc.) are often the very words ("stop words") that are removed.

Consequently, systems that use keyword extraction usually prefer pages with the very items that are meant to be excluded. An example of such a system is Alta Vista (Alta Vista, 2000). Below are the results of the query *What are some states besides Texas?*. In all cases, even where it cannot be seen in the summary, *Texas* appears in the page. In fact, it is worse than this because the results *only* refer to Texas and no other state.

(180)
```
1. No Title
   BACKGROUND: I was born on October 4, 1979...
2. State and Local Immigration Issues
3. billhicks.com
4. Re: Tax Lien Certificates advice.
5. www.artec-usa.com/San Francisco Chronicle
   What They Said About the Meyerson Symphony Center,
   Dallas,Texas.
6. No Title
   Confessional Study Groups. around the state of Texas.
7. About Me
   A Little About Me...  I am originally from El Paso, Texas
8. Leigh and Dan
   ...Our incredible hosts (and tourguides) in Austin, Texas.
```

More sophisticated techniques can be, and are, incorporated into keyword extraction: stemming, synonym and hyponym expansion, special analysis of the source documents, etc. However, none of this overcomes not recognizing a user's desire to exclude a certain set of answers from the results.

6.2.2 Pattern Recognition

The Electric Monk uses pattern recognition to attempt to classify a natural language query. It is not interested in understanding the question, *per se*, but rather tries to understand enough about the question to be able to transform it into a series of increasingly general Boolean queries that can be used with standard back-end retrieval techniques. The query is filtered through a hierarchy of regular expressions until it reaches the most general. At this point, hand-crafted rules are used to extract information from the query into Boolean expressions. The

regular expression matching is aided by an ISA hierarchy such that generalizations of keywords can be captured. For instance, if the query refers to a golden retriever, the Monk will successfully match against patterns about animals.

This technique suffers from the fact that, in order to be tractable, only a certain number of sentence types can be treated. Thus, important information in the query is often lost in the process. This can be seen in (181) where, as before, all results refer to Texas (although sometimes, states besides Texas as well).

(181)
```
State and Local Immigration Issues
  Immigrants tend to concentrate... Yet many states
  besides California, Texas and Florida are facing a
  major challenge from immigration.

Sound
  Besides being one of the continent's biggest under-rated
  concert attractions, all four of ZZ Top's albums...

WER: Wisconsin and Its Resources [Chapter 3]
  In regard to the value of improved lands in the new
  States... the same report shows that the average value
  is: in Illinois, 7.99; in Iowa, 6.09; in Texas, 1.09;

Complete Book: "PRESUMED GUILTY
  Roffman in this book states the charge explicitly: "When
  the Commissioners decided in advance that the wrong man
  was the lone assassin, whatever their intentions, they
  protected the real assassins.
```

6.2.3 Parsing

Other systems attempt to perform a full syntactic/semantic parse of the query so that relevant information can be extracted and used for the information retrieval stage of the system

Ask Jeeves is such a system. At this time, I do not know exactly how Ask Jeeves performs its analysis except for the following information, which, in all honesty, could mean anything:

> *Ask Jeeves attempts to understand the precise nature of the question by using a question-processing engine. Using natural language processing technology, Ask Jeeves determines both the meaning of the words in the question (semantic processing) as well as the meaning in the grammar of the question (syntactic processing).*
>
> Ask Jeeves (2000)

After the analysis, Ask Jeeves compares the result to a database of questions for which an answering page has been found by hand. These questions are then presented to the user. The results for our sample sentence (182) shows that their grammar does not appear to account for alternative markers like **besides**.

(182)
```
You asked:   What are some states besides Texas?
I know the answer to the following questions.
  Click the Ask! button next to the best one.

Where can I find the newspaper Abilene Reporter News?
Where can I find a landform map of the state Texas?
Where can I find government...  for the state Texas?
Where can I find the official Web site for cities in Texas?
Where is/are Texas?
```

While matching against hand-indexed queries might perform well for small domains, it is unclear how well it scales. A technique they seem to use is to have a small number of general pages that can answer a large number of questions. Consequently, questions with alternative phrases are sometimes accidentally answered appropriately because of the generality of the matching questions.

Parsing systems such as Ask Jeeves would not be an appropriate type of system to use Grok because Grok itself parses and therefore the work would be duplicated. However, if the system uses a lexical grammar, it is possible that it could incorporate the analysis of alternative phrases that Grok uses.

6.2.4 Machine Learning

There are also machine learning techniques for understanding natural language queries. There are several systems based on CHILL (Zelle and Mooney 1993). CHILL takes as input sentences and their associated semantic representation. It then produces a shift-reduce parser that performs the mapping from sentences into semantic representations. This system has been used for the more specific purpose of mapping natural language questions into database queries, in particular in a U.S. geography domain (Zelle and Mooney 1996) and a job search domain (Thompson *et al.* 1997). For Geoquery (Geoquery, 2000), our sample query produces the results in (183).

Obviously, this is the incorrect result. But this is perhaps not the fault of the methodology and is simply due to the lack of training data. But this pinpoints the general problem with supervised learning techniques—that they are intrinsically tied to a particular domain.

(183)
```
YOUR QUESTION:
What are some states besides Texas?

RESULT:
TEXAS

LOGICAL QUERY:
ANSWER(_2463,(STATE(_2463),CONST(_2463,STATEID(TEXAS))))
```

I have several goals regarding information retrieval, but none of them are meant to replace the methods described in this section. Instead I intend to show how these methods can be augmented by a front-end, such as Grok, that implements the ideas in this dissertation.

It is possible that some of the techniques in the previous section could implement some of the ideas in the dissertation. For instance, it is possible for machine learning techniques, like those built on CHILL, and for parsing techniques, such as Ask Jeeves, to understand some alternative phrases.

My approach is to provide a front end that converts a query to something a back-end NLIR system understands. This sometimes involves producing a query that is a hybrid between natural and artificial language. I will be using the Monk as my back-end system.

6.3 Anaphora Resolution

Given a system like Grok, it is natural to want to use its discourse capabilities for an NLIR system. Handling anaphoric reference is certainly one way of doing so. A certain amount of this (pronouns, reflexives, and reference in presupposition) has already been implemented in Grok. This section should be read as a gentle introduction into the next section which discusses alternative phrases.

I have already described in Section 5.7.2 how Grok performs anaphor resolution. The question, then, is how to use this information to produce a new query with the anaphors filled. One possible solution is to generate the sentence from scratch given that we have the logical form as well as the anaphoric referents. However, this is time-consuming and not necessary. Instead we use the derivation from the original parse. Consider the simple discourse in (184).

(184) Who is Madonna?
(search results)
What does she wear?

At this point, it must be reiterated that semantics in Grok are handled, not with the lambda calculus, but simply through unification in exactly the same way as the syntax. For example, whereas the analysis for **likes** was expressed as $\mathsf{S\backslash NP/NP} : \lambda x\lambda y.\mathtt{like}(y,x)$ in Section 2.2.2, it is really represented in Grok as $\mathsf{S} : \mathtt{like}(y,x)\backslash\mathsf{NP} : y/\mathsf{NP} : x$. Thus, (185) shows the derivation of *What does she wear?* as it is actually represented in Grok. This is important to keep the matching simple and to not have to employ higher-order unification techniques on lambda expressions.

After the parse, in addition to the derivation, we know how anaphoric references, e.g. *ana*, were resolved. In this case, we have $\{ana = madonna\}$. Our algorithm for producing the new query is to recursively descend the derivation, producing orthography at the leaves, and if a variable in our resolution set is encountered, we generate a description of the antecedent rather than the orthography of the anaphor. The algorithm is shown in Algorithm 1. Given, *What does she wear?* and the table given above, it will produce the string *What does Madonna wear?* since **she**'s semantics is found in the table and will therefore be replaced by *Madonna*. This can then be sent to the Monk (or any other NL search engine).

(185)

what	**does**	**she**		**wear**
$\mathsf{S}:(p\|x)/(\mathsf{S}:p/\mathsf{NP}:x)$	$\mathsf{S}:x/\mathsf{S}:x$	$\mathsf{NP}:ana$		$\mathsf{S}:\mathtt{wear}(y,x)\backslash\mathsf{NP}:y/\mathsf{NP}:x$
		$\mathsf{T}:p/(\mathsf{T}:p\backslash\mathsf{NP}:ana)$ (>T)		
		$\mathsf{S}:\mathtt{wear}(ana,x)/\mathsf{NP}:x$ (>)		
	$\mathsf{S}:\mathtt{wear}(ana,x)/\mathsf{NP}:x$ (>B)			
$\mathsf{S}:\mathtt{wear}(ana,x)\|x$ (>)				

Algorithm 1 (Anaphor Filling)
resolved = mapping from anaphors to their antecedents
function *query-xform(Constituent constituent)*
  if *resolved.containsKey(constituent.semantics)*
    return generate(resolved.get(constituent.semantics))
  else if *constituent.derivation.isEmpty()*
    return constituent.getOrthography()
  else
    answer = ""
    for-each *l:constituent.derivation.children*
      answer = append(answer, query-xform(l))
    return answer

6.4 Alternative Phrases

6.4.1 *The* Monk*'s Operational Semantics*

The previous section showed how pronouns in queries could be filled before the query was sent to a back-end retrieval engine. No special operational semantics is required from the retrieval engine to accommodate this transformed query.

This is not the case when transforming queries with alternative phrases. My approach to this task is to remove the alternative phrase from the query and present it independently of the resulting query in a way the back-end retrieval engine can understand. The retrieval engine must have some operational semantics for allowing this. For The Electric Monk, this is done with a hybrid query which combines a natural language query with further restrictions added in a system-specific language. The syntax is shown in (186a).

(186) a. NL Query :|: `ANSWER` [`NOT`|`PREF`] `NEAR`_num (| word list)

b. What are some web browsers? :|: `ANSWER NOT NEAR`_8 (| netscape)

The natural language query is separated from the restrictions by the|: symbol. The restrictions specify that the answer to the query must not be (or preferably be) `num` words near to certain words.

The hybrid query in (186b), for example, is a transformation of the original query *What are some web browsers besides Netscape?*. The Monk uses the natural language part of the query to initially locate possible answering documents. The rest of the query is used when gathering evidence for including a document in the final results. The Monk finds a location in the document that should answer the query and then compares it against the criteria appended to the end of the query. If it does not meet the criteria (that it not be within eight words of Netscape), another location is tried. If there are no more possible answers, the document is rejected.

This is, of course, not exactly what the original query meant. However, it is superior to queries like `‘‘browsers’’ AND NOT ‘‘netscape’’` which rejects all pages containing Netscape, even if they also contain other browsers. The evaluation in Section 6.6 shows that this operational semantics is sufficient to dramatically improve the results for queries with alternative phrases.

6.4.2 *The Algorithm*

The algorithm to produce these hybrid queries is similar to Algorithm 1. It is a postprocessing of the derivation created when parsing the sentence with the full

analyses given in this thesis. In addition to searching for anaphors and filling them, the algorithm searches for Boolean operations that result from alternative phrases and performs a simplification. For example, with excision words like **other** (**than**) and **besides**, we look for logical forms whose predicate is the $\wedge\neg$ operator, as in derivation (188) from the discourse in (187).

(187) What is the drinking age in Afghanistan?
(search results)
What is the drinking age in other countries?

(188)
$$\dfrac{\dfrac{\textbf{other}}{\mathsf{NP}:\lambda x.g(x)\wedge\neg f(x))/\mathsf{NP}:g} \quad \dfrac{\textbf{countries}}{\mathsf{NP}:\texttt{country}}}{\mathsf{NP}:\lambda x.\texttt{country}(x)\wedge\neg f(x)}>$$

Given such a logical form, in general $\lambda x.g(x) \wedge \neg f(x)$, we instantiate all anaphoric variables to their antecedents and then generate descriptions for f and g. Note that these anaphoric variables include the reference to the figure. Section 3.6 describes and evaluates how these antecedents are found. The first argument of the logical form is returned and the second argument is saved to be attached to the final string as extra information. In this example, "countries" would be returned and "Afghanistan" would be saved. The resulting query is shown in (189). Algorithm 2 is an extension of Algorithm 1 that does this.

(189) What is the drinking age in countries? :|: `ANSWER NOT NEAR` (| Afghanistan)

The natural language portion of this query is odd, but The Electric Monk's regular expression matching is not sensitive to this level of detail. The rest of the information in the query is used to return documents that contain answers that are not located near an occurrence of the word *Afghanistan*. Providing extra information in this way is very coarse, but being more specific causes problems. In this example, for instance, we should not require that the answer not be *equal* to Afghanistan since the answer will actually be a number—i.e., drinking age—not a country. This is why we use the more general `NEAR` operator.

Algorithm 2 (Simplifying Alternative Phrases)
resolved = mapping from anaphors to their antecedents
function *query-xform-help(Constituent constituent, Set notNear)*
    if *resolved.containsKey(constituent.semantics)*
        return generate(resolved.get(constituent.semantics))
    else if *isAndNot(constituent.semantics)*
        sem = fill(constituent.semantics, resolved)

```
        notNear.add(generate(sem.arg2))
        return generate(sem.arg1)
    else if constituent.derivation.isEmpty()
        return constituent.getOrthography()
    else
        answer = ""
        for-each l:constituent.derivation.children
            answer = append(answer, query-xform-help(l))
        return str

function query-xform(Constituent constituent)
    notNear = {}
    answer = query-xform-help(constituent, notNear)
    return append(answer, " | ANSWER NOT NEAR ", notNear)
```

6.4.3 Other Alternative Phrases

The above technique describes how to transform queries that use excision words—e.g. **besides**, **other**, **except**, etc. For questions with these alternative markers, the algorithms given above provide a way to find appropriate answers (see Section 6.6 for evaluation) without world knowledge. This can be more difficult with other alternative markers.

Such

The two varieties of **such**, for example, do not have a simple approximation in the limited operational semantics available in most NLIR systems. Non-restrictive **such**, as in *Where can I find shoes made by Buffalino, such as the Bushwackers?*, is used to give a representative example of the noun phrase it modifies. Such information might be used in NLIR systems with clustering techniques. That is, if *Bushwackers* is a representative example, it might be useful for the search to include other words that tend to appear near it. Without this information, the best that can be done is to pay special attention to pages which contain *Bushwackers* in the hope that other examples will appear on the same pages. Since most NLIR techniques already use the figures of alternative phrases as keywords, this will happen without any special processing.

The restrictive version of **such** requires even more world knowledge. In the query *Where can I find such shoes as the Bushwackers?*, it is necessary to identify the property, exemplified by *Bushwackers*, that the speaker is interested in: e.g. that they are made by Buffalino, are comfortable, etc. Sometimes this is available in the context, in which case it can be included in the query. More generally,

though, this requires a far more extensive knowledge base than is usually available.

Despite the difficulties these queries present, it is at least the case that an NLIR system can still learn world knowledge from them that can be used to better answer future queries (see Section 7.4.2).

Comparatives

Like restrictive **such**, comparatives require a great deal of world knowledge. For example, in the question *What are some bigger dogs than poodles?*, we are unlikely to be able to properly exclude all dogs smaller than poodles, but we can at least exclude poodles themselves. Unfortunately, a user will most likely be more surprised by finding a chihuahua in the results than pleased at not finding a poodle. However, as is evident in the evaluation in Section 6.2 and Section 6.6, standard NLIR techniques would return pages almost *exclusively* about poodles. Simply not making this mistake is an important part of handling these queries correctly.

In Chapter 7, I discuss the possibility of learning some of the information necessary to answer these questions more effectively. Such learning might be possible with questions like *Other than chihuahuas, what are some dogs smaller than poodles?*.

Specifiers

In contrast to the previous examples, alternative markers like **especially** and **in particular** are easy to handle. An example sent to The Electric Monk is *Tell me about religious cults, in particular Waco.* This indicates a preference for pages with *Waco*, although it does not exclude other pages about religious cults. This sort of information can be conveyed to the Monk with the `PREF NEAR` operator which indicates that the answer to the query preferably be near the figure, e.g. *Waco*.

This query is also a good example of when learned information may not be trustworthy. Waco is not a religious cult; rather it was the location of a religious cult. I discuss this further in Section 7.4.2.

Unlike, In addition to, etc.

Alternative markers such as **unlike** and **in addition to** are mostly useful in conjunction with other alternative makers to help identify the figure, as described in Section 3.6.1.

It might also be possible to use these phrases more directly. Consider the ques-

tion *What presidents, in addition to Lincoln, were assassinated.* The knowledge that Lincoln was assassinated could help point to pages about other assassinations. But how this might be done is very unclear. It is far more useful to not handle the alternative phrase *incorrectly.*

As I discussed in relation to comparatives, standard NLIR techniques tend to use the elements of alternative phrases as keywords. Simply excluding *Lincoln* from the search is a large improvement over this.

6.5 A Note on Implementation

Most of the ideas discussed in this thesis have been implemented, but some have not. In general, though, I have endeavored to ensure that everything relevant to NLIR is implemented. In particular, sentences like those in (1) are successfully parsed by Grok while accommodating the correct presuppositions.

Connected Alternative Phrases

Nearly all aspects of connected alternative phrases discussed in Chapter 3 have been implemented. This includes the various techniques used to identify the figure with alternative markers that depend on anaphora, e.g. **other** and **such**. The grammar only includes entries for **other**, **such**, and **besides**, but adding more is simply a matter of time rather than any theoretical issue.

I do not implement the reasoning portion of my discussion of scoping in Section 3.5 or the discussion of determiners in Section 3.7.

Free Alternative Phrases

This aspect of the theory is far less implemented. Entries are included in the lexicon for common examples like **other than** and **unlike**. However, I do not implement their full analyses. In particular, I do not implement the commonsense reasoning used in parts of the analysis as it is too tangential to the thrust of the thesis.

Query Transformation and The Electric Monk

I have implemented a program which performs Algorithm 2, communicates the resulting query to the Monk, and displays the final document set in a web browser. The implementation is restricted to handling excision alternative phrases like **another**, **other**, and **besides** because they are, by far, the most common. They account for 94% of the alternative phrases shown in Table 6.1. The issues in-

volved in implementing support for other alternative phrases are discussed in Section 6.4.3.

6.6 Evaluation

It is necessary to evaluate how well the techniques described in this chapter actually improve results for information retrieval.

6.6.1 Frequency of Alternative Phrases

First, it is useful to determine how many queries contain alternative phrases in order to judge how large a problem this really is. Unfortunately, this is complicated by the fact that users, in general, *know* that such constructions are not understood by search engines, so they avoid them. In fact, even in NLIR systems, users often use keywords even though doing so performs worse than asking natural language questions. They do this because they do not trust the system. Work will be necessary to improve users' awareness of NL capabilities through advertising and by implementing new user interfaces. A simple idea in this vein is to show a history of questions that have been asked so that the user is more aware that the system is remembering. Also, more accurately providing the question's answer in the document summary will allow users to avoid following too many links and losing track of the discourse. The fact that searching is an interactive process is often ignored, even in large-scale document-retrieval competitions such as the Text Retrieval Conference (TREC, 2000). As noted by Lewis and Sparck-Jones (1996), these results then do not necessarily reflect the experience many users have with retrieval systems.

In the meantime, I have attempted to find a baseline for this number by considering two corpora of human/human dialogues. The corpora are both from tutorial situations where a mentor helps a student through problems in the subject of Physics for one corpus (VanLehn *et al.*, 1998, in press), and Basic Electricity and Electronics (Rose *et al.*, 1999), for the other. Tutoring dialogues are an ideal place to look for data relevant to NLIR because they consist entirely of one party attempting to elicit information from the other. In some cases, it is the tutor eliciting information from the student, and in others it is the other way around.

Table 6.1 shows the frequencies of some alternative phrases that have been discussed in this thesis. A more illuminating statistic is how often alternative phrases appear in a single dialogue. I consider a dialogue to be a single *session*

	Physics		**EE**		**Total**
Alternative Phrase (AP)	Student	Tutor	Student	Tutor	
in addition to	0	0	0	3	3
besides	1	3	0	2	6
another	56	124	10	38	228
especially	0	1	0	0	1
except	0	17	1	1	19
other	107	484	36	59	686
in particular	0	6	0	0	6
such	2	18	3	10	33
unlike	0	0	0	1	1
Total	166	653	50	114	983
Alternative Phrases per Dialogue					
Dialogues	203	203	66	66	269
AP/dialogue	0.82	3.22	0.76	1.73	3.65
Alternative Phrases in Queries per Dialogue					
Query APs	51	261	16	86	414
Query AP/dialogue	0.25	1.29	0.24	1.3	1.54

Table 6.1: Frequency of Alternative Phrases in Dialogue

between the student and tutor where the discussion of each problem is considered a separate session. The table shows that in 269 total dialogues, each dialogue contained, on average, 3.65 alternative phrases. If one only considers alternative phrases that occur in the context of a question, there are, on average, 1.54 alternative phrases per dialogue. I consider an alternative phrase to be in a question context if it is in the same dialogue turn as a question. Because tutors ask questions in order to lead the listener to the answer, it is perhaps better to consider just the student data, where a quarter of the dialogues contained question contexts with alternative phrases.

This data is not meant to be considered a rigorous result, but it is a strong indication that any query-answering system has an excellent chance of having to deal with an alternative phrase during the course of interacting with a user. Furthermore, the data shows that during the course of the interaction it will be appropriate for the system to *respond* using an alternative phrase—see Section 7.3 for more on this.

It is also interesting to note that a wide variety of alternative phrases occur in this data. (190) contains some examples. Because excision words, especially **other**, are by far the most frequent, I will only consider them in the evaluation

of the next few sections.

(190) a. The battery is the green cylinder right? I don't see anything negative **other than** #5.

b. I guess until 0 or should I examine negative values...is there **such** a thing?

c. And what do you have **in addition to** voltage?

d. I don't exactly remember what the letters stand for, **except for** r.

e. Are there any **other** forces on that knob **besides** that one you've labeled W1?

6.6.2 Potential Improvement

I now informally test the potential for improvement for various search engines. That is, given a set of queries (cleaned up from The Electric Monk and listed in Appendix A) containing alternative phrases, how well do these systems perform now? Table 6.2 shows the performance of Alta Vista, Ask Jeeves, and the Monk on eight excision examples taken from the corpus of Monk queries. For a data entry "$x/y,\ z$", y is the total number of returned documents and x are those which contain an answer to the query. z are the number of answers that are wrong because they are about the subject that was explicitly being excluded in the query.

	AV	Jeeves	Monk
Cancer	2/9, 6	2/5, 3	2/3, 0
Bidfind	2/7, 5	1/6, 0	0/0, 0
Jobs	2/9, 0	2/5, 0	3/9, 6
Warts	0/9, 6	1/4, 2	0/1, 1
Drinking	4/10, 0	0/5, 1	0/2, 0
Browsers	2/8, 6	3/5, 2	0/9, 8
Witches	0/10, 10	0/5, 4	0/0, 0
Hondo	0/7, 6	1/6, 3	0/0, 0
Avr. Precision	17%	25%	20%
Avr. % false positives due to searching for the figure	67%	53%	58%

Table 6.2: Potential Improvement for NLIR Systems

As the data shows, none of the search engines fare particularly well. The precision for all three is around 20%. That is, only about one in five of the

responses contained the answer to the query. Furthermore, from a half to two thirds of the incorrect responses were specifically about the subject the query wanted to exclude, displaying little or no understanding of excision alternative phrases.

It is, of course, unreasonable to draw any conclusions about the relative merits of the search engines from this test. The systems were built for different purposes and have widely varying coverage—clearly factors which have an impact on the results. The important point is that each NLIR system shows room for improvement. Since I will demonstrate that improvement only for The Electric Monk, next, this shows that that improvement is not due to exceptionally bad prior performance by the Monk.

6.6.3 Evaluation of Section 6.4

Table 6.3 shows the results of asking the Monk questions in three different forms: without an alternative phrase, with an alternative phrase that has not been translated, and with the alternative phrase translated as described in Section 6.4. The first row of the table, for instance, refers to the questions in (191). The remaining sentences can be found in Appendix A. Although Grok is capable of performing the translation in Section 6.4, I did this by hand for this evaluation to abstract away from parsing issues.

(191) What are some works by Edgar Allan Poe?
What are some works by Edgar Allan Poe other than the Raven?
What are some works by Edgar Allan Poe? :|: `ANSWER NOT NEAR_8` (| raven)

Unfortunately, at the time of this evaluation, Electric Knowledge (the creators of The Electric Monk) had taken down their public portal in favor of providing search for the web pages of specific clients. This means that the large index used to process the queries in Table 6.2 was no longer available, and I therefore used different indices and different queries for this evaluation. I used indices of pages about American history and literature—each about 11,000 pages. These new, more specialized, indices have the benefit of abstracting this evaluation away from coverage issues (explaining the differences in precision between Table 6.2 and Table 6.3).

I created the questions in two ways. For several, I began by asking a question without an alternative phrase. I then added an alternative phrase in order to remove some responses I was not interested in. For example, when I asked `Who are the romantic poets?`, all responses were about female romantic poets. I there-

	Baseline			**Original Query**			**Translated by Algorithm 2**		
	Tot	Good	Top 5	Tot	Good	Top 5	Tot	Good	Top 5
Poe	12	6.5	3	10	0.5	0.5	10	5.5	2.5
Romantics	10	0	0	15	0	0	10	3	3
Witch Hunts	10	8	3	14	2	1	10	8	5
US Wars	15	12	2	0	0	0	16	13	4
Sonnets	15	10	5	10	2	0	10	8	4
Presidents	15	2	2	15	0	0	15	2	2
Epics	10	7	4	10	5	3	10	7	4
Dec of Ind	10	2	0	0	0	0	10	5.5	2
Avr. Precision		48%	47.5%		14.9%	15%		58%	66.3%

Table 6.3: Evaluation of Improvement for NLIR Systems

fore used the query `Who are the romantic poets not including women?` in the evaluation. Some queries were made without first trying the non-alternative phrase version: `What are some epics besides Beowulf?`, for example. This variation reflects the fact that it is unclear which is more common, excluding clutter from a set of responses or *a priori* excluding cases that the questioner is not interested in. The queries also vary in their syntactic structure, information requested, and alternative phrase used. The purpose of varying the queries in these ways is to ensure that the results do not simply reflect a quirk in the Monk's implementation.

For each query, Table 6.3 shows *total*, the number of documents returned; *good*, the number of true positives; and *top 5*, the number of true positives in the top five returned documents. A true positive was given to a document if it contained an answer to the question, and half a point was given to a document that contained an obvious link to a document with an answer to the question. Precision is computed for all documents and for the top five. It is important to note that the scores for the queries without alternative phrases are still computed with respect to the alternative phrase. That is, documents only about the "Raven" are considered false positives. In this way, we can view these scores as a baseline—what would have happened had the system simply removed the alternative phrase. This should be taken with a grain of salt because, in many cases, I chose the query *because* there were documents to remove (as in the *Romantics* example). However, a concern was that the transformed query would cause numerous false negatives. This is not the case as seen by the fact that the precision of the transformed query is not lower than the baseline. In fact, in no example was the precision less than the baseline, and at worse, the precision remained the same.

Performance on questions containing alternative phrases was quite poor, with an average of 15% precision. This is significantly worse than the transformed query, and even the baseline. The performance drop is due to the fact that the complex syntax of the query confuses the Monk's analysis. The Monk is forced to reduce the query to a simple set of keywords including the figure, which we were trying to avoid.

	Baseline		**Original Query**		**Translated by Algorithm 2**	
	FPF	FPF/Tot	FPF	FPF/Tot	FPF	FPF/Tot
Poe	1	.08	1	.1	1	.1
Romantics	8	.8	15	1	2	.2
Witch Hunts	2	.2	1	.07	0	0
US Wars	3	.2			3	.19
Sonnets	5	.33	8	.8	2	.2
Presidents	5	.33	10	.67	2	.13
Epics	0	0	0	0	0	0
Dec of Ind	3	.3			2	.2
		28.1%		44%		12.8%

Table 6.4: False Positives Containing Figure

Thus as predicted in the discussion of potential improvement, not accounting for alternative phrases can greatly increase the number of false positives containing the figure (FPF), the very thing the query is attempting to exclude. Table 6.4 shows that in the baseline case, where there is no alternative phrase, on average for 28% of the returned documents the only answer was the one we wanted to exclude. Adding the alternative phrase has the opposite of the intended effect as the percentage of FPFs increases to 44% for reasons described above. Transforming the query, on the other hand, causes the desired effect, more than halving the percentage of FPFs of the baseline.

6.7 Conclusion

This chapter has shown that an analysis of alternative phrases has significant practical application. In particular, I consider the task of natural language information retrieval. I first show that in human/human dialogues, alternative phrases are quite common, and we would expect the same to be true in NLIR if users truly believed that the system could understand their questions. I go on to show that current systems for NLIR perform poorly when faced with these constructions. In fact, they perform more poorly than if they had simply ignored the construction

altogether. Finally, I demonstrate that performance can be improved substantially even with simple operational semantics for the back-end retrieval system.

Much of this performance improvement can be achieved without implementing the full analysis of Chapter 3 and Chapter 4, but in order to implement approximations it is very helpful to understand the underlying theory. Otherwise the resulting heuristics are *ad-hoc* and can interact in unpredictable ways. My analysis can direct the construction of consistent, efficient systems for practical applications dealing with alternative phrases.

Chapter 7

Future Work

In this chapter, I discuss future directions in which I would like to take the research presented in this dissertation. First, I describe more work that must be done with the analysis of alternative phrases and with Grok and the OpenNLP project.

Also, I have been concerned here with the understanding of alternative phrases. However, generation is also an important issue which I have begun to explore. This can also be used to allow a search engine to inform the user of ways to improve the results of a query. For instance, if the results of the query *What are some web browsers?* are overwhelmed by pages about Netscape, the message, *Shall I search for web browsers other than Netscape?* could be generated as text or speech.

The acquisition of knowledge in an ISA hierarchy, discussed in Chapter 3, is also pertinent to NLIR. As described in Cooper (1997), The Electric Monk uses an ISA hierarchy to help classify queries. By inferring knowledge from users' questions topical information is automatically made available to the Monk to improve the results for future queries.

7.1 Further Analysis

Chapters 3 and 4 delve into the syntax and semantics of connected and free alternative phrases. In the course of my investigation, I made a number of interesting observations that I simply was unable to follow up on without straying too far from the goals of the thesis.

In particular, I focused primarily on what alternative phrases presuppositionally communicate about their figures. However, the assertional semantics deserves more attention in future work. For example, the assertions of restrictive **such** and **especially** were merely glossed over in this thesis. Considering that their inter-

pretations rely heavily on world and discourse knowledge, further investigation can also help in the construction of better knowledge representations in Grok.

I have also discussed some interactions of alternative phrases with themselves and other phenomena that seem to validate, to a certain extent, my analysis. For example, scope ambiguity accounts for different readings involving post-modifiers (Section 3.4.3) and my analysis makes the right predictions regarding different scopings of **such**, **other**, and comparatives. It would be very useful to follow up on these interactions and also to look for others.

Finally, a closer look at the syntax of alternative phrases is in order. In particular, I would like to produce a nicer account of commas in free alternative phrases. A possible avenue of research in this direction is Categorial Type Logic (CTL) and its connection to CCG. CTL, like CCG, is an extension of basic categorial grammar, but the two theories differ in their emphasis. CCG is concerned with restricting generative capacity and has been shown to have the minimum power necessary to account for observed linguistic phenomena. CTL, on the other hand, can support anything from context-free to Turing-complete power based on what *structural rules* are used in a given grammar. But, CTL has a logical framework (see Moortgat 1997 for details) which CCG lacks. These differences are reconciled in Kruijff and Baldridge (2000), who show that a fragment of CTL is weakly equivalent to CCG, thus providing a logical basis for CCG. The logic of CTL uses binary and unary *modalities*. The unary modality $\Box^{\downarrow}$ represents the lock and key mechanism and has been used to account for word order in Dutch (Oehrle, 1998), extraction (Morrill 1994, ch. 8; Hepple 1990), morphology (Heylen, 1999; Kruijff, forthcoming), and more. This mechanism can allow a simple account of commas, (192), and alternative phrases, (193).

(192) a. $\mathsf{X}|\Box^{\downarrow}\mathsf{X}$

b. $\mathsf{X}/,/\Box^{\downarrow}\mathsf{X}$

(193) a. $\Box^{\downarrow}((\mathsf{S}|\mathsf{NP})|(\mathsf{S}|\mathsf{NP}))$

b. $\Box^{\downarrow}((\mathsf{S}|(\mathsf{S}|\mathsf{NP}))|(\mathsf{S}|(\mathsf{S}|\mathsf{NP})))$

Research into the connection between CTL and CCG is still new, and the question of how much, if any, computation power such an analysis adds to the system requires further investigation.

7.2 More to Grok

There is much left to do in the Grok system. First, although the hand-built lexicon is extensive, it should be expanded. More closed-class items and more categories for open-class items should be added. This could be pursued by studying the most common entries in the acquired lexicon that do not appear in the hand-built lexicon. Also, more semantic distinctions (such as different types of adjectives and verbs) should be encoded, and the models for predicting semantics, described in Section 5.4.4, must be expanded to accommodate them. Finally, a more theoretically sound way should be found to assign probabilities (of a word having a particular category) to closed class entries in the hand-built lexicon that do not appear in the acquired lexicon.

The final goal can really only be considered when an accurate probabilistic model for CCG parsing is found. This will also open the door to more efficient parsing, and techniques will have to be developed to incorporate semantics into these new parsing techniques.

Finally, the knowledge-based component of Grok, while sufficient for most of this thesis, is quite lacking. The primary concern is that Grok has no reasoning capabilities, which would be useful in the implementation of generation (Section 7.3), a deeper instantiation of the "use existing objects" heuristic used throughout my analysis, and handling expectations as described in Section 4.3.3. The related issue of incremental processing must also be considered.

7.3 Generation

Whenever a linguistic analysis is developed for a phenomenon—particularly of discourse—and is shown to work in a robust natural language understanding system, it is worth considering how well it holds up in a generation system. In the case of alternative phrases, this is particularly interesting given their presuppositional and (sometimes) anaphoric nature. Generating with alternative phrases would, in certain situations, provide more textual economy (as described in Stone and Webber (1998)) than would be possible otherwise. For example, (194a) is clearly more compact than (194b).

(194) a. Cameron walked Fido and the other dogs.

b. Cameron walked the dogs. Fido was one of the dogs.

In this section, I discuss what would need to be done to allow successful generation of alternative phrases with the analysis provided in Chapter 3. I

propose a plan of attack for this problem.

I begin by listing the requirements for a generation system to generate alternative phrases — in particular for the analysis given in this dissertation. I then briefly discuss some standard generation techniques with respect to these requirements. Finally, I choose generation techniques compatible with my requirements and discuss what enhancements must be made.

7.3.1 An Alternative Phrase-Friendly Generator

1. **Multiple Communicative Goals**
   Any system that wants to use alternative phrases to exploit their textual economy must be able to entertain multiple communicative goals with respect to the same clause. Consider the examples in (194) where each is communicating two goals:

   (195) `walked(cameron, dogs)`
   `dog(fido)`

   Any generator that only considered fulfilling goals in distinct clauses would be unable to generate (194a).

2. **A Lexicalized System**
   As my analysis is given in terms of lexical semantics, the generation system must support some lexical grammar. Although the analysis can potentially be done in any lexical formalism, it is preferable for the generation system to be general enough to handle CCG.

3. **Presuppositions**
   The analysis requires presuppositions, and therefore the generator should be able to use presuppositions to satisfy goals through accommodation as well as testing whether an item's presuppositions are already satisfied. Furthermore, presuppositions must help drive lexical choice, as I discuss below.

4. **Anaphoric Reference**
   Such and **other** can refer anaphorically to the figure. To fully exploit textual economy, one would like to generate (196a) instead of (196b) when the context is appropriate.

   (196) a. What is the drinking age in other countries?
   b. What is the drinking age in other countries than Afghanistan?

A generation system must therefore support anaphoric reference (beyond simply choosing whether to realize an NP as a pronoun), and consequently, will most likely need access to a discourse model during the generation process.

7.3.2 Generation Techniques

Template Based

There are many types of generation systems that range from simple to very sophisticated. On the simple end of the spectrum are template-based generation systems, often known as *mail-merge* systems. These are commonly used for generating form letters and for other very constrained tasks, and work by substituting a small number of parameters into hand-crafted sentences. There are also more sophisticated techniques that use nested templates to produce more complex results (McKeown, 1985).

These methods suffer from the fact that they do not make use of a grammar that encodes linguistic analyses; in essence, they *are* the grammar and have compiled out the interesting features in a less flexible form. One consequence of this is that the same grammatical information used to understand, or parse, cannot be used for generation. From a generality standpoint, this and the fact that the possible generated sentences are very limited are serious drawbacks.

Of course, in constrained situations, template based techniques might be sufficient. Section 7.4.1 discusses this possibility for suggesting alternate queries in information retrieval.

Systemic Grammars

Another generation technique uses systemic grammars (Mann and Matthiessen 1983; The PenMan Project 1989). A systemic grammar is a set of hierarchies called *systems*. Systems are designed to choose different aspects of the structure of a sentences such as mood, tense, and voice. Each node in a system is a choice whose answer leads to newly relevant choices. Each choice is decided functionally by querying the knowledge base and other sources of information. Along the way, features are collected that will describe the final surface representation of the sentence.

This integration of the grammar with the control structure, while pinpointing many of the interesting distinctions that must be made in the generation process, makes it difficult to see how such a system could work directly with a lexicalized

grammar such as CCG. Furthermore, the control structure described in Mann and Matthiessen (1983) works from a single communicative goal, and it is unclear how multiple goals (and therefore lexical selection through presupposition) could be handled.

Semantic Head Driven Generators

Semantic head driven generation uses the structure of the logical form to guide the generation (Shieber *et al.* 1989). This is a bidirectional algorithm in that it traverses its derivation tree top-down and bottom-up. The input to the algorithm is a logical form. The algorithm moves down the tree by choosing a lexical head, a lexical item whose semantics unifies with the logical form. The algorithm then moves back up the tree by applying rules with this logical form as one of the right hand side arguments. The other elements of the right hand side are recursively generated. This step is continued until the resulting left hand side of the rule equals the original root.

Many extensions have been proposed to this algorithm to make it more general. van Noord (1993) proposes to relax the restriction that the semantics of a lexical head must unify with that of its parent. We will see that this becomes important for CCG. He also suggests extending the algorithm to take syntactic information into account when choosing the lexical head. This is important for handling lexical items that do not contribute to the semantics, but simply "pass them on". An example is the sentential complement *that* whose semantics is simply $\lambda x.x$. Additionally, many efficiency optimizations have been suggested, but I will not be concerned with those here.

Head driven generation is compatible with our requirements because it is mainly a search algorithm that requires little of an analysis except that one be able to locate lexical heads.

Spud

The Spud generator (Stone and Doran 1997; Stone 1998) works on a very different principle. Spud requires a grammar written in the Lexicalized Tree Adjoining Grammar (LTAG) formalism. LTAG is a tree-based formalism which supports two rules, *substitution* and *adjunction*. Substitution allows a tree to be attached to another tree's leaf that matches its root. With adjunction, a tree is inserted into an interior node of another tree. See Joshi *et al.* (1975) and Joshi and Schabes (1992) for more information.

Spud generates from a set of two types of goals. The first tells Spud to distinguish a semantic form (an entity or an event) using a particular syntactic category. The rest are communicative goals which tell Spud to include certain propositions in the description. Spud proceeds by building a tree that uniquely identifies the original semantic form while attempting to include information that satisfies the communicative goals. At each iteration, Spud chooses from a set of elementary trees that can substitute or adjoin into the current description. Preference is given to trees that satisfy more communicative goals. Interestingly, for our purposes, these goals can be satisfied through presupposition. Termination occurs when all goals are satisfied and there are no empty substitution sites in the tree.

As mentioned, Spud takes presupposition into account, but only syntax is used to find the set of possible elementary trees. Presupposition is simply used to choose among those trees: if a tree contributes to satisfying goals both compositionally and presuppositionally, it is more likely to be chosen. What we desire, though, is for the goal to be able to choose the lexical item with the presupposition even if it contributes no compositional information. While Spud's technique of data-directed generation does not ignore any possibilities, goal-directed generation would provide a more precise approach that is very possibly more efficient.

Also, the implementation of Spud's sentence realizer relies on a particular grammatical formalism, LTAG, and would have to be generalized to CCG. Finally, nothing prevents Spud from handling anaphoric reference, but it has not yet been implemented. Spud has the benefit that the generator can add content anywhere within the tree, and this flexibility can be used to enforce a left to right constraint on the description of entities. This would yield the appropriate salience updating for generating anaphoric reference. Spud implements support for this restriction for use in REA, an embodied conversational agent system, (Cassell *et al.*, 2000).[1]

In general, the Spud philosophy is consistent with the requirements for generating alternative phrases, but the implementation requires a number of changes.

7.3.3 Generating Alternative Phrases

In the discussion of generation techniques above, both semantic head-driven generation and Spud are consistent with generating alternative phrases. I will therefore continue with a discussion of what would be necessary to generate alternative phrases in a CCG analysis using a combination of semantic head driven generation and Spud techniques.

[1]Personal communication with Matthew Stone.

A few issues arise when doing head generation for CCG. One issue is alluded to by van Noord's suggested relaxation of head restrictions. In standard semantic head driven generation, the semantics of the lexical head must unify with the original logical form. However, this is not possible with standard CCG semantics because CCG semantic forms are often functions which are reduced through function application and β-reduction[2]. Therefore given the logical form in (197a), the lexical head, **likes**, would have the semantics in (197b) which, of course, does not unify with the original logical form. Note that ***this*** formatting denotes referables — i.e. entities and kinds.

(197) a. $\texttt{like}(\boldsymbol{john}, \boldsymbol{mary})$

b. $\lambda x \lambda y.\texttt{like}(y, x)$

c. $\texttt{like}(y, x)$

Van Noord states that in general, for non-standard head definitions it is necessary to compute the transitive and reflexive closures of the relations between parents and heads. Fortunately, this is a simple task for these relations as the desired logical form is simply the innermost body of the λ-expression, (197c). This, of course, still leaves the problem of lexical items that do not contribute any semantics of their own to the logical form — e.g. sentential complements. This problem is handled in Spud by attempting to use these lexical items if no other choices are possible.

The key problem regarding generation in this dissertation is how to use presuppositions to help direct lexical choice. Consider the two communicative goals in (198).

(198) $\texttt{have}(\boldsymbol{car}, \boldsymbol{engine})$
$\texttt{car}(\boldsymbol{volvo})$

(199) a. Cars have engines. A Volvo is a car.

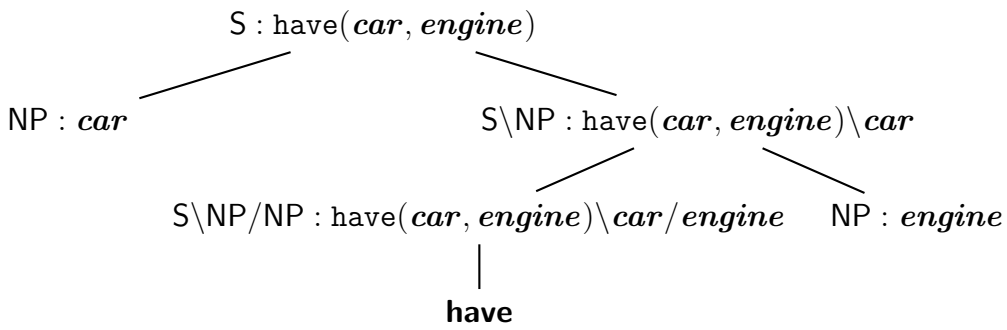
b. Cars, such as Volvos, have engines.

One way to realize these goals is (199a). However, as discussed in Section 7.3, this is not doing much for textual economy. I would much rather produce (199b). Through the normal techniques of head driven generation, even including the use of syntax for lexical choice, we cannot do this. There must be an added component to lexical choice that considers presuppositions as well as assertions.

[2]As noted in Section 2.2.2 and Section 6.3, this is not strictly true. However, in principle, the problem remains.

How, exactly, this should be done requires further research. I can give a hint, though, of how things might work out. The generation process must start with a particular goal. First, we perform semantic head driven generation, stopping when we reach a referable (an entity, kind, or event). This produces a structure that is very similar to an LTAG tree. The goal $\texttt{have}(\boldsymbol{car}, \boldsymbol{engine})$ produces the tree in (200), for example. Notice that generation has stopped at ***car*** and ***engine***, which are kinds.

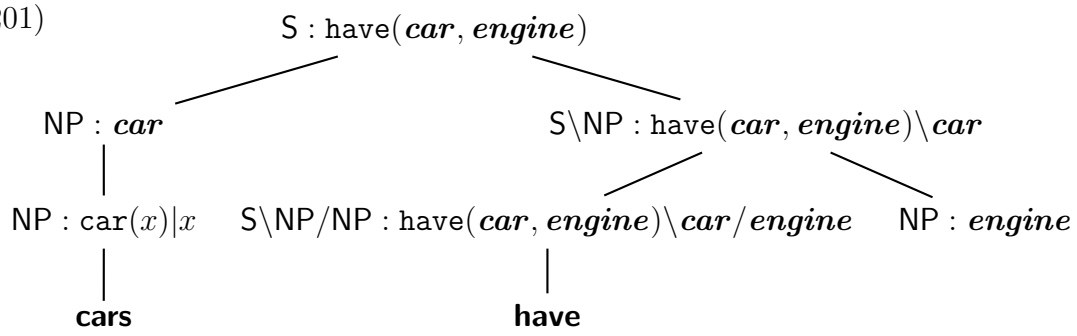
(200)



Now, we look at each unfinished node on the tree from left to right and proceed in a manner very similar to Spud. We attempt to add to the tree until the added content is equivalent to the semantics of the original node, where the trees being added are recursively created through semantic head driven generation. Syntactically, we can use operations very similar to substitution and adjunction. Substitution corresponds to expanding the referable to a description of like syntactic type (the entity ***john*** to the expression $\lambda x.\texttt{john}(x)$ or $\texttt{john}(x)|x$, for instance). Adjunction is equivalent to the CG operation application.

For example, a planner may decide that the best way to describe ***car***, in our tree, is simply as $\texttt{car}(x)|x$ — the car property. This is an NP and is thus attached through substitution:

(201)

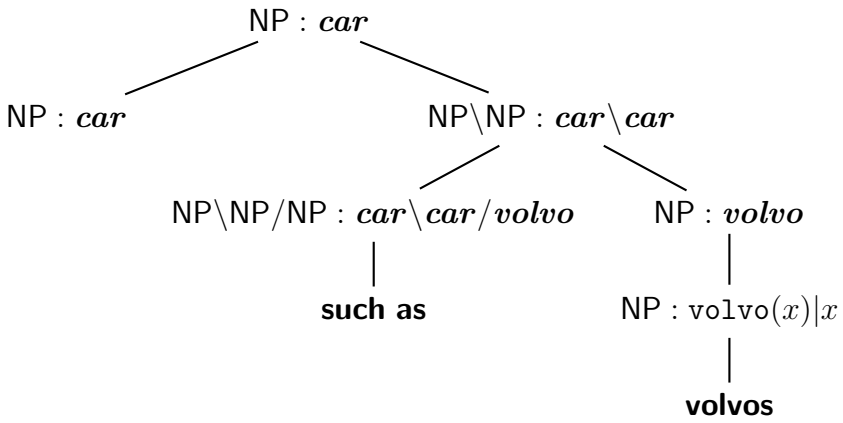


At the same time, we must also be opportunistically looking for ways to communicate our other goals, either through assertion or presupposition. For instance, we might have the goal $\texttt{car}(\boldsymbol{volvo})$ and notice that **such** has a presupposition that satisfies this goal. The original lexical entry is (53b) and is

instantiated through the goal to be (202). Recursively generating the goal with this as the lexical head produces the tree in (203).

(202) **such as** $\vdash$
$$\begin{cases} syn: & \mathsf{NP}_{\mathsf{eq}+,\mathsf{comp}-}\backslash\mathsf{NP}/\mathsf{NP} \\ sem: & \lambda S\lambda F \begin{cases} assert: & \boldsymbol{car} \cup \boldsymbol{volvo} = \boldsymbol{car} \\ presup: & \mathtt{car}(\boldsymbol{volvo}) \\ & alts(\boldsymbol{volvo}, \boldsymbol{car}) \end{cases} \end{cases}$$

(203)



This tree can now be adjoined into the original tree to produce the desired sentence. Furthermore, this technique allows us to distinguish between restrictive and non-restrictive **such**. The above example, since it does not reduce the cars being discussed, is non-restrictive, i.e. a parenthetical. However, we can also have sentences like *Cars such as Volvos do well on crash tests.* In this case, we are talking about cars that have some properties in common with Volvos — that they are safe, for instance.

The above algorithm will continue to add content to a node until the added content matches the semantics of the node. In our previous example, no more content needed to be added, so we used the non-restrictive **such** (which could be realized textually with commas or prosodically with a lower pitch). However, in the example about crash tests, we do need to restrict the cars we are talking about, so the best move is to pick the restrictive lexical entry whose semantics will reduce the set of cars. If instead we pick the non-restrictive entry, our tree will not be complete because the cars described will not be the same as the cars in the target node. This could be fixed using an adjective, producing a sentence like *Safe cars, such as Volvos, do well on crash tests.*

There are still questions left unanswered.

1. Does this method have the right level of generality? Perhaps it does not capture all of the phenomena we would like; perhaps it captures too many.

So far, I have only demonstrated a few alternative phrases—what about a wider variety?

2. How exactly do we order our lexical possibilities so that we try the most textually economic one first? This is especially interesting when we may want to use lexical items with anaphoric reference (which might be in the presuppositions).

3. I have left much unsaid about how semantics is evaluated as the descriptions are built. How does one use a description's semantics to determine what information should be added next? How does one check the effect of presuppositions on the hearer? Does it matter that the algorithm sketched above leaves added information local to the description subtree and does not percolate it to the rest of the tree? What about lexical items that have the identity semantics?

   Some of these issues are the focus of Stone (1998).

I began by presenting several requirements for a generator to handle my analysis of alternative phrases. I then reviewed several approaches to generation and discussed their merits regarding these restrictions. I concluded that a combination of semantic head driven generation and Spud techniques is a fruitful line of inquiry.

Although there are challenges to generating within CCG, integrating presupposition is the most interesting problem. I hypothesize that the issue comes down to efficient and effective use of presuppositions to direct lexical choice in a multi-goal environment. Much more work is required to test this hypothesis and work out the details.

7.4 More with NLIR and Queries

7.4.1 Suggesting Alternative Queries

In Section 7.3, I discussed how one can generate sentences containing alternative phrases and cited textual economy as the reason for wanting to do this. A further reason is to allow a NLIR system to interact with a user by suggesting alternative queries if their previous query generated unsatisfactory results.

For instance, if the search results are overwhelmed by one particular response (as can frequently happen if there has been a recent incident relevant to the

question), Grok can suggest an alternate query, as in (204)[3].

(204) User: What presidents have been impeached?
Monk: Shall I look for presidents other than Clinton?

Although not all NLIR systems would be capable of detecting such problems, the Monk can do it with its ISA hierarchy. The hierarchy would contain a list of presidents which could be compared against the documents returned by the query to determine if any one president occurred too frequently.

There are still several unanswered questions.

- By what metric does the Monk decide if an entity occurs in too many documents?
- What other things might a system want to communicate with a user? One possibility is clarifying questions which, for example, are ambiguous or anaphorically refer to an unknown antecedent.
- What protocol should be used for the NLIR system to send this information to Grok?
- Do we even need to perform full generation? Would template filling be sufficient? This depends on the answers to the previous questions. If there are many things that we might want to communicate to the user and they can be sent to Grok in a general way, then generation has more scalability than template filling. In addition, improvements in the grammar and generator will improve the responses, while templates are static and would need to be individually changed.

7.4.2 Learning from a Query

Chapter 3 and Chapter 4 not only show how to analyze the assertions of alternative phrases in order to direct search, they also show how properties of kinds and individuals can be learned from the presuppositions of alternative phrases. This is also very useful for systems like the Monk. Section 6.2.2 describes how the Monk uses an ISA hierarchy to answer questions. This runs into problems, however, if words appear in a query that are not in the hierarchy. Proper names are especially prone to this as new ones, such as **Clinton**, become topical all the time. So, if the system was unable to make the suggestion in (204), the user

[3]The other president was Andrew Johnson.

might follow up with *What about presidents other than Clinton?*. The system would then learn that Clinton is a president, which can be used both to classify questions, as described above, and to make suggestions the next time a question about presidents comes around.

There are still questions to be answered here, too:

- What information do we really need to remember?
  It would be useful to have a concept of "forgetting" so that if learned information is not referenced in some time, then it will leave the database to free up space. After all, it probably is not useful to remember that John Smith is a teacher in Kansas.

- When can we trust information we learn?
  Because this information is being learned through presuppositions, it is perhaps not particularly likely people would intentionally try to mislead the system. However, the system would certainly not be immune to such attacks. In addition, people innocently convey misinformation due to their own incorrect knowledge. We need a theory of trust so that we know when we can start using information we learn. In addition, it would be useful to know how harmful it would be to use incorrect information. In some cases, the effect might not even be noticed.

Chapter 8

Conclusions

The primary thrust of this thesis has been to provide an in-depth account of the semantics of alternative phrases using alternative sets, presupposition, and a "use existing objects" heuristic also used in abductive approaches to discourse interpretation.

Although I only treat a subset of alternative phrases, I still go significantly further than previous research. Until now, formal semantic analyses have been restricted to a few examples (**but** and **except for**) and are primarily concerned with the assertional semantics. Pattern-matching techniques which attempt to handle more examples are not effective for free alternative phrases and would need to be extended to account for alternative phrases with anaphoric reference.

I have discussed the connected alternative markers **besides**, **such** (**as**), and **other** (**than**) and the free alternative markers **besides**, **other** (**than**), **excluding**, **except for**, **in addition to**, **unlike**, **especially**, and more. This is significantly more breadth than previous semantic approaches. It is true that I do not display the same depth regarding assertional semantics (determining only the referred-to set), but I believe my approach can incorporate the previous work. The major new contribution of my analyses is what is being expressed about the *figure*, the NP argument of the alternative marker. These properties also put these analyses on a deeper theoretic level than the pattern matching approaches.

As this is a computational analysis, it is meant to be used in real systems. This thesis introduced such an NLP system, Grok. I discussed how Grok incorporates a large English CCG lexicon into a larger system that supports parsing, generation, and many aspects of knowledge representation. Preliminary experimentation with the system supports the belief that a well designed, modular system can produce satisfactory results even when the modules themselves are sometimes fast and simple solutions for complex problems. Furthermore, with a well designed system, these modules can be easily and transparently replaced with more interesting

modules to produce, hopefully, better results. Importantly, at this time, the system successfully supports most of the work presented in Chapter 3.

The Grok system is free, open-source software available to anyone from:

http://grok.sourceforge.net.

Grok, and the module interfaces of OpenNLP, are a contribution to the community meant to instigate greater modularity and reuse of code in the NLP community. At this time, Grok has already been used in several projects and is in the top 7% of the most active projects on SourceForge, a centralized host for more than 7000 open source projects.

I use Grok and my analysis of alternative phrases in the task of natural language information retrieval. I first show that in human/human dialogues, alternative phrases are quite common, and we would expect the same to be true in NLIR if users truly believed that the system could understand their questions. I go on to show that current systems for NLIR perform poorly when faced with these constructions. In fact, they perform more poorly than if they had simply ignored the construction altogether. Finally, I demonstrate that performance can be improved substantially even with simple operational semantics for the back-end retrieval system.

Until now, natural language search engines have largely ignored the semantics of discourse-related phenomena. However, if users are to ever have a truly effective natural language interaction with an information retrieval system, such constructions must be considered. This thesis takes a first step by showing that properly considering alternative phrases can have dramatic results. It is my hope that this research will inspire innovation in this area and in user interfaces so that users are better able to exercise their natural querying abilities.

Appendix A

Evaluation Sentences

Sentences for Table 6.2:

1. Does a pap smear test for anything other than cancer cells?
2. Are there other auction search engines besides BidFind?
3. How do I find job opportunities in other countries?
4. What growths can occur in the vaginal area other than warts?
5. What is the drinking age in other countries?
6. Where can I find web browsers other than netscape?
7. Were there any witch trials in America besides Salem?
8. Besides John Wayne, who was in Hondo?

Sentences for Table 6.3:

1. What are some works by Edgar Allan Poe?
   What are some works by Edgar Allan Poe other than the Raven?
   What are some works by Edgar Allan Poe? :|: ANSWER NOT NEAR_8 (| raven)
2. Who are the romantic poets?
   Who are the romantic poets not including women?
   Who are the romantic poets :|: ANSWER NOT NEAR_8 (| women)
3. Where did witch hunts occur?
   Where did witch hunts occur besides the Wyandot?
   Where did witch hunts occur? :|: ANSWER NOT NEAR_8 (| wyandot)

4. In what wars was the US involved?
   In what wars was the US involved besides the world wars?
   In what wars was the US involved? :|: ANSWER NOT NEAR_8 (| world)

5. Who wrote sonnets?
   Other than Shakespeare, who wrote sonnets?
   Who wrote sonnets? :|: ANSWER NOT NEAR_8 (| shakespeare)

6. What presidents were assassinated?
   What presidents were assassinated besides Kennedy?
   What presidents were assassinated? :|: ANSWER NOT NEAR_8
   (| kennedy)

7. What are some epics?
   What are some epics besides Beowulf?
   What are some epics? :|: ANSWER NOT NEAR_8 (| beowulf)

8. Who signed the declaration of independence?
   Who, excluding Georgians, signed the declaration of independence?
   Who signed the declaration of independence? :|: ANSWER NOT NEAR_8
   (| georgians)

Bibliography

Alta Vista (2000). http://www.altavista.com/.

Asher, N. and Morreau, M. (1995). What some generic sentences mean. In G. N. Carlson and F. J. Pelletier, editors, *The Generic Book*, chapter 7. The University of Chicago Press.

Ask Jeeves (2000). What is Ask Jeeves? http://www.askjeeves.com/docs/about/whatIsAskJeeves.html.

Bach, E. and Cooper, R. (1978). The np-s analysis of relative clauses and compositional semantics. *Linguistics and Philosophy*, **2**, 145–150.

Bach, E. and Partee, B. (1980). Anaphora and semantic structure. In *Papers from the Parasession on Pronouns and Anaphora, Tenth Meeting of the Chicago Linguistics Society*, Chicago. CLS.

Bangalore, S. (1997). *Complexity of Lexical Descriptions and Its Relevance to Partial Parsing*. Ph.D. thesis, University of Pennsylvania.

Beaver, D. (1995). *Presupposition and Assertion in Dynamic Semantics*. Ph.D. thesis, University of Edinburgh.

Beaver, D. (1997). Presupposition. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 939–1008. North Holland, Amsterdam.

Beesley, K. R. (1983). *On the Analysis of English Adjectives in a Montague Grammar*. Ph.D. thesis, University of Edinburgh.

Bierner, G. (1998). TraumaTalk: Content-to-speech generation for decision support at point of care. In *Proceedings of AMIA*. Hanley and Belfus.

Bierner, G. (1999). Inference through alternative-set semantics. In C. Monz and M. de Rijke, editors, *Inference in Computational Semantics*, pages 39–52.

Bierner, G. (2000). Inference through alternative-set semantics. *Journal of Language and Computation*, pages 259–274. To appear.

Black, A. and Taylor, P. (1997). Festival speech synthesis system: system documentation (1.1.1). Technical Report TR-83, Human Communication Research Centre.

Bos, J. (2000). http://www.coli.uni-sb.de/~bos/doris/.

Bowerman, M. (1982). Reorganizational processes in lexical and syntactic development. In E. Wanner and L. Gleitman, editors, *Language acquisition: The state of the art*. Cambridge University Press, New York.

Burnard, L. (1995). Users reference guide for the British National Corpus.

Carlson, G. N. and Pelletier, F. J., editors (1995). *The Generic Book*. The University of Chicago Press.

Carpenter, B. (1989). *Phrase Meaning and Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Carpenter, B. (1992). *Formal Grammar: Theory and Implementation, Vancouver Studies in Cognitive Science*, volume II, chapter Lexical and Unary Rules in Categorial Grammar. Oxford University Press.

Cassell, J., Stone, M., and Yan, H. (2000). Coordination and context-dependence in the generation of embodied conversation. In *First International Conference on Natural Language Generation*, pages 171–178.

Charniak, E. (1999). A maximum-entropy-inspired parser. Technical Report CS-99-12, Department of Computer Science, Brown University.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. MIT Press, Cambridge MA.

Chomsky, N. (1981). *Lectures on Government and Binding*. Foris, Dordrecht.

Cohen, W. (1995). Fast effective rule induction. In *Machine Learning: Proceedings of the Twelfth Int'l Conference*.

Collins, M. (1998). *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Cooper, E. (1997). *The On Point System: A Natural Language Search Engine for the World Wide Web*. Master's thesis, University of Chicago.

Copestake, A. (1999). The (new) LKB system. Technical report, CSLI, Stanford University.

Cunningham, H., Humphreys, K., Wilks, Y., and Gaizauskas, R. (1997). Software infrastructure for natural language processing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*.

Cunningham, H., Gaizauskas, R. G., Humphreys, K., and Wilks, Y. (1999). Experience with a language engineering architecture: Three years of GATE. In *Proceedings of the AISB'99 Workshop on Reference Architectures and Data Standards for NLP*, Edinburgh, U.K. The Society for the Study of Artificial Intelligence and Simulation of Behaviour.

Curry, H. B. and Feys, R. (1958). *Combinatory Logic: Vol I*. North Holland, Amsterdam.

Digest, R. (1991). *Reader's Digest New Complete Do-It-Yourself Manual*. Reader's Digest Publishing Co.

Geoquery (2000). http://www.cs.utexas.edu/users/ml/geo.html.

Grosz, B. J., Joshi, A. K., and Weinstein, S. (1995). Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, **21**(2), 203–225.

Grover, C., Carroll, J., and Briscoe, T. (1993). The Alvey Natural Language Tools Grammar (4th release). Technical report, Human Communication Research Centre, University of Edinburgh and Computer Laboratory, University of Cambridge.

Hearst, M. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, Nantes, France.

Hepple, M. (1990). *The Grammar and Processing of Order and Dependency: a Categorial Approach*. Ph.D. thesis, University of Edinburgh. Tech Report, Centre for Cognitive Science, University of Edinburgh.

Heycock, C. (1994). The internal structure of small clauses. In J. Beckman, editor, *Proceedings of NELS 25*, volume 1, pages 223–238, Amherst, Mass.

Heylen, D. (1999). *Types and Sorts: Resource Logic for Feature Checking*. Ph.D. thesis, Netherlands Graduate School of Linguistics.

Hobbs, J. R., Stickel, M., Martin, P., and Edwards, D. (1988). Interpretation as abduction. In *Proc. of the 28th Annual Meeting of the Association for Computational Linguistics. Buffalo, NY, 7–10 June 1988*, pages 95–103.

Hobbs, J. R., Stickel, M. E., Appelt, D. E., and Martin, P. (1993). Interpretation as abduction. *Artificial Intelligence*, **63**(1–2), 69–142.

Hockenmaier, J. (2000). Extraction of categorial lexica from an annotated corpus. Technical report, Division of Informatics, University of Edinburgh.

Hockenmaier, J., Bierner, G., and Baldridge, J. (2000). Providing robustness for a CCG system. In *Proceedings of the ESSLLI workshop on Linguistic Theory and Grammar Implementation*.

Hoeksema, J. (1995). The semantics of exception phrases. In J. van der Does and J. van Eijck, editors, *Quantifiers, Logic, and Language*, chapter 6, pages 145–177. Cambridge University Press.

Horn, L. (1989). *A Natural History of Negation*. University of Chicago Press.

Joshi, A. and Schabes, Y. (1992). Tree adjoining grammars and lexicalized grammars. In M. Nivat and M. Podelski, editors, *Definability and Recognizability of Sets of Trees*. Elsevier, Princeton.

Joshi, A. and Vijay-Shanker, K. (1999). Compositional semantics with lexicalized tree-adjoining grammar (ltag). In *Proceedings of the 3rd International Workshop on Computational Semantics, Tilburg, January*, pages 131–146. Computational Linguistics, Tilburg University.

Joshi, A., Levy, L., and Takahashi, M. (1975). Tree-adjunct grammars. *Journal of Computer Systems Science*, **10**, 136–163.

Karttunen, L. and Peters, S. (1979). Conventional implicature. In C. Oh and D. Dinneen, editors, *Syntax and Semantics 11: Presupposition*. New York, Academic Press.

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.

Keenan, E. L. (1996). The semantics of determiners. In S. Lappin, editor, *The Handbook of Contemporary Semantic Theory*, chapter 2, pages 41–63. Blackwell.

Klein, E. (1991). Comparatives. In A. von Stechow and D. Wunderlich, editors, *Semantics:An International Handbook of Contemporary Research*, chapter 32, pages 673–691. Walter de Bruyter.

Koller, A. and Niehren, J. (2000). On underspecified processing of dynamic semantics. In *COLING-2000*.

Kruijff, G.-J. M. (forthcoming). *A Categorial Architecture of Informativity: Dependency Grammar Logic & Information Structure*. Ph.D. thesis, Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic.

Kruijff, G.-J. M. and Baldridge, J. M. (2000). Relating categorial type logics and CCG through simulation. *Journal of Language and Computation*. In submission.

Kuhn, J., Eckle-Kohler, J., and Rohrer, C. (1998). Lexicon acquisition with and for symbolic NLP-systems – a bootstrapping approach. In *Proceedings of the First International Conference on Language Resources and Evaluation (LREC98)*, Granada, Spain.

Ladusaw, W. A. (1979). *Polarity sensitivity as inherent scope relations*. Ph.D. thesis, University of Texas at Austin.

Lagerwerf, L. (1998). *Causal Connectives have Presuppositions*. Ph.D. thesis, Catholic University of Brabant.

Lambek, J. (1958). The mathematics of sentence structure. *American Mathematical Monthly*, **65**, 154–170.

Landman, F. (1989). Groups, I. *Linguistics and Philosophy*, **12**, 559–605.

Lewis, D. (1979). Scorekeeping in a language game. *Journal of Philosophical Logic*, **8**, 339–359.

Lewis, D. and Sparck-Jones, K. (1996). Natural language processing for information retrieval. In *Communications of the ACM*, volume 39, pages 92–101.

Link, G. (1983). The logical analysis of plurals and mass terms. In R. Bäuerle, C. Schwarze, and A. von Stechow, editors, *Meaning, Use, and Interpretation in Language*, pages 302–323. de Gruyter, Berlin.

Mann, W. and Matthiessen, C. (1983). Nigel: A systemic grammar for text generation. Technical report, University of Southern California.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. (1993a). Building a large annotated corpus of English: the Penn Treebank. *Computational linguistics*, **19**, 313–330.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. (1993b). Building a large annotated corpus of English: the Penn Treebank. *Computational linguistics*, **19**, 313–330. Reprinted in Susan Armstrong, ed. 1994, *Using large corpora*, Cambridge, MA: MIT Press, 273–290.

McCawley, J. D. (1988). *The Syntactic Phenomena of English*. The University of Chicago Press.

McCord, M. C. (1990). Slot Grammar: A system for simpler construction of practical natural language grammars. In R. Studer, editor, *Natural Language and Logic*, Lecture Notes in Computer Science, pages 118–145. Springer, New York, NY.

McKeown, K. R. (1985). *Text Generation : Using discourse strategies and focus constraints to generate natural language text*. Cambridge University Press.

Mikheev, A. (1999). A knowledge-free method for capitalized word disambiguation. In *37th Annual Meeting of the Association for Compuational Linguistics*, pages 159–166.

Mikheev, A., Grover, C., and Moens, M. (1998). Description of the LTG system used for MUC-7. In *Proceedings of 7th Message Understanding Conference (MUC-7)*.

Miller, G. (1990). WordNet: an on-line lexical database. *International Journal of Lexicography*, **3**.

Monk (1999). http://www.electricmonk.com.

Montague, R. (1970). English as a formal language. In B. Visentini, editor, *Linguaggi nella Società e nella Technica*, pages 189–224. Edizioni di Comunità, Milan. Reprinted in Montague 1974, 188-221.

Moortgat, M. (1997). Categorial type logics. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 93–177. North Holland, Amsterdam.

Morrill, G. (1994). *Type-logical Grammar*. Kluwer, Dordrecht.

Oehrle, R. T. (1998). Multi-modal type-logical grammar. In R. Borsley and K. Borjars, editors, *Non-transformational Syntax*. Blackwell. To appear.

Palmer, M. and Wu, Z. (1995). Verb semantics for English-Chinese translation. Technical Report 95–22, IRCS, University of Pennsylvania.

Park, J. (1995). Quantifier scope and constituency. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, Boston*, pages 205–212, Palo Alto, Calif. Morgan Kaufmann.

Park, J. (1996). *A Lexical Theory of Quantification in Ambiguous Query Interpretation*. Ph.D. thesis, University of Pennsylvania. Tech Report MS-CIS–96-26/IRCS-96-27, University of Pennsylvania.

Pollard, C. and Sag, I. (1994). *Head Driven Phrase Structure Grammar*. CSLI/Chicago University Press, Chicago.

Prevost, S. and Steedman, M. (1994). Specifying intonation from context for speech synthesis. *Speech Communication*, **15**, 139–153.

Ratnaparkhi, A. (1997). A simple introduction to maximum entropy models for natural language processing. Technical Report IRCS-97-08, University of Pennsylvania, Institute for Research in Cognitive Science.

Ratnaparkhi, A. (1998). *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

Reynar, J. and Ratnaparkhi, A. (1997). A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the International Conference on Spoken Language Processing*, pages 16–19, Washington D.C.

RIALIST (2000). http://www.riacs.edu/research/detail/rialist/.

Rooth, M. (1985). *Association with Focus*. Ph.D. thesis, University of Massachusetts, Amherst.

Rooth, M. (1992). A theory of focus interpretation. *Natural Language Semantics*, **1**, 75–116.

Rose, C. P., Eugenio, B. D., and Moore, J. D. (1999). A dialogue based tutoring system for basic electricity and electronics. In *Proceedings of AI in Education*.

Sacerdoti, E. (1977). *A Structure for Plans and Behavior*. Elsevier/North-Holland, Amsterdam.

Shieber, S., van Noord, G., Moore, R., and Pereira, F. (1989). A semantic head-driven generation algorithm for unification-based formalisms. In *Proceedings of the 27th Conference of the Association for Computational Linguistics*.

Stalnaker, R. (1974). Pragmatic presuppositions. *Semantics and Philosophy*, pages 129–214.

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 Mass. Linguistic Inquiry Monograph, 30.

Steedman, M. (1999). Alternating quantifier scope in CCG. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, College Park, MD*, pages 301–308, San Francisco, CA. Morgan Kaufmann.

Steedman, M. (2000a). Information structure and the syntax-phonology interface. *Linguistic Inquiry*.

Steedman, M. (2000b). *The Syntactic Process*. The MIT Press, Cambridge Mass.

Stone, M. (1998). *Modality in Dialogue: Planning Pragmatics and Computation*. Ph.D. thesis, University of Pennsylvania. advisor: M. Steedman.

Stone, M. and Doran, C. (1997). Sentence planning as description using tree adjoining grammar. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Association for Computational Linguistics, Madrid, July*, pages 198–205.

Stone, M. and Webber, B. (1998). Textual economy through close coupling of syntax and semantics. In *International Workshop on Natural Language Generation*, pages 178–187, Niagara-on-the-Lake, Canada.

Strube, M. (1998). Never look back: An alternative to centering. In *Proceedings of COLING-ACL '98, Montreal*, pages 1251–1257.

The PenMan Project (1989). The penman documentation. Technical report, USC/Information Sciences Institute, Marina del Rey, CA.

Thompson, C. A., Mooney, R. J., and Tang, L. R. (1997). Learning to parse natural language database queries into logical form. In *Workshop on Automata Induction, Grammatical Inference, and Language Acuisition*.

TREC (2000). http://trec.nist.gov/.

van Benthem, J. (1986). *Essays in Logical Semantics*. Reidel, Dordrecht.

van Benthem, J. (1991). *Language in Action*. North Holland, Amsterdam.

van der Sandt, R. A. (1992). Presupposition projection as anaphora resolution. *Journal of Semantics*, **9**, 333–377.

van Genabith, J., Way, A., and Sadler, L. (1999). Semi-automatic generation of F-structures from Treebanks. In M. Butt and T. H. King, editors, *Proceedings of the LFG99 Conference*. CLSI Publication.

van Noord, G. (1993). *Reversibility in Natural Language Processing*. Ph.D. thesis, Universty of Utrecht.

VanLehn, K., Siler, S., Murray, C., and Baggett, W. (1998). What makes a tutorial event effective? In M. A. Gernsbacher and S. Derry, editors, *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society*, pages 1084–1089, Hillsdale, NJ.

VanLehn, K., Siler, S., Murray, C., Yamauchi, T., and Baggett, W. (in press). Human tutoring: Why do only some events cause learning? *Cognition and Instruction*.

Villavicencio, A. (1997). *Building a wide-coverage Combinatory Categorial Grammar*. Master's thesis, Cambridge.

von Fintel, K. (1993). Exceptive constructions. *Natural Language Semantics*, **1**(2), 123–148.

von Stechow, A. (1991). Current issues in the theory of focus. In A. von Stechow and D. Wunderlich, editors, *Semantics:An International Handbook of Contemporary Research*, chapter 39, pages 804–824. Walter de Bruyter.

Webber, B., Knott, A., and Joshi, A. (1999a). Multiple discourse connectives in a lexicalized grammar for discourse. In *Third International Workshop on Computational Semantics*, Tilberg, The Netherlands.

Webber, B., Knott, A., Stone, M., and Joshi, A. (1999b). Discourse relations: A structural and presuppositional account using lexicalised tag. In *Proceedings of the 37th Conference of the Association for Computational Linguistics*, pages 41–48, College Park, MD.

World Wide Web Consortium (1997). Extensible markup language (xml). Web Page. http://www.w3.org/XML/.

Xia, F. (1999). Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium(NLPRS-99)*.

XTAG-group (1999). A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-98-18, University of Pennsylvania.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time $O(n^3)$. *Information and Control*, **10**(2), 189–208.

Zelle, J. M. and Mooney, R. J. (1993). Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 817–823, Menlo Park, CA, USA. AAAI Press.

Zelle, J. M. and Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 1050–1055, Menlo Park. AAAI Press / MIT Press.

Index