Phrase Meaning and Categorial Grammar

Bob Carpenter

Ph.D.
University of Edinburgh
1989
I declare that this thesis has been composed by myself and that the research reported therein has been conducted by myself unless otherwise indicated.
Abstract

A substantial grammar of English is developed within a unification based categorial grammar and an intensional event-based algebraic semantics.

Feature-structure and first-order term unification phrase structure grammars are defined and some extensions and restrictions studied. A pair of theorems are proved, stating that the term domain is representationally more powerful, but that any phrase structure grammar written in one formalism is strongly equivalent to one in the other, both being Turing powerful. A tractable unification categorial grammar is then defined within the term system, allowing features only on basic categories.

An intensional event-based model theory is developed within the typed λ-calculus. The basic domains are taken to be algebraically structured sets of particulars and propositions. The domain of particulars is partitioned into domains of individuals, temporal intervals and eventualities. Mass and count nominals, as well as their verbal analogues, are treated as basic particulars and given a weak algebraic structure. The propositions are taken to form an algebra slightly weaker than a boolean algebra. Relations are constructed as propositional functions.

A compositional semantics is developed for a fragment of English using the intensional model theory and categorial grammar. Sentences and nouns are both interpreted as properties, sentences as properties of events and nouns as properties of individuals. This allows a uniform semantic treatment of modification and nominalisation. A syntax and semantics is also provided for auxiliaries, control, attitude verbs, naked infinitives and nominal and verbal adjuncts. It is also shown how the higher-order semantics can be encoded by first-order unification without extending the unification categorial grammar formalism.

A generalized quantifier system based on discourse representation theory and Cooper storage is employed to generate the acceptable readings of nominal and verbal phrases involving quantifiers and dependent elements. The interaction of quantification and tense with the event semantics is studied. Polyadic quantification in "donkey sentences", sloppy anaphora and the attributive/referential contrast are also captured.
## Contents

Acknowledgements vi

1 Introduction 1

1.1 Methodology 1

1.1.1 Generative Syntax 2

1.1.2 Linguistic Meaning and Interpretation 7

1.2 Overview 11

1.3 Mathematical Preliminaries 13

1.3.1 Set Theory 14

1.3.2 Formal Languages 19

1.3.3 Trees 20

1.3.4 Formal Languages 24

1.3.5 Trees 25

1.3.6 Partial Orderings and Lattices 28

1.3.7 Algebras and Lattices 32

1.3.8 Typed $\lambda$-Calculus 37

2 Phrases, Categories and Features 43

2.1 Phrase Structure 44

2.1.1 Phrase Structure Grammars 46

2.1.2 Simple English Phrase Structure 47

2.1.3 Interpretations of Phrase Structure Grammars 49

2.1.4 Equivalences and Restrictions 56

2.2 Categories, Rules and Partial Information 60

2.2.1 Subcategorisation and Polyadicity 61
3 Semantic Models

3.1 Philosophical Groundwork

3.1.1 Uniformities

3.1.2 Realistic Semantics

3.1.3 Models and Theories

3.2 Basic Domains

3.2.1 Individuals

3.2.2 Eventualities

3.2.3 Temporal Intervals

3.3 Propositions

3.3.1 Modality and Possible Worlds

3.3.2 Partial Information and Situations

3.3.3 The Algebra of Propositions

3.4 Propositional Functions

3.4.1 Typed Functions

3.4.2 Relations and Propositional Types

3.4.3 Individuating Relations

3.5 Higher Order Logic

3.5.1 Quantification
### 3.5.2 Generalised Logical Operations

#### 4 Meaning and Interpretation

4.1 Compositionality ............................. 163
4.1.1 Rule to Rule Compositionality ............... 164
4.1.2 Type-Driven Translation .................... 170
4.1.3 Application Driven Translation ............... 172

4.2 Unification and Logical Form .................. 174
4.2.1 Intermediate Translations ................... 175
4.2.2 Interpreted Unification Categorial Grammars .. 176

4.3 Naïve Event Semantics ........................ 182
4.3.1 Simple Nominals ........................... 182
4.3.2 Simple Verbs ............................... 185
4.3.3 Modifiers ................................. 187

4.4 Context and Discourse Referents ............... 192
4.4.1 Relational Meaning ........................ 192
4.4.2 Relational Categorial Grammar ............... 194
4.4.3 Utterance Events ........................... 197

4.5 English Relational Categorial Grammar ........... 199
4.5.1 Simple Noun Phrases ....................... 199
4.5.2 Common Nouns .............................. 201
4.5.3 Referential Determiners .................... 201
4.5.4 Resource Situations and Speaker Intentions .. 202
4.5.5 Simple Verbs ............................... 204
4.5.6 Auxiliary Verbs ............................ 209
4.5.7 Simple Modifiers ........................... 216
4.5.8 Complex Verbs ............................. 219
4.5.9 Complex Modifiers ........................ 228

### 5 Scope and Dependency

5.1 Quantification ............................... 231
5.1.1 PTQ ...................................... 233
5.1.2 Post-syntactic Processing .................. 238
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.3</td>
<td>Quantification and Relations</td>
<td>242</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Quantification and Events</td>
<td>248</td>
</tr>
<tr>
<td>5.1.5</td>
<td>Cooper Storage</td>
<td>252</td>
</tr>
<tr>
<td>5.1.6</td>
<td>Quantification in Unsaturated Clauses</td>
<td>257</td>
</tr>
<tr>
<td>5.1.7</td>
<td>De Dicto Nominals</td>
<td>269</td>
</tr>
<tr>
<td>5.1.8</td>
<td>Attributive Nominals</td>
<td>272</td>
</tr>
<tr>
<td>5.2</td>
<td>Dependency</td>
<td>278</td>
</tr>
<tr>
<td>5.2.1</td>
<td>PTQ</td>
<td>279</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Cooper Storage</td>
<td>281</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Polyadic Quantification</td>
<td>289</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Sloppy Anaphora</td>
<td>292</td>
</tr>
<tr>
<td>5.2.5</td>
<td>Other Dependents</td>
<td>295</td>
</tr>
<tr>
<td>A</td>
<td>English Relational Categorial Grammar</td>
<td>298</td>
</tr>
<tr>
<td>A.1</td>
<td>Grammar Rules</td>
<td>298</td>
</tr>
<tr>
<td>A.1.1</td>
<td>Categorial Application Rules</td>
<td>298</td>
</tr>
<tr>
<td>A.1.2</td>
<td>Unary Recursive Abstraction Rules</td>
<td>299</td>
</tr>
<tr>
<td>A.1.3</td>
<td>Quantifier Term Reduction Rules</td>
<td>299</td>
</tr>
<tr>
<td>A.1.4</td>
<td>Descriptive Term Reduction Rules</td>
<td>300</td>
</tr>
<tr>
<td>A.1.5</td>
<td>Anaphora Term Reduction Rules</td>
<td>300</td>
</tr>
<tr>
<td>A.2</td>
<td>Lexical Entries</td>
<td>302</td>
</tr>
<tr>
<td>A.2.1</td>
<td>Simple Noun Phrases</td>
<td>302</td>
</tr>
<tr>
<td>A.2.2</td>
<td>Common Nouns</td>
<td>303</td>
</tr>
<tr>
<td>A.2.3</td>
<td>Determiners</td>
<td>303</td>
</tr>
<tr>
<td>A.2.4</td>
<td>Nominal Modifiers</td>
<td>304</td>
</tr>
<tr>
<td>A.2.5</td>
<td>Simple Verbs</td>
<td>305</td>
</tr>
<tr>
<td>A.2.6</td>
<td>Auxiliary Verbs</td>
<td>306</td>
</tr>
<tr>
<td>A.2.7</td>
<td>Attitude Verbs</td>
<td>307</td>
</tr>
<tr>
<td>A.2.8</td>
<td>Perception Verbs</td>
<td>308</td>
</tr>
<tr>
<td>A.2.9</td>
<td>Control Verbs</td>
<td>308</td>
</tr>
<tr>
<td>A.2.10</td>
<td>Expletive Verbs</td>
<td>309</td>
</tr>
<tr>
<td>A.2.11</td>
<td>Verbal Modifiers</td>
<td>309</td>
</tr>
</tbody>
</table>
A.2.12 Prepositions ........................................ 310

Bibliography .................................................. 311
Acknowledgements

During my three years at the Centre for Cognitive Science at the University of Edinburgh, from Fall of 1984 until the summer of 1987, I was supported by a United States National Science Foundation Graduate Fellowship in Computer Science (grant number RCD-8651747) and by the Committee of Vice-Chancellors and Principals of the Universities of the United Kingdom Overseas Research Student Award Scheme (reference ORS/84281). While writing this dissertation in 1987/88, I was a visiting researcher at the Center for the Study of Language and Information at Stanford University. I would also like to thank my current employers, the Philosophy Department at Carnegie Mellon University and the Intelligent Systems Program at the University of Pittsburgh, whose members were understanding when this dissertation was completed two months late.

I initially planned to do a one year M.Sc. degree at the Center for Cognitive Science, but the research environment was so wonderful and the subject matter so compelling that I remained there as a Ph.D. student. A number of factors contributed to this decision, including good classes, seminar programs, excellent related departments and so on, but the main factor, for me, was the presence of other students and researchers as enthusiastic as myself. Most importantly, I would like to thank the other members of my class, Alex, Andy, Carol, Fairouz, Fiona, Glyn, Joe, Judy, George, Mukesh, and Remo.

Glyn Morrill was a constant source of inspiration in the domain of syntax, and my major collaborator during my stay at Edinburgh. I was also strongly influenced by the members of the ACORD Project at Edinburgh, whose goals were very similar to my own. The ACORD group consisted of Henk Zeevat, Marc Moens, Ewan Klein, Jo Calder and Mark Steedman.
My thoughts were shaped by the workshops which I attended regularly at the Centre for Cognitive Science. These were regular forums in which we discussed our own current research topics and studied those of others. My view of events and semantics was developed in the Tense and Discourse Workshop at Edinburgh. The members of that workshop included Marc Moens, Alex Lascarides, Barry Richards, Henk Zeevat, David McCarty, Jon Oberlander, Fairouz Kameradine and Robin Cooper. The major forum for computational linguistics discussion at Edinburgh was the Parsing Workshop. Regular participants in that group while I was there included Nick Haddock, Remo Pareschi, Henry Thompson, John Phillips, Robert Dale, Fred Popowich, Geraint Wiggins, Graeme Ritchie, Glyn Morrill, Mark Steedman, Jo Calder, Ewan Klein and Einar Jowsey.

A similar force was exerted on my thinking by the working groups at the Center for the Study of Language and Information. In the Situation Semantics Working Group at the Center for the Study of Language and Information, we struggled to figure out exactly what situation semantics was all about and how it related to other theories. In particular, I would like to thank Craige Roberts, Sally McConnell-Ginet, Bonnie Webber, Steve Levinson, Syun Tutiya, John Nerbonne, Mark Gawron, Annie Zaenen, Judy Crow and Doug Edwards. It was also very useful to participate in the Foundations of Grammar Group, of which I would like to thank Bill Rounds, Lauri Karttunen and Stu Shieber for helpful comments and discussions. I would also like to thank the members of the Mathematics of Situation Theory and Situation Theory and Situation Semantics Groups, including Jon Barwise, Gordon Plotkin, John Etchemendy, David Israel and John Perry.

Of course, no set of acknowledgements would be complete without a nod to the supervisors. In my case, this was Ewan Klein, Barry Richards and David McCarty. All of them have influenced my thinking profoundly and helped out in more ways, both academic and practical, than I could possibly mention.

For technical support, I would like to thank Doug Jones and Tom Wooding for helping me with the Hewlett-Packard “Bobcat” Workstations on which this thesis was produced. I am also indebted to Emma Pease, whose knowledge of the \LaTeX formatting system was invaluable. The thesis was actually finished at Carnegie Mellon University in the Laboratory for Computational Linguistics. Luckily, the
Laboratory has HP workstations and the transition was not too difficult, thanks to John Aronis. John is also responsible for the bibliography style.

On a more personal note, I would like to thank my parents, Mack and Carol Carpenter. Without their continuing support, this work would have not been possible. Finally, I would like to thank Kate Bielaczyc for being there when I needed her.
Chapter 1

Introduction

To say that someone knows a language is to say that they are attuned to the systematic relationship between expressions of the language and their meanings. The purpose of this thesis is to study this relationship.

This introductory chapter is meant to clarify exactly what we take the study of language to involve and to introduce some of the general issues we will be discussing in later chapters. In the first section we spell out the methodological assumptions that locate this research with respect to related work in the area of natural language. The second section is devoted to a quick overview of the remaining chapters. In the final section we provide the mathematical definitions and notations which we adopt.

1.1 Methodology

The work reported in this thesis falls under the broad umbrella of generative grammar. As we will use the term, generative grammar encompasses all areas of grammar, including phonology — the study of sound patterns, morphology — the study of word formation, syntax — the study of structural well-formedness, semantics — the study of literal meaning, and pragmatics — the study of interpretation in context. We will use the more restrictive terms generative syntax, generative morphology, generative semantics and so on to apply to the study of a particular aspect of grammar. In this thesis we will be especially concerned with syntax and semantics, as these two components provide the link between
compound expressions, or phrases, and their literal meanings. We consider each in turn in the next two sections.

1.1.1 Generative Syntax

The basic assumption made in generative syntax is that the syntactic portion of a language can be usefully viewed as a collection of expressions, where an expression is either a primitive expression or is a compound expression generated by combining other expressions in various ways. We will mainly be concerned with syntax, where the basic expressions are words. Of course, things other than written words, such as sign-language signs, pictographs, icons or phonological representations of words can also serve as basic expressions in syntax. The syntactician has the job of characterising the collection of so-called well-formed expressions. The linguistic notion of well-formedness is derived from generalising our intuitions concerning the acceptability of particular linguistic constructions.

A serious stumbling block for generative syntax is the lack of a pre-theoretic procedure for determining whether a given expression is well-formed or not. There is not even agreement among linguists as to what sort of heuristics should be employed in this task. Consequently, there is a great deal of freedom and controversy with respect to the decision as to which expressions are to be taken to be well-formed. What we will try and do here is provide some intuitive feeling for the sorts of judgements we will be making later with respect to well-formedness.

It has been assumed from the beginning of the generative enterprise that the well-formedness of an expression is independent of its meaning or lack thereof. Consider the following pair of expressions, from Chomsky (1957)

(1) a. Colourless green ideas sleep furiously.

b. *Furiously sleep ideas green colourless.

Although both of the expressions in (1) are meaningless, at least in the sense that they are impossible to interpret literally, Chomsky argued that the first is well-formed while the second is not. We will mark sentences which we judge to be ill-formed with a star, as we have done in (1)b. While this accords quite well
with the intuitions of native English speakers, there are more subtle examples in which the contribution of the meanings of the words plays an important role in intuitive judgements of acceptability.

Context also contributes to intuitions in the case of an expression's acceptability. Many expressions which are unacceptable when considered in isolation are perfectly all right when they are found in the appropriate context. As far as possible, we will attempt to factor out the contribution of meaning and context to syntactic well-formedness judgements. We will be liberal in our judgements, though, and make well-formedness an existential condition. That is, a construction will be taken to be well-formed if some instance of it is definitely acceptable in some context, rather than requiring it to be acceptable in every context. We then assume that the effect of meaning and context on acceptability should be accounted for by the semantic and pragmatic components of the grammar.

Another stumbling block on the road to determining whether or not an expression is well-formed is that speakers who are assumed to be part of the same linguistic community, where a linguistic community is just a group of people who are all assumed to speak a common language, will not always agree as to which expressions are acceptable. Not surprisingly, language users are reasonably consistent. If they were not, they would have an extremely difficult time understanding one another. In fact, it is usually only in cases where sentences are of borderline acceptability in the first place that speakers' intuitions begin to vary. Since the linguists' notion of well-formedness is very much an idealisation, the job of the linguist is usually taken to be that of capturing the difference between expressions which most speakers find acceptable and those which most find unacceptable. It is usually then left up to the theory to determine the well-formedness of the expressions of borderline acceptability.

A related difficulty is that of expressions which are considered unacceptable to most speakers, in the sense that they would never be used or would be next to impossible to understand in any context, but which nevertheless get classified as syntactically well-formed. A standard example of where this strategy is applied is in the case of centre-embedded relative clauses, as in
(2)  a. The mouse is hiding.
   b. The mouse that the cat bit is hiding.
   c. The mouse that the cat that the dog chased bit is hiding.
   d. The mouse that the cat that the dog that ran chased bit is hiding.

By (2)d, it is obvious that we have left the realm of intuitive acceptability. No one would use (2)d, in any context, nor would anyone understand it without a great deal of effort. The standard decision is to treat all of the examples in (2) as well-formed and draw a distinction between what is known as linguistic competence and performance. Linguistic competence is understood to be whatever it is that a person understands when they are taken to know a language. Performance, on the other hand, is taken to be the behaviour which is produced when actually using a language. The distinction seems to be at least plausible, if not completely uncontroversial, for a number of reasons stemming from the resource-boundedness of language users. People have finite short-term memories, are subject to distractions and behave differently when put in different situations. Language use, like any other aspect of human activity, is subject to these very real limitations. The task of the linguist is understood to be that of characterising linguistic competence, while linguistic performance is left to the psychologist (for some studies of performance factors, see Bever (1970), Kimball (1973), Tyler and Marslen-Wilson (1977), Crain and Steedman (1985) and Altmann (1986, 1987)). This introduces a large amount of noise into the empirical data, and unfortunately, opens another avenue of escape for the linguist. It might always be postulated that an observed distinction in linguistic behaviour is due to a deviance in actual performance from the ideal predicted by the theory of competence. We will try and exercise caution in invoking this sort of explanation.

Linguists, like any scientists, are concerned with characterisations of phenomena which are explanatory as opposed to being merely descriptive. An explanatory account of some phenomena would presumably lead to useful predictions about the way some system will behave in the future. The goal of a descriptive characterisation of some phenomena, not surprisingly, is to provide a complete and thorough
The way in which theories usually progress from descriptive stages to explanatory stages is by the unification of seemingly heterogenous phenomena.

The distinguishing feature of generative grammar, as opposed to other methods of linguistic theorising, is its commitment to the development of fully explicit and precise characterisations of linguistic phenomena. By employing the same sort of rigour found in the other empirical sciences, the claims embodied in a particular linguistic theory can be worked out in detail and hence evaluated at a more than superficial level. For historical reasons, such linguistic theories have come to be known as generative, because a generative syntax can be seen as generating the expressions it takes to be well-formed, while a generative semantics can generate the meanings of well-formed expressions, and so on. This is not intended to introduce a bias towards the generation or production as opposed to the recognition of languages, but merely reflects the order in which declarative grammar rules are usually written (see Section 2.1.4). Thus, generative grammar might very well be called mathematical or formal linguistics, although these terms already carry the connotation of separation from empirical data and a concentration on purely mathematical results concerning formal systems.

All natural languages have an infinite supply of well-formed expressions. We can see this by considering the case of centre-embedding found above, in which an extra relative clause can always be added to an existing noun complex. This means that a finite list of well-formed expressions will not suffice. Some means of recursively structuring expressions is needed if a such an infinite set of expressions is to be defined. Of course, there will only be a finite number of expressions which could ever be accepted by a human language user. This is because there are only a finite number of basic expressions in any language and there will be an upper bound on the length of expression that a person could comprehend or utter in one lifetime.

Generative syntacticians express their claims about acceptability in a particular language by specifying a so-called grammar for that language, where we now, somewhat ambiguously, use the term grammar to refer to what is really only the syntactic component of a complete grammar. There are a number of ways
in which a grammar for a language might be constructed and there are no hard and fast rules for what constitutes a grammar. Consequently, different grammars vary widely with regard to the primitives and the means of structuring them that are employed. Some grammars are algebraic in their structure (Ajdukiewicz 1935, Curry 1961, Steedman 1988), others use recursive production rules (Lambek 1958, 1961, Geach 1972), some provide transformational operations over trees of categories (Chomsky 1957, 1965, 1981, Joshi 1985), while still others are expressed solely in terms of logical relations between words (Hudson 1976). Recently, grammars have been expressed in a uniform format dealing with partial information known as unification (see Section 2.2).

With such wide scope for choice at such a basic level, one of the primary concerns of the generative grammarian or syntactician is not in the formulation of grammars for particular languages, but in the characterisation of the range of possible grammars, which would in turn characterise the range of possible natural languages. Hopefully, the discovery of invariant properties and mechanisms across languages will lead to a theory which provides some insight into the nature of human language, rather than just a description of the properties of particular languages.

The methodology most often employed in the search for so-called universal properties of languages is to construct a meta-grammar in which object grammars can be expressed. If such a meta-grammar can express all and only the possible natural languages, it is said to be a universal grammar. Similarly, Montague’s universal grammar system, which we discuss in Chapter 4.1, can be thought of as a universal grammar capturing not only all possible natural languages, but also all of the artificial languages, such as those employed in formal language theory, computer science and mathematics.

A number of proposals have been made as to the nature of universal grammar. The currently favoured approach of Chomsky (1982), is to explore a number of parameters along which languages are found to vary and associate each such parameter with a number of possible values or settings. A particular grammar, and hence a particular language, will then arise from fixing the values of these parameters. Alternatively, a universal grammar can be viewed as providing a collection
of universal operations, which can then be restricted or selected for grammars of particular languages. This is the approach taken in combinatory grammar (Steedman 1988), generalized phrase structure grammar (Gazdar, Klein, Pullum and Sag 1985) and also in Montague’s universal grammar system (Montague 1974c).

1.1.2 Linguistic Meaning and Interpretation

The primary purpose of language is communication. While this statement is fairly uncontroversial, much of the work in generative syntax and morphology has totally avoided the problems of connecting well-formed expressions up with meanings. This can be traced back to Chomsky’s (1957) arguments for the so-called autonomy of syntax. Examples such as (1) are supposed to display that syntax is an autonomous component of a natural language. From our point of view, even if the autonomy theory is partially correct, a theory of syntax or morphology is only interesting in so far as it tells us something useful about meanings and how they are derived from well-formed expressions. We will be taking a model theoretic approach to linguistic meaning. As its name implies, model theory is concerned with the construction of mathematical models which correspond to or represent observed phenomena in a natural way. In this section we introduce the basic methodology and concepts embodied in the model theoretic paradigm.

Natural languages display a large degree of context-dependence. What we mean by this is that the interpretation of a particular occurrence of an expression will be dependent on the context in which the expression is found. Adopting standard terminology, we will call occurrences of expressions utterances, even if they are written, signed or in some other way communicated without being uttered vocally. Coming with each utterance is an utterance context, which is just the situation in which the expression was used. The simplest form of contextual dependence is displayed by personal pronouns such as I, you and we, whose interpretations will depend on the speaker and addressee. Luckily, there is a systematic relationship between utterance contexts and possible interpretations of expressions. The use of I will invariably refer to the speaker, and you to the audience.
We follow the terminological lead of situation semantics (Barwise and Perry 1983) and refer to the invariant information an expression carries across different usages as its *linguistic meaning*. It is only by supplying additional contextual information that we can obtain an *interpretation* for an utterance of an expression. Of course, situation semantics was not the first theory to represent the meaning of an expression abstracted from particular utterances. Work in Montague's general framework, as exemplified by Lewis (1970), represents the meaning of an expression as a function from *indices* representing a context of use to interpretations. The study of linguistic meaning as we have defined it here is often referred to as *semantics*, while the study of contextual factors is called *pragmatics*. This use of the term *pragmatics* dates back to Bar-Hillel (1954), who originally introduced a formal theory of indices into the study of natural language pragmatics. It should be noted that this division of labour and terminology is not universal. The term *pragmatics* is often used to refer to the study of topics such as speech acts, conversational and conventional implicatures and the like, which we will not even touch upon in this thesis.

We also follow situation semantics in adopting a *relational* theory of meaning, although our analysis of context will differ from that of situation semantics. Under the relational theory of meaning, the meaning of an expression will be a relation between contexts in which the expression could be uttered and the possible interpretations of the expression in that context. One of the key features of the relational theory of meaning is that it is not only possible to fix the context and determine facts about the interpretation, but also to fix the interpretation and learn things about the context. The ability of language to operate in just this fashion is studied by Barwise and Perry (1983), and includes examples of when an addressee's knowledge of the intended interpretation of an utterance leads the addressee to previously unknown information about the utterance context.

On the relational view, it is impossible to do semantics without also considering pragmatics. This is because model-theoretic interpretations only arise as one side of a meaning relation, which is studied in total. Montague's (1974) approach can be seen as a restriction of the relational approach to functional relations which produce unique interpretations from expressions and utterance contexts. The
added generality of the relational approach stems from the possibility of assigning multiple interpretations to a single utterance. This is simply not possible in a theory where the meaning relation is assumed to be a function.

One of the most basic methodological principles of semantics is that of *compositionality*. The principle of compositionality, which is normally attributed to Frege (1892), is just that the meaning of an expression can be determined from the meaning of its parts and the way in which they were combined. Put this way, the principle of compositionality seems trivial. It is difficult to see how it could not hold due to the productive nature of language use. A finite number of basic expressions can be combined in a finite number of ways to produce an unbounded set of complex expressions. If the meanings of these complex expressions were not in some way derived from the meanings of their parts, it would seem impossible for people to interpret complex expressions in a principled way. While the basic principle of compositionality has not gone unchallenged, objections can usually be sidestepped by taking an appropriate view of what "meanings" are and what tools are available for "combination". With sufficiently rich meaning domains and operations, any well-defined relation between expressions and their meanings can be made to obey the letter of the compositionality principle, if not its intent. A theory of semantics usually takes the form of a proposal to restrict the sorts of things which can act as meanings and the operations available to generate new meanings. For instance, Montague (1974c) used entities from the higher-order modal interpretation of his intensional logic, situation semantics (Barwise and Perry 1983) employs situation theoretic objects, lexical-functional grammar (Kaplan and Bresnan 1982) an uninterpreted level of functional structure, or an interpretation of functional structure in a Montagovian model theory (Halvorsen 1983) or in a model of situation semantics (Fenstad, Halvorsen, Langholm and van Benthem 1987). The operations employed have varied with the sorts of objects used for representing meaning, but see Morrill and Carpenter (forthcoming), Steedman (1988), and Gazdar, Klein, Pullum and Sag (1985) for some specific principles.

We will consider a number of grammars and possible meaning domains. The grammar that we finally settle with will be such that the categories associated
with each phrase can be constructed from the categories associated with each constituent phrase. A category, for us, will contain not only syntactic information, but also semantic information, thus resembling the signs of Unification Categorial Grammar (Zeevat, Klein and Calder 1987) and Head-Driven Phrase Structure Grammar (Pollard and Sag 1987). Where our grammar will differ from Montague’s version of compositionality is in not associating each intermediate category with an interpretation in a model, although we will provide such an interpretation for finite declarative sentences. In Section 4.1, we will take up this issue in more detail, and investigate exactly which form of the compositionality principle our grammar respects.

A declarative sentence such as \textit{john ran} is used to make statements about the way the world is. The interpretations we will be proposing for the utterances of declarative sentences will be truth-conditional. Not surprisingly, truth-conditional interpretations will be intimately bound to the conditions under which a declarative sentence can be used truthfully. In (Carpenter, forthcoming), we propose interpretations for utterances of \textit{interrogatives} such as \textit{did john run?}, which are used to ask questions, and \textit{imperatives} like \textit{run!}, which are used to issue commands, which are derived from the semantics of their declarative counterparts.

The standard method by which truth-conditional semantics is carried out is by employing the mathematical tool of model theory. In model theory, mathematical models are constructed which represent possible ways in which the world and the objects in it could be arranged. Model theory dates back to Tarski (1935), who was concerned with the notion of the truth of logical statements. Montague (1974b) is usually cited as having brought significant model-theoretic ideas into the mainstream of linguistic semantics. For a recent article dealing with the advantages of models as opposed to strictly dealing with the logical relations that hold between various objects in a given domain, see Barwise (1984b).

When working in model theory, it is impossible to side-step the fundamental metaphysical issue concerning exactly what exists to be modeled. The study of what exists and how it is structured is called ontology. We will be adopting what is essentially a realist view of semantics. A realist semantics is one in which the meanings and interpretations of expressions are taken to be real in the sense
that they correspond to things which have some sort of existence, even if it is only abstract. This is in sharp contrast to the typical Montagovian approach to semantics (except as viewed by Lewis (1986)), which employs so-called *possibilia* such as possible worlds and individual concepts which are assumed to be merely possible objects. Of course, the objects used to do the modelling, which are almost invariably set-theoretic objects in mathematical model theory, should not be confused with the objects that are being modelled. It is the modelled objects which are assumed to be real, and this does not imply a platonistic view toward sets and other mathematical objects. In Chapter 3, we discuss the basic domains of objects we adopt, which will include singular and group individuals, basic and complex events, temporal intervals and propositions, as well as abstract functional and relational objects which can be constructed from the basic objects. While we will be interested in matters of ontology, we will be primarily concerned with the algebraic structure of these basic domains and the structure this induces on the constructed domains.

### 1.2 Overview

In the remainder of this chapter, we present the mathematical definitions and results which we will be using later.

In Chapter 2 we take up the question of syntactic structure in more detail. After introducing the phrase structure grammar formalism, we go on to tackle the problem of dealing with partial information in grammar. Following recent work in unification grammars, we develop a unification based phrase structure formalism based on the lattice of first-order terms. Most of the work in unification grammar has been carried out in an alternative formalism, namely that of directed acyclic attribute-value graphs, which we show to be a particular case of term-based unification. That is, every domain of categories which can be represented by means of directed acyclic graphs can be isomorphically embedded in a term lattice, but not conversely. It is somewhat surprising then, that we are able to prove that phrase structure grammars based on these non-equivalent category structures turn out to be strongly equivalent. This means that every term-based grammar
can be represented as an attribute-value grammar and conversely. We then cast some of the extensions of feature-based grammars, such as those dealing with conjunctive and negative information and sorting, into the term-based paradigm. Finally, we introduce the fundamental notion of categorial grammar on which our grammars are based, showing how agreement and subcategorisation information can be incorporated by means of extending the basic categories of pure categorial grammars to allow for partial information.

We begin Chapter 3 with a review of some philosophical approaches to the representation of linguistic meaning. We study the problems faced by realistic theories of meaning, such as situation semantics and our own. We also discuss the difference between a model and a theory of a given domain. We then turn to the basic domains that we employ in our semantics to represent the objects in the world that are left unanalysed in our theory. We introduce a domain of individuals based on that of Landman (1987). We set out the general principles of an event-based approach to the semantics of sentences and discuss the similarities in structure between the nominal and verbal domain and explain why we take the events to be a subset of the individuals. We also discuss how information stemming from tense and aspect can be incorporated into an event-based framework. We then lay out the basic domain of propositions we use to represent truth and intensionality. Our domain of propositions is based on that of Thomason (1980), but is stronger in the sense that our propositions are required to satisfy more restrictive axioms. Finally, we use these basic domains to build up a model of the typed λ-calculus which we will use to model the world and provide the basis for our representation of the interpretations of utterances.

In Chapter 4 we introduce Montague’s general theory of compositionality, work through type-driven translation and finally show how the compositional meaning assignment of categorial grammar follows from these general principles. We then show how terms of the λ-calculus can be represented in our term-based categorial grammar formalism and a type-driven translation can be implemented. This leads to the interpreted unification categorial grammars, which form the core of our syntactic and semantic grammar components. We then approach the task of providing a naïve event-based semantics for our interpreted unification categorial
grammar system. This grammar relates the treatment of events to the treatment of intensionality, but ignores the problems associated with the use of language in context. We then provide a general system for dealing with contextual information and introduce the relational categorial grammar system. It is at this point that we detail the fundamentals of our grammar system, including everything but quantified noun phrases and anaphoric reference.

Finally, in Chapter 5, we add additional grammar rules and lexical entries to deal with quantified noun phrases and anaphoric dependencies. We work with a unification-based version of the Cooper-storage system (Cooper 1983) and compare this approach to the term-insertion approach of Montague (1974d) and the post-syntactic approach of which Hobbs and Shieber (1987) is typical. In particular, we address the problems of capturing intensional objects, quantification within nominal constituents, quantification in unsaturated clauses and the interaction of quantification with an event-based semantics. We then turn to the treatment of dependent elements, and in particular, anaphoric pronouns. We augment our relational categorial grammar with additional rules which give the correct interpretations for clauses involving attributive nominals, polyadic quantifiers and the troublesome "donkey sentences" and even cases of so-called "sloppy" anaphora.

1.3 Mathematical Preliminaries

In this section we will outline the basic mathematical definitions and results which we will use. While it may be helpful to browse through this section before continuing on to the rest of the thesis, it is not necessary or even advisable to get bogged down here. Much of the content of the thesis can be understood without a total understanding of the mathematical concepts presented here. It is presented here for completeness and to provide precise definitions of the mathematical notions used in the sequel. None of the material is original and all of it can be found, albeit in a different format, in standard textbooks.
1.3.1 Set Theory

Basics

A set can be thought of as a collection of objects which can be grouped together and considered as an object in its own right. For a given set $S$ and object $x$, we will write $x \in S$ if $x$ is an element or member of $S$, and say $x$ is in $S$ or $S$ contains $x$. We write $x \notin S$ if $x$ is not an element of $S$. We write \{x_1, \ldots, x_n\} for the set with exactly the members $x_1, \ldots, x_n$. We write $\emptyset$ for the unique set {} with no members, which is called the empty set. A set with only one member is called a singleton. Where $\phi(x)$ is a formula or statement containing an occurrence of the variable $x$ and $S$ is a set, we will write \{x \in S \mid \phi(x)\} for the set of all objects $x \in S$ such that $\phi(x)$ is true. When $S$ is obvious from context, we will simply write \{x \mid \phi(x)\}.

For two sets $S$ and $T$, we write $S \subseteq T$ and say $S$ is a subset of $T$ if for every element $x \in S$ we have $x \in T$. Sets are inherently extensional, which means that two sets $S$ and $T$ are identical if and only if they have exactly the same members. So we have $S = T$ if and only if $S \subseteq T$ and $T \subseteq S$. We will write $S \subset T$ if $S \subseteq T$ and $S \neq T$ and say that $S$ is a proper subset of $T$. Two sets $S$ and $T$ are said to be disjoint if there is no $x$ such that $x \in S$ and $x \in T$.

We will define a number of operations over sets. Suppose that $S$ and $T$ are sets. The intersection of $S$ and $T$ is defined by

$$S \cap T = \{x \mid x \in S \text{ and } x \in T\},$$

the union of $S$ and $T$ by

$$S \cup T = \{x \mid x \in S \text{ or } x \in T\},$$

and the difference between $S$ and $T$ by

$$S \setminus T = \{x \mid x \in S \text{ and } x \notin T\}.$$  

We can extend the definitions of union and intersection to operate over sets of sets. Suppose $S$ is a set, all of whose members are sets. Then we let

$$\bigcup S = \{x \mid x \in T \text{ for some } T \in S\}.$$
We will often use the abbreviation $\bigcup_{\phi(S)} S$ for the set $\bigcup\{S \mid \phi(S)\}$, and similarly for other operations defined over sets of sets. When using this notation it will usually be in the form $\bigcup_{i \in I} S_i$, in which case we say that $S = \bigcup_{i \in I} S_i$ is indexed by the elements of $I$. If $s \in S_i$ then we say that $s$ has index $i$. We note that every element of an indexed set has at least one index.

We will write $(x_1, \ldots, x_n)$ for the ordered $n$-tuple of objects $x_1, \ldots, x_n$ with $n \geq 0$. We call an arbitrary $n$-tuple a sequence. We call an $n$-tuple an ordered pair if $n = 2$, an ordered triple if $n = 3$, and so on. Two $n$-tuples $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ are identical if and only if $x_i = y_i$ for each $i$ such that $1 \leq i \leq n$. For sets $S_1, \ldots, S_n$, we define their cross-product as

\[ S_1 \times \cdots \times S_n = \{(x_1, \ldots, x_n) \mid x_i \in S_i\}. \]

We will use the standard vector notation and write $\vec{s}$ for $(x_1, \ldots, x_n)$ where convenient. For a set $S$, we define its $n$-fold product to be

\[ S^n = \{(s_1, \ldots, s_n) \mid s_i \in S \text{ for } 1 \leq i \leq n\}. \]

We note that $S^0 = \{\emptyset\}$. We say that $s_m$ is the $m$th coordinate of the $n$-tuple $\vec{s} = (s_1, \ldots, s_n)$.

Of course, we could extend the notion of tuple to contain infinite sequences of objects, but we will have no need for infinite sequences. Sequences are often represented set-theoretically, where a pair $(a, b)$ is encoded as the set $\{a, \{a, b\}\}$ and the $n$-tuple $(x_1, \ldots, x_n)$ is represented by the ordered pair $(x_1, (x_2, \ldots, x_n))$, where $(x_2, \ldots, x_n)'$ is the representation of $(x_2, \ldots, x_n)$. We note that this encoding respects the identity conditions on tuples. It is easily verified that two tuples are identical if and only if their set-theoretic representations are identical sets. We define the head and tail of a tuple $\vec{x} = (x_1, \ldots, x_n)$ with $n > 0$ by

\[ \begin{align*}
\text{i. Head}((x_1, \ldots, x_n)) &= x_1 \\
\text{ii. Tail}((x_1, \ldots, x_n)) &= (x_2, \ldots, x_n).
\end{align*} \]
Functions and Relations

An \( n \)-ary relation over \( S_1, \ldots, S_n \) is any set \( R \subseteq S_1 \times \cdots \times S_n \). Again, if \( n = 2 \) we say that \( R \) is a binary relation, if \( n = 3 \), a ternary relation, and so on. We will sometimes use the shorthand \( R(s_1, \ldots, s_n) \) for \( (s_1, \ldots, s_n) \in R \) and say that \( (s_1, \ldots, s_n) \) are \( R \)-related or stand in the relation \( R \).

For two sets \( S \) and \( T \), a partial function from \( S \) into \( T \) is a relation \( f \subseteq S \times T \) such that for every \( x \in S \) there is at most one \( y \in T \) such that \((x, y) \in f \). We use the notation \( f : S \rightarrow T \) if \( f \) is a partial function from \( S \) into \( T \), and say that \( S \) is the domain and \( T \) is the range of \( f \). For a partial function \( f \) we will write \( \text{Dom}(f) \) for its domain and \( \text{Rng}(f) \) for its range. For a partial function \( f : S \rightarrow T \), we write \( f(x) = y \) if \((x, y) \in f \) and say that \( y \) is the image of \( x \) under \( f \) and that \( f \) is defined for \( x \). If \( f \) is not defined for \( x \), we say that \( f(x) \) is undefined. We will allow a slight abuse of notation and define the image of a subset \( U \subseteq \text{Dom}(f) \) by

\[
(11) \quad f(U) = \{f(y) \mid y \in U\}.
\]

If \( f : S \rightarrow T \) is a partial function which is defined for every \( x \in S \) we say that \( f \) is a total function, or just function. We will assume throughout that a function is total unless its partiality is explicitly specified. A (partial) function \( f : S_1 \times \cdots \times S_n \rightarrow T \) is called an \( n \)-ary (partial) function, and we use the shorthand \( f(s_1, \ldots, s_n) \) for \( f((s_1, \ldots, s_n)) \). For two sets \( S \) and \( T \), we write \( T^S \) for the set of total functions from \( S \) into \( T \), so that

\[
(12) \quad T^S = \{f \mid f : S \rightarrow T \text{ is total}\}.
\]

A partial function \( f : S \rightarrow T \) is said to be one-one or an injection if and only if for every \( t \in T \), there is at most one \( s \in S \) such that \( f(s) = t \), and is said to be onto or a surjection if for every \( t \in T \) there is at least one \( s \in S \) such that \( f(s) = t \). A one-one onto function is said to be a bijection. A total one-one function \( \sigma : S \rightarrow S \) is often called a permutation of \( S \). Of course, every permutation will be onto, as well. We will let \( \text{id}_S \) denote the total identity function on \( S \), where \( \text{id}_S(x) = x \) if \( x \in S \).

We will let

\[
(13) \quad \omega = \{0, 1, 2, \ldots\}
\]

16
be the set of natural numbers. We will use a standard encoding of the naturals as sets, setting
\[(14) \ n = \{m \mid m < n\}\]
for \(n \in \omega\), so that
\[(15) \ \omega = \bigcup_{n \in \omega} n.\]
A set \(S\) for which there is a one-one onto function from \(n\) to \(S\) is said to be of cardinality \(n\) and we write \(|S| = n\). A set \(S\) is said to be finite if it has a cardinality in \(\omega\) and is said to be infinite otherwise. If there is a one-one function from a set \(S\) into \(\omega\) then \(S\) is said to be countable or enumerable. Countable sets can be either finite or infinite. A set which is not countable is said to be uncountable, and all uncountable sets are infinite. We define the hereditary membership relation \(\in_h\) between objects and sets by assuming that \(x \in_h S\) if and only if either \(x = S\), \(x \in S\) or there is some set \(T \in S\) such that \(x \in_h T\). We say that a set has a property hereditarily if and only if all of its hereditary members have the property. For instance, a set is said to be hereditarily finite if it is finite and all of its members are hereditarily finite.

For a relation \(R \subseteq S_1 \times \cdots \times S_n\) we define its characteristic function to be the total function \(C_R : S_1 \times \cdots \times S_n \to 2\) such that \(C_R(\bar{x}) = 1\) if \(\bar{x} \in R\) and \(C_R(\bar{x}) = 0\) if \(\bar{x} \notin R\). Conversely, for any total function \(f : S_1 \times \cdots \times S_n \to \{0, 1\}\) we can define a relation \(R_f \subseteq S_1 \times \cdots \times S_n\) such that \(\bar{x} \in R_f\) if and only if \(f(\bar{x}) = 1\). We write \(2^S\) for the set of all subsets of \(S\), where a subset \(T \subseteq S\) is given by its characteristic function \(C_T : S \to 2\). Over tuples, there is a derived projection function \(\pi_m^n(\bar{s})\), where \(\pi_m^n(\bar{s})\) is defined to be the \(m\)th coordinate of the \(n\)-tuple \(\bar{s}\). More specifically, we set
\[(16) \ \pi_m^n((s_1, \ldots, s_n)) = s_m \text{ if } 1 \leq m \leq n.\]
A binary function \(f : S \times S \to S\) is often referred to as an operator over \(S\). It is standard to write operators in what is called infix notation (as opposed to the prefix notation we have used so far), writing \(f(x, y)\) as \(x \ f \ y\). An operator \(f\) over \(S\) such that for all \(x, y \in S\) we have \(x \ f \ y = y \ f \ x\) is called symmetric. An operator \(f\) over \(S\) such that
(17) \((x f y) f z = x f (y f z)\) (associative)

for every \(x, y, z \in S\) is said to be associative. We will usually drop parentheses around associative operators and shorten \((x f y) f z\) and \(x f (y f z)\) to \(x f y f z\). We will also use infix notation where convenient for binary relations as well as operators, writing \(x R y\) where \(R\) is a relation such that \((x, y) \in R\).

With two relations \(R \subseteq S \times T\) and \(R' \subseteq T \times U\), we define their composition to be the relation \(R' \circ R \subseteq S \times U\) such that \((s, u) \in R' \circ R\) if and only if there is some \(t \in T\) such that \((s, t) \in R\) and \((t, u) \in R'\). We note that if \(f : S \to T\) and \(g : T \to U\) are partial (total) functions then \(g \circ f : S \to U\) is a partial (total) function with domain \(S\) and range \(U\).

We say that \(x \in S\) is a fixed-point of the partial function \(f : S \to S\) if \(f(x) = x\). If \(f : S \to S\) is a partial function over \(S\), then for each \(n \in \omega\) we define the \(n\)th composition of \(f\) with itself to be that function \(f^n : S \to S\) such that \(f^0 = f\) and \(f^{n+1} = f \circ f^n\).

Suppose we have a set \(S\) and a number of operations \(F = \{f_1, \ldots, f_n\}\). We say that the closure of \(S\) with respect to \(F\) is the smallest set \(\mathcal{C}_F(S)\) such that

\begin{enumerate}
  \item \(S \subseteq \mathcal{C}_F(S)\)
  \item \(f(\bar{s}) \in \mathcal{C}_F(S)\) if \(f \in F\) is an \(n\)-place operation and \(\bar{s} \in \mathcal{C}_F(S)^n\).
\end{enumerate}

For a relation \(R \subseteq S \times T\), we define its inverse to be the relation \(R^{-1} \subseteq T \times S\) such that

\[(19) \quad R^{-1}(x, y) \iff R(y, x).\]

We note that if \(f\) is a partial function, then \(f^{-1}\) is a partial function if and only if \(f\) is one-one, and is total only if \(f\) is onto. If \(f : S \to T\) is a total bijection, then for every \(x \in S\), \(f^{-1}(f(x)) = x\) and for \(y \in T\), \(f(f^{-1}(y)) = y\). As notational shorthand, if we have a relation \(R\) such as \(\subseteq\), we will just turn the symbol around and write \(\subseteq^{-1}\) as \(\supseteq\). Similarly, \(\leq^{-1}\) is written \(\geq\), and so on.

There are a number of special types of relations we will be interested in. Suppose we fix a relation \(R \subseteq S \times S\). We say that \(R\) is
A relation which is transitive, reflexive and symmetric is said to be an equivalence relation. For any set $S$, we define the identity equivalence relation over $S$ to be the identity relation $\{(x, x) \mid x \in S\}$ and the complete equivalence relation over $S$ to be $S \times S$. If $R$ is an equivalence relation over a set $S$ and $x \in S$, we define the equivalence class of $x$ to be the set

$$ [x]_R = \{y \mid (x, y) \in R\} $$

and say that $x$ is a representative of $[x]_R$. We note that $(x, y) \in R$ if and only if $[x]_R = [y]_R$. For a given set $S$ and equivalence relation $R$ over $S$ we define the quotient set

$$ S/ R = \{[x]_R \mid x \in S\}, $$

which we call $S$ modulo $R$, to be the set of equivalence classes of $S$ with respect to $R$.

### 1.3.2 Formal Languages

For any set $S$, we define the Kleene-star of $S$ to be the set $S^*$ such that

$$ S^* = \bigcup_{n \in \omega} S^n. $$

The elements of $S^*$ are tuples of elements of $S$ of arbitrary finite length, where the length of an $n$-tuple is taken to be $n$. Elements of $S^*$ are often called sequences and in the context of linguistics are usually called strings. We will use the shorthand $s_1s_2\cdots s_n$ for $(s_1, s_2, \ldots, s_n)$, omitting the angle brackets and commas. In particular, we will often shorten the sequence $(c)$ to $c$ and write $(a, b)$ as $ab$. For the case where $n = 0$, we write the unique 0-tuple () as $e$, which is called the
null-string. Within all of the grammar formalisms we will be interested in, the only method for combining strings to form new strings is by concatenation. The concatenation operator \( \cdot \) over \( S^* \) is given by

\[
(s_1, \ldots, s_n) \cdot (t_1, \ldots, t_m) = (s_1, \ldots, s_n, t_1, \ldots, t_m).
\]

Of course, for any string \( \sigma \), we have \( \sigma \cdot e = e \cdot \sigma = \sigma \). The concatenation operation is associative, with \( (\sigma \cdot \tau) \cdot \pi = \sigma \cdot (\tau \cdot \pi) \) for any strings \( \sigma, \tau, \pi \). This being the case, we will usually omit the parentheses around concatenations as no ambiguity can arise. We will also take the step of suppressing the concatenation symbol and abbreviate \( \sigma \cdot \tau \) as \( \sigma \tau \). For two strings \( \sigma \) and \( \tau \) we say that \( \sigma \) is a substring of \( \tau \) if there are strings \( \pi \) and \( \rho \) such that \( \tau = \pi \sigma \rho \). We say that the string \( \sigma \) is a prefix of the string \( \tau \) if there is a string \( \pi \) such that \( \tau = \sigma \pi \), and we say that \( \sigma \) is a suffix of \( \tau \) if there is a string \( \pi \) such that \( \tau = \pi \sigma \).

We will call any subset \( \mathcal{L} \subseteq S^* \) a (formal) language over \( S \).

### 1.3.3 Trees

We define the set of trees over a set \( S \) to be the least set \( \text{TREE}(S) \) such that

\[
\begin{align*}
&i. \quad S \subseteq \text{TREE}(S), \\
&ii. \quad (s, (t_1, \ldots, t_n)) \in \text{TREE}(S) \text{ if } n \geq 1, s \in S \text{ and } t_i \in \text{TREE}(S) \text{ for } 1 \leq i \leq n.
\end{align*}
\]

Note that \( \text{TREE}(S) \), as we have defined it, will contain an infinite number of finite trees as long as \( S \) is non-empty. In fact, if \( S \) is countable then so is \( \text{TREE}(S) \). It would be straightforward to alter this and other definitions in this section to account for transfinite trees by taking something other than the minimal solution to the clauses in the definitions. But, since these have no recognised linguistic applications, we will ignore the transfinite cases here (but see Langendoen and Postal (1984)).

The reason trees are called trees is that they are usually pictured graphically. For instance, the tree

\[
\langle s_1, \langle s_2, \langle s_3, s_4 \rangle \rangle, s_5, \langle s_6, \langle s_7 \rangle \rangle \rangle
\]
whose structure is rather undecipherable, would be graphically displayed as

```
s1
  /\ 
 s2 /  \
  /   \
 s3 /s4 \
 s5  
    /   \
  s6  
    /    \
 s7      
```

which resembles a tree growing upside-down. Since it grows tiresome to typeset trees in this fashion, we will adopt some alternative methods for displaying trees. The first displays the tree above as

(27) \[
[[s_3 \ s_4]_{s_2} \ s_5 \ [s_7]_{s_6}]_{s_1}
\]

and will ordinarily be used within text. We will also display trees in an upside-down format, where our example tree would look like

(28) \[
\begin{array}{ccc}
s_3 & s_4 & s_5 & s_7 \\
s_2 &   &   & s_6 \\
    &   &   & s_1
\end{array}
\]

Suppose we have fixed a domain \(\text{TREE}(S)\) of trees defined over some set \(S\).

The nodes of a tree \(t\) are defined so that

(29) \begin{enumerate}
  \item \(\text{Node}(s) = \{s\}\) if \(s \in S\)
  \item \(\text{Node}((s,(t_1,\ldots,t_n))) = \{s\} \cup \bigcup_{1 \leq i \leq n} \text{Node}(t_i)\)
\end{enumerate}

The yield of a tree is defined recursively to be

(30) \begin{enumerate}
  \item \(\text{Yield}(s) = s\) if \(s \in S\)
  \item \(\text{Yield}((s,(t_1,\ldots,t_n))) = \text{Yield}(t_1) \cdots \text{Yield}(t_n)\).
\end{enumerate}

We define the root of a tree by

(31) \begin{enumerate}
  \item \(\text{Root}(s) = s\) if \(s \in S\)
  \item \(\text{Root}((s,(t_1,\ldots,t_n))) = s\).
\end{enumerate}

The set of leaves of a tree is defined so that
i. Leaf(s) = \{s\} if s ∈ S

ii. Leaf((s, \langle t_1, \ldots, t_n \rangle)) = \bigcup_{1 \leq i \leq n} Leaf(t_i).

The depth of a tree is defined so that

i. Depth(s) = 0 if s ∈ S

ii. Depth((s, \langle t_1, \ldots, t_n \rangle)) = 1 + \max_{1 \leq i \leq n} Depth(t_i).

The set of local trees of a tree is such that

i. Loc Tree(s) = \{s\} if s ∈ S

ii. LocTree((s, \langle t_1, \ldots, t_n \rangle)) = \{(s, \langle \text{Root}(t_1), \ldots, \text{Root}(t_n) \rangle)\} \cup \bigcup_{1 \leq i \leq n} \text{LocTree}(t_i).

Note that a local tree is always of depth 1 unless it comes from a leaf, in which case it is of depth 0. In a local tree t = \langle s, \langle s_0, \ldots, s_n \rangle \rangle we say that s is the mother and s_0, \ldots, s_n are the daughters of s, and we say that s immediately dominates each s_i. The set of subtrees of a tree is such that

i. SubTree(s) = \{s\} if s ∈ S

ii. SubTree((s, \langle t_1, \ldots, t_n \rangle)) = \{(s, \langle t_1, \ldots, t_n \rangle)\} \cup \bigcup_{1 \leq i \leq n} \text{SubTree}(t_i).

An element s ∈ S is said to be a descendant of or to be dominated by a s' ∈ S in the tree t if there is a subtree t' ∈ Subtree(t) of t such that Root(t) = s' and s ∈ Node(t), so that s is a node in a subtree of t rooted at s'.

The sample tree displayed above has a yield of s_3s_4s_8s_7, is rooted at s_1, has a depth of 2, and has local trees s_3, s_4, s_5, s_7, [s_2 s_5 s_6]_{s_1}, [s_3 s_4]_{s_2}, and [s_7]_{s_6}.

For a relation R ⊆ S_1 × \cdots × S_n we define its characteristic function to be the total function \( C_R : S_1 × \cdots × S_n → 2 \) such that \( C_R(\vec{x}) = 1 \) if \( \vec{x} \in R \) and \( C_R(\vec{x}) = 0 \) if \( \vec{x} \not\in R \). Conversely, for any total function \( f : S_1 × \cdots × S_n → \{0, 1\} \) we can define a relation \( R_f ⊆ S_1 × \cdots × S_n \) such that \( \vec{x} \in R_f \) if and only if \( f(\vec{x}) = 1 \). We write \( 2^S \) for the set of all subsets of S, where a subset \( T ⊆ S \) is given by its characteristic function \( C_T : S → 2 \). Over tuples, there is a derived projection function \( \pi^n_m(\vec{s}) \), where \( \pi^n_m(\vec{s}) \) is defined to be the mth coordinate of the n-tuple \( \vec{s} \). More specifically, we set
A binary function \( f : S \times S \to S \) is often referred to as an operator over \( S \). It is standard to write operators in what is called infix notation (as opposed to the prefix notation we have used so far), writing \( f(x, y) \) as \( x f y \). An operator \( f \) over \( S \) such that for all \( x, y \in S \) we have \( x f y = y f x \) is called symmetric. An operator \( f \) over \( S \) such that

\[(x f y) f z = x f (y f z) \quad \text{(associative)}\]

for every \( x, y, z \in S \) is said to be associative. We will usually drop parentheses around associative operators and shorten \( (x f y) f z \) and \( x f (y f z) \) to \( x f y f z \). We will also use infix notation where convenient for binary relations as well as operators, writing \( x R y \) where \( R \) is a relation such that \( (x, y) \in R \).

With two relations \( R \subseteq S \times T \) and \( R' \subseteq T \times U \), we define their composition to be the relation \( R' \circ R \subseteq S \times U \) such that \( (s, u) \in R' \circ R \) if and only if there is some \( t \in T \) such that \( (s, t) \in R \) and \( (t, u) \in R' \). We note that if \( f : S \to T \) and \( g : T \to U \) are partial (total) functions then \( g \circ f : S \to U \) is a partial (total) function with domain \( S \) and range \( U \).

We say that \( x \in S \) is a fixed-point of the function \( f : S \to S \) if \( f(x) = x \). If \( f : S \to S \) is a function over \( S \), then for each \( n \in \omega \) we define the \( n \)th composition of \( f \) with itself to be that function \( f^n : S \to S \) such that \( f^0 = f \) and \( f^{n+1} = f \circ f^n \).

Suppose we have a set \( S \) and a number of operations \( F = \{f_1, \ldots, f_n\} \). We say that the closure of \( S \) with respect to \( F \) is the smallest set \( Cl_F(S) \) such that

\[
\begin{align*}
\text{(i) } & S \subseteq Cl_F(S) \\
\text{(ii) } & f(\vec{s}) \in Cl_F(S) \text{ if } f \in F \text{ is an } n\text{-place operation and } \vec{s} \in Cl_F(S)^n.
\end{align*}
\]

For a relation \( R \subseteq S \times T \), we define its inverse to be the relation \( R^{-1} \subseteq T \times S \) such that

\[(R^{-1})(y, x) \iff R(y, x).\]

We note that if \( f \) is a partial function, then \( f^{-1} \) is a partial function if and only if \( f \) is one-one, and is total only if \( f \) is onto. If \( f : S \to T \) is one-one and onto,
then for every $x \in S$, $f^{-1}(f(x)) = x$ and for $y \in T$, $f(f^{-1}(y)) = y$. As notational shorthand, if we have a relation $R$ such as $\subseteq$, we will just turn the symbol around and write $\subseteq^{-1}$ as $\supseteq$. Similarly, $\subseteq^{-1}$ is written $\supseteq$, and so on.

There are a number of special types of relations we will be interested in. Suppose we fix a relation $R \subseteq S \times S$. We say that $R$ is

1. **reflexive** if $(x, x) \in R$ for every $x \in S$
2. **symmetric** if $(x, y) \in R$ if and only if $(y, x) \in R$, for every $x, y \in S$
3. **anti-symmetric** if $x = y$ whenever we have $(x, y), (y, x) \in R$
4. **transitive** if for every $x, y, z \in S$ such that $(x, y), (y, z) \in R$, we have $(x, z) \in R$.

A relation which is transitive, reflexive and symmetric is said to be an *equivalence relation*. For any set $S$, we define the *identity* equivalence relation over $S$ to be the identity relation $\{(x, x) \mid x \in S\}$ and the *complete* equivalence relation over $S$ to be $S \times S$. If $R$ is an equivalence relation over a set $S$ and $x \in S$, we define the *equivalence class* of $x$ to be the set

$$[x]_R = \{y \mid (x, y) \in R\}$$

and say that $x$ is a *representative* of $[x]_R$. We note that $(x, y) \in R$ if and only if $[x]_R = [y]_R$. For a given set $S$ and equivalence relation $R$ over $S$ we define the *quotient set*

$$S/R = \{[x]_R \mid x \in S\},$$

which we call $S$ *modulo* $R$, to be the set of equivalence classes of $S$ with respect to $R$.

**1.3.4 Formal Languages**

For any set $S$, we define the *Kleene-star* of $S$ to be the set $S^*$ such that

$$S^* = \bigcup_{n \in \omega} S^n.$$
The elements of $S^*$ are tuples of elements of $S$ of arbitrary finite length, where the
length of an $n$-tuple is taken to be $n$. Elements of $S^*$ are often called sequences
and in the context of linguistics are usually called strings. We will use the short-
hand $s_1 s_2 \ldots s_n$ for $\langle s_1, s_2, \ldots, s_n \rangle$, omitting the angle brackets and commas. In
particular, we will often shorten the sequence $\langle c \rangle$ to $c$ and write $\langle a, b \rangle$ as $ab$. For
the case where $n = 0$, we write the unique 0-tuple $\langle \rangle$ as $e$, which is called the
null-string. Within all of the grammar formalisms we will be interested in, the
only method for combining strings to form new strings is by concatenation. The
concatenation operator $\cdot$ over $S^*$ is given by

$$ (s_1, \ldots, s_n) \cdot (t_1, \ldots, t_m) = (s_1, \ldots, s_n, t_1, \ldots, t_m). $$

Of course, for any string $\sigma$, we have $\sigma \cdot e = e \cdot \sigma = \sigma$. The concatenation operation
is associative, with $(\sigma \cdot \tau) \cdot \pi = \sigma \cdot (\tau \cdot \pi)$ for any strings $\sigma, \tau, \pi$. This being the
case, we will usually omit the parentheses around concatenations as no ambiguity
can arise. We will also take the step of suppressing the concatenation symbol and
abbreviate $\sigma \cdot \tau$ as $\sigma \tau$. For two strings $\sigma$ and $\tau$ we say that $\sigma$ is a substring of
$\tau$ if there are strings $\pi$ and $\rho$ such that $\tau = \pi \sigma \rho$. We say that the string $\sigma$ is a
prefix of the string $\tau$ if there is a string $\pi$ such that $\tau = \pi \sigma$, and we say that $\sigma$ is
a suffix of $\tau$ if there is a string $\pi$ such that $\tau = \pi \sigma$.

We will call any subset $\mathcal{L} \subseteq S^*$ a (formal) language over $S$.

1.3.5 Trees

We define the set of trees over a set $S$ to be the least set $\text{TREE}(S)$ such that

$$ (45) \quad \begin{array}{l}
\text{i. } S \subseteq \text{TREE}(S), \text{ and } \\
\text{ii. } \langle s, \langle t_1, \ldots, t_n \rangle \rangle \in \text{TREE}(S) \text{ if } n \geq 1, s \in S \text{ and } t_i \in \text{TREE}(S). 
\end{array} $$

Note that $\text{TREE}(S)$, as we have defined it, will contain an infinite number of finite
trees as long as $S$ is non-empty. In fact, if $S$ is countable then so is $\text{TREE}(S)$. It
would be straightforward to alter this and other definitions in this section to
account for transfinite trees by taking something other than the minimal solution
to the clauses in the definitions. But, since these have no recognised linguistic
applications, we will ignore the transfinite cases here (but see Langendoen and Postal (1984)).

The reason trees are called trees is that they are usually pictured graphically. For instance, the tree

\[(s_1, (s_2, (s_3, s_4)), s_5, (s_6, (s_7)))\]

whose structure is rather undecipherable, would be graphically displayed as

```
     s1
    /   \
   s2   s5
  /     \
 s3     s6
```

which resembles a tree growing upside-down. Since it grows tiresome to typeset trees in this fashion, we will adopt some alternative methods for displaying trees. The first displays the tree above as

\[\left[\begin{array}{c}
  s_3 \\
  s_4
\end{array}\right]_{s_1} \left[\begin{array}{c}
  s_5 \\
  s_7
\end{array}\right]_{s_6}
\]

and will ordinarily be used within text. We will also display trees in an upside-down format, where our example tree would look like

\[
\begin{array}{c}
  s_3 & s_4 \\
  \hline
  s_2 & s_5 & s_6 & s_7
\end{array}
\]

Suppose we have fixed a domain \(\text{TREE}(S)\) of trees defined over some set \(S\). The nodes of a tree \(t\) are defined so that

\[(49)\]

i. \(\text{Node}(s) = \{s\} \text{ if } s \in S\)

ii. \(\text{Node}(\langle s, (t_1, \ldots, t_n) \rangle) = \{s\} \cup \bigcup_{1 \leq i \leq n} \text{Node}(t_i)\)

The yield of a tree is defined recursively to be

\[(50)\]

i. \(\text{Yield}(s) = s \text{ if } s \in S\)

ii. \(\text{Yield}(\langle s, (t_1, \ldots, t_n) \rangle) = \text{Yield}(t_1) \cdots \text{Yield}(t_n)\).
We define the root of a tree by

\begin{align*}
\text{(51) } & \text{ i. } \text{Root}(s) = s \text{ if } s \in S \\
& \text{ ii. } \text{Root}((s, \{t_1, \ldots, t_n\})) = s.
\end{align*}

The set of leaves of a tree is defined so that

\begin{align*}
\text{(52) } & \text{ i. } \text{Leaf}(s) = \{s\} \text{ if } s \in S \\
& \text{ ii. } \text{Leaf}((s, \{t_1, \ldots, t_n\})) = \bigcup_{1 \leq i \leq n} \text{Leaf}(t_i).
\end{align*}

The depth of a tree is defined so that

\begin{align*}
\text{(53) } & \text{ i. } \text{Depth}(s) = 0 \text{ if } s \in S \\
& \text{ ii. } \text{Depth}((s, \{t_1, \ldots, t_n\})) = 1 + \max_{1 \leq i \leq n} \text{Depth}(t_i).
\end{align*}

The set of local trees of a tree is such that

\begin{align*}
\text{(54) } & \text{ i. } \text{LocTree}(s) = \{s\} \text{ if } s \in S \\
& \text{ ii. } \text{LocTree}((s, \{t_1, \ldots, t_n\})) \\
& \quad = \{(s, (\text{Root}(t_1), \ldots, \text{Root}(t_n)))\} \cup \bigcup_{1 \leq i \leq n} \text{LocTree}(t_i).
\end{align*}

Note that a local tree is always of depth 1 unless it comes from a leaf, in which case it is of depth 0. In a local tree \( t = (s, \{s_0, \ldots, s_n\}) \) we say that \( s \) is the mother and \( s_0, \ldots, s_n \) are the daughters of \( s \), and we say that \( s \) immediately dominates each \( s_i \). The set of subtrees of a tree is such that

\begin{align*}
\text{(55) } & \text{ i. } \text{SubTree}(s) = \{s\} \text{ if } s \in S \\
& \text{ ii. } \text{SubTree}((s, \{t_1, \ldots, t_n\})) \\
& \quad = \{(s, (t_1, \ldots, t_n))\} \cup \bigcup_{1 \leq i \leq n} \text{SubTree}(t_i).
\end{align*}

An element \( s \in S \) is said to be a descendant of or to be dominated by a \( s' \in S \) in the tree \( t \) if there is a subtree \( t' \in \text{Subtree}(t) \) of \( t \) such that \( \text{Root}(t) = s' \) and \( s \in \text{Node}(t) \), so that \( s \) is a node in a subtree of \( t \) rooted at \( s' \).

The sample tree displayed above has a yield of \( s_3 s_4 s_5 s_7 \), is rooted at \( s_1 \), has a depth of 2, and has local trees \( s_3, s_4, s_5, s_7, [s_2 s_5 s_6]_{s_1}, [s_3 s_4]_{s_2} \), and \( [s_7]_{s_6} \).
A transitive reflexive anti-symmetric relation \( \leq \) over a set \( S \) is called a *partial ordering* of \( S \). If \( x \leq y \) then we say that \( x \) is *less than or equal to* \( y \) or looking the other way, \( y \) is *greater than or equal to* \( x \) which we write \( y \geq x \). We will write \( x < y \) if \( x \leq y \) and \( x \neq y \) and say that \( x \) is *less than* \( y \) or \( y \) is *greater than* \( x \).

We will say that \( x \) *covers* \( y \) and write \( x \geq y \) or \( y \leq x \) if \( y < x \) and there is no \( z \in S \) such that \( y < z < x \). We will extend \( \leq \) to sets of elements \( T \subseteq S \) by writing \( x \leq T \) if and only if \( x \leq t \) for every \( t \in T \), and similarly \( T \leq y \) if and only if \( t \leq y \) for every \( t \in T \), and \( T \leq U \) if and only if \( t \leq u \) for every \( t \in T \) and \( u \in U \). A partial ordering \( \leq \) of \( S \) is then such that for all \( x, y, z \in S \),

\[(56) \begin{align*}
  &i. \ x \leq x \text{ (reflexivity)} \\
  &ii. \text{ if } x \leq y \text{ and } y \leq z \text{ then } x \leq z \text{ (transitivity)} \\
  &iii. \text{ if } x \leq y \text{ and } y \leq x \text{ then } x = y \text{ (anti-symmetry)}
\end{align*}\]

We will call a pair \((S, \leq)\) a *partially-ordered set* when \( \leq \) is a partial ordering of \( S \). Where \( \leq \) is understood, we will often abuse our terminology and refer to the set \( S \) as a partially ordered set.

Suppose \( S \) is a set partially ordered by \( \leq \) such that \( x, y \in S \). The *intervals* of \( S \) with endpoints \( x \) and \( y \) are defined by

\[(57) \begin{align*}
  &i. \ [x, y] = \{ z \mid x \leq z \leq y \} \\
  &ii. \ (x, y) = \{ z \mid x < z < y \} \\
  &iii. \ [x, y) = \{ z \mid x \leq z < y \} \\
  &iv. \ (x, y] = \{ z \mid x < z \leq y \}.
\end{align*}\]

If an interval contains its endpoint then it is said to be *closed* for that endpoint, and is otherwise said to be *open* for that endpoint. The interval \([x, y]\) is simply referred to as a *closed interval* and \((x, y)\) as an *open interval*.

If \( x, y \in S \) are such that neither \( x \leq y \) or \( y \leq x \) then we say \( x \) and \( y \) are *incomparable*. A partial ordering \( \leq \) on \( S \) is said to be a *linear ordering* or *total ordering* if

\[(58) \ x \leq y \text{ or } y \leq x \text{ (linearity)} \]
for every $x, y \in S$.

Suppose we have a set $S$ partially ordered by $\leq$. For a subset $T \subseteq S$, we say that $\bigvee T$ is its least upper bound and $\bigwedge T$ is its greatest lower bound when

\begin{align*}
(59) \quad & \text{i. } \bigvee T = x \text{ if } T \leq x \leq y \text{ for all } y \in S \text{ such that } T \leq y \quad \text{(lub)} \\
& \text{ii. } \bigwedge T = x \text{ if } y \leq x \leq T \text{ for all } y \in S \text{ such that } y \leq T \quad \text{(glb)}
\end{align*}

are satisfied. So $x$ is a least upper bound of a set of elements $T$ if and only if it is the least element which is greater than all of the elements of $T$. Similarly, $x$ is a greatest lower bound of a set of elements $T$ if and only if it is the greatest element which is less than all of the elements of $T$. Of course, there may be subsets of partial orders without least upper bounds, greatest lower bounds, or both. But note that bounds are unique if they are defined.

We define two partial operators, the join $\bigvee$ and the meet $\bigwedge$ over a partially ordered set $S$ by setting $x \bigvee y = \bigvee\{x, y\}$ and $x \bigwedge y = \bigwedge\{x, y\}$. We note the the meet and join operators satisfy

\begin{align*}
(60) \quad & \text{i. } x \bigvee x = x \bigwedge x = x \quad \text{(idempotence)} \\
& \text{ii. } x \bigvee (x \bigwedge y) = x \quad \text{(absorption)} \\
& \text{iii. } x \bigvee (y \bigvee z) = (x \bigvee y) \bigvee z \quad \text{(associativity)} \\
& \text{iv. } x \leq y \text{ if and only if } x = x \bigwedge y \text{ if and only if } y = x \bigvee y. \\
& \text{v. } x \bigwedge y \leq x \leq x \bigvee y
\end{align*}

where all of the relevant bounds exist.

For any subset $T \subseteq S$ we say that an element $t \in T$ is minimal (maximal) in $T$ if and only if there is no $t' \in T$ such that $t' < t$ ($t' > t'$). When they exist, we call $\bigwedge T \in T$ the minimum or least element of $T$ and $\bigvee T \in T$ the maximum or greatest element of $T$. If there is a minimum element of the whole set $S$ we call it the bottom of $S$ and write it $\bot = \bigwedge S$, and if there is a maximum we call it top and write $\top = \bigvee S$. We note that $\bigvee \emptyset = \bot$ and $\bigwedge \emptyset = \top$ if the bounds exist. A partial ordering is said to be bounded below if it has a bottom element, bounded above if it has a top element or simply bounded if it has both. We define the bounding of a partially ordered set $S$ to be the smallest bounded set $S^+$ containing $S$. $S^+$ will simply be the result of appending an element $T$ to $S$ which is greater than every
element of $S$ if $S$ is not bounded above and appending an element $\bot$ to $S$ less than every element of $S$ if $S$ is not bounded below.

There is no guarantee that if we fix a set and a partial ordering that every two elements will have a well-defined meet and join. We will be interested in partial orderings which produce meet and join operations which are total. A partially ordered set $\langle S, \leq \rangle$ is said to be a join semilattice if $x \lor y$ is defined for every $x, y \in S$ and is said to be a meet semilattice if $x \land y$ is defined for every $x, y \in S$. Joins, and by a similar definition meets, may be extended to finite sets by the recursive definition

\begin{enumerate}
  \item $\lor \{x_1, \ldots, x_n\} = \lor \{x_2, \ldots, x_n\} \lor x_1$ if $n > 1$
  \item $\lor \{x_1\} = x_1$
\end{enumerate}

This is possible because $\lor$ and $\land$ are associative, idempotent and commutative, thus insuring bounds for arbitrary finite subsets of a semilattice. A partially ordered set $\langle S, \leq \rangle$ is said to be a lattice if it is both a meet and join semilattice. While a lattice guarantees the existence of least-upper bounds and greatest-lower bounds for finite subsets, we still have no guarantee that there will be least upper bounds and greatest lower bounds for arbitrary (possibly infinite) sets. A join semilattice $\langle S, \leq \rangle$ is said to be a complete if for every $T \subseteq S$ we have some $x \in S$ such that $x = \lor T$, and similarly for meet semilattices. A lattice is said to be complete if it is both a complete meet and join semilattice. Somewhat surprisingly, any complete join semilattice forms a complete lattice. It is only necessary to define the meet operation over arbitrary sets by

\begin{equation}
\land T = \lor \{x \mid x \geq T\}
\end{equation}

Dually, any complete meet semilattice forms a complete lattice.

A lattice $L$ is said to be modular if it meets the two equivalent conditions

\begin{enumerate}
  \item $(x \land y) \lor (x \land z) = x \land (y \lor (x \land z))$
  \item $(x \land y) \lor z = x \land (y \lor z)$ if $z \leq x$.
\end{enumerate}

A lattice $L$ is said to be distributive if it meets the equivalent conditions
We note that every distributive lattice is modular, but not conversely.

If \( x \) and \( y \) are elements of the bounded lattice \( L \) we say that \( x \) is a complement of \( y \) if \( x \lor y = T \) and \( x \land y = \bot \). A lattice is said to be complemented if every element of the lattice has a complement. A lattice is said to be a boolean lattice if it is both complemented and distributive.

We note that an element \( x \) of a distributive lattice can have at most one complement, which we will label \( x' \) if it exists. Distributive lattices also respect DeMorgan’s Laws

\[
\begin{align*}
(64) & \quad \text{i. } (x \land y) \lor (x \land z) = x \land (y \lor z) \\
& \quad \text{ii. } (x \lor y) \land (x \lor z) = x \lor (y \land z) \\
& \quad \text{iii. } (x \lor y) \land z \leq x \lor (y \land z).
\end{align*}
\]

Suppose \( (S_1, \leq_1) \) and \( (S_2, \leq_2) \) are partially ordered sets. A total function \( \phi : S_1 \to S_2 \) is said to be monotonic if

\[
(66) \quad x \leq_1 y \implies \phi(x) \leq_2 \phi(y) \quad \text{(monotonicity)}
\]

for every \( x, y \in S_1 \). A one-one monotonic function is called an order embedding of \( S_1 \) in \( S_2 \). A one-one onto monotonic function \( \phi \) is said to be a order isomorphism of \( S_1 \) onto \( S_2 \). In this case, \( S_1 \) and \( S_2 \) are said to be order isomorphic, or equivalent up to isomorphism, and \( \phi \) is said to witness their isomorphism. If \( S_1 \) is isomorphic to \( S_2 \), we write \( S_1 \cong S_2 \). The order isomorphism relation between orderings is such that

\[
(67) \quad \text{i. } \text{id}_S \text{ is an isomorphism on any partially ordered set } S, \\
\quad \text{ii. } \text{If } \phi : S_1 \to S_2 \text{ is an isomorphism, then so is } \phi^{-1} : S_2 \to S_1, \\
\quad \text{and} \\
\quad \text{iii. } \text{If } \phi : S_1 \to S_2 \text{ and } \psi : S_2 \to S_3 \text{ are isomorphisms then so is } \psi \circ \phi : S_1 \to S_3.
\]
From this it directly follows that \( \equiv \) is an equivalence relation on the collection of partial orderings.

Just as we did with sets, we can define operations on orderings which form new orderings. Suppose \( S = (S, \leq_S) \) and \( T = (T, \leq_T) \) are partial orders, and consider the following partial orders constructed from \( S \) and \( T \)

\[
S \otimes T = (S \times S', \leq_{S \times S'}) \text{ where } (s, t) \leq_{S \times S'} (s', t') \text{ if and only if } s \leq_S s' \text{ and } t \leq_T t'.
\]

\[
S \oplus T = ((S \times \{1\}) \cup (S \times \{2\}) \cup \{\perp, \top\}, \leq_{S \oplus T}) \text{ such that } x \leq_{S \oplus T} y \text{ if and only if } x = (s, 1), y = (s', 1) \text{ and } s \leq_S s',
\]

or \( x = (t, 2), y = (t', 2) \) and \( t \leq_T t' \), or \( x = \perp \) or \( y = \top \).

We call \( S \otimes T \) the product of \( S \) and \( T \) and \( S \oplus T \) their sum. We note that \( \otimes \) and \( \oplus \) are associative and symmetric operations up to isomorphism, so that for arbitrary partial orderings \( S, T, U \) we have

\[
S \otimes (T \otimes U) \cong (S \otimes T) \otimes U \text{ (associativity)}.
\]

We note that if \( S \) and \( T \) are (complete) lattices then so are their sum and product.

### 1.3.7 Algebras and Lattices

A functional signature \( \Upsilon \) is an indexed set \( \Upsilon = \bigcup_{n \in \omega} \Upsilon_n \) of function symbols where we say that \( f \in \Upsilon_n \) is a function symbol of arity \( n \). Constants correspond to function symbols of arity 0 in \( \Upsilon_0 \).

An algebra over the functional signature \( \Upsilon \) is a pair \( (S, O) \) where \( S \) is a set called the carrier and \( O = \{ \sigma_f \mid f \in \Upsilon \} \) is a set of operations where \( \sigma_f : S^n \to S \) is an \( n \)-place operation if \( f \in \Upsilon_n \).

We will assume from now on that we are working with a fixed functional signature \( \Upsilon = \{ f_1, \ldots, f_n \} \), since the notions we will be interested in relate algebras of a given signature rather than cutting across signatures. We will write an algebra over \( \Upsilon \) as \( (S, \sigma_1, \ldots, \sigma_n) \), where \( \sigma_f = \sigma_i \), or even as \( S \) when the operations are understood.
We say that the algebra \((T, \tau_1, \ldots, \tau_n)\) forms a subalgebra of \((S, \sigma_1, \ldots, \sigma_n)\) if \(\tau_i(\vec{x}) = \sigma_i(\vec{x})\) for every sequence \(\vec{x} \in T^n\) of length \(n\) over \(T\) if \(\sigma_i\) and \(\tau_i\) are \(n\)-place operations.

If \(S\) is an algebra and \(T \subseteq S\) we set

\[(T) = \bigcap \{L \mid T \subseteq L, L \text{ a subalgebra of } S\}\]

and note that \([T]\) is the smallest subalgebra of \(S\) containing \(T\). We call \([T]\) the subalgebra generated by \(T\). We say that \([T]\) is independently generated by \(T\) if there is no proper subset \(U \subset T\) such that \([T] = [U]\).

Suppose that \((T, \tau_1, \ldots, \tau_n)\) and \((S, \sigma_1, \ldots, \sigma_n)\) are both algebras over the functional signature \(T\). A total function \(\phi: S \rightarrow T\) is said to be a homomorphism if

\[(70) \quad \phi(\sigma_i(s_1, \ldots, s_n)) = \tau_i(\phi(s_1), \ldots, \phi(s_n))\]

if \(\sigma_i\) and \(\tau_i\) are \(n\)-place operations and \(s_k \in S\) for \(1 \leq k \leq n\). If \(\phi\) is onto then we say that \(T\) is a homomorphic image of \(S\). A one-one homomorphism is called an embedding. A homomorphism which is both one-one and onto is called an isomorphism. We write \(S \cong T\) if there is an isomorphism from \(S\) to \(T\), or equivalently from \(T\) to \(S\), and say that \(S\) is isomorphic to \(T\). Isomorphism will be an equivalence relation, since again, the identity function is an isomorphism, inverses of isomorphisms are isomorphisms and the composition of isomorphisms is an isomorphism.

An equivalence relation \(\Theta\) on the carrier set \(S\) of an algebra \((S, \sigma_1, \ldots, \sigma_n)\) is said to be a congruence relation if

\[(72) \quad \sigma_i(s_1, \ldots, s_k) = \sigma_i(t_1, \ldots, t_k)\]

whenever \(\sigma_i\) is an \(k\)-place operation and \(s_j, t_j \in S\) for \(1 \leq j \leq k\). We call the equivalence class \([s]_\Theta\) of \(s\) its congruence class when \(\Theta\) is a congruence relation.

Suppose \(\Theta\) is a congruence relation over \(S\). Then \((S/\Theta, \tau_1, \ldots, \tau_n)\) forms an algebra under the definitions

\[(73) \quad \tau_i([s_1]_\Theta, \ldots, [s_k]_\Theta) = [\sigma_i(s_1, \ldots, s_k)]_\Theta\]
where \( \tau_i \) is a \( k \)-place relation. Note that the operation \([\cdot]_{\Theta} \) is a homomorphism from \( S \) onto \( S/\Theta \).

Lattices can be characterised as algebras with operations corresponding to meet and join. We will say that an algebra \( \langle S, o \rangle \) with a binary operator \( o \) is a semilattice if \( o \) is idempotent, commutative and associative. This means that if \( \langle S, o \rangle \) is a semilattice we know that for every \( x, y, z \in S \)

\[
(74) \quad \begin{align*}
\text{i. } x \circ x &= x \quad \text{(idempotency)} \\
\text{ii. } x \circ y &= y \circ x \quad \text{(commutativity)} \\
\text{iii. } x \circ (y \circ z) &= (x \circ y) \circ z \quad \text{(associativity)}.
\end{align*}
\]

If \( \langle S, o \rangle \) is a semilattice, then \( \langle S, o \rangle^{P_{\lor}} = \langle S, \leq_{\lor} \rangle \) is a join semilattice and \( \langle S, o \rangle^{M_{\land}} = \langle S, \leq_{\land} \rangle \) is a meet semilattice, where

\[
(75) \quad \begin{align*}
\text{i. } x \leq_{\lor} y &\quad \text{if and only if } x \circ y = y \\
\text{ii. } x \leq_{\land} y &\quad \text{if and only if } x \circ y = x.
\end{align*}
\]

If \( \langle S, \leq \rangle \) is a join semilattice, then \( \langle S, \leq \rangle^{A_{\lor}} = \langle S, \lor \rangle \) is a semilattice, where \( \lor \) is the total least upper bound operation in \( \langle S, \leq \rangle \). Similarly, if \( \langle S, \leq \rangle \) is a meet semilattice, then \( \langle S, \leq \rangle^{A_{\land}} = \langle S, \land \rangle \) is a semilattice, where \( \land \) is the total greatest lower bound operation in \( \langle S, \leq \rangle \). Furthermore, \( \langle \langle S, \leq \rangle^{A_{\lor}} \rangle^{P_{\lor}} = \langle S, \leq \rangle \) if \( \langle S, \leq \rangle \) is a join semilattice, and \( \langle \langle S, o \rangle^{P_{\lor}} \rangle^{A_{\lor}} = \langle S, o \rangle \) if \( \langle S, o \rangle \) is a semilattice, and similarly for meet semilattices.

We say that the algebra \( \langle S, \lor, \land \rangle \) is a lattice if \( \langle S, \lor \rangle \) and \( \langle S, \land \rangle \) are semilattices and the absorption identities

\[
(76) \quad \begin{align*}
\text{i. } x \land (x \lor y) &= x \quad \text{(absorption)} \\
\text{ii. } x \lor (x \land y) &= x \quad \text{(absorption)}
\end{align*}
\]

are satisfied.

If \( \langle S, \lor, \land \rangle \) is a lattice, then \( \langle S, \lor, \land \rangle^{P} = \langle S, \leq \rangle \) is a lattice, where we set \( x \leq y \) if and only if \( x \lor y = y \), which, in the context of lattices, holds if and only if \( x \land y = x \). We define the lattice \( \langle S, \leq \rangle^{A} = \langle S, \lor, \land \rangle \) similarly, with \( \lor \) and \( \land \) taken to be least upper bound and greatest lower bound respectively. Furthermore, we have \( \langle \langle S, \lor, \land \rangle^{P} \rangle^{A} = \langle S, \lor, \land \rangle \) where \( \langle S, \lor, \land \rangle \) is a lattice and \( \langle \langle S, \leq \rangle^{A} \rangle^{P} = \langle S, \leq \rangle \).
where \( (S, \leq) \) is a lattice. With this in mind, we will assume that a lattice comes with an ordering relation and corresponding meet and join operations, since either can be derived from the other.

We note that lattice homomorphisms are monotonic in the partial orderings corresponding to the semilattices, but not conversely.

In a semilattice \( (S, \circ) \) we say that \( i \in S \) is an identity on \( S \) if

\[
i \circ x = x \circ i = x \quad \text{(identity)}
\]

for every \( x \in S \). If identities exist then they are unique. We note that if \( (S, \circ) \) is a semilattice with identity \( i \) then \( i \) is the least element of \( (S, \circ)^{P\circ} \) and greatest element of \( (S, \circ)^{P\circ} \). Similarly, if \( (S, \leq) \) is a meet or join semilattice with \( \bot \) or \( \top \) then we note that \( \top \) is the identity of \( (S, \leq)^{A\leq} \) and \( \bot \) is the identity of \( (S, \leq)^{A\leq} \).

A boolean algebra is a tuple \( (B, \lor, \land, ', \bot, \top) \) where \( (B, \lor, \land) \) is a complemented distributive lattice with a unary complement operation ' and least and greatest elements \( \bot \) and \( \top \), which we may think of as nullary operations with arity 0. In the context of boolean algebras, \( \bot \) and \( \top \) are sometimes called the zero and unit elements and written as 0 and 1. An element \( x \) of a boolean algebra is said to be an atom if \( \bot < x \).

By a ring of sets, we mean a set \( T \) of sets such that \( x \cup y, x \cap y \in T \) if \( x, y \in T \). Every distributive lattice is isomorphic to some ring of sets. Suppose \( L \) is a distributive lattice and \( T \) is a ring of sets isomorphic to it. Now consider \( 2^UT \) which forms a boolean lattice under the subset relation \( \subseteq \). Since the identity function is an embedding of \( T \) in this boolean lattice, it follows that any distributive lattice can be embedded in a boolean lattice. A field of sets is a ring of sets \( T \) such that \( x \in T \) only if \((\cup T \setminus x) \in T \). It is a well-known theorem of boolean algebras, due to Stone (1936), that any boolean algebra is isomorphic to some field of sets.

Suppose \( S \) is a lattice. A subset \( I \) of \( S \) is said to be an ideal of \( S \) if

\[
\begin{align*}
\text{i.} & \quad x \lor y \in I \text{ if } x, y \in I \\
\text{ii.} & \quad x \in I \text{ if } x \in S, y \in I, \text{ and } x \leq y.
\end{align*}
\]

An ideal \( I \) of \( S \) is said to be proper if \( I \neq S \). A proper ideal \( I \) of \( S \) is said to be prime if we have \( x \in I \) or \( y \in I \) whenever we have \( x \land y \in I \). If \( T \subseteq S \), we let \( [T]_I \)
be the smallest ideal containing \( T \), and say that \( T \) generates \([T]_1\). Such a smallest ideal can be found since \( S \) is an ideal of \( S \) containing \( T \) and the intersection of an arbitrary set of ideals is itself an ideal. We will shorten \([\{x\}]_1\) to \([x]_1\) or just \( \downarrow x \), which is called the principal ideal generated by \( x \), and note that

\[
79 \quad \downarrow x = \{ y \mid y \leq x \}.
\]

We let \( \mathbf{I}(L) \) denote the set of all ideals of \( L \) and note that it forms a lattice under \( \supseteq \). If \( L \) is bounded below, then \( \mathbf{I}(L) \) is a complete lattice. \( \mathbf{I}(L) \) is also modular (distributive) if \( L \) is modular (distributive). As \( L \) may not be bounded below, we let

\[
80 \quad \mathbf{I}_1(L) = \mathbf{I}(L)^+ = \mathbf{I}(L) \cup \{\emptyset\},
\]

which is always a complete lattice. Let \( L \) be a lattice and, as usual, set

\[
81 \quad \downarrow L = \{ \downarrow x \mid x \in L \}.
\]

We notice that \( \downarrow L \) forms a sublattice of \( \mathbf{I}(L) \) under \( \supseteq \). Furthermore, the natural mapping \( \downarrow : L \to (\downarrow L) \) is in fact an isomorphism of \( (L, \leq) \) onto \( (\downarrow L, \supseteq) \) and hence an embedding of \( L \) in \( \mathbf{I}(L) \) and \( \mathbf{I}_1(L) \), so that any lattice may be naturally embedded in a complete lattice.

Switching \( \lor \) for \( \wedge \), \( \leq \) for \( \geq \) and \( \uparrow \) for \( \downarrow \) in the above definitions leads to what are called filters, prime filters, principal filters and so on. For instance, the principal filter generated by \( x \) is given by \( \uparrow x = \{ y \mid y \geq x \} \) In general, carrying out such a switch on a statement leads to what is called the dual of a statement.

In the theory of lattices it is useful to note that any statement that holds of all lattices is such that its dual holds of all lattices. This is true because if \( (L, \lor, \land) \) is a lattice then so is \( (L, \land, \lor) \). Partial orders respect the duality principal, as well, since \( (S, \geq) \) is a partially ordered set if \( (S, \leq) \) is.

For a more detailed introduction to the theory of lattices, see Grätzer’s (1971) book. For a thorough treatment of algebras, see Cohn’s (1965) book.
1.3.8 Typed $\lambda$-Calculus

Types and Terms

To begin, we fix some non-empty set $\text{BASTYP}$ of basic type symbols. We then define the set $\text{TYP}(\text{BASTYP})$ of type symbols freely generated from $\text{BASTYP}$ to be the smallest set such that

\[(82) \begin{aligned} 
\text{i. } & \text{BASTYP} \subseteq \text{TYP}(\text{BASTYP}) \\
\text{ii. } & \langle \sigma, \tau \rangle \in \text{TYP}(\text{BASTYP}) \text{ if } \sigma, \tau \in \text{TYP}(\text{BASTYP}), 
\end{aligned} \]

so that $\text{TYP}$ is the minimal solution to the equation

\[(83) \text{TYP} = \text{BASTYP} \cup (\text{TYP}(\text{BASTYP}) \times \text{TYP}(\text{BASTYP})). \]

A type $\langle \sigma, \tau \rangle$ is called a functional type. We will write $\text{TYP}(\text{BASTYP})$ as $\text{TYP}$ when $\text{BASTYP}$ is understood.

We next fix a set of variables $\mathcal{V} = \bigcup_{\tau \in \text{TYP}} \mathcal{V}_\tau$ indexed by type, where the set of variables of type $\tau \in \text{TYP}$ are given by $\mathcal{V}_\tau = \{ v^n_\tau \mid n \in \omega \}$. We will also fix a similar set of constant symbols $\mathcal{C} = \bigcup_{\tau \in \text{TYP}} \mathcal{C}_\tau$ where the constants of type $\tau \in \text{TYP}$ are given by $\mathcal{C}_\tau = \{ c^n_\tau \mid n \in \omega \}$.

We now define the set $\mathcal{T} = \bigcup_{\tau \in \text{TYP}} \mathcal{T}_\tau$ of $\lambda$-terms indexed by type by simultaneous recursion, taking the smallest sets $\mathcal{T}_\tau$ of terms of type $\tau$ such that

\[(84) \begin{aligned} 
\text{i. } & \mathcal{C}_\tau \subseteq \mathcal{T}_\tau \\
\text{ii. } & \mathcal{V}_\tau \subseteq \mathcal{T}_\tau \\
\text{iii. } & \alpha(\beta) \in \mathcal{T}_\tau \text{ if } \alpha \in \mathcal{T}_{(\sigma, \tau)} \text{ and } \beta \in \mathcal{T}_{\sigma} \\
\text{iv. } & \lambda x.\alpha \in \mathcal{T}_{(\sigma, \tau)} \text{ if } x \in \mathcal{V}_{\sigma} \text{ and } \alpha \in \mathcal{T}_{\tau}. 
\end{aligned} \]

We will use the following abbreviations when writing $\lambda$-terms

\[(85) \begin{aligned} 
\text{i. } & \lambda x_1.\lambda x_2.\ldots.\lambda x_n.\alpha \text{ for } \lambda x_1.(\lambda x_2.\ldots. (\lambda x_n.\alpha)\ldots) \\
\text{ii. } & \alpha_0(\alpha_1)(\alpha_2)\cdots(\alpha_n) \text{ for } (\cdots((\alpha_0(\alpha_1))(\alpha_2))\cdots)(\alpha_n). 
\end{aligned} \]

We define the free variables $\text{FreeVar}(t)$ of a $\lambda$-term $t$ inductively by
\( \text{i. } \text{FreeVar}(c) = \emptyset \text{ if } c \in \mathcal{C} \)

\( \text{ii. } \text{FreeVar}(v) = \{v\} \text{ if } v \in \mathcal{V} \)

\( \text{iii. } \text{FreeVar}(\alpha(\beta)) = \text{FreeVar}(\alpha) \cup \text{FreeVar}(\beta) \)

\( \text{iv. } \text{FreeVar}(\lambda x.\alpha) = \text{FreeVar}(\alpha) \setminus \{x\} \)

\( \lambda \)-terms with no free variables are said to be \textit{closed}. We call a \( \lambda \)-term without any constants a \textit{pure} term, and note that it is possible for pure terms to be closed. An occurrence of \( x \in \text{FreeVar}(\alpha) \) is said to be \textit{bound} by \( \lambda x \) in \( \lambda x.\alpha \). Every variable \( x \) which occurs in a \( \lambda \)-term \( t \) is either free or bound by some \( \lambda x \). We will say that \( \lambda x \) is a \textit{vacuous binding} in \( \lambda x.\alpha \) if there are no free occurrences of \( x \) in \( \alpha \). We will equate two terms that differ only in the names of their free variables. This means that, for instance, \( \lambda x.x = \lambda y.y \), and \( \lambda x.z = \lambda y.z \).

A \textit{substitution} is an indexed function \( \phi = \bigcup_{r \in \text{Typ}} \phi_r \) where \( \phi_r : \mathcal{V}_r \rightarrow \mathcal{T}_r \) is a function from variables of type \( \tau \) to terms of type \( \tau \). If \( \alpha \) is a term, we write \( \alpha[\phi] \) for the result of replacing every free occurrence of a variable in \( \alpha \) with its image under \( \phi \) in the usual way, as long as no free variables in the range of \( \phi \) subsequently become bound in \( \alpha[\phi] \). That is,

\begin{align*}
(86) & \quad \text{i. } c[\phi] = c \text{ if } c \in \mathcal{C} \\
& \quad \text{ii. } v[\phi] = \phi(v) \text{ if } v \in \mathcal{V} \\
& \quad \text{iii. } \alpha(\beta)[\phi] = \alpha[\phi](\beta[\phi]) \\
& \quad \text{iv. } (\lambda x.\alpha)[\phi] = \lambda x.(\alpha[\phi_x])
\end{align*}

where, in general, \( \phi_x \) is the substitution differing from \( \phi \) only in that it maps \( x \) to \( \alpha \), so that

\[ \phi_x = (x,\alpha) \cup (\phi \setminus (x, \phi(x))). \]

We also write \([v_1, \ldots, v_n]/(\alpha_1, \ldots, \alpha_n)] \) for the substitution mapping \( v_i \) to \( \alpha_i \) and every other variable to itself, as long as \( v_i \neq v_j \) if \( i \neq j \).

When we compute \( \alpha[x/\beta] \), we can simply rename any variables so that the variables occurring in \( \alpha \) and \( \beta \) are disjoint, so that, for instance, \((\lambda x.y)[y/f(x)] = \lambda z.f(x)\), where we have replaced the term \( \lambda x.y \) with the identical \( \lambda z.y \).
Theory

If \( \alpha, \beta \) are \( \lambda \)-terms, then we say that \( \alpha \leftrightarrow \beta \) is a \( \lambda \)-formula or \( \lambda \)-statement. A \( \lambda \)-formula \( \alpha \leftrightarrow \beta \) states that the terms \( \alpha \) and \( \beta \) are the same function. A statement is open or closed based on its terms.

A rule of inference is of the form \( \Gamma \Rightarrow \gamma \) where \( \Gamma \) is a set of statements and \( \gamma \) is a statement. \( \Gamma \) is the antecedent and \( \gamma \) the consequent in the rule \( \Gamma \Rightarrow \gamma \). A \( \lambda \)-axiom is a rule of inference with an empty antecedent. Rules or axioms with free variables are often called rule schemata or axiom schemata. This is because they do the same work as the set of all of their substitutions would do. In the context of linguistic semantics, a \( \lambda \)-axiom is often called a meaning postulate. We often write a rule of the form \( \{ \gamma_1, \ldots, \gamma_n \} \Rightarrow \gamma \) as

\[
(89) \quad \gamma_1, \ldots, \gamma_n \Rightarrow \gamma
\]

and an axiom of the form \( \{ \} \Rightarrow \gamma \) simply as \( \gamma \).

A set of statements \( T \) is said to be closed under the rule of inference \( \Gamma \Rightarrow \gamma \) if \( \Gamma[\phi] \subseteq T \) implies \( \gamma[\phi] \in T \) for every substitution \( \phi \).

Let \( R \) be a set of rules, \( \Gamma \) be a set of formulas and \( \gamma \) be a formula. We define the consequences of \( \Gamma \) in \( R \) to be the minimal set \( \text{THR}(\Gamma) \) such that \( \Gamma \subseteq \text{THR}(\Gamma) \) which is closed under the rules in \( R \).

We write \( \Gamma \vdash_R \gamma \) and say that \( \gamma \) is a consequence of \( \Gamma \) in \( R \) if \( \gamma \in \text{THR}(\Gamma) \).

We write \( \vdash_R \gamma \) for \( \{ \} \vdash_R \gamma \). We say that \( \alpha \leftrightarrow \beta \) are \( R \)-equivalent if \( \vdash_R \alpha \leftrightarrow \beta \).

A set \( R \) of rules is often called a theory and is often identified with the set \( \text{THR} = \text{THR}(\{ \}) \) of its consequences. A theory is consistent if the set of its consequences is a proper subset of the set of terms.

We can give a positive account of \( \text{THR}(\Gamma) \) by setting

\[
(90) \quad \text{THR}(\Gamma) = \bigcup_{n \in \omega} \text{THR}(\Gamma)_n
\]

with the inductive definitions

\[
(91) \quad \begin{align*}
&i. \quad \text{THR}(\Gamma)_0 = \Gamma \\
&ii. \quad \text{THR}(\Gamma)_{n+1} = \text{THR}(\Gamma)_n \cup R(\text{THR}(\Gamma)_n)
\end{align*}
\]
where

\( R(T) = \{ \gamma[\phi] \mid \Gamma[\phi] \subseteq T, \ \Gamma \Rightarrow \gamma \in R \} \).

We think of \( R(T) \) as the set of theorems that can be proved from \( T \) in \( R \) in one step. A proof of a theorem \( \gamma \) from the assumptions \( \Gamma \) in the theory \( R \) is a sequence \( p_1, \ldots, p_n \) of formulas, where \( \gamma = p_n \) and

\[ p_i \in \Gamma \cup R(\{p_k \mid k < i\}) \tag{93} \]

for \( 1 \leq i \leq n \). Obviously, a formula \( \gamma \) is a consequence of \( \Gamma \) in \( R \) just in case there is a proof of \( \gamma \) from \( \Gamma \) in \( R \).

The \( \lambda \)-calculus consists of the axioms

\[ \tag{94} \]

i. \((\lambda x.\alpha)(\beta) \leftrightarrow \alpha[x/\beta] \) (\( \beta \) conversion)

ii. \( \delta \leftrightarrow \lambda x.\delta(x) \) (\( \eta \) conversion)

iii. \( \alpha \leftrightarrow \alpha \) (identity)

where \( x, y \in \mathcal{V}_r, \beta \in \mathcal{T}_r, \alpha \in \mathcal{T}, \) and \( \delta \in \mathcal{T}_{(\sigma, \tau)} \), and \( \sigma, \tau \in \mathcal{TYP}, \) and the rules of inference

\[ \tag{95} \]

i. \( \frac{\alpha_1 \leftrightarrow \alpha_2}{\alpha_2 \leftrightarrow \alpha_1} \) (symmetry)

ii. \( \frac{\alpha_1 \leftrightarrow \alpha_2, \alpha_2 \leftrightarrow \alpha_3}{\alpha_1 \leftrightarrow \alpha_3} \) (transitivity)

iii. \( \frac{\alpha_1 \leftrightarrow \alpha_2, \beta_1 \leftrightarrow \beta_2}{\beta_1(\alpha_1) \leftrightarrow \beta_2(\alpha_2)} \) (congruence)

iv. \( \frac{\alpha_1 \leftrightarrow \alpha_2}{\lambda x.\alpha_1 \leftrightarrow \lambda x.\alpha_2} \) (congruence)

v. \( \frac{\alpha \leftrightarrow \beta}{\alpha[\phi] \leftrightarrow \beta[\phi]} \) (substitution)

where \( \alpha_i \) is a term of type \( \tau \) and \( \beta_i \) is a term of type \( (\tau, \sigma) \) and \( x \) is a variable of type \( \rho \) for some types \( \rho, \sigma \) and \( \tau \), and \( \phi \) is any substitution. We will write \( \Lambda \) for the rules of the \( \lambda \)-calculus and \( \Lambda + R \) for \( \Lambda \cup R \) if \( R \) is a set of rules. We note that the theory \( \Lambda \) is consistent, as it will obviously not contain any formula of the form \( c \leftrightarrow c' \) where \( c \) and \( c' \) are distinct constants.
We note that every λ-term has a normal form. We define the normal form $\text{Norm}(\alpha)$ of a term $\alpha$ inductively so that

\begin{align*}
\text{(96)} & \\
\text{i. } & \text{Norm}(\alpha) = \alpha \text{ if } \alpha \text{ is a constant or variable} \\
\text{ii. } & \text{Norm}(\lambda x.\alpha(\beta)) = \text{Norm}(\alpha[x/\beta]). \\
\text{iii. } & \text{Norm}(\alpha(\beta)) = \text{Norm}(\alpha)(\text{Norm}(\beta))
\end{align*}

Further note that every term is provably equivalent to its normal form in $\Lambda$ so that

\begin{align*}
\text{(97)} & \\
& \vdash_{\Lambda} \alpha \leftrightarrow \text{Norm}(\alpha)
\end{align*}

for every term $\alpha \in T$.

Models

A frame for the types $TYP$ is an indexed set $D = \bigcup_{\tau \in TYP} D_\tau$ where $D_\tau$ is the domain of possible denotations for terms of of type $\tau$, where

\begin{align*}
\text{(98)} & \\
& D_{(\sigma, \tau)} = D_\tau^{D_\sigma}.
\end{align*}

Note that the $D_\tau$ may be chosen arbitrarily for the basic types $\tau \in \text{BASTYP}$ and such a choice will fully determine the domains for the functional types.

We can now define a model for the typed λ-calculus to be a pair $M = (D, \nu)$ where $D$ is a frame and where $\nu$ is an indexed valuation function $\nu = \bigcup_{\tau \in TYP} \nu_\tau$ where $\nu_\tau : C_\tau \rightarrow D_\tau$ is a total function for each $\tau \in TYP$.

Before giving the definition of the value of a term in a model, we first define an assignment to be an indexed function $a = \bigcup_{\tau \in TYP} a_\tau$ where $a_\tau : \mathcal{V}_\tau \rightarrow D_\tau$ is some total function for each $\tau \in TYP$.

The value or denotation of a λ-term in a model $M$ under the assignment $a$ is then given inductively by

\begin{align*}
\text{(99)} & \\
\text{i. } & [c]_a^M = \nu(c) \text{ if } c \in \mathcal{C} \\
\text{ii. } & [x]_a^M = a(x) \text{ if } x \in \mathcal{V} \\
\text{iii. } & [\alpha(\beta)]_a^M = [\alpha]_a^M([\beta]_a^M) \\
\text{iv. } & [\lambda x.\alpha]_a^M = \{(s, t) \in D_\tau \times D_\sigma \mid t = [\alpha]_s^\sigma\}
\end{align*}
where \( a^x \) is just like \( a \) except that it maps \( x \) to \( s \), or more precisely

\[
(100) \quad a^x = (a \setminus \{(x, a(x))\}) \cup \{(x, s)\}.
\]

It can be routinely verified that the value of a closed \( \lambda \)-term in a model is not dependent on the assignment function, so that if \( a \) is closed, \( [\alpha]^M_a = [\alpha]^M_{a'} \) for all assignment functions \( a \) and \( a' \). We use the shorthand \( [\alpha]^M \) for this value. Similarly, a pure term without any constants will only depend on the frame \( D \) and not on the valuation function, and we write \( [\alpha]^D_a \) for the value of such a term under the assignment \( a \), and simply \( [\alpha]_a \) when the frame \( D \) is understood. Similarly, if \( \alpha \) is both closed and pure, we write \( [\alpha]^P \) for the denotation of \( \alpha \) in the frame \( D \), and simply \( [\alpha] \) if \( D \) is understood.

We define the satisfaction relation so that

\[
(101) \quad \vdash^M \alpha \leftrightarrow \beta \text{ iff } [\alpha]^M_a = [\beta]^M_a \text{ for every assignment } a
\]

and say that \( \alpha \) and \( \beta \) are equivalent in \( M \) and that the formula \( \alpha \leftrightarrow \beta \) is satisfied by \( M \).

Suppose we fix a collection \( M \) of models. We write \( \models^M \alpha \leftrightarrow \beta \) and say that \( \alpha \) and \( \beta \) are \( M \)-equivalent and that the formula \( \alpha \leftrightarrow \beta \) is \( M \)-valid if \( \models^M \alpha \leftrightarrow \beta \) for every model \( M \in M \). We will usually be interested in the collection \( M \) of models of a particular \( \lambda \)-theory \( \Lambda + T \).

Suppose we have fixed a collection of models \( M \) and a theory \( T \). We say that \( T \) is sound with respect to \( M \) if

\[
(102) \quad \vdash^T \gamma \text{ implies } \models^M \gamma \quad \text{(soundness)}
\]

and say that it is complete if we have the converse

\[
(103) \quad \models^M \gamma \text{ implies } \vdash^T \gamma \quad \text{(completeness)}
\]

We note that the \( \lambda \)-calculus is sound with respect to any collection \( M \) of models.

The typed \( \lambda \)-calculus was introduced by Church (1940) and the main results concerning its syntax and semantics were proved by Henkin (1950). For more details, see Andrews (1986).
Chapter 2

Phrases, Categories and Features

In this chapter, we will introduce three of the primary objects used in natural language grammars, namely categories, features and phrase structure rules.

In the first section, we introduce standard phrase structure grammars. After presenting the basics of phrase structure and looking at a simple example, we discuss three interpretations of phrase structure grammars. First, we discuss an interpretation of the rules as rewriting or analysis axioms in a deductive system. We then discuss an interpretation of the rules as admissibility conditions on parse trees. The last interpretation we consider is by means of a direct denotational semantics for the phrase structure formalism itself, which provides insight into the notion of category employed in modern grammars. The last part of the first section deals with possible restrictions on the phrase structure formalism, and in particular with the restriction of finiteness that leads to the context-free grammars. Then we define the concepts of strong and weak equivalence which provide alternative measures for deciding when two grammars represent the same language.

The second section deals primarily with the encoding of partial information in the categories and rules of a phrase structure grammar. The ability to represent partial information leads to a perspicuous encoding of categories which can appear in a number of different contexts and allows for a sharp reduction in the number of rules necessary to analyse a language. We develop a now standard method of encoding partial information in linguistic descriptions by means of categories based
on features and their partial values. We then provide a grammar formalism based on these categories and show how it can be reduced to a strongly equivalent phrase structure grammar. Next, we introduce first-order functional terms common to logical programming languages such as PROLOG and show how partial information embodied in the feature-value formalism can be translated into logical terms. We show that every feature structure category domain can be represented as a term domain, but also show how the converse fails. Using a standard grammar formalism for these terms based on definite clause grammars, we prove that every feature structure grammar is strongly equivalent to some term grammar and conversely. We also discuss some extensions to the unification formalism dealt with in the literature by means of feature structures and show how they can be naturally encoded in extensions of the term domains.

In the final section, we introduce the categorial grammar framework. Starting with the pure categorial grammars, we provide a grammar weakly equivalent to our phrase structure grammar example. Then we show how partial information can be naturally encoded in categorial grammars. By restricting feature information to basic categories, we have a grammar formalism which is powerful enough to capture complex agreement and semantic information, but is also effectively parsable. This contrasts with the general term and feature grammars, which are powerful enough to encode any recursively enumerable language.

2.1 Phrase Structure

We will begin with a discussion of what we take phrase structure to be in the pre-theoretic sense and then introduce the fundamental definition of a phrase structure grammar. By way of example, we include an elementary phrase structure grammar for a small fragment of English. We next consider three possible interpretations for the phrase structure formalism and finally turn to a discussion of the properties of phrase structure grammars and one of their important subclasses, the context-free grammars.

In most grammar formalisms, it is assumed that there is some fixed set of primitive or basic expressions and means for combining them to form more com-
plex expressions. In morphology, the basic expressions are *morphemes*, which are assumed to be the most primitive meaningful units of language. For instance, the morphemes *-ly, un-, -ate*, and *fortune* would all be considered basic expressions, which can combine to form complex expressions such as *fortunate, unfortunate, fortunately*, and *unfortunately*. These acceptable expressions contrast with the unacceptable strings of morphemes found in *fortunately* and *unfortune*. In syntax, the basic expressions are words in the simplest case, but can also consist of strings of words in the case of idioms such as *put up, shut up*, and so on. Here we will be using the *orthographic* representation of words as strings of characters, but it would also be possible to carry out grammar construction on the basis of phonological representations of words. What is essential is just that there is some fixed finite stock of basic expressions.

It is important to note that the complex expressions of morphology make up the basic expressions of syntax. Similarly, the complex expressions of phonology make up the basic expressions of morphology. Dividing the labour between different components of a grammar is a general principle of modular grammar design. Deciding precisely where to draw the line between components is thus a significant design decision. This is especially true when one is restricting attention to a single component of grammar such as syntax. It is deceptively easy to suppose that a distinction found in the data is to be accounted for at another level of the grammar. When we make claims about the appropriate division of labour, say between semantics and pragmatics, we will be particularly careful to motivate and justify our decisions.

In both syntax and morphology, it is useful to develop some means for classifying expressions on the basis of the role they play in forming complex expressions. One useful method of categorisation used in syntax is in terms of traditional *parts-of-speech*, where the word *fortune* is classified as a noun, *fortunate* as an adjective, *fortunately* as an adverb, and so on. Similarly, we could divide sentences into subjects and predicates, just as in traditional "schoolbook" grammar (see, for instance, Quirk, Greenbaum, Leech and Svartik (1985), for a comprehensive traditional grammar). With such a categorisation, simple rules can be stated such as 'a sentence consists of a subject followed by a predicate'. Of course,
these simple categorisations are very coarse-grained and finer divisions will be necessary for serious investigation of linguistic structure. This chapter will be in part concerned with developing mathematical tools for providing such fine-grained categorisations.

Once our basic expressions are fixed and some method of categorisation is chosen, some means must be provided for stating rules that govern how complex expressions are built up out of basic expressions. The second concern of this chapter is with phrase structure and categorial grammars, which are two particular formalisms developed for precisely this purpose.

2.1.1 Phrase Structure Grammars

Phrase structure grammar, as its name implies, is a formalism designed to study the structure of phrases. Phrases are usually taken to be grammatical expressions composed of basic expressions and smaller phrases. Phrase structure grammars employ rules which characterise the structure of an expression in terms of the categorisations of the subexpressions out of which it is composed. For instance, a sentence is a type of phrase which can be composed of a subject phrase followed by a predicate phrase.

A particular phrase structure grammar will specify a set of basic expressions as well as a set of category symbols which represent the categories used in the classification of phrases based on their positional distribution. A phrase structure grammar will also specify what is called a lexical assignment, which provides categorisations of the basic expressions of the language. A lexical assignment is necessary in order to ground the recursive structure of a language in terms of the categorisations of its basic expressions. These are features that the phrase structure formalism shares with most grammar formalisms. What sets it apart is the way in which complex expressions are categorised. A phrase structure grammar will also contain, along with basic expressions, category symbols and a lexical assignment, a set of phrase structure rules. A particular phrase structure rule characterises a possible categorisation for an expression composed of a string of subphrases by inspecting the possible categorisations of each subphrase.
in the sequence. Taken together, the phrase structure rules and lexical assignment determine the ways in which strings of basic expressions may be categorised.

More formally, we take a phrase structure grammar to be a quadruple $\Gamma = (\Sigma, \Delta, \Phi, \Lambda)$, where

\[
\begin{align*}
(1) \quad & \Sigma \quad \text{set of category symbols} \\
& \Delta \quad \text{set of basic expressions} \\
& \Lambda \quad \text{a lexical assignment consisting of a} \\
& \quad \text{relation } \Lambda \subseteq \Sigma \times \Delta \\
& \Phi \quad \text{a subset } \Phi \subseteq \Sigma \times \Sigma^* \\
& \quad \text{of phrase structure rules}
\end{align*}
\]

We will use use the notation $s \overset{\Phi}{\rightarrow} s_1 \cdots s_n$, suppressing the $\Phi$ when it is understood, when the phrase structure rule $(s, (s_1, \ldots, s_n))$ is an element of $\Phi$ and say that $s$ can be re-written as $s_1 \cdots s_n$, or that $s_1 \cdots s_n$ can be reduced to $s$. The reason for this terminology is that we often think of a rule such as $a \rightarrow b \ c$ as stating that an expression of category $a$ can be composed of an expression of category $b$ followed by an expression of category $c$. This notion is made more precise in the section below concerning interpretations of the arrow $\rightarrow$. If $(c, w) \in \Lambda$ we say that $c$ is a lexical category of $w$ or that $w$ is a lexical entry of the category $c$. We will write $\Lambda(c)$ for the set of all expressions of category $c$, so that

\[
(2) \quad \Lambda(c) = \{ w \mid (c, w) \in \Lambda \}.
\]

### 2.1.2 Simple English Phrase Structure

Before continuing, it will be helpful to inject some concreteness into our theoretical discussion with an example. This section contains the definition of a particular phrase structure grammar which models some important structural properties of English syntax. We will assume the following correspondence between categories and parts of speech
We then take the lexical assignment

The intended interpretation is that the categories on the left are lexical categories of all of the words opposite them. This information could equally well have been represented by listing the lexical entries of each word in the fashion of a dictionary.
These rules tell us, for instance, that a sentence can be composed of a noun phrase followed by a verb phrase, and that a noun phrase, in turn, can itself be composed of a determiner followed by a noun, and so on. We will use this example grammar to illustrate the definitions in the next section.

### 2.1.3 Interpretations of Phrase Structure Grammars

There are many ways in which a phrase structure grammar can be interpreted. We will consider three of these, all of which will be of some use to us later. The first interprets the phrase structure rules as rewrite rules and construes a phrase structure grammar as a particular kind of deduction system. The second interprets the rules as so-called “tree-admissibility conditions” as used in Generalized Phrase Structure Grammar (Gazdar and Pullum 1981, Gazdar, Klein, Pullum and Sag 1985), where the rules act as a filter on the set of admissible parse trees. Finally, we will provide a denotational semantics for the phrase structure formalism itself of the sort found in Pereira and Shieber (1984). For a general introduction to phrase structure grammars and other related grammar formalisms, see Hopcroft and Ullman (1979), Lewis and Papadimitriou (1981), or Chomsky’s original papers (1956,1959,1963).

Throughout this section, we will assume that some phrase structure grammar $\Gamma = (\Sigma, \Delta, \Phi, \Lambda)$ has been fixed. All of our definitions will be relative to this grammar and when we wish to notate this explicitly, we include a subscript $\Gamma$. 

\[
(5) \quad s \rightarrow np \ vp \\
np \rightarrow det \ n \\
n \rightarrow n \ pp \\
n \rightarrow adj \ n \\
v_p \rightarrow iv \\
v_p \rightarrow tv \ np \\
v_p \rightarrow bv \ np \ np \\
v_p \rightarrow adv \ vp \\
v_p \rightarrow vp \ adv \\
v_p \rightarrow vp \ pp \\
pp \rightarrow p \ np
\]
Deductive System Interpretation

The first method we give for interpreting phrase structure grammar is as a deductive system with rules of inference corresponding to the rewrite rules of the grammar.

We will be concerned with deductions involving the derives relation which holds between an expression and a category just in case there is a proof or deduction that the expression is of the given category. Before defining this relation more precisely, we must define an auxiliary relation derives-in-one-step, written $\Rightarrow$, over $(\Sigma \cup \Delta)^* \times (\Sigma \cup \Delta)^*$ by the taking the minimal relation such that

\[(\sigma \tau \pi \Rightarrow \sigma \tau' \pi) \text{ if } \tau \rightarrow \tau' \text{ or } \Lambda(\tau, \tau').\]

In general, a string $\sigma$ can be derived in one step from the string $\tau$ just in case $\tau$ is the result of replacing a category symbol appearing anywhere in $\sigma$ with a basic expression of that category or with a string of category symbols that it can be rewritten as according to some phrase structure rule in the grammar.

We call a sequence $\langle \sigma_1, \ldots, \sigma_n \rangle$ such that $\sigma_i \Rightarrow \sigma_{i+1}$ for $1 \leq i < n$ a derivation of $\sigma_n$ from $\sigma_1$ in $\Gamma$, which we write as

\[(\sigma_1 \Rightarrow \sigma_2 \Rightarrow \cdots \Rightarrow \sigma_n).\]

With respect to our Simple English Phrase Structure Grammar, which we will call $G$, we get the derivations

\[
(8) \quad \begin{array}{l}
s \Rightarrow_G np \; vp \\
\quad \Rightarrow_G opus \; vp \\
\quad \Rightarrow_G opus \; iv \\
\quad \Rightarrow_G opus \; ran
\end{array}
\]

\[
(9) \quad \begin{array}{l}
np \Rightarrow_G det \; n \\
\quad \Rightarrow_G det \; adj \; n \\
\quad \Rightarrow_G the \; adj \; n \\
\quad \Rightarrow_G the \; silly \; n \\
\quad \Rightarrow_G the \; silly \; penguin
\end{array}
\]

50
One of the important features of phrase structure grammars is their lack of context-dependence. No matter where a category symbol shows up it can always be re-written as exactly the same things. This feature of phrase structure grammars should be obvious from the way we defined the derives-in-one-step relation. It seems important to have a way to characterise all of the possible strings that a category could be re-written as. This is easy enough, and we do so by defining a derivation relation $\Rightarrow$ over $(\Sigma \cup \Delta)^* \times (\Sigma \cup \Delta)^*$ by the obvious closure of $\Rightarrow$ under chains of steps. More formally, we let $\Rightarrow$ be the smallest relation such that $\sigma \Rightarrow \tau$ if

(11) i. $\sigma = \tau$, or
ii. $\sigma \Rightarrow \tau$, or
iii. there is some $\pi$ such that $\sigma \Rightarrow \pi$ and $\pi \Rightarrow \tau$.

If $\sigma \Rightarrow \tau$, we say that $\tau$ can be derived from $\sigma$. It should be obvious from the definition that $\Rightarrow$ will be the smallest transitive and reflexive relation containing $\Rightarrow$.

For any category symbol $c \in \Sigma$, we let

(12) $\mathcal{L}(c) = \{\sigma \in \Delta^* \mid c \Rightarrow \sigma\}$

be the set containing all and only the expressions in $\Delta^*$ which can be derived from $c$. This is really the relation that we are after, since $\mathcal{L}(c)$ is the set of all expressions or phrases which we think of the grammar $\Gamma$ as classifying as being of the category $c$. In the Simple Phrase Structure Grammar we presented earlier, we see that $\mathcal{L}(s)$ is the set of expressions which are derivable sentences, $\mathcal{L}(np)$ is the set of derivable noun phrases, and so on.

We will often wish to ignore a particular lexical assignment. Thus, we will define another set
which contains all of the sequences of category symbols in $\Sigma^*$ which can be derived from $s \in \Sigma$. We note that $\mathcal{C}(c)$ is a language over the strings of categories in $\Sigma$.

**Tree Admissibility Conditions**

We now turn to our second interpretation of the phrase structure grammar formalism, which is stated in terms of conditions on admissible trees. This is more or less the approach taken by generalized phrase structure grammar (Gazdar and Pullum 1981, Gazdar, Klein, Pullum and Sag 1985). Derivations, as defined above, have the unsavoury characteristic of linearity. There are many derivations of one string from another which only vary in the order in which rules are applied. For instance, consider the two distinct derivations

(14) $\begin{align*}
vp \Rightarrow_\mathcal{G} & tv \ np \\
\Rightarrow_\mathcal{G} & hit \ np \\
\Rightarrow_\mathcal{G} & hit \ milo
\end{align*}$

and

(15) $\begin{align*}
vp \Rightarrow_\mathcal{G} & tv \ np \\
\Rightarrow_\mathcal{G} & tv \ milo \\
\Rightarrow_\mathcal{G} & hit \ milo
\end{align*}$

These both derive the verb phrase *hit milo* using exactly the same rewrite rules, varying only in the order in which they are applied. Fortunately, there are well-developed tools in mathematical linguistics for dealing with this problem. The most important of these is the *tree*. Trees can be viewed as providing an abstract order-independent characterisation of derivations.

We will say that a tree $t \in \text{TREE}(\Sigma \cup \Delta)$ is *admissible* with respect to $\Gamma$ if and only if

(16) $\begin{align*}
i. & \ t \in \Sigma \cup \Delta, \\
ii. & \ t = [t_1 \cdots t_n]_c, \text{ and for } 1 \leq i \leq n \text{ we have that each } t_i \text{ is admissible, } c_i \text{ is the root of } t_i, \text{ and } c \rightarrow c_1 \cdots c_n, \text{ or} \\
iii. & \ t = [w]_c \text{ and } \Lambda(c, w).
\end{align*}$
An admissible tree $t$ is said to be a *parse-tree* of the string $\sigma$ from the category symbol $c$ just in case $t$ is rooted at $c$ and $\sigma$ is the yield of $t$. It should be noted that the string $\sigma$ can be an arbitrary element of $(\Sigma \cup \Delta)^*$ and so may consist of a mixture of category symbols and basic expressions. If necessary, we could restrict attention to parse trees with yields in either $\Sigma^*$ if we wish to ignore the lexical assignment or in $\Delta^*$ if we are interested in parse-trees for expressions.

It should be noted that the second condition in the definition of tree admissibility requires that all of the local trees within a tree be admissible. This admissibility check for local trees is, of course, carried out independently of the context in which they occur. From this it follows that a tree whose leaves are made up of basic expressions is admissible if and only if all of its local trees are admissible and its leaves are lexical entries for their mother categories.

The example we considered earlier of the two derivations of the verb phrase *hit milo* correspond to the single parse-tree

(17) \[
\begin{array}{c}
hit \\
tv \\
vp
\end{array}
\begin{array}{c}
milo \\
np
\end{array}
\]

The other examples correspond to the parse-trees

(18) \[
\begin{array}{c}
opus \\
np
\end{array}
\begin{array}{c}
ran \\
iv \\
vp
\end{array}
\]

(19) \[
\begin{array}{c}
the \\
det
\end{array}
\begin{array}{c}
silly \\
adj
\end{array}
\begin{array}{c}
penguin \\
n
\end{array}
\
np
\]

(20) \[
\begin{array}{c}
hit \\
tv \\
vp
\end{array}
\begin{array}{c}
steve \\
np
\end{array}
\begin{array}{c}
in \\
p
\end{array}
\begin{array}{c}
bloom county \\
np
\end{array}
\begin{array}{c}
pp \\
vp
\end{array}
\]

We note that these are the unique parse-trees corresponding to each of these strings, whereas in the case of the string *the silly penguin* there are 8 distinct
derivations. There are 36 derivations of *hit steve in bloom county*. In fact, the number of derivations of a derivable string is greater than the factorial of its length, since we at least have the freedom to derive the lexical categories of a string in any order whatsoever. For a discussion of the efficiency of using trees in parsing, see Lewis and Papadimitriou (1981).

Even though we have eliminated derivational ambiguity, it is still possible for there to be distinct parse trees of a given string from a single category. When an expression has more than one parse-tree from a category, the expression is said to be *structurally ambiguous*. This sort of ambiguity is quite different from the ambiguity we developed trees to avoid. We see an example of structural ambiguity in the parse trees for the string *milo saw the penguin in bloom county* from $S$

$$
(21) \begin{array}{cccc}
\text{milo} & \text{ saw the penguin} & \text{ with } & \text{binkley} \\
np & vp & p & np \\
\hline
vp \\
pp \\
s
\end{array}
$$

$$
(22) \begin{array}{cccc}
\text{milo} & \text{ saw } & \text{the } & \text{penguin} & \text{ with binkley} \\
np & tv & det & n & pp \\
\hline
n \\
np \\
vp \\
s
\end{array}
$$

Structural ambiguity will usually signal the possibility of semantic ambiguity. The sentence *milo saw the penguin with binkley* could mean that Milo was with Binkley when the seeing took place or that the penguin with Binkley was the penguin that Milo saw, among other things. The first of these meanings is signaled by the prepositional phrase being a daughter of the verb phrase, while the second comes from taking the prepositional phrase to be a daughter of the nominal $n$.

Such structural ambiguity is an important part of the phrase structure formalism as applied to natural language analysis. In the realm of computer science, where the interest is in computer languages, which are often given by means of phrase structure grammars, it is almost always undesirable to allow structural ambiguity. This is because in computer science, non-determinism would lead to
unpredictable or non-deterministic behaviour of algorithms expressed in the language. Consequently, classes of phrase structure rules which are not structurally ambiguous have been studied in detail (see Aho and Ullman (1972)). Natural languages are loaded with structural ambiguity and human beings are spectacular in their ability to understand and interpret structurally ambiguous sentences correctly in context (see Bever (1970), Kimball (1973), Tyler and Marslen-Wilson (1977), Crain and Steedman (1985), and Altmann (1986, 1987)) for psycholinguistic studies of the effects of syntactic, semantic and pragmatic factors on human parsing). Unfortunately, structural ambiguity is one of the major stumbling blocks in the way of developing efficient computational parsers for natural languages. This is a topic we will come back to in more detail later when we present a more complete lexicon for a fragment of English and study its ambiguity properties.

Denotational Semantics

It is quite simple to provide a clean denotational semantics for the phrase structure grammar formalism. By a denotational semantics for phrase structure grammars, we do not mean a semantics in the sense of meaning and interpretation for natural languages. Rather, we will provide a semantics for the phrase structure formalism itself of the sort Pereira and Shieber (1984) provide for feature-value grammars. The key idea here is to provide a category with a meaning that consists of the set of expressions which can have the category. This view of categories is arguably the right one for context-free grammars, but is violated by transformational grammars (see Gazdar (1981), which includes reactions by Chomsky).

Suppose we have a category symbol $c \in \Sigma$ and we are only interested in the set of expressions $\sigma \in \Delta^*$ such that $c \Rightarrow^* \sigma$. We can define the language $\mathcal{L}(c)$ directly by taking the smallest set such that

\begin{equation}
(23) \begin{array}{l}
i. \quad \Lambda(c) \subseteq \mathcal{L}(c) \\
ii. \quad \mathcal{L}(c_1) \times \cdots \times \mathcal{L}(c_n) \subseteq \mathcal{L}(c_0) \text{ if } c_0 \rightarrow c_1 \cdots c_n
\end{array}
\end{equation}

Thus, the denotation of a category contains all of the lexical entries of the category and also contains all of the strings composed by concatenating elements of the denotations of categories to which the category can be rewritten. For instance,
by clause i., \(milo \in \Lambda(np)\) so \(milo \in \mathcal{L}(c)\) and \(ran \in \Lambda(iv)\) by the same reasoning.

By clause ii. of the definition and the fact that \(vp \rightarrow iv\) in the grammar, we have \(ran \in \mathcal{L}(iv) \subseteq \mathcal{L}(vp)\) and \(milo \ ran \in \mathcal{L}(np) \times \mathcal{L}(vp) \subseteq \mathcal{L}(s)\), since \(s \rightarrow np \ vp\).

In the same way, we directly define the set \(C(c)\) of category strings which can be derived from \(c\), by taking the least set satisfying

\[
\begin{align*}
(24) \ i. \ & c \in C(c) \\
& \text{ii. } C(c_1) \times \cdots \times C(c_n) \subseteq C(c_0) \text{ if } c_0 \rightarrow c_1 \cdots c_n
\end{align*}
\]

The denotational semantics of the phrase structure formalism makes clear the view of categories as representatives of sets of expressions. Two categories get the same denotation if and only if they can be rewritten as the same set of expressions. Categories with the same denotations are said to be extensionally equivalent.

It is equally straightforward to give a similar definition for the set of parse-trees \(T(c)\) of the category \(c\) by taking

\[
\begin{align*}
(25) \ i. \ & \{c\} \times \Lambda(c) \subseteq T(c) \\
& \text{ii. } \{c\} \times (T(c_1) \times \cdots \times T(c_n)) \subseteq T(c_0) \text{ if } c_0 \rightarrow c_1 \cdots c_n
\end{align*}
\]

Of course, while we may have \(\mathcal{L}(c) = \mathcal{L}(c')\) or \(C(c) = C(c')\) for some \(c \neq c'\), we will always have \(T(c) \neq T(c')\) if \(c \neq c'\).

2.1.4 Equivalences and Restrictions

It is fairly straightforward to prove that the three interpretations of phrase structure grammars given above are equivalent in that the denotation of a category is equivalent to the set of yields of parse-trees of the category which is in turn equivalent to the strings derivable from the category. The proofs are quite tedious and will not be presented here. Interested readers should be able to reconstruct the proofs themselves. A simple structural induction suffices in all cases.

Homomorphisms and Equivalences

An important notion in phrase structure grammars is that of weak and strong equivalence. We will first define a notion of weak homomorphism for phrase structure grammars.
A pair of total functions $\varphi = (\varphi_\Sigma, \varphi_\Delta)$ such that $\varphi_\Sigma : \Sigma \to \Sigma'$ and $\varphi_\Delta : \Delta \to \Delta'$ is said to be a *weak phrase structure homomorphism* with respect to $c$ from $\Gamma$ to $\Gamma'$ just in case

$$\varphi_\Delta(\mathcal{L}_\Gamma(c)) = \mathcal{L}_{\Gamma'}(\varphi_\Sigma(c))$$

where we extend $\varphi_\Delta$ to strings in the usual way by defining

$$\varphi_\Delta(\tau \sigma) = \varphi_\Delta(\tau) \varphi_\Delta(\sigma).$$

For instance, consider the two simple grammars:

1. $np \rightarrow det\ adj\ n\ pp$
2. $adj \rightarrow adj\ adj$
3. $pp \rightarrow p\ np$
4. $pp \rightarrow pp\ pp$

and

1. $np \rightarrow det\ n$
2. $n \rightarrow adj\ n$
3. $n \rightarrow n\ p\ np$

with lexical entries

1. $np\ \text{opus, binkley}$
2. $n\ \text{penguin, kid}$
3. $det\ \text{the, every}$
4. $adj\ \text{short, smart}$
5. $p\ \text{in, by}$

The pair consisting of the identity functions on the categories and lexical entries is a weak homomorphism from the first grammar to the second, and also from the second to the first for the category $np$. It is not a homomorphism in either direction for any other category. The reason it fails on the other categories is that the set of strings of basic expressions assigned by the grammars to any other category is different.

A pair of functions $\varphi$ as above is said to be a *strong phrase structure homomorphism* from $\Gamma$ to $\Gamma'$ with respect to $c$ just in case for every parse tree $t \in T_\Gamma(c)$ we have
(31) i. $\varphi(t) \in T_{\Gamma'}(\varphi_{\Sigma}(c))$

ii. $\varphi_{\Delta}(\text{Yield}(t)) = \text{Yield}(\varphi(t))$

where the image $\varphi(t)$ of a tree $t$ under $\varphi$ is

(32) i. $\varphi_{\Sigma}(t)$ if $t \in \Sigma$,

ii. $\varphi_{\Delta}(t)$ if $t \in \Delta$

iii. $[\varphi(t_1) \cdots \varphi(t_n)]_{\varphi_{\Sigma}(c)}$ if $t = [t_1 \cdots t_n]_c$.

A homomorphism $\varphi = \langle \varphi_{\Sigma}, \varphi_{\Delta} \rangle$ is said to be an embedding if $\varphi_{\Sigma}$ and $\varphi_{\Delta}$ are both one-one. We will say that two grammars $\Gamma$ and $\Gamma'$ are weakly (strongly) equivalent or weakly (strongly) isomorphic with respect to some $c \in \Sigma$ just in case there is a weak (strong) homomorphism from $\Gamma$ to $\Gamma'$ whose inverse is a weak (strong) homomorphism from $\Gamma'$ to $\Gamma$, where we take the inverse of $\langle \varphi_{\Delta}, \varphi_{\Sigma} \rangle$ to be $\langle \varphi_{\Delta}^{-1}, \varphi_{\Sigma}^{-1} \rangle$.

This is a relativisation of the usual definitions of weak and strong equivalence to a specific category. This is necessary because our phrase structure grammars do not come with a "distinguished symbol". We note that the strong equivalence conditions do indeed presuppose the weak equivalence conditions.

We say that a category $c'$ is accessible from $c$ in a grammar $\Gamma$ just in case $c \rightarrow^* c'$. We further note that if $\varphi$ is a strong homomorphism from $\Gamma$ to $\Gamma'$ with respect to $c$ and $c'$ is accessible from $c$ in $\Gamma$, then $\varphi$ must also be a strong homomorphism with respect to $c'$. This entails that if a pair of functions is to be a strong homomorphism then it can be nothing more than a way of renaming the categories and basic expressions for all categories accessible from the category for which it is a strong homomorphism.

We will say that two grammars $\Gamma$ and $\Gamma'$ are weakly (strongly) equivalent or weakly (strongly) isomorphic just in case they are weakly (strongly) equivalent with respect to every category $c \in \Sigma$. It follows from the remarks above, that a strong equivalence or isomorphism is nothing more than a way of uniquely renaming the categories and expressions in a grammar.
Generative Capacity

The generative capacity of phrase structure grammars as we have defined them is such that for any language $L \subseteq \Delta^*$ there is a phrase structure grammar $\Gamma = \langle \Sigma, \Delta, \Phi, \Lambda \rangle$ such that $L = L_\Gamma(c)$ for some $c \in \Sigma$. This should be obvious since we have made no restriction on the cardinality of the sets $\Sigma, \Delta, \Phi$ and $\Lambda$ and they could in fact be infinite, with a unique category for each basic expression $w \in \Delta$ and a unique rule for each string $\sigma \in L$.

Context-Free Grammars

A phrase structure grammar $\Gamma = \langle E, A, \mathcal{R}, \Lambda \rangle$, where each of the sets $E, A, \mathcal{R}$ and $\Lambda$ is finite is said to be a context-free grammar. The class of context-free grammars forms an interesting and much studied subclass of phrase structure grammars. Many important results concerning context-free grammars have been proven and their computational properties are well understood. For now, most of the grammars that we will consider will be context-free and we will freely draw upon existing results from formal language theory concerning these grammars.

We can easily expand the notion of context-freeness from grammars to languages in the following way. A language $L \subseteq \Delta^*$ is said to be a context-free language if and only if there is some context-free grammar $\Gamma = \langle \Sigma, \Delta, \Phi, \Lambda \rangle$ and some category $c \in \Sigma$ such that $L = L_\Gamma(c)$. The set of all context-free languages over a set $\Delta$ of basic expressions can then be expressed by

$$\text{CFL}_\Delta = \{ L_\Gamma(c) \mid \Gamma = \langle \Sigma, \Delta, \Phi, \Lambda \rangle \text{ is a CFG and } c \in \Sigma \}.$$  

We note without proof that $\text{CFL}_\Delta$ is not simply the set $2^\Delta$ of all possible languages over $\Delta$. For instance, with $\Delta = \{a, b, c\}$, the language $\{a^n b^n c^n \mid n \text{ an integer}\}$ over $\Delta$ can not be generated by any context-free grammar. See Hopcroft and Ullman (1979), and Lewis and Papadimitriou (1981) for details concerning this and other languages which are provably non-context-free.

We will sometimes want to fix a set $\Sigma$ of category symbols as well as a set $\Delta$ of basic expressions and consider the set

$$\text{CFL}_{\Sigma,\Delta} = \{ L_\Gamma(c) \mid \Gamma = \langle \Sigma, \Delta, \Phi, \Lambda \rangle \text{ is a CFG and } c \in \Sigma \}.$$  

59
of all context-free languages over the categories \( \Sigma \) and basic expressions \( \Delta \). The set \( \text{CFL}_{\Sigma,\Delta} \) will be a monotonic function of the cardinality of \( \Sigma \). This is because for any set \( \Delta \) of basic expressions, if \( \| \Sigma \| \leq \| \Sigma' \| \) then \( \text{CFL}_{\Sigma,\Delta} \subseteq \text{CFL}_{\Sigma',\Delta} \). The actual symbols that are used are inconsequential and with more category symbols more distributional distinctions can be made.

The topic of whether or not natural languages are context-free is still debated in the linguistics community. The general consensus these days seems to be that they are not, but the arguments are rather technical. We will again, remain agnostic on this general topic, but see Culy (1985), Gazdar (1983), Pullum and Gazdar (1982), Gazdar and Pullum (1985), Shieber (1985), Kac, Manaster-Ramer and Rounds (1987).

### 2.2 Categories, Rules and Partial Information

In this section we will look at techniques available for extending the expressiveness of the basic phrase structure formalism. We carry this out by allowing the set of category symbols to be taken from a domain which is ordered by means of the amount of information categories give about the expressions they classify. We first outline the motivation for extending the grammar formalism in this way. Then we contrast the popular feature structure or directed acyclic graph encoding of hierarchical information with a representation based on first-order functional terms like those found in logic programming languages such as PROLOG. We then turn to the expressiveness of phrase structure grammars employing both sorts of structured categories and see that there is a grammar of each type for any recursively enumerable language. In fact, we will prove the more powerful equivalence result that there is a strong isomorphism from a grammar in either formalism to the other which respects the informational properties of the category domains. We will then consider some extensions of the basic system which have been proposed and explore their structural properties.
2.2.1 Subcategorisation and Polyadicity

A serious drawback to the phrase structure grammar formalism is its representation of categories by means of atomic symbols. It is often useful to present the information associated with expressions in a more structured manner. The simple English grammar fragment that we presented only encompasses third-person singular noun phrases such as *opus, every kid* and *the senator in bloom county*, excluding plurals such as *men, kids* and *bands* and non third person singular pronouns like *I, you* or *them*. There is also no distinction made between noun phrases of different case such as *he* and *him*. Similarly, only simple finite verb phrases like *runs, played, loved steve* and *cheated opus* are included, with no allowance for present participles such as *running, loving* and *knowing* or past participles such as *known, loved* and *eaten* and so on.

To account for such distinctions within the phrase structure formalism, it would be necessary to enlarge the set of category symbols, with a separate symbol for each subcategory of noun and verb. This is because the simple phrase structure formalism distinguishes sets of expressions with different distributions by giving them distinct categorisations. Unfortunately, following this rather simple path leads to an unstructured proliferation of atomic categories and brings with it the feeling that the natural organisation of the information has been missed. Harman (1963) introduced the idea of using non-terminal symbols that were themselves structured objects able to represent fine-grained distinctions between subcategories.

In the ideal situation, we would like to account for the fact that singular and plural noun phrases are both noun phrases and share the properties that all noun phrases have, but differ in what we will call *subcategorisation*. For instance, the class of verbs can be further subcategorised on the basis of the number and category of possible *complements* the verb takes, as well as on the basis of which auxiliaries and adverbs it may co-occur with. Consider the simple contrast in allowable complements we have between intransitive and transitive verbs in...
Natural languages display a wide range of agreement, which is when the categories of expressions in two positions within a phrase covary. Intuitively, this means that if one is changed, the other one must be changed so that they “agree”. We will not try to present a concrete definition of agreement, but will provide a number of examples. An example of agreement in English is the number agreement that occurs between the subject and predicate in a sentence, as seen in

(36) a. \{ * A man \} run.
    \{ All men \}

b. \{ A man \}
    \{ * All men \} runs.

In (36) we see that the number marking of the subjects is required to agree with the number marking of the predicates. The singular a man only occurs with the singular runs while the plural all men occurs only with the plural run. A much more subtle case of agreement can be found in

(37) a. \{ This is \}
    \{ * Those are \}
    \{ it was \}
    \{ * they were \}.

b. \{ * This is \}
    \{ Those are \}
    \{ * it was \}
    \{ they were \}.

where there is multiple agreement occurring across clause boundaries. For a study of agreement in the kinds of systems we are considering, see Barlow (1988,1988b) and Pollard and Sag (1988).

In some languages the case marking of noun phrases is crucial in determining the meaning and acceptability of a sentence. This is usually true of languages
which allow a large degree of freedom in the order of their words. In English, the only expressions which appear to be case-marked are pronouns, as in

\[
\begin{align*}
\text{He} & \quad \text{* Him} \\
\text{John} & \quad \text{him} \\
\end{align*}
\]

(38) \text{saw} \quad \text{He \quad * him}

In addition to expressions which are elements of a unique subcategory there are also expressions which act as if they had multiple subcategorisations. Consider the determiner \text{a} which only occurs with singular nouns and the determiner \text{all} which occurs only with plural nouns. Compare this with the behaviour of the determiner \text{the} which occurs with both kinds of nouns. The first two determiners are restricted in a way that \text{the} is not. The determiner \text{the} is said to be \textit{polymorphic} and it is in some sense more general than either of the more specialised sorts of determiners. The same is true of the noun \text{sheep} which can occur with singular or plural determiners as contrasted with nouns such as \text{penguin} and \text{penguins} which are restricted to occurring with singular or plural determiners and verbs respectively. We will also need some way to represent the fact that the compound expression \text{the sheep} is itself polymorphic and can occur as the subject of sentences with singular or plural predicates while \text{the penguin} occurs only with singular verb phrases and \text{all sheep} occurs only with plural verb phrases.

2.2.2 Subsumption and Unification

Section 2.2

Unification grammar formalisms were developed to remedy the problems faced by simple phrase structure grammars in representing subcategorisation and agreement information. See Shieber (1984,1986), Pollard and Sag (1987), Sag, et al. (1986), Gazdar, et al. (1988) and Johnson (1987) for general introductions to unification based approaches to grammar formalisms. We will begin our discussion of unification grammar with some foundational issues involved in the representation of partial information.

We will think of information about objects as coming in chunks. What you get when you take only some part of the information about an object is partial
information about the object. We will also assume that one piece of information can be more specific or general than another piece. More general information about an object is said to subsume less general information. Suppose one piece of information about an object is more general than another piece. Then everything which is compatible with the information in the more specific category is also compatible with the information in the more general category. So at least as many objects will fit a more general description than a more specific one. Looked at this way, more general information is less restrictive in that it rules out fewer possibilities. This general view of partial information descends from the denotational semantics paradigm used to model computer languages and data types, which is due to Scott and Strachey (Scott 1970, 1972, 1976, 1982 and Stoy 1977).

Two chunks of information may be such that neither one subsumes the other. This arises when each chunk contains information that the other doesn't. If this is the case, we will say that the chunks of information are incomparable with respect to the subsumption ordering.

We will assume that there is some method for combining partial information into more specific information. We call this act of merging partial information unification. The term stems from the unification operation introduced by Robinson (1965) in the context of automated theorem proving. The result of unifying two pieces of partial information is the most general information which is more specific than each. So the unification operator is just the least upper-bound operator in the subsumption ordering on partial information. Unification corresponds to the result of combining two sources of information into one unified chunk. Various methods have been developed to represent this kind of partial information in linguistics and we will discuss three of them in this section.

In some instances the operation of unification may not be well defined. It may be the case that there is not a most general piece of information subsumed by two given chunks of information. If two chunks of information can not be unified, we will say that they are incompatible.
2.2.3 Features and Values

Suppose we thought of the subcategorisation process as that of providing partial information about the values of a number of features of a category. For instance, a noun would have a number feature while a noun phrase would have additional features for case and person. The noun *penguin* would be classified as a noun with singular number while *penguins* would have a plural number value. The category of the polymorphic noun *sheep* would not restrict the value of the number feature.

**Simple Categories**

Suppose we fix a set of features and give them labels such as *number*, *case*, *person* and so on. A category could then be thought of as providing some subset of features values from a domain of atomic values such as SINGULAR, THIRD and OBJECT. For instance, we could take some pairing of feature labels and values such as

\[
\begin{array}{l}
major : \text{NP} \\
\text{number} : \text{SINGULAR} \\
\text{person} : \text{THIRD} \\
\text{case} : \text{OBJECT}
\end{array}
\]

(39)

to represent the category of the object pronoun *him*. While representations such as these were common in work on phonology, Chomsky (1965) introduced them into the phrase structure base component of transformational grammar.

For concreteness, we fix two finite basic domains of feature labels and atomic values

\[
\begin{array}{l}
\text{FEAT} \quad \text{finite set of feature labels} \\
\text{ATOM} \quad \text{finite set of atomic values}
\end{array}
\]

We could then think of a category as a partial function from FEAT to ATOM, so that the domain of simple functional categories would be given by

\[
\text{SIMCAT} = \text{ATOM}^{\text{FEAT}}.
\]

(41)

A category under this conception will specify the atomic values of some atomic features. In the category above we say that the *value* of the feature number is
SINGULAR and the value of person is THIRD, just as with ordinary functional application.

We note that SIMCAT as we have constructed it will be finite with cardinality

\[ ||\text{SIMCAT}|| = (||\text{ATOM}|| + 1)||\text{FEAT}||. \]

We will write \( c \subseteq c' \) and say that \( c \) properly subsumes \( c' \) if and only if the partial information in \( c \) is more general than the information in \( c' \). If \( c \subseteq c' \) or \( c = c' \) we will write \( c \sqsubseteq c' \) and say that \( c \) subsumes \( c' \). \( c \) subsumes \( c' \) if the information in \( c' \) is at least as specific as the information in \( c \). We will always require that \( c = c' \) if \( c \subseteq c' \) and \( c' \subseteq c \), so that the subsumption relation is antisymmetric. To define the subsumption relation on SIMCAT we set \( c \subseteq c' \) if and only if \( c \subseteq c' \) as a partial function. This makes sense as \( c \subseteq c' \) if and only if \( c' \) is defined for more elements of FEAT than \( c \) and \( c(f) = c'(f) \) for those \( f \) for which \( c \) is defined. The bounded domain \( \text{SIMCAT}^+ \), which comes from adding a top element to \( \text{SIMCAT} \), forms a complete lattice with respect to the subsumption ordering. SIMCAT has the empty function as a bottom element, since the empty function is at least as general than any other function under our definition. Therefore, the operation of unification, which we will write as \( \cup \), is well-defined and is just the least upper-bound operator on the lattice. In fact, the unification of two categories under this representation will just be their union as partial functions if the result is a partial function and \( T \) otherwise. Some formalisms, for instance HPSG turn the lattice upside down and make \( T \) the most general category (Pollard and Sag 1987). Of course, this is only a notational variant, due to the duality of lattices.

For instance, we have the subsumption relations

\[ \bot \sqsubseteq [a : b] \sqsubseteq \begin{bmatrix} a : b \\ c : d \end{bmatrix} \sqsubseteq T \]

and unifications

\[ [a : b] \cup [a : b] = [a : b] \]

\[ \begin{bmatrix} a : b \\ c : d \\ g : h \end{bmatrix} \cup \begin{bmatrix} e : f \\ g : h \end{bmatrix} = \begin{bmatrix} a : b \\ c : d \\ e : f \\ g : h \end{bmatrix} \]
We see that two categories are incompatible in the ordering only if they contain contradictory information about the value of at least one feature, as in the last example above. Otherwise, they could be unified into another category in SIMCAT.

**Nested Categories**

While this simple functional model of information is attractive by virtue of its simplicity, we sometimes want to represent information with a more deeply nested structure. For this purpose, we might suppose that the value of a feature label in a category could not only be an atom in ATOM, but could also be a complex bit of information itself. The simplest way to do this is to suppose that the value of a feature may itself be a category, since we use categories as our representations of partial information. This new domain **NESTCAT** of categories could then be given as the least domain **NESTCAT** such that

\[
\text{i. } \text{ATOM} \subseteq \text{NESTCAT}, \text{ and} \\
\text{ii. } \text{NESTCAT}^{\text{FEAT}} \subseteq \text{NESTCAT}.
\]

We are again restricting our attention to finite domains. Very little hinges on this decision besides computational implementation. We could just as easily work with a category set which is a larger fixed-point of the domain equations. Note that we have also smuggled in the assumption that a category may be a single atom in ATOM as well as a partial function from feature labels to values. The idea of letting a feature take a value which is itself a category is originally due to Kay (1979), but soon became standard in non-transformational theories such as generalized phrase structure grammar (Pullum and Gazdar 1982) and lexical-functional grammar (Kaplan and Bresnan 1982).

We define a subsumption ordering on the bounded domain **NESTCAT** such that \( c \subseteq c' \) if and only if one of

(46) \[
\begin{bmatrix}
  a : b \\
  c : d \\
\end{bmatrix} \cup 
\begin{bmatrix}
  e : f \\
  c : g \\
\end{bmatrix} = T
\]
As a consequence of this definition, a member of ATOM is incompatible with everything except itself and \( \bot \). Under this subsumption ordering we again have a lattice which yields a well-defined total operation of unification. More specifically, we have constructed NEST\,CAT so that

\[
\text{NEST\,CAT}^+ \cong \text{ATOM} \oplus (\text{NEST\,CAT}^{\text{FEAT}}).
\]

where we take ATOM to be partially ordered so that all of its elements are pairwise incomparable, and NEST\,CAT^{\text{FEAT}} to be ordered so that \( f_1 \sqsubseteq f_2 \) for \( f_1, f_2 \in \text{NEST\,CAT}^{\text{FEAT}} \) if and only if \( \text{Dom}(f_1) \subseteq \text{Dom}(f_2) \) and \( f_1(x) \sqsubseteq f_2(x) \) for every \( x \in \text{Dom}(f_1) \).

Unification is now slightly harder to spell out, although it still just the taking of least upper bounds in the lattice. For \( c, c', c'' \in \text{NEST\,CAT}^+ \) we have \( c \sqcup c' = c'' \) if

\[
\begin{align*}
i. & \quad c = c' \\
ii. & \quad c(x) \sqsubseteq c'(x) \text{ for every } x \in \text{Dom}(c) \\
iii. & \quad c' = \top \\
iv. & \quad c = \bot.
\end{align*}
\]

Consider the subsumption relations

\[
\begin{align*}
\bot & \sqsubseteq \text{ATOM} \\
\bot & \sqsubseteq [a : [b : c]] \sqsubseteq [a : [b : c] \sqsubseteq [a : [b : c]] \\
\bot & \sqsubseteq [a : [g : H]] \\
\bot & \sqsubseteq [e : F]
\end{align*}
\]
and the unifications

\[
(53) \quad \begin{array}{l}
\left[ \begin{array}{l}
a : [c : D] \\
b : C
\end{array} \right] \cup \\
\left[ \begin{array}{l}
a : [e : F] \\
i : H
\end{array} \right] = \\
\left[ \begin{array}{l}
a : [c : D] \\
b : C \\
i : H
\end{array} \right]
\end{array}
\]

\[
(54) \quad A \cup [b : c] = T
\]

\[
(55) \quad [a : [c : D]] \cup [a : [c : E]] = T
\]

In SimCat, a category \( c \) had to be defined for more feature labels than a category \( c' \) if \( c \sqsubseteq c' \). Now that we allow features to take on complex values we may have \( c \sqsubseteq c' \) where \( c \) and \( c' \) are defined for exactly the same feature labels. In this case, the value of the category \( c \) at every feature \( f \) for which it is defined must be at least as general as the value of the category \( c' \) at \( f \) and more general for some \( f \). Partial information is still represented as the lack of knowledge of feature values, but it is now possible to find the feature-value pairs in question arbitrarily nested within the values of other features.

Now that features can nest, we will introduce a useful notation to get at the value of a nested feature. We will define a path over \( \text{FEAT} \) to be an arbitrary sequence of feature labels \( (f_1, \ldots, f_n) \in \text{FEAT}^* \).

We define the path value operation \( \langle \rangle \) which produces the value of a category at a path inductively by

\[
(56) \quad \begin{array}{l}
i. \quad c \langle \rangle = c, \text{ and} \\
\quad \quad \quad \text{ii. } \quad c \langle f_1, \ldots, f_n \rangle = c(f_1) \langle f_2, \ldots, f_n \rangle.
\end{array}
\]

In general, if the value of a category at a path is defined, then

\[
(57) \quad c \langle f_1, \ldots, f_n \rangle = (\cdots (c(f_1))(f_2) \cdots )(f_n).
\]

For instance, with the category

\[
(58) \quad c = \begin{bmatrix}
\begin{array}{l}
c : D \\
a : [j : K] \\
g : H \\
b : C \\
i : H
\end{array}
\end{bmatrix}
\]

69
we have the path values

(59) \( c \downarrow (i) = \Pi \)

(60) \( c \downarrow (a, c) = D \)

(61) \( c \downarrow (a) = \left[\begin{array}{l}
  c : D \\
  e : [j : K] \\
  g : H
\end{array}\right] \)

(62) \( c \downarrow (a, e) = [j : K] \)

(63) \( c \downarrow (a, e, j) = K \)

The specification of path values plays an important role in the PATR-II grammar writing formalism (Shieber et al. 1983, Shieber 1984, Karttunen 1986) and also in the specification of lexical-functional grammars (Kaplan and Bresnan 1982) and functional unification grammars (Kay 1984, 1985).

Without changing our conception of categories, we could change their mathematical representation so that we could think of them as path-value sets rather than as feature-value sets. This is the approach taken by Pereira and Shieber (1984) in providing a denotational semantics for the PATR-II grammar writing formalism (Shieber et al. 1983, Karttunen 1986) and by Moshier and Rounds in their work on data types (Moshier and Rounds 1987, Moshier 1988). In fact, we do not even need to record the value of every path. Much of that information would be redundant. Instead, we will only need to specify the values of maximal paths which take values in ATOM. A path \( p \) is said to be maximal in the category \( c \) if \( c \downarrow p \in \text{ATOM} \). The set of maximal paths in a category is hence given by

\[
\text{Path}(c) = \{p \in \text{FEAT}^* \mid c \downarrow p \in \text{ATOM}\}.
\]

Keeping in mind that atomic categories provide a path value for the null string, we note that \( c \sqsubseteq c' \) if and only if

\[
\begin{align*}
\text{i. } & \text{Path}(c) \subseteq \text{Path}(c'), \text{ and} \\
\text{ii. } & \text{for every } p \in \text{Path}(c) \text{ we have that } c \downarrow p = c' \downarrow p.
\end{align*}
\]
Suppose we wanted to represent categories as pairings between maximal paths and atomic values. We have already seen that this is enough information about a category to determine its subsumption behaviour. Unfortunately, not every path-value set will represent a category. Only those path-value functions which could be derived from some functional category in NESTCAT should be allowed. Such a path-value set will be said to be consistent. A partial function \( f \) from paths to atomic values will be consistent if and only if it is finite and there are no \( p, p' \in \text{FEAT}^* \) such that \( p \) is a proper prefix of \( p' \) and \( f(p) \) and \( f(p') \) are both defined. The finiteness requirement comes from the fact that the elements of NESTCAT are hereditarily finite. We write

\[
(66) \text{CATPATH} = \{ c \mid c : \text{FEAT}^* \rightarrow \text{ATOM} \text{ is consistent} \}
\]

for the domain of consistent functions from paths to values. The unification of compatible categories in CATPATH is given by their union as partial functions, with incompatible categories producing unions which are either not functions or inconsistent functions.

**Equational Categories**

Suppose that in addition to local restrictions on the values of paths in a category, we also have the information that two paths share the same value. What this means is that any information gained about the value of one path can be taken to be information about the value of the other. Consequently, we are assured that if two paths \( p \) and \( p' \) share a value, then for any path \( p'' \), we can conclude that \( p \cdot p'' \) and \( p' \cdot p'' \) also share a value. This just says that extending any two shared paths by the same path always leads to paths that are shared. Such path sharing is allowed in all of the feature based grammar formalisms we have discussed except generalised phrase structure grammar (Gazdar et al. 1985, Gazdar et al. 1988).

In purely informational terms, if we have two categories which are identical except that with one we know that the values of two paths are equivalent, then the category with the shared paths is subsumed by the more general category for which there is no sharing between the paths in question.
We will construct a domain CATEQ to represent this kind of shared information based on Moshier and Rounds (1987) and similar to the treatment of Pereira and Shieber (1984) and Ait-Kaci (1984). We suppose that an element of CATEQ is modeled by a pair \((c, \equiv)\), where \(c \in \text{NESTCAT}\) represents the known information about path values and \(\equiv\) is a relation on paths which holds between paths which share values. Again, not any relation \(\equiv\) and category \(c\) in NESTCAT will represent something sensible, so we require \((c, \equiv)\) to meet the consistency conditions satisfied.

(67) i. \(\equiv\) is an acyclic right-invariant equivalence relation

ii. if \(p, p' \in \text{Path}(c)\) are such that \(p \equiv p'\) we have \(c \parallel p = c \parallel p'\)

if either exists.

An equivalence relation \(R\) on strings in \(S^*\) is said to be right invariant if for every \(\sigma, \tau \in S^*\), if \(\sigma R\tau\) then \(\sigma\pi R\tau\pi\) for every string \(\pi \in S^*\). We require that \(\equiv\) be right invariant so that the values of extensions of equivalent paths are forced to be equivalent if they are defined. A relation \(R\) on paths in \(S^*\) is said to be acyclic if there are no \(p, p' \in S^*\) such that \(p\) is a proper prefix of \(p'\) and \(p\) is \(R\)-related to \(p'\).

As before, we will adjoin a \(T\) element to CATEQ to produce the bounded domain \(\text{CATEQ}^+\). The bottom element will simply be the pair \((\bot, \{(p, p) \mid p \in \text{FEAT}^*\})\) where we have the bottom element of NESTCAT and the minimal path equivalence relation. We now take the subsumption relation to be such that for two categories in CATEQ we get

(68) \(\langle f, \equiv \rangle \sqsubseteq \langle f', \equiv' \rangle\) iff \(f \sqsubseteq f'\) and \(\equiv \sqsubseteq \equiv'\).

Of course, for every \(c \in \text{CATEQ}\) we have \(\bot \sqsubseteq c \sqsubseteq T\) by definition. It is routine to verify that \(\text{CATEQ}^+\) forms a lattice.

Again, we take unification to correspond to the least upper bound operation in the lattice. To unify two categories it is still necessary to unify their path sets and values, but now we must also unify their path equivalences. The result will be the smallest right invariant equivalence relation which is a superset of the two given relations. We will also require a category to meet the consistency requirement. If
such a construction is not possible, the unification of the two categories is taken to be \( T \) by default. Working in \( \text{CAT EQ} \) we now have \( \langle f, = \rangle \cup \langle f', =' \rangle = \langle f'', ='' \rangle \) if and only if

\[
\begin{align*}
\text{(69)} & \quad ='' \text{ is the smallest right invariant equivalence relation which} \\
& \quad \text{is a superset of } = \cup =', \text{ and} \\
& \quad \text{ii. } f'' \in \text{NEST} \text{CAT} \text{ is the minimal element of NEST} \text{CAT consistent with } ='' \text{ such that } f \cup f' \subseteq f''.
\end{align*}
\]

Before moving on, let’s see a few examples of these new kinds of categories and consider their subsumption and unification behaviour. First we need to develop some notation for representing path equivalences. Where two paths are equivalent, we will co-index their values. For example

\[
\begin{align*}
\text{(70)} & \quad \begin{bmatrix}
a : [1] \\
b : [2]
\end{bmatrix}
\end{align*}
\]

is the category where we know that the values of \( a \) and \( b \) are identical. We will also display categories in the form

\[
\begin{align*}
\text{(71)} & \quad \begin{bmatrix}
a : [1] \\
c : [D] \\
e : [F] \\
b : [g : [2]]
\end{bmatrix}
\end{align*}
\]

where we omit the redundant information about the value of paths prefixed by \( (b, g) \), since they are required to have the same value as paths prefixed by \( (a) \). What we have done is added so-called tags to certain paths which are taken to be equivalent. This gives us a finite means of presenting an essentially infinite equivalence relation. We assume that we have a countably infinite supply of distinct tags, but the actual tags that we use are insignificant. By this we mean that

\[
\begin{align*}
\text{(72)} & \quad \begin{bmatrix}
a : [1] \\
b : [1]
\end{bmatrix}
\end{align*}
\]

and

\[
\begin{align*}
\text{(73)} & \quad \begin{bmatrix}
a : [2] \\
b : [2]
\end{bmatrix}
\end{align*}
\]
are to be taken to represent the same category. The tags are not part of categories as we have defined them, but merely a means to display them.

This new domain gives us more representational power than we had previously with NEST CAT alone and also gives us a more structured domain of categories. In fact, NEST CAT and SIM CAT can both be embedded in CATEQ. The domain of simple partial functions SIM CAT is embedded in the domain NEST CAT of complex values by the identity function \( id_{\text{CATSIM}} \). We can also embed NEST CAT in CATEQ by the one-one homomorphism \( \phi \) such that

\[
\phi(c) = \langle c, \{(p, p) \mid p \in \text{FEAT}^*\} \rangle.
\]

An element of NEST CAT is mapped onto the pair consisting of itself and the minimal equivalence relation consistent with it, which is always the identity relation over \( \text{FEAT}^* \). From the proper subsumptions,

\[
\begin{align*}
(a: 1) & \subseteq (a: 1) \\
(b: 1) & \subseteq (b: 1) \\
(c: 2) & \subseteq (c: 3) \\
(d: 1) & \subseteq (d: 3) \\
(e: 1) & \subseteq (e: 3) \\
(f: 1) & \subseteq (f: 3)
\end{align*}
\]

it should be obvious that one-one homomorphisms could not be constructed in the other direction so that the domains are not isomorphic. Now consider the unifications

\[
\begin{align*}
(a: \{c: D\}) \cup (a: \{e: F\}) \cup (b: \{e: G\}) &= (a: \{c: D\}) \\
(b: \{c: D\}) &
\end{align*}
\]

Consider what happens in the case of
We might have thought that

\[(81) \begin{bmatrix} a : \square \\ b : [c : \square] \end{bmatrix} \cup \begin{bmatrix} a : [c : \square] \\ b : \square \end{bmatrix} = \begin{bmatrix} a : \square [c : \square] \\ b : \square [c : \square] \end{bmatrix}\]

but this would lead us to a cyclic set of path equivalences, which is explicitly excluded by the consistency conditions, as can be seen by performing the expansion

\[(82) \begin{bmatrix} a : \square [c : \square] \\ b : \square [c : \square] \end{bmatrix} = \begin{bmatrix} a : \square [c : \square [c : \square]] \\ b : \square \end{bmatrix}\]

Trying to unify this cyclic category with something like

\[(83) [a : F]\]

would lead to chaos for the present system, as the resulting categories would have values for an infinite number of paths, which is outside the domain NESTCAT of hereditarily finite partial functions. We do not find such circularity particularly desirable in our linguistic categories, but the interested reader should consult Ait-Kaci (1984), Ait-Kaci and Nasr (1986), Pollard and Sag (1987), and Pollard and Moshier (forthcoming) for possible applications in syntactic representation and Barwise and Etchemendy (1987) for applications in semantic representation. As Pereira and Shieber (1984) point out, it is a simple matter to extend a framework like the present one to incorporate such circular information.

Recall that the categories form a lattice and so the operations of unification in all of our domains is associative, symmetric and idempotent with identity $\perp$. This means that for all categories $c, c', c''$ we have

\begin{align*}
(84) &. c \cup (c' \cup c'') = (c \cup c') \cup c'' \quad \text{(associativity)} \\
&. c \cup c' = c' \cup c \quad \text{(symmetry)} \\
&. c \cup c = c \quad \text{(idempotence)} \\
&. c \cup \perp = \perp \cup c = c \quad \text{(identity)}
\end{align*}

and we note that the subsumption relation is anti-symmetric, transitive and reflexive, so that for all categories $c, c', c''$ we have
(85)  i. if \( c \sqsubseteq c' \), and \( c' \sqsubseteq c'' \) then \( c \sqsubseteq c'' \) (transitivity)

ii. \( c \sqsubseteq c \) (reflexivity)

iii. if \( c \sqsubseteq c' \) and \( c' \sqsubseteq c \) then \( c = c' \). (anti-symmetry)

Rounds and Kasper (1986) and Pollard and Moshier (forthcoming) contain a number of additional identities satisfied by category domains with additional operations. Gazdar et. al. (1988) contains a study of the logic of identities satisfied by a more restrictive category system with a more expressive logical apparatus.

**Alternative Representations**

There are many ways besides the one which we have chosen to represent the same domains. Another popular method is in terms of directed rooted graphs which have arcs labeled with elements of FEAT and have some final nodes labeled with elements of ATOM. See, for instance, Pereira and Shieber (1984) Pollard and Sag (1987) and Kay (1984,1985). Nodes of the graph are taken to correspond to values which are elements of ATOM or graphs themselves. The value of the whole graph corresponds to the value corresponding to its root. If two paths have the same value in CATEQ then they will both lead to the same node in the graph representing the category. Acyclic graphs lead to non-circular terms, while arbitrary graphs lead to circular terms. Trees, on the other hand, lead to elements of NESTCAT, as there is no structure sharing.

A second alternative is to use finite-state automata whose transitions are labeled with elements of FEAT and some terminal final states labeled with elements of ATOM. This approach is fully developed by Moshier and Rounds (1987) and Moshier (1988). States of the machine will be taken to represent elements of the domain, with arcs representing features.

Yet another way to represent categories is in terms of Aczel's non-well founded sets, as in Aczel (1988) which allows an elegant representation of cyclic feature structures.

The nice thing about all of these alternative representations is that there is no need to check consistency conditions on equivalence relations. The consistency conditions are simply built into the objects used to do the modeling.
Unification Grammar

Now that we have seen what categories look like in the feature based approach to information we turn to the definitions of grammars using these categories. Grammars that have been proposed for these sorts of category systems have mainly been driven by a context-free backbone of some kind. The notable exception is head-driven phrase structure grammars (Pollard and Sag 1987) which are based on the head grammars of Pollard (1984), which are themselves a close relative of context-free grammars. The standard strategy is to allow some finite set of phrase structure rules to be given with some kind of augmentation that affects the way features of categories in a tree or derivation are related. These augmentations are usually called feature passing conventions. This is the strategy adopted in generalized phrase structure grammar (Gazdar et al. 1984) lexical-functional grammar (Kaplan and Bresnan 1982) functional unification grammar (Kay 1984,1985) and the PATR-II grammar writing system (Shieber et al. 1983, Shieber 1984, Karttunen 1986). Ideally, we would like the set of phrase structure rules over our structured categories to follow the pattern of the categories themselves and be given some sort of informational structure, as argued for by Kay (1984,1985) and later incorporated in head-driven phrase structure grammar (Pollard and Sag 1987). This means that there would be a subsumption relation over the rules and a derived operation of rule unification.

We proceed in a way similar to that we took in defining simple phrase structure grammars and say that a unification phrase structure grammar is a quintuple

\[
\Gamma = \langle \text{FEAT}, \text{ATOM}, \Delta, \Lambda, \Phi \rangle
\]

where

\[
\begin{align*}
\text{FEAT} & \quad \text{finite set of feature labels} \\
\text{ATOM} & \quad \text{finite set of atomic values} \\
\Delta & \quad \text{set of basic expressions} \\
\Lambda & \quad \text{a lexical assignment relation} \\
\Lambda & \subseteq \text{CATEQ} \times \Delta \\
\Phi & \quad \text{a set of feature-value phrase structure rules} \\
\Phi & \subseteq \{ (c, \sigma, \Rightarrow) \mid c \in \text{CATEQ}, \sigma \in \text{CATEQ}^*, \} \\
& \Rightarrow \subseteq \omega \times \text{FEAT}^*
\end{align*}
\]

The augmentation \(\Rightarrow\) on a rule is meant to specify path equivalences within as well as across categories. The numbers \(n \in \omega\) are used to code up the information
about which category in the rule is being referred to. The mother category receives an index of 0 and the daughters are indexed from left to right using successive natural numbers beginning with 1. We introduce the notation \( c_0 \rightarrow c_1 \cdots c_n \| \Rightarrow \), for a rule \( \langle c_0, (c_1, \ldots, c_n), \Rightarrow \rangle \), where the index of \( c_i \) is \( i \). Paths beginning with the index of a category provide information about that category.

The easiest sort of interpretation to give for this sort of grammar is in terms of tree admissibility. When we give these conditions, the rather mysterious equivalence relation attached to a rule should become clear. An auxiliary definition we will need is that of a reduction of a right invariant equivalence relation \( \Rightarrow \) along a path \( p \), which is given by

\[
\Rightarrow_p = \{ (p', p'') \mid p \cdot p' \Rightarrow p' \cdot p'' \}
\]

The reduction of a right invariant equivalence relation is another right invariant equivalence relation.

Again, we assume a tree \([w]\) is admissible if \( w \) is a lexical entry of category \( c \) and that elements of CATEQ and \( \Delta \) are themselves acceptable trees. The new case is where

\[
\Rightarrow(c_1, \Rightarrow_1) \cdots (c_n, \Rightarrow_n) \Rightarrow(c_0, \Rightarrow_0)
\]

is admissible if there is some rule

\[
d_0 \Rightarrow d_1 \cdots d_n \Rightarrow_R
\]

such that

\[
\langle \begin{bmatrix}
0 : e_0 \\
1 : e_1 \\
\vdots \\
n : e_n
\end{bmatrix}, \Rightarrow \rangle = \langle \begin{bmatrix}
0 : d_0 \\
1 : d_1 \\
\vdots \\
n : d_n
\end{bmatrix}, \Rightarrow_R \rangle \cup \langle \begin{bmatrix}
1 : c_0 \\
\vdots \\
n : c_n
\end{bmatrix}, \Rightarrow_T \rangle
\]

where

\[
\Rightarrow_T = \{ (p, p) \mid p \in \text{FEAT}^* \} \cup \{ (\langle n, p \rangle, \langle n, p' \rangle) \mid (p, p') \in \Rightarrow_n \}.
\]

What we do to combine a string of categories is look for a rule whose daughters will simultaneously unify with the string. If we find such a rule, the resultant
mother category can be derived from the effects of the other unifications on the mother position. We need to take this complicated route of stripping the mother category off afterwards, since we might gain more information about the mother by unifying the categories with the rule. We have separated the feature-passing mechanism, which is embodied in our indexed equivalence relation on paths in categories, and actual consistency requirements which are embodied in the categories on the phrase structure part of the rule. The original idea to encode grammar rules in the same format as categories is due to Kay (1984) and has been taken up in other unification grammar formalisms such as head-driven phrase structure grammar (Pollard and Sag 1987).

A tree $t$ will again be called admissible if all of its local trees are admissible. An admissible tree $t$ rooted at a category $c$ with a yield of $\sigma$ is again said to be a parse-tree of $\sigma$ from $c$.

We can now define what it means for one rule to subsume another. We put

$$(92) \quad d_0 \rightarrow d_1 \cdots d_n \equiv \sqsubset \quad d'_0 \rightarrow d'_1 \cdots d'_n \equiv'$$

if and only if

$$(93) \quad \left[ \begin{array}{l}
0 : d_0 \\
n : d_n
\end{array} \right] \equiv \left[ \begin{array}{l}
0 : d'_0 \\
n : d'_n
\end{array} \right]$$

as a category. This, in turn, gives us a well-defined unification operation over rules. We note that if a grammar contains two rules, one of which subsumes the other, the more specific rule could be eliminated with the result being a weakly equivalent grammar. Of course, there is a loss in strong generative power under our informational approach to labeling trees, where a category is only labeled with as much information as is derivable directly from its daughters and the rule used to combine them.

Consider a simple grammar with the rules

$$(94) \quad \left[ \begin{array}{l}
cat : S \\
cat : NP
\end{array} \right] \rightarrow \left[ \begin{array}{l}
cat : NP \\
agr : [1]
\end{array} \right] \left[ \begin{array}{l}
cat : VP \\
agr : [1]
\end{array} \right]$$

$$\left[ \begin{array}{l}
cat : NP \\
agr : [1]
\end{array} \right] \rightarrow \left[ \begin{array}{l}
cat : DET \\
agr : [1]
\end{array} \right] \left[ \begin{array}{l}
cat : N \\
agr : [1]
\end{array} \right]$$

79
that will account for the number agreement in noun phrases and sentences like we discussed earlier for the lexical items

(95) \[\text{Word} \quad \text{Lexical Entry}\]

\begin{tabular}{ll}
the & [cat : DET] \\
every & [cat : DET, agr : [num : SING]] \\
all & [cat : det, agr : [num : PLU]] \\
sheep & [cat : N] \\
man & [cat : N, agr : [num : SING]] \\
men & [cat : N, agr : [num : PLU]] \\
ate & [cat : VP] \\
eats & [cat : VP, agr : [num : SING]] \\
eat & [cat : VP, agr : [num : PLU]]
\end{tabular}

Note that we have used the same sort of notation for the equivalence on rules that we used for categories, which allows us to avoid the explicit indexing of daughter and mother positions. Our grammar will admit the parse trees

(96) \[
\begin{array}{c}
\text{all} \\
\text{sheep} \\
\text{ran}
\end{array}
\]

\[
\begin{array}{c}
\text{cat : DET} \\
\text{agr : [num : PLU]} \\
\text{cat : NP} \\
\text{agr : [num : PLU]} \\
\text{cat : S}
\end{array}
\begin{array}{c}
\text{cat : N} \\
\text{cat : VP}
\end{array}
\]
We can see from these examples that the information associated with a node in an admissible tree is gained directly from its daughters. Another way of looking at these rules is to require actual structure sharing in the tree. In effect, this would mean employing all of the path equivalences globally throughout a tree rather than locally. Such global sharing is carried out in all of the other unification grammar formalisms we have discussed. While this may be desirable for efficient computational applications, we wish to maintain that the category associated with a node must be derivable from the categories of its daughters. This is a much more strictly compositional approach where information constraints are enforced locally. That is, we are truly sticking to a local tree admissibility theory of grammar. In structure-sharing across nodes in a tree, the value of a noun's number agreement feature might be determined by the value of an arbitrarily distant verb's number agreement feature. We would rather assume that the noun's number feature simply remains unspecified even after the entire tree is constructed. This will not affect the weak generative capacity of the grammars in any way, but simply provide an alternative convention for displaying parses.

2.2.4 Categories as Functional Terms

Up to this point, we have been representing the information in a category as supplying values for some of the features a category could have. We could just as easily represent the information structure we get from these categories by means of first-order functional terms. The way in which we build a functional term is
by providing a function and some sequence of arguments. In fact, if we think of the arguments as values of features, this approach looks very similar to the feature-value systems.

To make things more precise, suppose \( \Upsilon = \bigcup_{n \in \omega} \Upsilon_n \) is a finite indexed set of function symbols, where a function symbol \( f \in \Upsilon_n \) is said to have an \textit{arity} of \( n \). The arity of a function specifies how many arguments it takes. Such a set \( \Upsilon \) of function symbols with arities is called a \textit{signature}.

The function symbols \( f \in \Upsilon_0 \) with arity 0 will correspond to functions without any arguments, which are also called \textit{constants} or \textit{atoms}. We will also fix a countably infinite set \( \mathcal{V} \) of \textit{variables} which are distinct from everything else.

We say that the set of \textit{terms} over \( \Upsilon \) and \( \mathcal{V} \) is the least set \( \mathcal{T}_{\Upsilon, \mathcal{V}} \) such that

\[
(99) \quad \begin{array}{l}
  \text{i. } \mathcal{V} \subseteq \mathcal{T}_{\Upsilon, \mathcal{V}} \\
  \text{ii. } f(t_1, \ldots, t_n) \in \mathcal{T}_{\Upsilon, \mathcal{V}} \text{ if } f \in \Upsilon_n \text{ and } t_i \in \mathcal{T}_{\Upsilon, \mathcal{V}}.
\end{array}
\]

which we will simply write as \( \mathcal{T} \) if \( \Upsilon \) and \( \mathcal{V} \) are understood.

The task now, in our information based approach, is to define a notion of subsumption for terms. We think of our variables as standing in for values which are not known. Consequently, we want variables to be more general than other terms. Also, if the same variable shows up in two locations in a term, we want this to mean that the values are taken to be the same. This is standard in the logic programming literature (see, for instance, Sterling and Shapiro (1986)) and stems from Robinson (1965).

Before we define the subsumption ordering on first-order terms more formally, we must first provide a number of preliminary definitions. A partial function \( \theta : \mathcal{V} \to \mathcal{T} \) is said to be a \textit{substitution}. For a term \( t \) and substitution \( \theta \) we will write \( t[\theta] \) for the result of replacing the variables in \( \text{Dom}(\theta) \) with their images in \( \mathcal{T} \). More formally, \( t[\theta] \) is given by

\[
(100) \quad \begin{array}{l}
  \text{i. } f(t_1, \ldots, t_n)[\theta] = f(t_1[\theta], \ldots, t_n[\theta]) \\
  \text{ii. } v[\theta] = \theta(v) \text{ if } v \in \text{Dom}(\theta) \\
  \text{iii. } t[\theta] = t \text{ otherwise.}
\end{array}
\]

82
The result of applying a substitution to a term yields what is called an instance of the term. For two terms \( s, t \in T \) we say that \( s \) subsumes \( t \) and write \( s \sqsubseteq t \) if and only if there is some substitution \( \theta \) such that \( s[\theta] = t \). So \( s \) subsumes \( t \) if and only if \( t \) is an instance of \( s \). We again write \( s \sqsubseteq t \) if \( s \sqsubseteq t \) and \( s \neq t \). Following the PROLOG convention of writing variables in upper case and function symbols in lower case, we have

(101) \( a(X,Y) \sqsubseteq a(Z,Z) \sqsubseteq a(b,b) \)

where we use the substitution \( (X \mapsto Z, Y \mapsto Z) \), in the first case and \( (Z \mapsto b) \) in the second. We will write finite substitutions in this manner, assuming they are undefined at elements not explicitly listed. We also have

(102) \( a(X,Y) \sqsubseteq a(c(Z),W) \sqsubseteq a(c(a(U,V)),b) \)

on the basis of the substitutions \( (X \mapsto c(Z), Y \mapsto W) \) and \( (Z \mapsto a(U,V), W \mapsto b) \). Also consider what happens to shared variables under subsumption

(103) \( a(X,X) \sqsubseteq a(a(Y,Z),a(Y,Z)) \sqsubseteq a(a(b(W),V),a(b(W),V)) \)

(104) \( a(X,X) \nsubseteq a(b(Y),b(Z)) \)

(105) \( a(X,X) \nsubseteq a(a(U,V),a(U,Z)) \).

The subsumption relation \( \sqsubseteq \) is both transitive and reflexive, but is not anti-symmetric. If \( s \sqsubseteq t \) and \( t \sqsubseteq s \) we say that \( s \) and \( t \) are alphabetic variants and write \( s \models t \). Two terms \( s \) and \( t \) will be alphabetic variants just in case there are substitutions \( \theta, \gamma \) such that \( s[\theta] = t \) and \( t[\gamma] = s \). This will make the pairs \( a(X) \models a(Y) \) and \( b(c(X),Y,Z) \models b(c(Y),Z,X) \) alphabetic variants by the obvious substitutions.

Since alphabetic variance is obviously an equivalence relation, we could work in the quotient domain \( T/\models \) of terms modulo alphabetic variance, where, as usual, we would put

(106) \( T/\models = \{[t]_\models \mid t \in T \} \)

and extend the subsumption relation \( \sqsubseteq \) on \( T \) to \( T/\models \) in the usual way by setting
We choose not to work with elements of the quotient domain directly, since it is a bother to carry around a whole equivalence class. Instead, we will work with representative elements of the equivalence classes, which are simply terms in $\mathcal{T}_{T,V}$.

It can be easily verified that $\mathcal{T}/\simeq$ under our new $\subseteq$ forms a meet semilattice. We will also need to add an additional $T$ element to the semilattice of terms modulo alphabetic variance to produce a lattice, noting that the single equivalence class $[v]_{\simeq}$ is the bottom element where $v \in V$ is some variable.

Now that we have seen what subsumption looks like, unification is fairly straightforward to define. A substitution $\theta$ such that $t[\theta] = s[\theta]$ is called a unifying substitution or unifier of $s$ and $t$. If $\theta$ is a unifier of $s$ and $t$ then we say that $s[\theta] = t[\theta]$ is a common instance of $s$ and $t$, since it is a term subsumed by both. A unifier $\theta$ of $s$ and $t$ is called a most general unifier of $s$ and $t$ if and only if for any other unifier $\gamma$ of $s$ and $t$ we have $s[\theta] = t[\theta] \subseteq s[\gamma] = t[\gamma]$. We note without proof, that for any two unifiable terms, a most general unifier exists. We also note that if $\theta$ and $\gamma$ are both most-general unifiers of $s$ and $t$ then $s[\theta] = s[\gamma] = t[\theta] = t[\gamma]$.

There is a slight complication in finding unifying substitutions for pairs of terms with variables in common, since the substitution applied to one will affect the other. So that, while the terms $a(X,Y)$ and $a(Y,X)$ are both alphabetic variants of the term $a(W,Z)$, the result of applying a most general unifier will result in an alphabetic invariant of $a(U,U)$, since $X$ must map onto a variant of the image of $Y$ and conversely. We will avoid this problem by only considering unification over pairs of terms without shared variables. If $s$ and $t$ do not have any variables in common, then we take

\[(108) \ s \cup t = s[\phi] = t[\phi] \]

if $\phi$ is a most general unifier of $s$ and $t$. Note that this is only defined up to alphabetic invariance, since there might be a number of distinct terms $u$ such that $s \cup t = u$. Fortunately, all such $u$ will be alphabetic variants. We are really thinking of terms as representing their equivalence class in $\mathcal{T}/\simeq$, since we have

\[(109) \ [s \cup t]_{\simeq} = [s]_{\simeq} \cup [t]_{\simeq} \]
if $s$ and $t$ do not share any variables, so that $\sqcup$ respects the information ordering on the domain of terms.

The term representation of information is also acyclic. Consider the following terms which cannot be unified

$$a(X, b(X)) \sqcup a(b(Y), Y) = T$$

and

$$a(X, b(Y)) \not\equiv a(b(Y), Y).$$

If we try and unify the terms, we just proceed in circles, getting

$$a(X, b(X)), a(b(b(X)), b(X)), a(b(b(X)), b(b(b(X))))\ldots$$

See Ait-Kaci (1984) for an extension to the term domain allowing such circularity.

A term $t$ without any variables is called a ground term. The set of ground terms made up of elements of a signature $T$ is often called the Herbrand universe of $T$, which we write $\text{HERB}(T)$. Ground terms and equivalence classes of ground terms are maximal in their respective orderings, not considering the $T$ element, of course. This is obvious in the case of terms since the only instance of a ground term is itself. For equivalence classes of terms it follows from the fact that $[t] = \{t\}$ if $t$ is a ground term, and singletons are obviously maximal.

We note that $\text{HERB}(T)$ is non-empty if and only if $T_0$ is non-empty. Fix some $T$ with $T_0 \neq \emptyset$. We define a function from terms to sets of ground terms in the Herbrand universe in the obvious way, such that

$$(113) \text{GROUND}(t) = \{t_g \in \text{HERB}(T) \mid t \sqsubseteq t_g\}$$

for $t \in T$. We now note that

$$(114) t \sqsubseteq t' \text{ iff } \text{GROUND}(t) \supseteq \text{GROUND}(t').$$

One term subsumes another if and only if there are at least as many ground terms which it can be unified with. It is also the case that

$$(115) t \sqcup t' = t'' \text{ iff } \text{GROUND}(t) \cap \text{GROUND}(t') = \text{GROUND}(t'').$$

Of course, this means that we have the isomorphism
We note that it is not the case that for every set $T \subseteq \text{HERB}(T)$ of ground terms there is a term $t \in T$ such that $\text{GROUND}(t) = T$.

Recalling the use of filters, we could represent a term $t$ by the principal filter consisting of all of its instances, since we have

$$\langle T, \sqsubseteq \rangle \cong \langle \text{GROUND}(T), \supseteq \rangle.$$

In this way, we can identify a category with all of the categories it subsumes or is more general than. This set corresponds to all of the possible extensions of the partial information about an expression given by a category to more complete information.

Finally, for a term $t$, we define the set $\text{COMPAT}(t)$ of terms which are said to be compatible with $t$ by

$$\text{COMPAT}(t) = \{ t' \mid t \sqcup t' \neq T \}.$$

Yet again, we have the results

$$t \sqsubseteq t' \iff \text{COMPAT}(t) \supseteq \text{COMPAT}(t')$$

$$t \sqcup t' = t'' \iff \text{COMPAT}(t) \cap \text{COMPAT}(t') = \text{COMPAT}(t'')$$

This means that one category subsumes another if and only if it is compatible with at least as many other categories.

Taken together, these results imply that we can model the sort of domain we are dealing with by sets of compatible ground terms, by sets of instances of terms or by sets of compatible terms.

We will now turn our attention to providing a definition of a grammar based on these sorts of categories. The definition, again, will follow previous ones, and a term phrase structure grammar is defined to be a quintuple $\Gamma = \langle \Delta, \Upsilon, \Psi, \Lambda, \Phi \rangle$ where

$$\Delta \quad \text{set of basic expressions}$$

$$\Upsilon \quad \text{indexed set of function symbols } \Upsilon = \bigcup_{n \in \omega} \Upsilon_n$$

$$\Psi \quad \text{a countably infinite set of variables}$$

$$\Lambda \quad \text{a lexical assignment } \Lambda \subseteq T_\Upsilon, \Psi \times \Delta$$

$$\Phi \quad \text{a set of term phrase structure rules } \Phi \subseteq T_\Upsilon, \Psi \times (T_\Upsilon, \Psi)^*.$$
Based on this definition of a term grammar, we will say that a tree \([c_1 \cdots c_n]_0\) of depth 1 is admissible with respect to a grammar \(\Gamma\) if there is some \(d_0 \rightarrow d_1 \cdots d_n\) in \(\Phi\) such that

\[
\begin{align*}
&\text{i. there are alphabetic variants } c'_i \text{ of the } c_i \text{ such that no variable occurs in two distinct } c'_i \text{ and } c'_j, \\
&\text{ii. } \theta \text{ is the most general unifier such that } c'_i [\theta] = d_i [\theta] \text{ for } 1 \leq i \leq n, \text{ and} \\
&\text{iii. } c_0 = d_0 [\theta].
\end{align*}
\]

A tree \([w]_c\) where \(w\) is a lexical entry of category \(c\) will also be admissible, as will any element of \(T\) or \(\Delta\). We will again say that an arbitrary tree \(t \in \text{TREE}(\Delta \cup T)\) is admissible with respect to \(\Gamma\) if and only if every one of its local trees is admissible with respect to \(\Gamma\). The term phrase structure grammars as we have presented them are a variant of the definite-clause grammars, for which Pereira and Warren (1980) is the standard reference. In fact, a depth-first parser for definite-clause grammars is a standard feature of most PROLOG interpreters. Definite clause-grammars are themselves descended from the metamorphosis grammars of Colmerauer (1978) and horn-clause logic grammars of Kowalski (1974,1980).

We are still assured that the information in a mother comes from the information in the daughters and not from elsewhere in the tree. In this way, information is guaranteed to flow from the leaves to the root.

### 2.2.5 Equivalence Theorems

We now turn to some results concerning the expressive power of the term domain as compared to the feature-value domains we considered earlier. Our first equivalence theorem concerns the fact that every feature structure domain is isomorphic to some term lattice. More specifically, for a feature value lattice defined in terms of an arbitrary set of features and atoms there is some set of functional symbols and constants such that the term lattice it induces is isomorphic to the feature structure lattice. We state this formally as the
(123) Term Lattice Representation Theorem

Suppose $\text{FEAT} = \{f_1, \ldots, f_k\}$ and $\text{ATOM} = \{v_1, \ldots, v_m\}$ induce the feature structure lattice $\text{CATEQ}$. Then $\mathcal{T}_\mathcal{T} \cong \text{CATEQ}$, where the signature $\mathcal{T}$ is defined by taking $\mathcal{T}_k = \{c\}$, $\mathcal{T}_0 = \{a_1, \ldots, a_m\}$ and $\mathcal{T}_i = \emptyset$ if $i \neq 0$ and $i \neq k$.

The function symbol $c$ will be used to construct complex categories and the $a_i$ used to represent atomic values in the term domain. We will use the $i$th argument position of a functional term to represent the value of the feature $f_i$, which is why we chose $c$ to take exactly many arguments as there are feature symbols. In fact, for a term $t \in \mathcal{T}_\mathcal{T}$ we can define its value at a path $p \in \text{FEAT}^*$ by setting

(124) i. $t \vdash () = t$

ii. $c(t_1, \ldots, t_n) \vdash (f_1, g_1, \ldots, g_k) = t_i \vdash (g_1, \ldots, g_k)$.

We will now construct a lattice isomorphism $\varphi : \text{CATEQ} \to \mathcal{T}$ by setting

(125) $\varphi((c, \equiv)) = \chi(c) \cup \psi(\equiv)$

where

(126) i. $\chi(c)$ is the least term such that if $c \vdash p = v_i \in \text{ATOM}$ then $\chi(c) \vdash p = a_i \in \mathcal{T}_0$, and

ii. $\psi(\equiv) = \bigcup\{\psi'(p, q) \mid p \equiv q\}$ where for two paths $p, q \in \text{FEAT}^*$ $\psi'(p, q)$ is the least term such that $\psi'(p, q) \vdash p = \psi'(p, q) \vdash q$.

The minimal terms in question are, in fact, simple enough to construct and will always exist. It should also be fairly obvious that the mapping preserves the subsumption ordering as well as the generalization and unification operations, and is invertible.

The consequence of this is that if we fix sets $\text{FEAT}$ and $\text{ATOM}$ of feature labels and atomic values, we can find a term domain isomorphic to it. Suppose $\text{CATEQ}$ is fixed and we have constructed an isomorphic term lattice $\mathcal{T}$. We can also derive a number of corollaries to this theorem, by looking at the effect of $\varphi$ on subdomains of $\text{CATEQ}$ and $\mathcal{T}$. First of all, the closure of the Herbrand universe $\text{HERB}(\mathcal{T})^+$,
of ground terms of $\mathcal{T}$ forms a flat sublattice of $\mathcal{T}$. $\varphi^{-1}(\text{HERB}(T))$ turns out to be the subset of NESTCAT containing atoms and functional categories such that every path value is either an atom or a total function. Looking the other way, $\varphi(\text{NESTCAT})$ is just the sublattice of terms without shared variables.

It should be kept in mind that what we have done is taken fixed sets FEAT and ATOM from which we determine a lattice CATEQ. With this lattice in hand, we were able to produce a signature $\mathcal{T}$ such that $\text{CATEQ} \cong \mathcal{T}_I$. Somewhat surprisingly, it is not possible to reverse this process. We give this result as the

(127) **Feature Structure – Term Domain Inequality Theorem**

There exists a signature $\mathcal{T}$ such that there is no feature structure domain CATEQ such that $\mathcal{T}_T \cong \text{CATEQ}$.

Simply consider any signature $\mathcal{T}$ with some $f \in \mathcal{T}_n$ and $g \in \mathcal{T}_m$ for $m, n > 0$ and $f \neq g$. Suppose, by way of contradiction, that we can find some ATOM and FEAT which produce a domain CATEQ where there is an isomorphism $\phi : T_T \rightarrow \text{CATEQ}$. Consider the terms in $T_T$ of the form $t_f = f(X_1, \ldots, X_n)$ and $t_g = g(Y_1, \ldots, Y_m)$ where the $X_i$ and $Y_i$ are distinct variables. Since $f \neq g$, we have $t_f \cup t_g = T$ and $t_f \cap t_g = \bot$ in the lattice $(T_T, \subseteq)$. Consequently, we must have $\phi(t_f) \cup \phi(t_g) = T$ and $\phi(t_f) \cap \phi(t_g) = \bot$ in the lattice CATEQ, since $\phi$ is an isomorphism. Suppose the image of $t_f$ was an atom. But, we have $\bot \subseteq f(X_1, \ldots, X_n) \subseteq f(X_1, \ldots, X_1)$. Atoms, on the other hand, cover $\bot$ and are covered by $T$ in the sense that there are no elements in the order strictly between them. So, the image $\phi(t_f)$ can not be an atom, and by a similar argument, $\phi(t_f)$ must not be an atom. So, $\phi(t_f)$ and $\phi(t_g)$ must either have a non-empty category or non-trivial path equivalence relation. In either case, there will be a category which is strictly between the category and $\bot$, which leads to a contradiction, since there is no element between $t_f$ and $t_g$ and $\bot$. So, no sets FEAT and ATOM can be such that CATEQ is isomorphic to $T_T$.

While every feature-value lattice with sharing is isomorphic to some term lattice, as we saw, the converse does not necessarily hold. What does in fact hold, is that every term lattice can be isomorphically embedded in some feature-value lattice. We state this as the
(128) **Term Lattice Embedding Theorem**

For every term lattice $T_T$ there is some feature-value lattice $\text{CATEQ}$ such that there is a one-one join homomorphism (join embedding) $\phi : T_T \rightarrow \text{CATEQ}$.

Suppose $T$ is a finite signature. There will be a maximum arity, call it $n$, among the functions in $T$. We then fix sets

(129) i. $\text{FEAT} = \{m \in \omega \mid m \leq n\} \cup \{\text{rel}\}$

ii. $\text{ATOM} = T$

of feature labels and atomic values. We can then encode a term $f(t_1, \ldots, t_k) \in T_T$ as the category taking $f \in \Upsilon_k \subseteq \text{ATOM}$ as the value for rel and taking the encoding of $t_i$ as the value of the label $i$. Sharing is then encoded in exactly the same way as before, with two identical variables in the term leading to shared paths in the category. For instance, we would encode the term $f(a(X), b(c, X))$ as the feature structure

(130) $\begin{bmatrix}
\text{rel} : f \\
1 : \begin{bmatrix}
\text{rel} : a \\
1 : 1 \\
\text{rel} : b \\
2 : 1 : c \\
2 : 1
\end{bmatrix}
\end{bmatrix}
$

This general approach to encoding functional terms as feature structures can be found in Pollard and Sag (1987) and Zeevat, Klein and Calder (1987). It is easy to verify that this is a join embedding in the sense that for every term $s, t \in T$,

(131) $\phi(s \sqcup t) = \phi(s) \sqcup \phi(t)$.

Significantly, this is not a meet homomorphism, since for instance,

(132) $\phi(f(a) \sqcap f(b)) = \phi(\perp) \neq \phi(f(a)) \sqcap \phi(f(b)) = \begin{bmatrix} \text{rel} : f \end{bmatrix}$.

In a sense, the feature structure domain provides more informative meets, but equivalent joins.

With this result, we are assured that for every feature phrase structure grammar there will be a strongly equivalent term phrase structure grammar and conversely. This can be stated as
Term Grammar and Feature Grammar Equivalence

Theorem

For every term unification grammar $\Gamma = (\Delta, \Sigma, \mathcal{N}, \lambda, \Phi)$ there is a strongly equivalent feature structure unification grammar

$\Gamma' = (\text{FEAT}, \text{ATOM}, \Delta, \lambda, \Phi)$, and conversely.

We will not present the actual construction here, as it is carried out by the obvious extension of our previous embedding of categories to rules. This brings us to the conclusion that there is no theoretical significance to the choice between term and feature-value domains, since the grammars produced in either domain have strongly equivalent analogues in the other domain. The decision as to which to use will be merely a matter of notational convenience.

As we noted earlier, for an arbitrary language over some set of basic expressions, a phrase structure grammar can be found that generates it. Of course, it follows that recognition in phrase structure grammars is not, in general, decidable. Somewhat more surprisingly, recognition is not even decidable for arbitrary finite term grammars or feature-value grammar. We present this as the more powerful

Turing Machine Enumeration Theorem

For any language recognized by a turing machine, there is a finite term unification (and hence a finite feature structure unification) grammar that generates it for some category.

Thus the set of languages generated by unification grammars are exactly the recursively enumerable languages. To see that this is so, consider an encoding of a Turing machine tape within the features of a category, which can most easily carried out with three features for the part of the tapes to the left and right of the scanner and one feature for the value being scanned. The rules can then be chosen such that they correspond to the transitions of any particular Turing machine, including the universal machine. A more formal proof is given in Andreka and Nemeti (1976). For details of Turing machines and decidability issues, see Hopcroft and Ullman (1979) or Lewis and Papadimitriou (1981).
2.2.6 Simple English Term Grammar

To extend the fragment of English we gave before, we present a grammar using simple first-order functional terms as that seems the most perspicuous. We will use the categories

\[
\begin{array}{ll}
\text{Category} & \text{Parts of Speech} \\
n(N) & \text{noun with number } N \\
np(N,C) & \text{noun phrase with number } N \text{ and case } C \\
det(N) & \text{determiner with number } N \\
tv(N) & \text{transitive verbs with number } N \\
bv(N) & \text{ditransitive verbs with number } N \\
iv(N) & \text{verb phrases with number } N \\
ad(L) & \text{adverbs with location } L \\
adj & \text{adjectives} \\
p & \text{prepositions} \\
pp & \text{prepositional phrases} \\
s & \text{sentences}
\end{array}
\]

with the lexical entries
Category | Words
---|---
\(n(X)\) | sheep, fish
\(n(\text{sing})\) | kid, penguin, cat, senator, band
\(n(\text{plu})\) | kids, penguins, cats, senators, bands
\(\text{np}(\text{sing},X)\) | it
\(\text{np}(\text{sing,subj})\) | he, she
\(\text{np}(\text{sing,subj})\) | him, her
\(\text{np}(\text{plu},X)\) | you
\(\text{np}(\text{plu,subj})\) | they
\(\text{np}(\text{plu,subj})\) | them
\(\text{det}(X)\) | the, some
\(\text{det}(\text{sing})\) | a(n), every
\(\text{det}(\text{plu})\) | most, three
\(\text{iv}(X)\) | sneezed, ran, played, ate
\(\text{iv}(\text{sing})\) | sneezes, runs, plays, eats
\(\text{iv}(\text{plu})\) | sneeze, run, play, eat
\(\text{tv}(X)\) | loved, cheated, knew, saw
\(\text{tv}(\text{sing})\) | loves, cheats, knows, sees
\(\text{tv}(\text{plu})\) | love, cheat, know, see
\(\text{bv}(X)\) | gave, sent, loaned
\(\text{bv}(\text{sing})\) | gives, sends, loans
\(\text{bv}(\text{plu})\) | give, send, loan
\(\text{adv}(X)\) | slowly, gracefully, quickly
\(\text{adv}(\text{pre})\) | necessarily, probably
\(\text{adv}(\text{post})\) | yesterday, today
\(\text{adj}\) | tall, short, silly, cool
\(p\) | in, on, under, by

and the rules

\[
\begin{align*}
137 & \quad s & \rightarrow & \quad \text{np}(\text{N,subj}) \quad \text{vp}(\text{N}) \\
\text{np}(\text{N,X}) & \rightarrow & \quad \text{det}(\text{N}) \quad \text{n}(\text{N}) \\
\text{n}(\text{N}) & \rightarrow & \quad \text{adj} \quad \text{n}(\text{N}) \\
\text{n}(\text{N}) & \rightarrow & \quad \text{n}(\text{N}) \quad \text{pp} \\
\text{vp}(\text{N}) & \rightarrow & \quad \text{tv}(\text{N}) \quad \text{np}(\text{X,subj}) \\
\text{vp}(\text{N}) & \rightarrow & \quad \text{bv}(\text{N}) \quad \text{np}(\text{X,subj}) \quad \text{np}(\text{Y,subj}) \\
\text{vp}(\text{N}) & \rightarrow & \quad \text{vp}(\text{N}) \quad \text{adv}(\text{post}) \\
\text{vp}(\text{N}) & \rightarrow & \quad \text{adv}(\text{pre}) \quad \text{vp}(\text{N}) \\
\text{vp}(\text{N}) & \rightarrow & \quad \text{vp}(\text{N}) \quad \text{pp} \\
\text{pp} & \rightarrow & \quad \text{p} \quad \text{np}(\text{X,subj})
\end{align*}
\]
The following parse trees are admissible with respect to this grammar

(138)  
\[
\begin{array}{c}
\text{every} \\
\text{det(sing)} \\
\text{n(sing)} \\
\text{ate} \\
\text{vp(Y)} \\
\text{np(sing, X)} \\
\text{s}
\end{array}
\]

(139)  
\[
\begin{array}{c}
\text{the} \\
\text{det(X)} \\
\text{sheep} \\
\text{n(Y)} \\
\text{ate} \\
\text{vp(Z)} \\
\text{np(U, V)} \\
\text{s}
\end{array}
\]

(140)  
\[
\begin{array}{c}
\text{the} \\
\text{det(X)} \\
\text{man} \\
\text{n(sing)} \\
\text{eats} \\
\text{vp(sing)} \\
\text{np(sing, X)} \\
\text{s}
\end{array}
\]

which are similar in their information structure to the ones we saw for the feature-value category parse trees, except that we have now added case information to our noun phrase category, which allows for the following simple distinctions in the case of noun phrases

(141)  
\[
\begin{array}{c}
\text{him} \\
\text{np(sing, obj)} \\
\text{saw} \\
\text{tv(B)} \\
\text{john} \\
\text{np(sing, C)} \\
\text{vp(A)} \\
\text{s}
\end{array}
\]

(142)  
\[
\begin{array}{c}
\text{john} \\
\text{np(sing, A)} \\
\text{saw} \\
\text{tv(C)} \\
\text{he} \\
\text{np(sing, obj)} \\
\text{vp(B)} \\
\text{s}
\end{array}
\]

(143)  
\[
\begin{array}{c}
\text{he} \\
\text{np(sing, subj)} \\
\text{saw} \\
\text{tv(C)} \\
\text{him} \\
\text{np(sing, obj)} \\
\text{vp(B)} \\
\text{s}
\end{array}
\]

2.2.7 Extended Information Systems

In this section, we will consider various extensions of the basic feature systems we have presented so far and consider some of their properties.
Sorted Signatures

In the term lattice, each function symbol comes with an associated arity which determines the number of arguments it must be given to form a term. No restriction is placed on what sorts of terms may show up in which places. For example, consider the categories from our simple term grammar given in (135). We used the binary function symbol np to construct noun phrases, where the first argument picks out the number and the second argument picks out the case. As we have defined things, there is nothing that requires the number slot to be filled with values like sing and plu rather than getting values like obj or subj which would be appropriate for filling the case slot, or even complex terms like np(bv(p), det(s)). It just so happens, that the way we set up the lexicon, such terms will never be categories in a parse-tree.

The easiest way to place 'restrictions on what sorts of terms can apply as arguments is by means of a sorting placed on the function symbols in a functional signature. We define a sorted signature to be a pair \((T, \Theta)\) where \(\Theta\) is a set of basic sort symbols to represent our sorts and \(\Upsilon = \bigcup_{\theta \in \Theta} \Upsilon_\theta\) is an indexed set of function symbols. We think of a function symbol \(f \in \Upsilon_{(\theta, \theta_1, \ldots, \theta_n)}\) as itself forming terms of sort \(\theta\) and taking \(n\) arguments of sorts \(\theta_1, \ldots, \theta_n\). Elements of \(\Upsilon_{(\theta, e)}\) will correspond to constants of sort \(\theta\). We will also fix a countable set of variables, indexed by sort, with \(\mathcal{V} = \bigcup_{\theta \in \Theta} \mathcal{V}_\theta\) where \(\mathcal{V}_\theta = \{v^\theta_i \mid i \in \omega\}\).

We can give an inductive definition of the sets \(\mathcal{T}_\theta\) of well-formed terms of sort \(\theta \in \Theta\) by taking the least sets such that

\[
(144) \quad \mathcal{T}_\theta = \mathcal{V}_\theta \cup \{f(t_1, \ldots, t_n) \mid f \in \Upsilon_{(\theta, (\theta_1, \ldots, \theta_n))}, t_i \in \mathcal{T}_{\theta_i}\}
\]

The subsumption relation on terms of these sorts is analogous to the subsumption relation on unsorted terms. In the case of sorted terms, we require a substitution \(\sigma : \mathcal{V} \rightarrow \mathcal{T}\) indexed by sort so that \(\sigma = \bigcup_{\theta \in \Theta} \sigma_\theta\) where \(\sigma_\theta : \mathcal{V}_\theta \rightarrow \mathcal{T}_\theta\) is an assignment of \(\theta\) sort terms to \(\theta\) sort variables. Again, a term will be called an instance of a second term if it is the result of applying a substitution to the second term. A term subsumes its instances, and unification is definable just as before.
By way of example, we provide a sort assignment for our term phrase structure grammar. We first fix a set of sort symbols

\( \Theta = \{ \text{num, case, loc, cat} \} \).

We can then classify our function symbols by sort, taking

\begin{align*}
(145) & i. \text{sing, plu} \in \Gamma(\text{num}, \emptyset) \\
& ii. \text{subj, obj} \in \Gamma(\text{case}, \emptyset) \\
& iii. \text{pre, post} \in \Gamma(\text{loc}, \emptyset) \\
& iv. \, n \in \Gamma(\text{cat}, \text{num}) \\
& v. \, np \in \Gamma(\text{cat}, \text{num}, \text{case}) \\
& vi. \, s \in \Gamma(\text{cat}, \text{num})
\end{align*}

For instance, sing is a basic term of sort num, n is a function symbol that takes a term of sort num to produce a term of sort cat. It should be obvious that all of our lexical assignments for expressions are terms of sort cat and that our rules operate over terms of sort cat.

Suppose we fix a sorted signature \((T, \Theta)\). We define a sort accessibility relation \(\prec\) on \(\Theta\), where we assume that for \(\theta, \tau \in \Theta\), we have \(\theta \prec \tau\) if and only if some term of sort \(\theta\) can occur within some term of sort \(\tau\). More precisely, we set \(\theta \prec \tau\) if and there is some function symbol \(f \in T(\tau_i, \ldots, \tau_n)\) such that for some \(i, 1 \leq i \leq n\), either \(\theta = \tau_i\) or \(\theta \prec \tau_i\). A sort \(\tau_0 \in \Theta\) is said to be an infinite sort if there is some infinite sequence \(\tau_0, \tau_1, \ldots, \tau_n, \ldots\) such that \(\tau_n \prec \tau_{n+1}\) for each \(n \in \omega\). A sort is called a finite sort if it is not infinite. We chose the names infinite and finite for the good reason that \(T_\Theta\) will contain an infinite number of ground terms if and only if \(\theta\) is an infinite sort, which is easily proved by contradiction. A signature \((\mathcal{Y}, \Theta)\), such that every \(\theta \in \Theta\) is finite is said to be a finite signature. It turns out that a signature \((\mathcal{Y}, \Theta)\) is a finite signature if and only if there is some stratification function \(s : \Theta \rightarrow \omega\) such that if \(f \in T(\theta_i, \theta_{i+1}, \ldots, \theta_n)\) we have \(s(\theta_i) < s(\theta)\) for \(1 \leq i \leq n\).

Suppose we take a term grammar based on a finite sorted signature \((\mathcal{Y}, \Theta)\) so that \(\mathcal{Y}\) is finite and every sort \(\theta \in \Theta\) is finite. Furthermore, suppose that we also fix a finite set of basic expressions. Each term grammar that can be defined
over such a signature and set of basic expressions will turn out to be strongly equivalent to some simple context-free grammar, since there will only be a finite number of unique categories. For instance, our example sorting provides a finite signature, which shows that our example term grammar is strongly equivalent to a context-free grammar.

Ordered Basic Domains

Suppose that the set \( \text{ATOM} \) of atomic values in \( \text{CATEQ} \) or the set \( \Gamma \) of function symbols was given a primitive subsumption ordering of its own. We could then extend the corresponding unifications over the whole domains in a very natural way. In the case of the term lattice, we simply extend the definition of subsumption so that \( a_1(t_1, \ldots, t_n) \sqsubseteq a_2(t_1, \ldots, t_n) \) if and only if \( a_1 \sqsubseteq a_2 \) in the ordering on function symbols and each \( t_i \).

The representational power of this kind of system will depend on what kind of primitive subsumption relations are allowed on the function symbols or atoms. Ait-Kaci (1984) contains a thorough discussion of the properties of these systems along with a number of possible applications. One motivation for this kind of primitive ordering on the atomic values is its usefulness in encoding primitive sort and inheritance information of any kind at a basic level, thereby separating it from the partial information encoded in the values of features. This has been found to lead to a great speed-up in processing time, both for natural language systems and general logic-programming systems. For instance, see Ait-Kaci and Nasr (1986) for a logic programming language based on an ordered domain of function symbols. Other studies of sorting for linguistic category descriptions can be found in Calder (1987) Pollard and Sag (1988) and Pollard (1988b).

Additional Values

It is certainly plausible to allow the value of a feature value or filler of an argument position to be something other than a complex category or an atomic value. One such "extension" would be to allow finite sequences or lists of categories. Suppose we want to add list values to a functional signature \( \Gamma \). To add lists to \( \Gamma \), we add
a distinguished constant \texttt{NIL} to \( \mathcal{T}_0 \) and a binary list constructor \texttt{.} to \( \mathcal{T}_2 \). We then encode the empty list \( \langle \rangle \) as the atom \texttt{NIL}, and encode a non-empty list \( \langle t_1, \ldots, t_n \rangle \) as \( \langle t_1, \text{tail} \rangle \), where \texttt{tail} is the encoding of \( \langle t_2, \ldots, t_n \rangle \). So, for instance, the list \( \langle t_1, t_2, t_3 \rangle \) would be encoded as \( \langle t_1, \langle t_2, \langle t_3, \texttt{NIL} \rangle \rangle \rangle \). With this encoding for lists, it turns out that if we have lists \( \langle t_1, \ldots, t_n \rangle \) and \( \langle s_1, \ldots, s_n \rangle \) composed of terms \( s_i \) and \( t_i \) encoded in this fashion, then

\[
(147) \langle t_1, \ldots, t_n \rangle \cup \langle s_1, \ldots, s_n \rangle = \langle t_1 [\theta], \ldots, t_n [\theta] \rangle
\]

if \( \theta \) is the most general substitution which is a unifier for each pair \( t_i \) and \( s_i \). This is arguably the notion of list we want, where two lists will be incompatible if they do not have the same number of elements. This encoding of lists is the standard encoding used by the \textsc{PROLOG} interpreter and can be found in any introductory text (see Sterling and Shapiro (1986)) For an equivalent encoding of lists in terms of features, see Pollard (1984) or Gazdar, et al. (1988).

It might also be desirable to give some kind of encoding of sets in our category systems, so that a slot could be filled with a set of values rather than just one. In our term-based formalism we can simply take sets of terms to themselves be well-formed terms. We then extend our definition of subsumption so that

\[
(148) \{t_1, \ldots, t_n\} \sqsubseteq \{s_1, \ldots, s_m\}
\]

if and only if there is some substitution \( \sigma \) such that for every \( t_i \) there is some \( s_k \) such that \( t_i [\sigma] = s_k \). The subsumption relation now behaves as in

\[
(149) \begin{array}{l}
i. \{a(b)\} \sqsubseteq \{a(X), a(b)\} \sqsubseteq \{a(b)\} \\
ii. \{a(Y), a(X)\} \sqsubseteq \{a(Z)\} \\
iii. \{a(b, X), a(Y, c)\} \sqsubseteq \{a(b, c)\} \\
iv. \{a(X), b(X)\} \sqsubseteq \{a(c), b(c)\} \\
v. \{a(X), b(X)\} \not\subseteq \{a(c), b(Y)\}.
\end{array}
\]

The weakness associated with this encoding of sets is that there is no way in which to count the elements of sets as we have represented them here, as should be obvious from the above subsumptions. There is simply no way to put a restriction on a set term that limits the number of elements in a possible
extension. This is the same ordering placed on sets as was used by Rounds (1988) in the domain of feature structures. A different ordering is employed by Pollard and Moshier (forthcoming) which does not allow unification to add elements to a set, so that a set with \( n \) elements will only unify with other sets of \( n \) elements. To add elements to sets, Pollard and Moshier involve operations other than direct unification. For linguistic applications of set values in linguistic formalisms, see Kaplan and Bresnan (1982) and Pollard and Sag (1987).

**Additional Operations**

Besides adding values, we can also add additional operations to our lattice of terms. The most obvious choice for this would be the so-called *generalisation* operator, which is simply the dual of unification, in that it corresponds to greatest lower bounds in the lattice of categories rather than least upper bounds. Unfortunately, while unification corresponds to our intuitive notion of information conjunction, generalisation will not correspond to our intuitive notion of information disjunction. Simply consider the subsumptions

\[(150) \ a(b) \cap a(c) = a(X) \subseteq a(d)\]

which is somewhat counterintuitive. The possibility of using generalisation for representing disjunctive information was first recognised by Pereira and Shieber (1984). See Karttunen (1984) for another early study of a computational mechanism for incorporating disjunctive information into feature structures.

Another approach to disjunctive feature information, which due to Pollard and Moshier (forthcoming), is to model categories not in the standard information lattice \( L \), but in the domain of finite unions of principal filters of our ordinary terms, which we will call \( \text{FUPF}(L) \). More precisely, we have

\[(151) \ \text{FUPF}(L) = \{ \cup F \mid F \subseteq \uparrow L, \ F \text{ finite} \}\]

ordered by \( \supseteq \). Since \( \uparrow \) is still an isomorphic embedding of \( L \) in \( \text{FUPF}(L) \), we could think of categories as members of \( \text{FUPF}(L) \). In fact, the image of the unification of two elements of \( L \) is just the intersection of their principal filters in \( \text{FUPF}(L) \), which turns out to be the filter generated by their unification, so that
\[(152) \uparrow (c \sqcup d) = (\uparrow c) \cap (\uparrow d)\]

for \(c, d \in L\). We can now encode the information in the disjunction of two categories as the union of their encodings, which will fall in \(\text{FUPF}(L)\) which is obviously closed under finite unions by definition. This will not be the image of generalisation, though, since we find

\[(153) \uparrow (c \cap d) \neq (\uparrow c) \cup (\uparrow d)\]

for \(c \neq d\). The move to the domain of \(\text{FUPF}(L)\) fills in gaps in the original domain by providing values for disjunctive information which are not images of elements from the original domain \(L\). A construction of this sort was first proposed by Smyth (1978) in the context of domains for denotational semantics. For linguistic applications of disjunctive information of this sort, see Pollard and Sag (1987). A logic was developed by Kasper and Rounds (1986) and Rounds and Kasper (1986) to deal with disjunctive information. Kasper (1987) introduced an efficient algorithm for computing the disjunction of feature structures. A semantics for the logic in terms of automata and feature structures was then developed by Moshier and Rounds (1987) and Moshier (1988).

The exact same problem develops in the case of negative information, since even though the term lattice is complemented, complements are not unique. Consider

\[(154)\]

\[\begin{align*}
\text{i. } & (a(b) \cap a(c)) = (a(b) \cap a(d)) = \bot \\
\text{ii. } & a(b) \cup a(c) = a(b) \cup a(d) = T.
\end{align*}\]

This time, even the move to the domain of finite unions of partial filters will not help out, as that domain is also not complemented in the way we would like it to be. Pollard (1984) introduced the idea of negation and disjunction for atomic features in feature structures. For a study of negation in feature structures based on negation in intuitionistic logic, see Moshier and Rounds (1987) and Moshier (1988). Karttunen (1984) discusses the computational problems faced by systems attempting to integrate partial information and negation. Johnson (1987) contains an alternative logic and semantics for feature structures with disjunction and negation that very closely parallels that of first-order logic, along with a general discussion of the usefulness of such feature structures in linguistics.
Restricted Domains

Another approach to extending the representational power of a given system is to restrict attention to subdomains which have categories respecting certain properties. The usual strategy is to provide some set of restrictions on which features and possible values can show up as part of the same category. This idea was introduced by Chomsky and Halle (1968) in the domain of phonology, but was quickly assimilated into the generalized phrase structure treatment of features (Gazdar, Pullum and Sag 1982, Gazdar et al. 1985). In generalized phrase structure grammars, a logic of feature-coocurrence restrictions was developed and then later extended by Gazdar et al. (1988). For instance, if a category had a value for the verb form feature then its value for the major feature had to be verb (Gazdar et al. 1985).

The major strength of these systems is that they allow a very general expression of negative information, disjunctive information and even information requiring some features to have values. Similarly, any of these requirements could be enforced recursively throughout a category and restricted to particular categorial contexts based on the values or existence of values for other features. The major restriction of these systems is that they lack a mechanism to express shared values, which we will find crucial for many of our needs later on in both syntax and semantics. The other shortcoming of this approach is that the domain of categories can be restricted in such a way that it does not form a semilattice due to the exclusion of too many possible categories needed for bounds.

2.3 Pure Applicative Categorial Grammar

In this section, we introduce the simplest kinds of categorial grammars. We see what categories look like in categorial grammar and how they combine by functional application. We introduce a particular method of encoding features in categorial grammars and of encoding categorial grammars as a simple sort of term phrase structure grammar with two fixed rules. We will finally consider the generative power of categorial grammars and their extensions incorporating
feature information. Unlike the extensions of context-free grammars to features, which turn out to be undecidable, the natural feature based extensions to categorial grammars turn out to be not only decidable, but to have efficient parsing algorithms. Along the way, we pause to give two categorial grammar fragments, the first weakly equivalent to the Simple English Phrase Structure Grammar, and the second weakly equivalent to the Simple English Term Grammar.

2.3.1 Categories and Application

In categorial grammars, which were introduced by Ajdukiewicz (1935), syntactic categories are thought of as being essentially functional in nature. A category will either be chosen from a set of basic categories or be a function from categories to categories. Restrictions on the domain and range of a functional category will determine what sort of categories will serve as arguments and what the result of applying the function will be.

To be more precise, we fix a set $\text{BasCat}$ of basic categories. We then take the full set $\text{Cat}(\text{BasCat})$ of categories to be the smallest such that

\begin{align*}
\text{(155)} \quad &\text{i. } \text{BasCat} \subseteq \text{Cat}(\text{BasCat}), \text{ and} \\
&\text{ii. } \alpha / \beta \in \text{Cat}(\text{BasCat}) \text{ and } \alpha \backslash \beta \in \text{Cat}(\text{BasCat}) \text{ if } \alpha, \beta \in \text{Cat}(\text{BasCat}).
\end{align*}

The idea is that an expression of the category $\alpha / \beta$ followed by an expression of the category $\beta$ will form an expression of category $\alpha$. Similarly, an expression of the category $\alpha \backslash \beta$ preceded by an expression of the category $\beta$ will also form an expression of the category $\alpha$. We note that as long as $\text{BasCat}$ is non-empty, the set $\text{Cat}(\text{BasCat})$ will be infinite. We will often write $\text{Cat}$ for $\text{Cat}(\text{BasCat})$ when $\text{BasCat}$ is understood. The introduction of two directional slash operators to represent complement order is due to Bar-Hillel (1950,1953), whose categorial grammars are essentially what we are calling pure categorial grammars.

A category $\alpha \backslash \beta$ or $\alpha / \beta$ is called a functor category and is said to have a domain or argument category of $\beta$ and a range or result category of $\alpha$. For instance, if $\text{BasCat} = \{a, b\}$, then $a \backslash b$, $(a \backslash b) / b$, $((a \backslash b) / b) \backslash a$ and $(a \backslash b) / (a \backslash b)$ are all in $\text{Cat}$. From now on, we will omit parentheses within categories, taking the
slashes to be left-associative operators, so that \( a \backslash b / b = (a \backslash b) / b, a \backslash b / b \backslash a = ((a \backslash b) / b) \backslash a \) and \( a \backslash b / (a \backslash b) = (a \backslash b) / (a \backslash b) \).

We will now define a pure categorial grammar over a set BASCAT of basic categories to be a triple \( \Gamma = (\Delta, \text{BASCAT}, \Lambda) \) such that

(156) \[
\begin{align*}
\Delta & \quad \text{set of basic expressions} \\
\text{BASCAT} & \quad \text{set of basic categories} \\
\Lambda & \quad \text{a lexical assignment} \\
\Lambda & \subseteq \text{CAT(BASCAT)} \times \Delta
\end{align*}
\]

Suppose we fix a pure categorial grammar \( \Gamma = (\Delta, \text{BASCAT}, \Lambda) \). We could convert \( \Gamma \) into a phrase structure grammar

(157) \[
\text{PSG}(\Gamma) = (\text{CAT(BASCAT)}, \Delta, \Lambda, \text{PSGRule(BASCAT)})
\]

where for a given set \( \Sigma \) of basic categories we define the set of categorial application rules over \( \Sigma \) to be

(158) \[
\text{PSGRule}(\Sigma) = \{ \alpha \rightarrow \alpha / \beta | \alpha, \beta \in \text{CAT}(\Sigma) \} \\
\cup \{ \alpha \rightarrow \beta \alpha \backslash \beta | \alpha, \beta \in \text{CAT}(\Sigma) \}
\]

As long as \( \Sigma \) is non-empty, the set \( \text{PSGRule}(\Sigma) \) will be infinite. But, if the lexical relation is finite, then we can find a finite subset of \( \text{PSGRule(BASCAT)} \) which provides a grammar strongly equivalent to the one with the full set, since only a finite number of categories could ever enter into parse trees. This follows from the finiteness of the lexicon and the fact that the only non-lexical categories that could enter a tree are subcategories of the lexical categories, of which there will only be finitely many. As a result, any finite categorial grammar will be strongly equivalent to a context-free grammar. We will simply inherit the notion of tree and derivation from those given for phrase structure grammars, so that it will make sense to talk about derivations and trees for pure categorial grammars.

Considering the restriction of phrase structure grammars to categorial grammars, we see that the set \( \mathcal{L}(c) \) of expressions of category \( c \) is the least such that

(159) \[
\begin{align*}
\text{i. } & \Lambda(c) \subseteq \mathcal{L}(c) \\
\text{ii. } & \mathcal{L}(c / c') \times \mathcal{L}(c') \subseteq \mathcal{L}(c) \\
\text{iii. } & \mathcal{L}(c') \times \mathcal{L}(c \setminus c') \subseteq \mathcal{L}(c)
\end{align*}
\]
for every pair of categories \( c, c' \in \text{CAT} \). As before, we take the set \( \mathcal{C}(c) \) of strings of categories that \( c \) can be rewritten as to be the least such that

\[
\begin{align*}
\text{(160)} & \quad \text{i. } c \in \mathcal{C}(c) \\
& \quad \text{ii. } \mathcal{C}(c/c') \times \mathcal{C}(c') \subseteq \mathcal{C}(c) \\
& \quad \text{iii. } \mathcal{C}(c') \times \mathcal{C}(c \setminus c') \subseteq \mathcal{C}(c)
\end{align*}
\]

Similarly, we can relativise the definition of the set \( \mathcal{T} \) of admissible trees to categorial grammars.

It should be apparent that this definition of derivation is just the restriction of the ordinary one to the case of categorial phrase structure rules, so that if \( \Gamma \) is a categorial grammar, then for every \( c \in \text{CAT(BASCAT)} \) we have

\[
\begin{align*}
\text{(161)} & \quad \text{i. } \mathcal{L}_\Gamma(c) = \mathcal{L}_{\text{PSG}(\Gamma)}(c) \\
& \quad \text{ii. } \mathcal{G}_\Gamma(c) = \mathcal{G}_{\text{PSG}(\Gamma)}(c) \\
& \quad \text{iii. } \mathcal{T}_\Gamma(c) = \mathcal{T}_{\text{PSG}(\Gamma)}(c)
\end{align*}
\]

### 2.3.2 Equivalences and Variants

Quite surprisingly, it has been proved by Gaifman that the every context-free grammar is weakly equivalent to some categorial grammar with a finite number of lexical entries (see Bar-Hillel, Gaifman and Shamir (1960), Zielonka (1978)). We present this as the

\[
\text{(162) Context-Free — Categorial Grammar Equivalence Theorem (Gaifman)}
\]

For any language for which there is a context-free, grammar, there is a weakly equivalent finite pure categorial grammar, and conversely.

Suppose we fix a context-free grammar. We can convert it to Greibach normal form where every rule is of the form

\[
\text{(163) } c \rightarrow a \ x_1 \cdots x_n
\]
for some $n \geq 0$ and $a$ is a pre-terminal category. A \textit{pre-terminal} category is one which does not show up on the left hand side of any rewrite rule, but is only allowed to have lexical entries. Every context-free grammar is weakly equivalent to some context-free grammar in Greibach normal form (see Greibach (1965) or Hopcroft and Ullman (1979) for a proof). We can then find a weakly equivalent categorial grammar by taking a lexical category

\begin{equation}
(164) \ c/x_\delta/x_1
\end{equation}

for each basic expression $\delta \in \Lambda(a)$ assigned to the category $a$ in the Greibach normal form grammar. This categorial grammar will be weakly equivalent to the original grammar with respect to every category that shows up on the left hand side of some rewrite rule, as can be proved by a simple induction on the depth of admissible trees. We should note that an algorithm was developed by Rosenkrantz (1967) that produces a weakly equivalent Greibach normal form grammar which contains no more than $n^2$ rules, where $n$ is the number of rules in the original grammar.

Consequently, if we are thinking of languages solely in terms of syntactic well-formedness, then there is nothing to choose between categorial grammars and context-free grammars. Of course, when one considers extensions to the basic frameworks, one is obviously led in different directions when starting from the different grammars. For instance, the treatment of unbounded dependencies and coordination in the context-free tradition has led to the development of extrapolation grammars (Pereira 1981), generalized phrase structure grammars (Gazdar 1981b), and head-driven phrase structure grammars (Pollard and Sag 1987). Similar trends in categorial grammar began earlier with Lambek's (1958,1961) associative and non-associative syntactic calculi and Geach's (1972) recursive extension to categorial grammar, followed by Steedman's (1985,1987,1988) combinatory categorial grammars. These extensions to the basic systems have quite different properties.

Similarly, if we take syntactic investigation to involve more than finding a grammar that generates the right set of strings, then a means might be discovered for distinguishing between the context-free and finite categorial grammars. Most
categorial grammarians are primarily interested in semantics, as can be seen most transparently in the work of Montague (1974b,1974c) and Bach (1980,1983,1984). As we will see in Chapter 4, categorial grammars can be naturally interfaced with a functional theory of semantics.

In the construction of a categorial grammar equivalent to a phrase structure grammar, we only need categories, and hence rules, containing slashes of one direction, as we saw above. The system of categorial grammar with slashes of one direction is not the same as what is known as unidirectional categorial grammar. The unidirectional system will take a set $\Sigma$ of basic categories and admit complex categories of the form $\alpha | \beta$ for categories $\alpha$ and $\beta$. In the unidirectional system, functors apply in either direction so that we have the rule schemata

\begin{align*}
1. \text{(ufa)} & \quad \alpha \rightarrow \alpha | \beta \beta \\
2. \text{(uba)} & \quad \alpha \rightarrow \beta \alpha | \beta
\end{align*}

Unidirectional categorial grammars are primarily of interest to those interested only in semantics, as additional mechanisms are obviously required to get the syntactic details of a configurational language like English right. Pollard (1988) has argued that such a division of labour is actually desirable in the form of a categorial system converted to ID/LP format. The ID/LP system, introduced by Gazdar and Pullum (1981), factors phrase structure rules into two independent grammar components. The first component of immediate dominance (ID) rules determines possible daughters for a category. The second component consists of linear precedence (LP) constraints that determine the possible orderings of categories in any local tree. A local tree admissible in an ID/LP grammar must have daughters which are introduced by some ID rule arranged in an order consistent with all of the linear precedence constraints.

### 2.3.3 Simple English Pure Categorial Grammar

By way of example, we will now present a fragment of English in pure categorial grammar and see that it is in fact, weakly equivalent to the simple English Phrase Structure grammar we gave in Section 2.1.2. We will assume the basic categories
Notice that this is just a small subset of those in the phrase structure grammar.

We then employ the lexical assignment

Of course, we need not provide any rules, since these will be determined from BASCAT. Parse trees for the expressions we saw earlier are

```
(168)          opus    ran
              np            s \ np
                   s

(169)        the    silly    penguin
              np / n   n / n       n
                   n
                    np

(170)    hit    the    penguin    in    Bloom County
              s \ np / np   np / n   n   n \ n / np    np
                                  n \ n
                                       n
                                        np
                                         s \ np
```
hit the penguin in bloom county

where we see that the phrase hit the penguin in bloom county is given the same two tree structures as in the phrase structure grammar.

Up until now, we have not discussed the ditransitive verb category which we provide for verbs taking two object complements, such as gave and loaned. In the simple context-free grammar, these would form verb phrases such as in the tree

In the corresponding categorial grammar, the ditransitive verb loaned receives the category $s \ np / np / np$. The corresponding parse tree for the simple categorial grammar is

A tree is said to be binary branching if every one of its local trees consists of a mother category dominating either a basic expression or exactly two other categories. From the form of the rule schemata, it should be obvious that all parse-trees in a categorial grammar must be binary branching. In the example above, we see that the categorial grammar analysis has the expression loaned opus categorised as $s \ np / np$. What this means is that the expression loaned opus will behave in all respects like a transitive verb such as saw, loved or hit. This is an important fact for the categorial analysis of unbounded dependencies and coordination. Note that there are no parses of the expression loaned opus in the phrase structure grammar. Of course, the phrase structure rule

could be replaced by the rule
which would not affect the set of well-formed verb phrases and sentences in the grammar, but would increase the set of well-formed transitive verbs, as well as increasing the similarity to the categorial binary branching analysis of constituency in the complements of ditransitive verbs.

2.3.4 Features in Categorial Grammar

We can extend categorial grammars in the exact same ways as we extended the more general phrase structure grammars. That is, we can add features and feature-passing information to the categories and rules of a categorial grammar. We carry out this development in the term system, but wish to reinforce the fact that this is merely an aesthetic decision.

To be more precise, we will assume that we are dealing with a functional signature $\Upsilon$ such that $\Upsilon_2$ contains the two slash constructors / and \. Such a signature will be called a directionally slashed signature. To work within the sorted framework, it is merely necessary to assume that the sort of the two slash constructors is $(\text{cat}, (\text{cat}, \text{cat}))$, so that they take two objects of the category sort to produce another object of the category sort.

We can then write down the two simple term phrase structure rules

\begin{align*}
(176) \quad & \text{i. (fa)} \quad X \rightarrow X / Y \quad Y \\
& \text{ii. (ba)} \quad X \rightarrow Y \quad X \backslash Y
\end{align*}

which correspond to the forward and backward application schemata. Following categorial grammar tradition, we write the slash operators in infix notation, but it should be remembered that they are really only distinguished elements of $\Upsilon_2$. We note that the power of term phrase structure grammars is such that we can have forward and backward application each corresponding to a single rule rather than a collection of rules generated from a schematic application rule.

To preserve this simple notion of application we have in effect forced features down to the level of basic categories. For a directionally slashed signature $\Upsilon$, we say that a category $c \in \Upsilon$ is forward looking if $c = \alpha / \beta$, backward looking if
\( c = \alpha \setminus \beta \), and basic, otherwise. A forward looking category \( \alpha / \beta \) or backward looking category \( \alpha \setminus \beta \) is said to be a functor or slash category with domain or argument \( \beta \) and range or result \( \alpha \).

Of course, this is not the only possibility for the addition of feature information to categorial grammars. As it is, we either have a basic category or information on the two subcategories of a functor category. The important thing to notice is that there is no way to put information on a slash category as a whole, since all of the information must go into one of the subcategories. Some approaches to categorial grammars allow a more generous distribution of features. For instance, categorial grammars employing feature structures, such as those employed by Zeevat Klein and Calder (1987), Uszkoreit (1986) and Karttunen (1986b), allow features to attach to complex categories as well as basic categories.

Even though term grammars are in general undecidable, restricting attention to the case of term categorial grammars leads to a simple decision procedure. Since unification is decidable and only binary phrase structure rules are used, the possible binary branching analysis trees can be enumerated and tested.

### 2.3.5 Simple English Agreement Categorial Grammar

In this section we present a term categorial grammar for English which is weakly equivalent to our earlier English term grammar. We take the basic categories

\[
\begin{align*}
\text{Category} & \\
\text{n(N)} & \\
n(N)/n(N) & \\
n(N) \setminus n(N) / np(X, obj) & \\
np(N,C) & \\
np(N,X) / n(N) & \\
s & \\
s \setminus np(N, subj) / np(X, obj) & \\
s \setminus np(N, subj) / np(Y, obj) / np(Z, obj) & \\
s \setminus np(N, subj) & \\
s \setminus np(X, Y) \setminus (s \setminus np(X, Y)) & \\
s \setminus np(X, Y) / (s \setminus np(X, Y)) & \\
s \setminus np(X, Y) \setminus (s \setminus np(X, Y)) / np(Z, subj) & \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>Parts of Speech</th>
<th>(177)</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>noun</td>
<td>n(N)</td>
<td></td>
</tr>
<tr>
<td>adjectives</td>
<td>n(N)/n(N)</td>
<td></td>
</tr>
<tr>
<td>nominal prepositions</td>
<td>n(N) \setminus n(N) / np(X, obj)</td>
<td></td>
</tr>
<tr>
<td>noun phrase</td>
<td>np(N,C)</td>
<td></td>
</tr>
<tr>
<td>determiner</td>
<td>np(N,X) / n(N)</td>
<td></td>
</tr>
<tr>
<td>sentences</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>transitive verbs</td>
<td>s \setminus np(N, subj) / np(X, obj)</td>
<td></td>
</tr>
<tr>
<td>ditransitive verbs</td>
<td>s \setminus np(N, subj) / np(Y, obj) / np(Z, obj)</td>
<td></td>
</tr>
<tr>
<td>verb phrases</td>
<td>s \setminus np(N, subj)</td>
<td></td>
</tr>
<tr>
<td>backward adverbs</td>
<td>s \setminus np(X, Y) \setminus (s \setminus np(X, Y))</td>
<td></td>
</tr>
<tr>
<td>forward adverbs</td>
<td>s \setminus np(X, Y) / (s \setminus np(X, Y))</td>
<td></td>
</tr>
<tr>
<td>verbal prepositions</td>
<td>s \setminus np(X, Y) \setminus (s \setminus np(X, Y)) / np(Z, subj)</td>
<td></td>
</tr>
</tbody>
</table>

110
where the variable \( N \) is the number of a category and \( C \) is the case agreement value.

We then take the lexical assignment

\[
\begin{align*}
(178) & \quad \text{Cats} & \quad \text{Words} \\
& n(X) & \text{sheep, fish} \\
n(sing) & \text{kid, penguin} \\
n(plu) & \text{kids, penguins} \\
n(N)/n(N) & \text{tall, cool} \\
n(N)\setminus n(N)/np(X, obj) & \text{in, under}
\end{align*}
\]

for categories producing nouns after getting their complements,

\[
\begin{align*}
(179) & \quad \text{Cats} & \quad \text{Words} \\
n(sing, X) & \text{it} \\
n(sing, subj) & \text{he, she} \\
n(sing, obj) & \text{him, her} \\
n(plu, X) & \text{you} \\
n(plu, subj) & \text{they} \\
n(plu, obj) & \text{them} \\
n(N, X)/n(N) & \text{the} \\
n(sing, X)/n(sing) & \text{every, a(n)} \\
n(plu, X)/n(plu) & \text{most, three}
\end{align*}
\]

for categories producing noun phrases, and finally, for the verbal system, we take

\[
\begin{align*}
(180) & \quad \text{Cats} & \quad \text{Words} \\
& s\setminus np(X, subj) & \text{sneezed, ate} \\
& s\setminus np(sing, subj) & \text{sneezes, eats} \\
& s\setminus np(plu, subj) & \text{sneeze, eat} \\
& s\setminus np(N, subj)/np(M, obj) & \text{loved, cheated} \\
& s\setminus np(sing, subj)/np(M, obj) & \text{loves, cheats} \\
& s\setminus np(plu, subj)/np(M, obj) & \text{love, cheat} \\
& s\setminus np(N, subj)/np(Y, obj)/np(Z, obj) & \text{gave, loaned} \\
& s\setminus np(sing, subj)/np(Y, obj)/np(Z, obj) & \text{gives, loans} \\
& s\setminus np(plu, subj)/np(Y, obj)/np(Z, obj) & \text{give, loan} \\
& s\setminus np(X, Y)\setminus(s\setminus np(X, Y)) & \text{slowly, yesterday} \\
& s\setminus np(X, Y)/(s\setminus np(X, Y)) & \text{slowly, probably} \\
& s\setminus np(X, Y)\setminus(s\setminus np(X, Y))/np(Z, subj) & \text{in, beside}
\end{align*}
\]

The following parse trees are then admissible with respect to this grammar

111
(181) three kids eat
\[ np(plu, V1) / n(plu) \]
\[ n(plu) \]
\[ s \n \{ np(plu, subj) \} \]
\[ np(plu, V0) \]

(182) he saw him
\[ np(sing, subj) \]
\[ s \n \{ np(V1, subj) / np(V2, obj) \} \]
\[ np(sing, obj) \]
\[ s \n \{ np(V0, subj) \} \]

(183) give him
\[ s \n \{ np(plu, subj) / np(V1, obj) / np(V2, obj) \} \]
\[ np(sing, obj) \]
\[ s \n \{ np(plu, subj) / np(V0, obj) \} \]

(184) give him them
\[ s \n \{ np(plu, subj) / np(V0, obj) \} \]
\[ np(plu, obj) \]
\[ s \n \{ np(plu, subj) \} \]

(185) kids by him
\[ n(plu) \]
\[ n(V3) \n \{ n(V3) / np(V4, obj) \} \]
\[ np(sing, obj) \]
\[ n(V2) \n \{ n(V2) \} \]
\[ n(plu) \]
Chapter 3

Semantic Models

We begin our discussion of semantics with a brief survey of the philosophical tradition behind the models which we introduce in the remainder of the chapter. We begin our model construction with the domain of individuals, which we assume to contain all of the usual individuals. We also take the basic collection of individuals to include the groups of individuals and also the event-like objects we use to model the semantics of verbs. Similarly, we will also include the temporal moments and intervals in our domain of individuals. We argue that these domains are similar enough to warrant treatment as a unified whole. After developing models of the domain of individuals, we turn our attention to the study of propositions. We are primarily concerned with the intensionality of propositions and the algebraic structure they have under the conjunction, disjunction and negation operations. Our models of relations are then simply models of the typed $\lambda$-calculus over the basic domains of propositions and individuals. Such models contain their basic domains and all possible higher-order functions constructible from the basic domains. This leads to a simple characterisation of higher-order logical operations as pointwise extensions of the propositional operations.

3.1 Philosophical Groundwork

We uphold a longstanding tradition by beginning our discussion of semantics with a look at some philosophical puzzles concerning the relation of language to the world. While this may seem like a lofty peak from which to begin, many seemingly
esoteric philosophical issues lie at the heart of the semantic endeavour to uncover the nature of linguistic meaning.

3.1.1 Uniformities

All but the most primitive animals depend for their survival on their sensitivity to different patterns in their environment. They distinguish individuals according to their properties and are attuned to various relationships between individuals. Our use of individual will be the literal one, where an individual is simply something which can be individuated from other things.

Consider a hunting animal such as a hawk. The hawk must be sensitive to wind conditions in order to fly. Of course, the hawk need not be conscious of this activity, but it is correcting for the wind nonetheless. At the same time, the hawk must be able to distinguish things on the ground which are possible prey from those that are not. The hawk's life is dependent on its ability to effectively carry out such a search. A hawk that attacked lions would not get very far. It is also obvious that different animals classify the world in different ways. What is dangerous for a mouse is not necessarily dangerous for a cat. The visual system of a frog is sensitive to different stimuli than those of a fly. The frog's visual system is set up to detect insects that it can eat, while a fly will detect even the smallest movements from almost any angle so that it can escape being eaten.

The classification of animals in terms of the uniformities in their environment that they perceive and respond to was begun by Gibson (1979) and the psychological school of ecological realism. Of course, people are also attuned to uniformities in their environment. Not only do people recognise concrete individuals such as prey and attackers, footballs, pencils, computers, books, telephone poles and koala bears, but they also recognise abstract individuals like particular symphonies, concerts, meetings, governments, thoughts, emotions and so on.

Not only do we recognise particular individuals in our environment, but we also recognise properties of individuals such as being a symphony, being boring, being long, and so on, as well as complex relations that hold between them such as being an inspiration of, loving, realising, believing and so on. We think of properties as
relations which only take one argument. In general, each relation will determine how many arguments it needs. For instance, the relation of loving basically holds between two people, while the giving relation involves three arguments.

Just as different animals respond to different uniformities in their environments, people in different cultures, almost by definition, divide their world up along different dimensions. This was first brought up by Whorf (1956), who pointed out that the Hopi view of space and time was relativistic, while the so-called Standard Average European view was Newtonian. In the case of the Hopi, Whorf found that this difference in world view was reflected in the Hopi language. Studying natural language semantics may thus lead to insight into the world view of a particular linguistic community.

It is a defining property of a linguistic community that language be used uniformly and consistently within it. If this were not so, communication would be impossible. For instance, when two different speakers of English use the term dog, it can be assumed that they share a similar notion of what it is to be a dog, which we might call the property of doghood. We will only be concerned with the fact that any linguistic community shares common assumptions as to how the world is to be divided into individuals and relations. Of course, we do not wish to imply that every member of a linguistic community has exactly the same knowledge of the world, for this is simply not the case. In fact, the main purpose of language is for conveying information from one person to another, which would not be possible if everyone always had exactly the same knowledge. When we say that the individuals and relations are held commonly in a linguistic community, we mean that each language user knows the conditions under which the relations would hold as well as the conditions under which individuals can be picked out and discriminated. This, of course does not imply that the language user has complete knowledge about which relations hold of which individuals or even which individuals exist. Assuming that there is shared knowledge of relations and individuation conditions alleviates the problem of having to model the world views of each person separately and then trying to find common ground between world views. We will simply assume that the common presuppositions of a given language are fixed and understood. Where this is not the case, the use of language
breaks down, and as we have said before, we will, in general, avoid all performance related issues such as we believe this to be. In most formalisms, this assumption of a common ground is implicitly built into the theory by not allowing for variation across language users to be represented.

We will not discuss the effects of discourse on the world views of its participants, since this does not affect the meanings of expressions in the language, but simply the non-linguistic knowledge of some of its speakers. We will also not be concerned with the way in which a language evolves over time, except in terms of the generation of new lexical entries from existing ones, which is discussed by Dowty (1979) and Carpenter (1987).

With a fixed stock of shared intuitions concerning individuals and relations, a potentially infinite collection of new relations can be constructed by recombining the existing individuals and relations in various ways. For instance, from the properties of being a penguin and living on an iceberg, we can form the property of being a penguin living on an iceberg. From the relation of loving and the individual Margaret Thatcher, we can form the property of loving Margaret Thatcher. From the property of walking and chewing gum, we can form the property of walking while chewing gum and so on. This can be iterated indefinitely, as we can see from properties such as being a mouse who was chased by a cat while looking for some cheese because it was hungry.

Combining a possibly complex relation with the appropriate number of arguments, produces what we will call a proposition. We assume that propositions are bivalent, so that every proposition must either be true or false. This assumption of bivalence implies that every relation either holds or does not hold of any appropriate sequence of arguments. Furthermore our assumption that the members of a linguistic community share a common understanding of relations leads us to conclude that every member of the linguistic community is able to determine when a relation holds of some arguments, in the sense of being able to give verification conditions for that relation. For instance, the relation of there being more titanium in the United States than in the USSR, while not immediately verifiable by most members of the English speaking community, is verifiable in the sense that an English speaker knows what the world must be like at an abstract level.
for the relation to hold. Making this assumption leads us into trouble in the case of so-called vague relations such as being bald or being next to. There is no set distance at which two objects are said to be next to one another, even if some common scale of comparison has been fixed. There is no rigidly fixed number of hairs that a person must have to be considered hirsute rather than bald. A common approach to the model theoretic representation of vagueness is in terms of the supervaluations of van Fraassen (1969) and Fine (1975). In the supervaluation model, each vague relation is modeled by the collection of its possible extensions to a bivalent relation. Unfortunately, this presupposes the ability to determine which extensions are possible and thus begs the question it sets out to answer.

Just as we can form new relations out of old relations, we can form new propositions out of old propositions in a number of ways. Suppose we have two propositions p and q. Then we can form the negation of p, which is a proposition that is true if and only if p is false. Similarly, we can form the disjunction of p and q which will be true just in case either p or q or both are true, and their conjunction, which is true just in case they are both true. We will come back to the analysis of propositions and just what it is that determines when propositions are true in Section 3.3.

3.1.2 Realistic Semantics

In the introduction, we claimed that we would be taking a realist view of natural language semantics. This means that we are committing ourselves to the realism of the objects corresponding to the model theoretic objects that we employ to model the meanings of expressions. Of course, this will not commit us to any particular kind of semantic theory, but will simply force us to take a particular ontological stand once the models have been fixed.

The most obvious way in which a realist theory can fail is by incorporating objects which do not correspond to anything in the real world. Of course, a semantic theory which did not have realist pretensions could not be criticised along these lines. This is the case for so-called possible worlds semantics, where a set of merely possible worlds is introduced, each of which determines the truth
of every proposition in that world. Now only one of these worlds is intended to correspond to the real world, so that the others can be viewed as unreal, except possibly as viewed by Lewis (1986). We will come back to problems associated with possible worlds in Section 3.3.1.

Another way in which a realistic semantics could fail is as a theory of semantics. This can happen most easily if two expressions with different meanings are conflated or if two terms with the same meanings are not identified. The classic example from Frege (1892) is that of the Babylonians and their beliefs about the morning star and the evening star, which both happen to be the planet Venus seen at different times of the day. Consider the sentences

(1)  
   a. The morning star is visible.  
   b. The evening star is visible.

Now one of these sentences may be uttered truthfully just in case the other one can. But consider what happens when these sentences serve as the complement to a verb like believes

(2)  
   a. Socrates believes the morning star is visible.  
   b. Socrates believes the evening star is visible.

These sentences have independent truth conditions — where the truth of one may not be determined from the truth of the other. From this, we conclude that there must be a difference in meaning between the expressions morning star and evening star, since this is the only difference between these sentences. Of course, we can not simply suppose that there are two different individuals, one of which is the morning star and one of which is the evening star, since this is simply not the case. Supposing that there were two different individuals might solve the semantic puzzle, but would be a violation of realism. Of course, it is always possible for utterances of expressions with different meanings to receive the same interpretations, even in different contexts. For instance, when I utter the expressions the current president and Ronald Reagan in California during February 1988, they might both contribute the same thing to the interpretation of the utterance in which they occurred.
Note that we have not identified meaning with intension and interpretation with extension. The intension of a term such as unicorn or men is meant to be the abstract property we understand by the noun. The extension is just the set of objects with the property. Similarly, the intension of a proposition can be viewed as carrying content while the extension of a proposition is a simple truth value. The idea to explicitly split the study of model theoretic semantics into a study of extensions and intensions is due to Quine (1961). This topic will be taken up in Section 3.3.

There is no reason to suppose that for something to be real it must also be a material or physical object. It might seem that we could reduce all of our talk about abstract objects such as debts, pains, beliefs and jobs to some set of ontologically primitive objects such as sub-atomic particles out of which everything else is composed.

There are two arguments against such a reductionist strategy. The first is that when push comes to shove, even simple things like my favourite tree in the courtyard, which is about as physical as any other object in the world, can not be identified with their physical composition at any one time, since the physical composition of the tree is changing from moment to moment. The tree does not even have a physical constancy at coarse grained levels since it would be the same tree even if it had all of its branches chopped off and it grew new ones. There simply seems to be no way to describe my favourite tree in terms of its molecular make up, or any group of physical properties that it has. It could be moved, tipped over, burned, genetically altered and so on, all while remaining the same tree. If we think of the tree as being purely physical we miss out on some of its essential properties. For instance, I have beliefs about the tree which are not reducible to beliefs about its sub-atomic structure. For further arguments along similar lines see Boyd (1980).

The second argument is due to Hilary Putnam (1970,1975) and hinges on the explanatory necessity of objects posited at different levels of abstraction. His simplest example involves explaining why a square peg will not fit in a round hole. On the reductionist view of pegs and holes, we are dealing with a simple collection of small particles which is no more than the sum of its parts. Reasoning on a sub-
atomic level, the reason the square peg does not fit is that its component particles are not aligned properly. While this massive calculation on the position and momentum of particles is not impossible in principle, carrying it out would still not explain the generalisation regarding square pegs and round holes which seems to hold in the world. Instead, if we are willing to admit the properties of squareness, roundness, size and rigidity we can provide a perfectly adequate explanation which can not be reduced to atom-talk even in principle. The lower level "prediction" is simply not explanatory in that the next situation with a square peg and a round hole requires a complete re-calculation down to the particle level. The reason such higher levels of abstraction are necessary for explaining human behaviour is that humans are resource-bounded agents, who could not, even in principle, carry out the reductionist strategy of dealing with their environment on a particle by particle basis.

Another argument for the reality of seemingly abstract objects comes from their contribution to our mental states. Uniformities that people are attuned to become real in the sense that the belief in the uniformity can be seen to guide their behaviour. Certain stimuli can cause an organism to enter a particular mental state. In this way, mental states reflect the properties of the real world as well as being part of the real world, and should thus be accepted as real objects. Also, it seems that the best way in which to describe the action of the mind is in terms of mental states which interact with one another and cause other mental states and bodily functions to occur. This view of mental states has come to be known as functionalism due to its emphasis on function rather than form in distinguishing mental states (see Block (1980) for an introduction).

3.1.3 Models and Theories

There are really two angles from which to approach the semantic enterprise. Either way, some domain of study must be delineated. The first approach is to develop a theory of the objects in the domain and the relationships that hold among them. A theory is usually constructed by specifying a primitive collection of statements, called axioms. Axioms specify how some of the objects under study are related.
Then some rules of inference are provided, which allow us to infer that additional statements, called theorems, follow from our axioms. Theories cast in this format are often referred to as Hilbert systems, after their inventor. To be useful, a theory must provide theorems which are useful in the sense of making accurate predictions about the domain. A theory in which every derivable theorem is actually true of the domain is said to be sound. If every true statement about the domain can be proved from the axioms using the rules of inference the theory is said to be complete. The theoretician is obviously in search of sound and complete theories. If restrictions are put on the inference rules and axioms, say that they are recursively enumerable, then it is not always possible, even in theory, to provide sound and complete theories of a given domain, as was proved by Gödel in his famous Incompleteness Theorem (1931).

An alternative to constructing a theory of a particular domain is to construct a model of it. For something to be considered as a model, it must be taken to be a model of something. Mathematical models were introduced by Tarski (1944) to develop a conception of truth for formal languages. Models can be useful in understanding the domain that is being modeled only if there is some correspondence between the model and the object(s) being modeled. For an example due to Barwise (1984b), consider the utility of modeling airplanes in wind tunnels. It is much more economical, in terms of both effort and expense, to construct a model of an airplane than it would be to construct a full-size version. Testing the model in the wind tunnel tells the aeronautical engineer useful things about the way the plane would behave if it were to be built. A theory of aerodynamics, on the other hand, would most likely be cast in terms of a set of initial or boundary conditions and some system of partial differential equations from which air flows at future times can be calculated, and the behaviour of the plane could be determined.

We will be concerned with building mathematical models of meanings in natural language, since it is usually easier to build a model than provide a theory. The usual reason for this is that if we use well-understood mechanisms to construct the models, we will be able to determine their effects quite readily. For instance, if we assume that the collection of propositions forms a boolean algebra, we will automatically be able to derive a number of conclusions concerning the behaviour
of the models of propositions. Of course, this brings out the possibility of side-effect, which might be artifacts of our particular representation. For an example due to Link (1987), consider the standard encoding of the natural numbers we gave in the introduction (page 17), where every natural number corresponds to the set of natural numbers less than it. This has side-effects such as \(2 \subseteq 8\) and \(123 \subseteq 144\), which are sometimes exploited in other definitions. It would be a mistake to suppose that this encoding commits us to these facts about natural numbers, so care must be taken to separate the models from the objects being modeled, even if both are abstract. In this regard, theories are more perspicuous, as a theory may relate to a number of different domains. For instance, a theory of aerodynamics stated in terms of differential equations might be applied to more general fluid dynamics problems. The major drawback of theories is that they require some link with the world if they are to be useful. Models provide just that link.

Working with models often highlights questions of ontological precedence. In the possible-worlds semantics of Montague (1974a,1974d), it is assumed that the sets of individuals, times and possible worlds are basic elements of the model, in the sense of not having any internal structure. From these basic elements, Montague constructed models of propositions as the sets of possible worlds in which they are true and modeled properties as functions from possible worlds to sets of individuals. Of course, this does not commit Montague to the conclusion that individuals and possible worlds are somehow more basic than propositions or properties or that propositions are sets of possible worlds. These are just features of the model.

One of the distinguishing features of the so-called algebraic approach to semantics (see Link (1983,1987)), which we have more or less adopted, is its relative lack of concern with the difficult issues of ontological precedence. We take it as given that there is some collection of relations, individuals, propositions and so on, and take it as our job to spell out various relationships that hold among them.

Algebraic semantics derives its name from the fact that it employs various notions from modern algebra, where domains are modeled as algebras, with their structural properties being determined by the operations of the algebra. Of course,
there is also a theory involved in the study of any algebra. Consider semilattices for a moment. A set and operation is only a semilattice if the operation is idempotent, commutative and associative. We can think of these conditions as being restrictions that a set and operation must meet if they are to form a semilattice. On the other hand, we can treat these restrictions as axioms, and together with some simple logical axioms and rules of inference, we can derive a number of theorems predicting the behaviour of semilattices in general.

3.2 Basic Domains

We begin our formal development of a model theoretic semantics by considering the nature of the basic domains of particulars individuated by English speakers.

3.2.1 Individuals

Perhaps the most natural place to begin a discussion of the various objects presupposed by English is with the interpretations of various utterances of simple non-quantificational noun phrases such as john, the cake, the chairman, my arm, john's thoughts, the destruction of the city, john's destroying the city and so on.

In English and many other languages, there is a fundamental distinction drawn between so-called count and non-count or mass nouns which is reflected in their meanings and syntactic distribution. Consider the common nouns in

\[
\begin{array}{ccc}
\text{Type} & \text{Agreement} & \text{Examples} \\
\text{mass} & \text{singular} & \text{water, furniture, anger} \\
\text{mass} & \text{plural} & \text{entrails, goods} \\
\text{count-singular} & \text{singular} & \text{man, sheep, song, injury} \\
\text{count-singular} & \text{plural} & \text{scissors, pants} \\
\text{count-plural} & \text{plural} & \text{men, sheep, songs, injuries} \\
\end{array}
\]

In terms of their meaning, mass nouns can be distinguished from count nouns by the way in which they are viewed by language users. Count nouns are used to pick out individuals which can be counted, whereas non-count nouns pick out individuals which are being viewed as an undifferentiated mass. The distinction between singular and plural nouns is then based on how many individuals
are being considered. For instance, we have noun phrases like *three books* and *one newspaper*, but it does not make sense to put a numerical determiner like *three* or *one* with mass nouns such as *milk* or *despair*, when used with their usual mass interpretations.

The agreement of a noun will indicate which form of verb must be used with it. Consider the contrast between the well-formed and ill-formed expressions in

(4) a. The \( \{ \text{water, man} \} \) \{ * are messy \}.

b. The \( \{ \text{entrails, men, scissors} \} \) \{ are messy \}.

We are treating agreement here as a purely syntactic phenomenon, but it should be noted that agreement also has a well-defined pragmatic role (see Section 5.2).

Count nouns usually have separate singular and plural forms such as *man* and *men* and *song* and *songs*, but there are count nouns such as *sheep* and *fish*, when used to refer to the whole animals, rather than the food, which have the same singular and plural forms. Most non-count nouns can be used to pick out kinds of objects, so that a non-count noun like *wine* could also be used as a count-noun synonymous with *kind of wine*, with the plural *wines* meaning the same thing as *kinds of wine*. Similarly, most count nouns can also be used as mass nouns, which can be seen in food terms such as *elephant*, which is normally a count noun, but may also be used to refer to the undifferentiated stuff produced from grinding up elephants. There are also a class of so-called *syntactic plurals*, such as *scissors* and *pants*, which are used to pick out single objects, but occur with plural verbs. Similarly, while most mass nouns occur with singular verbs, some mass nouns such as *entrails* and *goods* take plural agreement.

The classification of nouns, and hence individuals, on the basis of whether they can be counted is something that varies from language to language. For example, the English mass nouns *spaghetti* and *information* are translated in Italian as count nouns (see Quirk et al. (1985)).
Basic Individuals

As we hinted at above, we simply assume a fixed set $\text{IND}$ of basic individuals. The actual members of $\text{IND}$ will be a matter for an empirical investigation which we will not pretend to have carried out. Since the set $\text{IND}$ is supposed to contain actual individuals in the world, it will contain me, my mother, this thesis, the act of writing this thesis, the fact that I wrote it and the piece of paper that it is printed on. On the other hand, it will not contain any unicorns or square circles.

It is also important to keep in mind that the elements of $\text{IND}$ are not taken to be by-products of the use of any particular language, but rather objects which language helps to differentiate in the real world. This is how speakers of different languages can be assumed to be talking about the same things in the world. For instance, a Frenchman and Scot may be standing on the Lawnmarket in Edinburgh looking at the castle, which the Frenchman may refer to as \textit{le château} and the Scot may refer to as \textit{the castle}. This does not mean that they are talking about different objects, since it is taken for granted that there is really only one castle visible from the Lawnmarket.

This realism at the object level does not preclude us from assuming that \textit{le château} and \textit{the castle} have different meanings. This is how we can reconcile the fact that one language would use a mass term where a second language would use a count noun to describe the same object in the real world. The nouns in the two languages have different meanings but they pick out the same objects in the real world.

Consider the conditions under which the following sentences can be truthfully uttered

\begin{enumerate}
\item Mary leaped over the hurdles.
\item My mother called last night.
\item Mary sat an exam.
\end{enumerate}

In uttering these expressions, the noun phrases can be interpreted referentially. A referential use of a noun phrase presupposes the existence of an individual in the world that has properties fixed by the utterance. A referential utterance of the
first sentence can only be true if there is someone named Mary who has in fact leaped over some real hurdles, be they abstract or real. The speaker must have a mother for the second sentence to be used truthfully and finally, both Mary and an exam must exist if the final sentence is to be used correctly. The existence of an individual will be modeled by membership in IND in our model.

Now consider the non-referring definite noun phrases, such as Santa Claus, James Bond and the planet closer to the sun than Mercury. Members of the English community know what it would mean for there to be a Santa Claus and for James Bond to exist and for there to be a planet closer to the sun than Mercury. This is part of our assumption that members of a linguistic community know the meanings of expressions in their language. But consider the sentences

\[(6)\]  
a. Santa Claus has a beard and rides in a sleigh.  
b. James Bond is licensed to kill.

It is not clear whether these sentences could ever be uttered truthfully if there are no individuals named Santa Claus or James Bond. In the models we develop, these sentences could only be uttered truthfully if there were some individuals in IND named Santa Claus and James Bond. In Montague’s possible worlds models, such “possible individuals” are taken to be individuals in the model domain. The reason that we can not take such an approach here is that this would violate our philosophical intuitions with respect to realism, since presumably there are not actually such individuals, and consequently there is nothing for the model-theoretic individuals to represent in the real world.

By these comments, we do not wish to imply that referentiality is a property of a noun phrase in isolation, but rather that the interpretation it is given when uttered will determine whether it is referential or not. Consider the case of

\[(7)\]  
a. A man ran.  
b. Mary believes a man ran.

Here, an utterance of the first sentence could only be true if there were a man and that man ran, while the second sentence, in which the entire first sentence is embedded, could be uttered truthfully regardless of the existence of any man. We
come back to the analysis of singular noun phrase interpretations in Section 4.5.3 and Section 5.1.7.

Distributions

Consider the minimal pair of sentences

(8) a. The lecture was in the seminar room.
    b. The lectures were in the seminar room.

While the first sentence could only be used truthfully if there was in fact a lecture in the seminar room, the second of the pair could be used truthfully if there is some collection of at least two lectures, each of which was in the seminar room. This interpretation of a plural noun phrase is said to be distributive, since the property of being in the seminar room is something that is required to hold individually of a whole collection of objects, that is, the property distributes over elements of the collection. Such distributive uses of plurals have the so-called cumulative reference property, as first noticed by Quine (1960). For example, this means that if we know that the syntax lectures are in the seminar room, and the semantics lectures are in the seminar room, then the semantics and syntax lectures are in the seminar room. Similarly, if the lecturers were tired and the students were tired, then the lecturers and students were tired.

As is pointed out by Landman (1987), this behaviour can be accurately modeled with simple sets of individuals in IND with more than one element (note that this sort of modeling dates back to the calculus of individuals of Leonard and Goodman (1940)). The set of all such sets of individuals with more than one individual can be given by

(9) \( \text{DIST}(\text{IND}) = \{ S \mid S \subseteq \text{IND}, \|S\| \geq 2 \} \).

We call elements of \( \text{DIST}(\text{IND}) \) distributions of individuals. We note that (8)b can be used truthfully if there is some element of \( \text{DIST}(\text{IND}) \), each member of which is a lecture that was in the seminar room. In general, we will suppose that a property will hold of a distribution just in case it holds of each member of the distribution. It should now be clear why the cumulative reference property

127
is satisfied. For instance, if the students is used with the interpretation \( d_1 \) and the lecturers is used with the interpretation \( d_2 \), where \( d_1, d_2 \in \text{DIST(IND)} \), then the students and lecturers can be used with the interpretation \( d_1 \cup d_2 \). Now a property will hold of \( d_1 \cup d_2 \) just in case it holds of all of its members, which will be true if and only if the property holds of \( d_1 \) and of \( d_2 \).

In our model of distributions, the converse of the cumulative reference property, called the homogeneity condition, will also hold, so that the students and lecturers will be in the room if and only if the students are in the room and the lecturers are in the room.

Groups

Now consider the sentence

(10) The students carried a computer downstairs.

One possible interpretation for this sentence is the distributive one, where there must be some member of \( \text{DIST(IND)} \), whose members are all students who individually carried a computer downstairs. But this is not the only possible reading of this sentence. There is also the so-called collective interpretation of the students. Under the collective interpretation, (10) could be uttered truthfully if there was some group of students who acted collectively to carry the computer downstairs, possibly because the computer was too heavy for any individual student to lift.

Consider the meanings of the two sentences

(11) a. The frog was separated by the biologist.

b. The frogs were separated by the biologist.

(11)a could only be uttered truthfully if the biologist had separated a single frog into more than one piece. On the distributive reading of its subject the frogs, (11)b could only be uttered truthfully if there were some collection of frogs each member of which was individually separated into more than one piece. On the collective interpretation of the frogs, (11)b could also be used truthfully if there happened to be a group of twenty frogs which is later broken down into smaller groups.
The properties and relations that a group stand in can not, in general, be determined solely by reference to the individuals that make up the group. For instance, consider the sentences in (12) (due to Landman (1987)), which have very different meanings

(12)  

a. The cards below seven and the cards from seven up have been separated.

b. The cards below eight and the cards from eight up have been separated.

It is possible to use the noun phrases the cards below seven and the cards from seven up and the cards below eight and the cards from eight up to pick out two different groupings of the same pack of cards. But, both of these groupings are made up of exactly the same individuals, namely the 52 cards in the pack. This means that any treatment of groups in terms of the individuals that make up the group will require some additional mechanism to distinguish between two groups with the same members (see Landman (1987) for just such an analysis).

In our semantic models, we treat groups just like any other individuals that language users pick out, namely as primitive members of IND. We treat groups this way because, unlike distributions, the properties of groups can not be determined from the properties of their members. This move also allows us to define relations solely in terms of the propositions they form when they take arguments in IND. The proposition formed by applying a property to a distribution will be determined solely on the basis of the propositions formed by applying the property to each of the members of the distribution.

We will assume that there is a composition relation, say \( \downarrow \), which will be assumed to hold between a group \( g \in \text{IND} \) and an individual \( x \subseteq \text{IND} \), which we write \( g \downarrow x \), if the group \( g \) is has \( x \) as a constituent. So, if \( d \) is a distribution, we will have \( g \downarrow d \) just in case every member of the distribution \( d \) is a constituent of the group \( g \). With this composition relation in hand, we will simply allow any noun phrase which could pick out a distribution \( d \in \text{DIST(IND)} \) to pick out some group \( g \in \text{IND} \) such that \( g \downarrow d \). The switch to the group reading is always possible, and is not explicitly marked in the syntax of English, on either nouns or
the verbs. Taking this approach, we will produce mass interpretations for noun phrases in sentences such as *three boys breathed*, which will produce sentence interpretations which are false, since presumably breathing is not something that can be performed by a group.

With our collapsing of groups into IND, nothing precludes the existence of a group which is itself composed of other arbitrarily complex groups. Hoeksema (1981) introduced this sort of nested group model in his analysis of non-boolean nominal conjunctions. Similarly, we do not encounter cardinality problems, since we are not requiring every possible set of individuals to make up some group, in the same way that we do not require every definite noun phrase to pick out some real individual. This is different from the approaches of both Landman (1987) and Link (1983, 1984, forthcoming), where arbitrary collections of individuals always form an actual group in the model. While it is always possible for a group to be composed out of any members of IND, arbitrary subsets of IND will not necessarily make up even one group. This means that \( \downarrow \) will not necessarily relate a group to every subset of individuals, nor relate every group to a different subset of individuals.

Landman (1987) has pointed out that it is more likely to find group readings of plural terms with additional pragmatic or semantic content. His example concerns the Talking Heads, which is a pop group consisting of the members Chris, Jerry, Tina and David. With this knowledge, we could give utterances of the sentences in (13) the same interpretation

(13) a. Chris, Jerry, Tina and David put out a new album.

    b. The Talking Heads put out a new album.

The knowledge that Chris, Jerry, Tina and David constitute the Talking Heads allows us to interpret the subjects of (13)a and (13)b as the same group. Of course, we could always interpret (13)a distributively as making the claim that each of the four performers put out a new album.

A further interesting class of phenomena are the so-called *cumulative* interpretations of plurals (the term is due to Scha (1984)). Consider the examples
modeled on examples of Scha (1984). The claim is that a sentence such as (14)a could be uttered truthfully if there were a number of groups, each gathering in different parts of the country, for which the cardinality of the union of all of their members totaled half a million. Similarly, (14)b might be used truthfully if there were 600 Dutch firms and 2000 American computers, with some owning relation holding among them, possibly with small groups of Dutch firms, each owning groups of computers, with the total cardinalities working out appropriately. Finally, it might be argued that it is possible to interpret an utterance of (14)c as implying that each of the men lifted exactly one of the four tables (Link 1987).

Similarly, consider another of Scha's examples

(15) The sides of this square parallel the sides of that square.

This sentence could not be uttered truthfully on the distributive readings of the sides of this square and the sides of that square, since that would imply that all of the sides of the first square parallel all of the sides of that square, which could never be the case.

If these interpretations are possible for utterances of the sentences in (14) and (15), then it remains to be seen whether or not these readings could be captured by interpreting the noun phrases in question as groups. This is the basic strategy of Link (1987), and the one we will follow here. If the group interpretations turn out to be inadequate, it should be relatively simple to add additional cumulative interpretations, although this should be a last resort, as the additional readings would significantly increase the already large number of interpretations possible for any plural noun phrase.

Masses

Mass nouns share some of the properties of singular nouns and some of the properties of plural nouns. For instance, a mass noun phrase will almost always take the
same agreement as a singular noun phrase, but mass nouns can be used without
determiners in the same way as plural nouns. For example, consider

\[(16) \quad \begin{align*}
    a. & \quad * \text{Cook is in the kitchen.} \\
    b. & \quad \text{Cooks are in the kitchen.} \\
    c. & \quad \text{Water is in the kettle.}
\end{align*} \]

Similarly, there are determiners such as *much* which only occur with mass nouns
and *many* which only occur with count nouns.

Just as we did with groups, we will suppose that masses are another sort of
ting that can be elements of IND. Masses will have some interesting properties,
though. First of all, they are such that if we simply take the material join of two
masses, it will again be a mass of the same sort. For example, if we take two
masses of water, and combine them, the result is also a mass of water. This is
like the cumulative reference property for distributive interpretations of plurals
(see Quiné (1960)). In fact, we can take this as a defining condition of when a
noun will take mass agreement. So *water* normally takes mass agreement, while
*five gallons of water* takes count agreement.

It is Parson's (1970) proposal, following Quine (1960), to include an operation
that maps individuals onto the mass out of which they are composed in the same
way that a group is mapped onto its members. But, as Bach (1986) points out, it is
simply not the case that such a well-defined operation exists. His example involves
a snowman, which can be viewed as composed of a number of different objects. On
the one hand, we can view a snowman as being composed of snow. In this case, we
would want our composition function to map the individual snowman onto a mass
of snow. But, we could also look at the snow as being composed of water, and
by transitivity of composition, conclude that the snowman was also composed of
water. We could then break the water down into masses of $\text{H}_2\text{O}$ molecules, along
with some impurities, and so on right down to the atomic level. By putting the
masses together in certain ways, they take on additional properties. The snow
has properties that the water does not share, and similarly for the snowman.

We will not follow this strategy, but will instead simply suppose that our
composition relation $\Downarrow$ is also defined for masses, so that $x \Downarrow y$ will also hold if
x is a mass of stuff which has a mass y among the stuff that makes it up. Thus we can treat the composition of groups in terms of their individual members in the same way as we deal with masses in terms of their compositions. Nothing significant in our model hinges on the use of one relation and a second relation could be introduced to deal with the composition relations between mass terms.

Of course, we do not want to assume that homogeneity holds for mass terms due to the minimal parts problem with mass terms such as fruitcake (the term is due to Quine (1960)). A particular object may be a fruitcake without all of its material parts also being fruitcakes, since a single raisin will be a material part of a fruitcake, but will not be a fruitcake itself. Similarly, some material parts of snow, such as hydrogen atoms will not themselves be snow.

3.2.2 Eventualities

The term eventuality, due to Bach (1981), is meant to encompass objects of many different sorts, including static states of affairs and more dynamic events and processes. There are many reasons for the inclusion of a collection of particulars corresponding to these objects in our models. In this section, we will consider a number of these, along with motivation for a relatively coarse-grained approach to the individuation of events.

Nominalisation

The primary motivation for treating eventualities as particulars comes from various nominalisation phenomena, where verb-like expressions are used as nouns. For instance, consider the sentences

(17) a. John predictably gave Mary an award yesterday.
    
    b. John's giving of an award to Mary yesterday was predictable.
    
    c. The giving of the award by John to Mary yesterday was predictable.
All of the sentences in (17) can be used to get across the same basic point, namely that it was predictable that John gave Mary an award yesterday. The subjects of (17)b and (17)c are in some sense derived from the verb gave as used in (17)a. In general, there is a gerund such as giving corresponding to every verb such as give which, in English, always has the same phonological realisation as the present participle form of the verb. We suppose that definite noun phrases, like the subjects of (17)b and (17)c, require that some object exist for sentences in which they occur to be used truthfully, as we saw in the case of (5) and (6). The most natural candidate for such an object in the case of (17)b and (17)c seems to be some kind of event. We will suppose that these sentences can be uttered truthfully if and only if there is some eventuality which is a giving event, had the participants as specified by the prepositional phrases, and was predictable.

Consider the sentences (after Parsons (1985))

(18) a. Every burning consumes oxygen.
   b. John burned some newspapers.
   c. Oxygen was consumed.

An utterance of (18)a can be used to make a claim that every eventuality which is a burning consumes oxygen. If this is true, then if someone were to use (18)b truthfully, we would be led to infer that (18)c could also be used truthfully. The reason for this is that if John burned some newspapers, then there had to be a burning event. In this way, verbs can be seen to introduce eventualities of the same sort as those introduced by their gerundive counterparts.

A slightly different example, which also involves the use of nominals to make claims about events, would be the utterance of the three sentences

(19) John did not finish his homework. That was not very wise. It caused him to fail his class.

The demonstrative that and pronoun it are linked to the eventuality introduced by the utterance of the verb finish in the first sentence. In order to preserve the uniformity of the treatment of nominals, it seems necessary to introduce a domain of events, whose elements are taken to be real in the same sense as the elements
Not surprisingly, we will simply suppose that there is a distinguished subdomain \( \text{EVENT} \subseteq \text{IND} \) of the set of individuals corresponding to the domain of events. One reason we make it a subdomain of the domain of individuals is that by nominalisation, any event that can be introduced by a sentence can also be introduced by its nominalised counterpart.

Causation, Levels, and Excuses

Explicit theories of causation more often than not make reference to some conception of eventuality. This should be evident from a quick glance through either the philosophical or artificial intelligence literature dealing with action, causality and reasoning about change. For instance, it is often said that it was the event of Brutus stabbing Caesar that caused the event of Caesar's death, or that the short circuit caused the fire (both examples due to Davidson (1967b)). It seems that if sense is to be made of causality at all, it must be closely tied to a theory of eventualities.

Examples such as the following were introduced by Goldman (1970)

\[(20) \quad \text{a. John stopped traffic by waving his arm.} \]
\[(20) \quad \text{b. Brutus killed Caesar by stabbing him.} \]

Goldman takes it to be the case that there is an asymmetrical, not necessarily transitive, relation between eventualities which he calls *level-generation*. That is, if John stopped traffic by waving his arm, it will not be the case that he waved his arm by stopping traffic. This is closely related to causality, which is often assumed to hold between various eventualities. An analogy could be made between the behaviour of the stopping event and the waving event and the snowman and the snow out of which it is composed. More precisely, the stopping event is composed out of the same physical actions as the arm waving event, but two events are individuated. This is similar to the case of the snowman. The snowman is made up of snow, but the individual corresponding to the snowman and the individual corresponding to the snow are not identified. They can not be identified because they have very different properties. Just as the snowman could be new and the
snow could be old, the waving could be intentional and the traffic stopping could be unintentional. An approach based on this model is adopted by Link (1987).

This is not the only feasible approach to the individuation of eventualities. Instead, we will follow Davidson in identifying the traffic stopping and arm waving eventualities. We assume that the clauses in (20)a contribute different event descriptions to the meaning of the sentence. We can then model the intentions of an agent as relations to properties of events. We assume that what is intentional is bringing about an event which is an arm waving versus bringing about an event which is a traffic stopping. Our approach is based on the so-called situation types found in situation semantics (Barwise and Perry 1983, 1985, Barwise 1984, Barwise and Etchemendy 1987). A situation type in situation theory corresponds to what we will call a property of events. We will discuss this issue in more detail after we have developed a general theory of properties in Section 3.4 and when we introduce semantics for basic lexical entries in Section 4.3.

Another domain in which events and types of events play an important role is in the generation of excuses. As Davidson (1967) points out, excuses can only be made sense of if we think of them as providing different descriptions of the same event. More precisely, if I was accused of killing the bank president, and I admit that I shot the escaping murderer, my excuse rests on the fact that I did not know that the events were actually identical.

This is our first clue for the treatment of the so-called propositional attitudes. Rather than treat propositional attitudes as relations to propositions, as in the Montague (1974d) tradition, we will suppose that beliefs are relations between individuals and properties or descriptions of events. Consider truth of the sentences

(21)  a. John believes he killed the bank president.

b. John believes he shot the escaping criminal.

Suppose we were to model beliefs, and consequently utterances about beliefs, as being about events directly. Now suppose that the bank president was the very same person as the escaping criminal who was shot and killed by John. Then John could not believe that he shot the escaping criminal without believing that he killed the bank president. Another alternative would, of course, be to assume
that these events were not in fact the same.

Individuating Eventualities

With the introduction of eventualities, the natural ontological question arises as to how eventualities should be individuated. It seems fairly uncontroversial to assume that the sentences in (22) (from Davidson (1967)) could all be used to pick out the same event

(22) a. John buttered the toast.
    b. John buttered the toast at midnight.
    c. John buttered the toast in the bathroom.
    d. John buttered the toast in the bathroom at midnight.

It is examples such as the following (also due to Davidson (1967)), which play a crucial role in deciding the matter

(23) a. Mary swam the channel quickly.
    b. Mary crossed the channel slowly.

Opinion seems to be divided on the issue of whether utterances of the sentences in (23) can be taken to pick out the same event in the same way as the nominals pick out the same individual. We favour the original approach of Davidson's (1967), where eventualities are given a coarse-grained structure, and the difference between (23)a and (23)b is dealt with in the same manner as nominal modification in noun phrases such as

(24) a. the small animal
    b. the big ant

It seems that exactly the same thing is going on in (24) and (23). The two noun phrases in (24) obviously have different meanings, but it is almost universally assumed that they can be used to pick out the same object. It then seems rather odd to suppose that the verb phrases in (23) could not be used to pick out the same event. When we come to treat properties in Section 3.4, we will deal with
eventualities introduced by verbs on a par with individuals introduced by nominals. This is even more appealing in light of adnominal modification of gerundive nominals in noun phrases such as

(25)  

a. the slow crossing  
b. the fast swimming

Fine-grained theories of eventualities will usually associate eventualities with objects resembling propositions. This strategy can be found in Cresswell (1985), Dowty (1988), Link (1987), Bach (1986) and Moore (1988). Under these fine-grained theories, verbs with different meanings will introduce distinct eventualities. This means that utterances of the two sentences

(26)  

a. John bought the book from Bill.  
b. Bill sold the book to John.

will introduce two necessarily distinct eventualities, one of which is a buying event, and the other a selling event. Whether situation semantics falls into this group will depend on the nature of the involvement relation (see Section 3.4 below).

The coarse-grained approach to eventuality individuation has usually been founded on the spatio-temporal properties of eventualities. The so-called Lemmon criterion is widely used (see Lemmon (1967) and Bennett (1985)). The individuation criterion is that two eventualities that occupy the same space and time are identical. Aside from problems of determining the exact location of an event in space and time (see Thomson (1971)), this seems to be far too coarse-grained. We leave open the possibility that two distinct events could occupy exactly the same space and time. What is truly important is the collection of properties of the eventualities. If there is a property that holds of one eventuality and not of the other, then we must conclude that there are two objects. It does seem, though, that approaches to individuation of events based on some pre-defined set of attributes, such as participants and type, predicate and agent, and so on seem doomed to fail, just as it seems doomed to fail for other individuals. The fact of the matter seems to be that events, like other individuals, can be individuated on any grounds that are deemed relevant.
There is also the question of whether or not events should be viewed as universals rather than particulars. This comes from the notion of repeatability, where it might be thought that an event can occur again and again. We take this view to be fundamentally misguided in the same way that assuming more normal individuals like my computer are universals because it plays a role in a number of different events. It is not the computer that is a universal, but the properties that it has. We take a similar strategy for events. There can be many events which are of the same type. That is, there can be any number of thinking, burning or killing events, but the same thinking event can not be repeated again and again.

**Aspectual Classes of Events**

So far, we have not mentioned any of the structure of the event domain itself. In fact, all the structure we need for events is already present in the domain of individuals.

Recall that there is a general mechanism by which distributions of individuals can be reified and treated as a new individual with its own properties. The same is true for events. For instance, consider the sentence (due to Thomason and Stalnaker (1973))

(27) John skillfully caught the ball and made the throw.

There are at least two interpretations an utterance of (27) could receive. Under either interpretation, there will be two events — one being a catching event and the other being a throwing event. The difference comes in what John did skillfully. Under the first interpretation, John could have skillfully caught the ball and skillfully made the throw, but the sentence could also be used to imply that it was the execution of the combination of the two events which was in fact skillful. This is exactly like the distinction between the collective and distributive reading of noun phrases. To account for these distinctions, we will again simply suppose that an expression which can be used to pick out a distribution of events can be used to pick out a group event which is composed of the elements of the distribution.

The event domain also inherits from the larger domain of individuals the notion of countability. Events which are viewed as having definite culminations are said
to be *telic*, while events which are not viewed in this way are called *atelic*. The sentences

(28)  
   a. Opus likes Binkley.  
   b. Opus walked.  
   c. Opus was in Bloom County.  
   d. John slept last night.

would normally be used to describe atelic events, while those in

(29)  
   a. Bill died.  
   b. Bill arose.  
   c. Oliver built a computer.  
   d. Oliver wrote a computer program.  
   e. Cutter arrived in Bloom County.

would usually be used to describe telic events. The reason for the qualification in the above statement is that sentences which are normally used to describe telic events can be used to describe atelic events and vice-versa with added contextual information in much the same way as mass nouns can be used in a count context and conversely, as we saw above. The possibilities for such aspectual shift in the event domain are often induced by adverbial modification or inflectional morphology, and are studied in Dowty (1979), Moens (1987) and Moens and Steedman (1987).

Another point worth mentioning is that descriptions of atelic events share the same cumulative reference properties as mass and plural count nouns. That is, a group event composed of two separate running events is still a running event. What we have with a group event consisting of two building of a house events is two buildings of a house rather than one.

There is a standard puzzle called the *imperfective paradox* stemming from the pairs of sentences
(30)  a. John was building a house.
     b. John built a house.

and

(31)  a. John was running.
     b. John ran.

From the fact that John was building a house, we can not conclude that he
built a house, since he may have been interrupted and never completed it. But,
if we know that John was running then we can conclude that John ran. This
difference is usually accounted for by the aspectual difference between the events
that could be described by utterances of the sentences in (31) and (30). This sort
of aspectual analysis was introduced in this form by Vendler (1967). The building
of a house is a telic event, while a running event would normally be atelic, while
a running to the store would in turn be telic, since it would have a definite end
point. See Vlach (1981), Parsons (1985) and Dowty (1979) for more details on
the imperfective paradox. Bach (1986) has pointed out that the same behaviour
is found in the nominal domain in the case of partitive constructions such as

(32)  a. This is part of a symphony.
     b. We found part of a Roman aqueduct.

The puzzle here is that the first of these sentences could be uttered truthfully
even if there was no completed symphony which the part could be drawn from,
and similarly for the Roman aqueduct. On the other hand, consider

(33)  a. This is a piece of cake.
     b. These are some of the papers.

With the mass noun cake and plural noun papers, we can conclude that if these
sentences are used truthfully, then there is some cake and there is some collection
of papers. A solution to one of these problems would carry over to a solution to
the other on our account.
These and other parallels between mass and count nominals and telic and atelic events have been studied by many authors (see, for example, Allen (1966), Bach (1986), Link (1987) or Mourelatos (1978). We feel that there is sufficient evidence for treating events as simply a special sort of individual. We will return to our particular analyses of the semantics of the constructions we have introduced in this section in Section 4.3.

3.2.3 Temporal Intervals

While the spatial and temporal location of an event may not be enough to identify it uniquely, the time at which an eventuality takes place is overtly marked on verbs in the form of a tense morpheme. There is a significant distinction to be drawn between verbs which are tensed and the so-called tenseless verbs. A tensed verb will place restrictions on the time of occurrence of the event which an utterance of the sentence could describe.

Moments

In dealing with the semantics of time, the issue arises as to whether times should be taken as basic or constructed out of events. Many researchers construct the domain of times out of the domain of events (see Kamp (1979) and Barwise and Perry (1983)). While there may be compelling reasons to view the event domain as ontologically prior to the domain of times, we will not adopt this strategy here. The main reason for this is that neither ontology nor the analysis of time are among our major concerns here.

Instead, we will simply assume that there is some set MOMENT of moments of time which is linearly ordered by the temporal precedence relation \( \preceq \), where we write \( m_1 \preceq m_2 \) if moment \( m_1 \) is before or identical to moment \( m_2 \). As usual, we will write \( m_1 < m_2 \) if \( m_1 \preceq m_2 \) and \( m_1 \neq m_2 \), and say that \( m_1 \) is strictly before \( m_2 \).

There are a number of additional restrictions we could place on the moments other than the requirement that they be linearly ordered. For a start, we could suppose that there is no beginning point or no ending point to the set of moments,
which would lead to the following restrictions on the domain of moments

(34) for every $x$, there is some $y$ such that $x < y$

and

(35) for every $x$, there is some $y$ such that $x \succ y$.

This would be the same as supposing that there is no least upper bound or greatest lower bound to the set of all moments.

We could also suppose that the moments are dense so that

(36) $x < y < z$ for some $y$ if $x < z$.

The opposite of density, discreteness assumes that for every $x$ there is some $y$ such that

(37) $x < y$ and if $x \leq y \leq z$ then $z = y$ or $z = x$.

We might also employ a continuity requirement, so that every sequence of moments had a limit. This could be enforced by requiring that the ordering on the times is such that every bounded set of moments has a least upper bound and greatest lower bound, or more precisely

(38) $\bigwedge M'$ exists if there is some $t \leq M'$

and

(39) $\bigvee M'$ exists if there is some $t \geq M'$.

It will not be necessary, for our purposes here, to choose among these axioms. We would like our semantic models to reflect the conceptions of the world held by language users, and this seems to be a matter open for empirical investigation, so we will simply assume the domain of moments is some linearly ordered set. For a more in-depth look at temporal logics, see Prior (1967) or Rescher and Urquhart (1971).
Intervals

With our moments in hand, we can construct a domain \( \text{TIME} \) of so-called times by taking closed intervals of moments, as in

\[
\text{TIME} = \{ [m_1, m_2] \mid m_1, m_2 \in \text{MOMENT}, \ m_1 \leq m_2 \}.
\]

We can then derive a complete temporal precedence relation on the set \( \text{TIME} \) of times by setting

\[
t_1 \preceq t_2 \text{ if } m_1 \preceq m_2 \text{ for every } m_1 \in t_1, \ m_2 \in t_2
\]

The other temporal relation we will be interested in is that of temporal overlap, which is defined by

\[
t_1 \circ t_2 \text{ if } m \in t_1, \ m \in t_2 \text{ for some } m \in \text{MOMENT}.
\]

If \( t_1 \circ t_2 \) then we say that \( t_1 \) overlaps \( t_2 \).

Finally, we will suppose, following Link (1987), that there is a so-called temporal trace function \( \tau : \text{EVENT} \to \text{TIME} \) from the eventualities into the times such that \( \tau(e) \) is the time at which the eventuality \( e \) takes place. Again, while the location of an event might in fact be vague (see Thomson (1971)), just as the location of a person, we will assume for the sake of simplicity that it is well-defined.

The use of intervals for the locations of events is by now well established (see, for instance Dowty (1979), Kamp (1979), McDermott (1982), Allen (1984) and Kowalski and Sergot (1986). One of the primary motivations for adopting intervals is the observation that both telic and atelic events take place during intervals rather than at moments. For instance, during sufficiently short intervals of a running event, the runner may be pausing to rest before continuing, while the running is still described by language users as taking place over the whole interval. Similarly, some telic events will occupy some non-empty stretch of time before culminating, while others are viewed as pure culminations (see Parsons (1985) and Moens and Steedman (1987).

We note that the interval-based approach is in fact more general than the moment-based approach, since \( \text{MOMENT} \) can be embedded in \( \text{TIME} \) by the obvious order isomorphism mapping \( t \) onto \([t, t]\). In general, we allow events to have
temporal traces corresponding to such singleton intervals. It is not clear if there are any such durationless eventualities, although eventualities like crossing a finish line or reaching the top of a mountain are obvious candidates. Whether or not an eventuality is viewed as having a definite endpoint will affect its aspectual classification (see Vendler (1967), Dowty (1979), Moens (1987) and Moens and Steedman (1987).

We will also assume that $\text{TIME} \subseteq \text{IND}$ so that every interval is also a basic individual. We again take this step because basic common nouns such as $\text{time}$, $\text{day}$, $\text{hour}$ and so on can be used to pick out temporal intervals.

Following Link (1987), we further require group events to have temporal traces which correspond to the closure of the union of their individual traces. That is, if $e \in \text{EVENT}$ is a group event, we require

$$\tau(e) = \bigvee \{\tau(d) \mid e \downarrow d\}$$

where we can partially order intervals by set inclusion, thus forming a complete, but non-modular lattice out of the domain of times.

An alternative to the approach we have sketched here would be to allow times to be arbitrary sets of moments rather than connected closed intervals, so that the domain of times would form an atomic boolean algebra with atoms corresponding to the singleton moments. But, such a move does not seem warranted by natural language temporal semantics, so we assume the more restrictive model of times as connected closed intervals.

We can extend our precedence and overlap relations to events in the obvious way, by setting

$$e \preceq e' \text{ if } \tau(e) \preceq \tau(e')$$

and

$$e \circ e' \text{ if } \tau(e) \circ \tau(e').$$

In fact, we will generally allow events to show up where intervals would be expected, with the understanding that they are simply standing in for their temporal traces.
To define the basic moments and intervals in terms of events, it is necessary to start with the \( \preceq \) and \( \circ \) relations defined on the domain of events, and then construct the moments out of maximal sets of pairwise overlapping events, and times in the same way as we do (see Kamp (1979) for details).

Finally, we will also follow Link (1987) in assuming that the duration of a time is measurable in the strict sense. That is, for each unit of time \( \alpha \), such as seconds, minutes, hours and so on, there will be a so-called additive measure function \( | \cdot |_\alpha \), such that

\[
(46) \quad | [m_1, m_3] |_\alpha = | [m_1, m_2] |_\alpha + | [m_2, m_3] |_\alpha
\]

A consequence of this definition is that for any time unit \( \alpha \) it must be the case that \( | [m, m] |_\alpha = 0 \). We need to introduce such a function to provide an absolute measure for the semantics of such expressions as

\[
(47) \quad \begin{align*}
\text{a. John has been president for three years.} \\
\text{b. Mary finished the race in under two minutes.}
\end{align*}
\]

3.3 Propositions

As we hinted at in our discussion of realism in Section 3.1.2, we think of propositions as the result of supplying a relation with an appropriate number of arguments. Since we are assuming that the domain of propositions is basic, we put off introducing our models of relations themselves until the next section. In this section, we simply suppose that relations are the kinds of things that come with arguments and once the arguments are supplied, a proposition is determined.

3.3.1 Modality and Possible Worlds

Probably the most well-known approach to the semantics of propositions is the so-called possible worlds approach introduced by Montague (1974a, 1974b, 1974c). In possible worlds semantics, a primitive set of so-called possible worlds is assumed. The idea behind possible worlds is that they are taken to be alternative possible ways the world could be or could have been. A proposition is then modeled as
a function from possible worlds to truth values or equivalently as sets of possible worlds at which the proposition is true. This idea stemmed from Carnap’s (1956) idea to treat a proposition as a function from models to truth values in those models and Kaplan’s (1964) subsequent development of Carnap’s ideas. A proposition is true at or in a possible world if and only if the possible world is a member of the proposition. Underlying this approach is the assumption that a possible world is simply something that determines the truth of every proposition in that world.

Possible worlds semantics was introduced by Kripke (1959, 1963) to model the logic of modality. The intention was to model the possibility and necessity of certain propositions about the world. This is done by positing a basic accessibility relation between worlds which is meant to pick out which worlds are possible relative to others. A proposition is then said to be necessary in a world if it is true in all worlds accessible from that world and is possible if it is true in some world accessible from that world. By singling out one possible world as the actual world, propositions making modal claims about possibility and necessity in the actual world can be modeled. The power and flexibility of Kripke models comes from the ability to place different sorts of restrictions on the accessibility relation (see Hughes and Cresswell (1968)). For instance, in Montague’s semantics, the accessibility relation was simply such that every world was accessible from every other.

There are a number of drawbacks to the possible worlds approach to modeling propositions. The first of these has to do with realism. It is far from obvious what exactly is being modeled by possible worlds themselves. Since the actual world determines the truth value of every proposition about the real world, the merely possible worlds are inherently unreal (but consider again the radical position of Lewis (1986) concerning the reality of possible worlds).

A stronger motivation for rejecting the possible worlds model of the semantics of propositions is that every proposition which is true in exactly the same worlds is equated, since the only individuation condition on propositions is the set of worlds at which they are true. In particular, propositions which are necessarily true are all equated, as are propositions which are necessarily false. For instance,
consider any two logical truths or theorems of mathematics. Presumably, these will be true in exactly the same possible worlds, namely all of them. This means that all of the logically valid propositions turn out to be not just necessarily true but identical. Of course, by introducing so-called impossible worlds (see Cresswell (1973)), where two logically equivalent propositions could be given different truth values, then the propositions could be distinguished, but this would take us even farther from our demands of realism. Another alternative would be to assume that propositions concerning necessary truths can somehow be given additional structure (see Stalnaker (1984) for an argument along these lines).

Individuals in a possible worlds model will usually, following the advice of Scott (1970b), be taken to be possible individuals, which have an existence at some possible world. The alternative would be to have a different domain of individuals in every world and some counterpart relation between them (see Quine (1960,1961) and Lewis (1968)).

We should also point out that it would be possible to retain a possible worlds treatment of modality, while maintaining a distinction between propositions and sets of possible worlds (see Thomason (1980) for just such a model).

### 3.3.2 Partial Information and Situations

An interesting view of situation theory has been recently put forward by Perry (1986) in response to Stalnaker's (1984,1986) attempts to salvage the possible worlds model of propositions. Stalnaker maintains that possible worlds are adequate to model propositions, but that mathematical truths are really truths about expressions in mathematical notation. This introduces a number of problems related to those involved in treating propositions as syntactic constructs (see Thomason (1977) for arguments and references). Perry has come to think of the situations of situation semantics as partial possible (and impossible) worlds. Since situation theory is supposed to model information, and most information is inherently partial, it is a small step to embrace partial possible ways that things could be. These possible ways can be modeled as partial functions from propositions to truth values. In fact, in many models of the theory, no total ways will exist
(see Barwise (1985)). In this way, a situation or way is simply a partial possible world. Muskens (1987) has worked out an elegant relational model of type theory in which possible worlds can be made partial in just this way, avoiding the mathematical obstacles associated with making the standard functional type theory partial.

The situation semantics literature has always vacillated between two views of situations. The first is the one sketched above, where situations are treated as partial possible worlds, while the second treats situations on an ontological par with our eventualities. In Barwise and Perry (1983,1985) and Cooper (1986), situations are taken to be real chunks of the world which play a role in causal relations, conditional information and so on.

A very important feature of situation semantics, and probably the one that has received the least formal attention, is the generalised notion of a constraint on the way the world is structured. Constraints in situation theory are modeled by so-called involvement relations. In general, involvement holds between types of eventualities (or situation types in the situation semantics terminology), rather than between eventualities themselves. That is, an involvement relation will be of the form

\[ T_1 \Rightarrow T_2 \]

which is assumed to hold if any situation of type \( T_1 \) involves the existence of a situation of type \( T_2 \). Their favourite example of involvement is that every kissing situation involves a touching situation. What this means is that the existence of a situation which happens to be a kissing of \( x \) by \( y \) implies the existence of a situation which is a touching of \( x \) by \( y \). We will ignore the parameters \( x \) and \( y \) and their behaviour, even though it is necessary to insure that the same individuals \( x \) and \( y \) participate in the touching that participate in the kissing, since the theory of parameters is even less fleshed out than the theory of involvement.

From this general notion of involvement, Barwise and Perry go on to develop a general theory of meaning in which it is possible to describe things like natural laws, such as the presence of smoke indicating the presence of fire, where it is assumed that if there is a situation in which there is smoke, there must be another
related situation in which there is also fire. Similarly, involvement is used to explain linguistic conventions, such as the fact that the word *cookie* is used to pick out cookies. This will require the introduction of a number of different kinds of involvement relations, such as logical involvement, nomic involvement, conventional involvement, and so on (see Barwise (1984) and Barwise and Perry (1983)). Consider now the case of logical involvement, which is notated $\Rightarrow$. This relation is meant to model logical consequence, and is defined so that $T_1 \Rightarrow T_2$ holds only if every situation of type $T_1$ is also a situation of type $T_2$. With this addition to situation semantics, and it seems to be a necessary one from the point of view of situation theory, the collection of facts supported by situations can be taken to be closed under logical consequences. This simply re-introduces the problems faced by the possible worlds approach, since logically equivalent facts will be supported by exactly the same situations. So, even the move to situation types will be of no use since types which hold of exactly the same situations are identical in the theory. Similarly, if situation semantics is to avoid the trap of requiring there to be a distinct buying and selling events for every item that is sold, we need to assume that an involvement relation holds between buying and selling events which behaves like a logical involvement in the sense that every situation which is a buying is also a selling. While this is arguably necessary from the point of view of situations as eventualities, it is undesirable as a theory of situations as providing partial information about the world, since the facts in a situation will be closed under logical consequence among other things. States of information available to people are not closed in this way (see Landman (1984)).

### 3.3.3 The Algebra of Propositions

In keeping with our general approach to modeling natural language semantics, we will simply take the domain of propositions as given and investigate the minimal algebraic structure that it displays. We will call this basic domain of propositions $PROP$ and call the elements of it propositions. This approach to modeling propositions is based on the intentional logic of Thomason (1980) and the data semantics of Veltman (1984) and Landman (1986b), while retaining many of the useful prop-

The first thing to notice about the domain of propositions is that it is closed under the so-called logical operations of conjunction, disjunction and negation. We use the following notation for these logical operations on propositions

(i) conjunction \( p \land q \)
(ii) disjunction \( p \lor q \)
(iii) negation \( \sim p \)

We think of \( p \land q \) as being the proposition which corresponds to the proposition that \( p \) and \( q \) hold, with \( p \lor q \) being the proposition that \( p \) or \( q \) hold, and \( \sim p \) being the proposition that \( p \) does not hold. This means that the domain of propositions can be taken as an algebra defined over the signature containing these operations, as in

(50) \( \text{PROP} = (\text{PROP}, \land, \lor, \sim) \).

In this definition, we have abused notation in a method familiar from algebra, conflating an algebra \( \text{PROP} \), which is formally a tuple, with its carrier domain \( \text{PROP} \), which is a set. We will continue to refer to algebras by the same name as their carriers, as the convenience outweighs the possible confusion.

The second thing to notice is that propositions are the sorts of things which are either true or false. The domain of truth values contains two elements, namely true and false, which we represent as 1 and 0, respectively. We make the standard assumption that the truth values form a boolean algebra with 1 as the top element and 0 as the bottom element. More specifically, this is the unique two element boolean algebra

(51) \( \text{TRUTHVAL} = (\text{TRUTHVAL}, \land, \lor, \sim, 0, 1) \)

with \( \text{TRUTHVAL} = \{0, 1\} \). We think of the boolean operations of meet, join and complement as conjunction, disjunction and negation in the truth value domain. This leads to the following behaviour of the operations
We then suppose that there is some operation $U : PROP \rightarrow TRUTHVAL$ which maps propositions onto their truth values. We require that $U$ be a homomorphism from the algebra $(PROP, \&, \cup, \neg)$ onto the algebra $(TRUTHVAL, \&, \vee, \neg)$. This is the minimum restriction we need so that for all propositions $p, q \in PROP$ we have

\begin{align*}
(53) \quad & i. \ U(p \& q) = (U p) \& (U q) \\
& ii. \ U(p \cup q) = (U p) \vee (U q) \\
& iii. \ U(\neg p) = \neg(U p)
\end{align*}

As it stands, we have an even weaker domain of propositions than that of Thomason’s (1980) intentional logic. For instance, there is no guarantee that

\begin{align*}
(54) \quad & p \cup q = \neg(\neg p \& \neg q).
\end{align*}

For a start, we will require that $\&, \cup$ and $U$ be semilattice operations on the domain of propositions. From this, we are assured that for all propositions $p, q, r \in PROP$, we have

\begin{align*}
(55) \quad & i. \ p \& p = p \cup p = p \quad \text{(idempotency)} \\
& ii. \ p \& q = q \& p \quad \text{(commutativity)} \\
& \quad \quad p \cup q = q \cup p \\
& iii. \ (p \& q) \& r = p \& (q \& r) \quad \text{(associativity)} \\
& \quad \quad (p \cup q) \cup r = p \cup (q \cup r)
\end{align*}

Thomason does not actually require that his operations of disjunction and conjunction meet these restrictions, although whether this was intentional, or merely an oversight is not clear, as these restrictions are not incompatible with his purposes.

Now that we have some semilattices, we can investigate the orderings that these put on the set of propositions. In the standard way, we put
We think of $p \geq q$ and $p \supseteq q$ as being two different ways in which $p$ can contain the information that $q$. The first way is by saying that the information that either $p$ or $q$ is no more general than the information that $q$. The second way in which we can think of $p$ containing the information that $q$ is by being such that adding the information in $q$ to that in $p$ leaves nothing more than $p$. Since we have not supposed that the set of propositions together with conjunction and disjunction form a lattice there is really no connection between conjunction and disjunction, so that it would be possible to have $p \geq q$ without $p \supseteq q$ and vice-versa.

We will assume that the operations $\cap$ and $\cup$ are complete. This means that we can define disjunction and conjunction over arbitrary sets of propositions. If $Q \subseteq \text{PROP}$, we will follow standard notation and write $\cap Q$ and $\cup Q$ for the conjunction and disjunction of all of the propositions in $Q$, respectively. This will also entail that $\langle \text{PROP}, \geq \rangle$ and $\langle \text{PROP}, \supseteq \rangle$ form bounded lattices with least and greatest elements, but again, these lattices are not necessarily identical or even isomorphic.

To remedy this last complication, we will assume that $\langle \text{PROP}, \cap, \cup \rangle$ is a lattice, so that we also get the absorption identities

$$
\text{(57) i. } p \cap (p \cup q) = p \quad (\cap \text{ absorption}) \\
\text{ii. } p \cup (p \cap q) = p. \quad (\cup \text{ absorption})
$$

for all $p, q \in \text{PROP}$. This will entail that $\geq$ and $\supseteq$ are the same relation, which we will simply write $\geq$. Note that $\geq$ is not an operation, but rather a relation over the domain of propositions. We think of $\geq$ as our implication relation.

Following data semantics (Landman 1986b), we will say that a proposition $p$ is possible just in case there is no proposition $q \in \text{PROP}$ such that

$$
\text{(58) } p \geq (q \cap \sim q)
$$

and otherwise say that $p$ is impossible. Two propositions $p, q \in \text{PROP}$ are then said to be incompatible if their conjunction, $p \cap q$, is impossible.
For the purpose of the task at hand, this is really all the structure on the domain of propositions that we will require. For the rest of this section, we will outline some further restrictions that might be placed on the domain of propositions and their consequences.

Another natural restriction on models of propositions would be distributivity, so that for \( p, q, r \in \text{PROP} \) we would have

\[
\begin{align*}
\text{i. } & p \land (q \lor r) = (p \land q) \lor (p \land r) \quad (\land \text{ distribution}) \\
\text{ii. } & p \lor (q \land r) = (p \lor q) \land (p \lor r) \quad (\lor \text{ distribution})
\end{align*}
\]

Along the same lines, we might wish to enforce DeMorgan's laws, so that for all propositions \( p, q \in \text{PROP} \) we would have

\[
\begin{align*}
\text{i. } & \sim(p \land q) = (\sim p) \lor (\sim q) \quad (\land \text{ DeMorgan}) \\
\text{ii. } & \sim(p \lor q) = (\sim p) \land (\sim q). \quad (\lor \text{ DeMorgan})
\end{align*}
\]

We could also assume that the law of double negation held, so that for every proposition \( p \in \text{PROP} \) we had

\[
(\sim \sim p) = p \quad (\text{double negation})
\]

With double negation and DeMorgan's laws, we would be assured that

\[
(\sim(p \land q)) = (\sim p) \lor (\sim q)
\]

which is a basic restriction Thomason explicitly puts on his models, by defining \( \lor \) in exactly this way.

Similarly, we could define an implication connective \( \supset \), where we take \( p \supset q \) to be the proposition that the truth of \( p \) implies the truth of \( q \). If we simply take the usual definition of implication in terms of negation and disjunction, with

\[
p \supset q = (\sim p) \lor q
\]

then we will be assured that \( \lor(p \supset q) \) is true if and only if \( \lor(p) > \lor(q) \), where implication for truth values is given, in the usual fashion, by

\[
p > q = (\sim p) \lor q.
\]
A lattice with an additional negation operator which is distributive and satisfies both DeMorgan's laws and double negation is called a DeMorgan lattice. An extended DeMorgan lattice, as defined by Landman (1986b), is a DeMorgan lattice with distinguished sets of propositions \( \bot \) and \( \top \) which are called contradictions and tautologies with the two restrictions

\[
(65) \quad \text{i. } (q \cup \sim q) \geq (p \cap \sim p)
\]

\[
\text{ii. } f \geq p \text{ if } f \geq (p \cap t), \quad t \in \top \text{ and } f \in \bot.
\]

The second condition says that if a proposition conjoined with a tautology provides the information that some contradiction holds, then the proposition must have already contained the information that the contradiction holds. This will only be satisfied if the set of contradictions and the set of tautologies are disjoint. If this is so, there will also be no fixed point \( p \in \text{PROP} \) of the negation operator such that \( p = \sim p \). We note that \( \bot \) is an ideal and \( \top \) is a filter in \( \text{PROP} \). Furthermore, for every proposition \( p \in \text{PROP} \), we have

\[
(66) \quad \text{i. } (p \cap \sim p) \in \bot, \quad \text{and}
\]

\[
\text{ii. } (p \cup \sim p) \in \top, \quad \text{and}
\]

\[
\text{iii. } (\sim p) \geq p \text{ if } p \in \bot
\]

The latter asserts that a contradiction \( p \) contains the information that its negation holds.

If we were to require that both \( \bot \) and \( \top \) were singletons, we would have a boolean algebra with \( \sim \) as the complement operation, so in a sense, an extended DeMorgan lattice is just slightly more general than a boolean algebra. The obvious reason for the added generality is to allow for the fact that two contradictions might be distinguishable propositions, and the same goes for tautologies. The boolean semantics of Keenan and Faltz (1985) is a theory in which the domain of propositions is simply taken to be a boolean algebra. In possible worlds semantics, the domain of propositions forms a complete atomic boolean algebra with the atoms being the singleton sets of worlds.

In data semantics (Landman 1984) it is also assumed that there is a set \( \text{FACT} \) of facts which independently generates the set of propositions and meets the following two conditions for every finite subset \( B \subseteq \text{FACT} \) and \( p, q \in \text{PROP} \)
\[(67) \quad \text{i. } (p \geq \bigcap B) \text{ or } (q \geq \bigcap B) \text{ if and only if } ((p \cup q) \geq \bigcap B) \]

\[(67) \quad \text{ii. } \bigcap (B \cup \{p\}) \text{ is possible for some if for some } q \text{ with } (\sim q \geq p), \]

\[\text{if } \bigcap (B \cup \{\sim q\}) \text{ is possible.} \]

The first condition requires a proposition to contain the information that a disjunctive proposition supports the conjunction of a set of facts if and only if one of the disjuncts supports the conjunction. The second condition requires a fact to be compatible with every set of facts compatible with some negative proposition. This insures that the negations are grounded in the facts.

Another requirement on the domain \(\text{PROP}\) of propositions in data semantics (Landman 1984) is that it forms a so-called completion lattice which is an extended DeMorgan lattice satisfying the property that for all propositions \(p, q \in \text{PROP}\)

\[(68) \quad p \cap q \text{ is impossible if } p \text{ is possible and } q \text{ is possible, but } p \cap r \]

\[\text{and } q \cap r \text{ are impossible for all } r \in \text{PROP}.\]

Completion lattices which are independently generated by a set of facts are called constructive completion lattices. The domain of propositions in data semantics is taken to form a constructive completion lattice.

The purpose of data semantics is to model information and information growth. An information state is defined to be a filter generated by some pairwise compatible set of propositions. An information state is said to be total if every proposition or its negation is in the state. Information states meet the important consistency condition which means that if some proposition \(p \in \text{PROP}\) is not in an information state then that information state is a subset of an information state which contains either \(p\) or \(\sim p\). Note that boolean algebras meet the stronger independence condition, where any information state not containing some \(p \in \text{PROP}\) can be extended to an information state containing \(\sim p\).

### 3.4 Propositional Functions

In this section we introduce our models of relations as functions into the domain of propositions.
3.4.1 Typed Functions

We will carry out our semantic model building in a restricted subset of a standard model of the typed lambda calculus over our basic domains. Recall that we have introduced the following domains of objects in this chapter, listed with the type symbols we will use for them

\[
\begin{align*}
\tau & \quad D\tau & \quad \text{Members} \\
 i & \quad \text{IND} & \quad \text{unrestricted individuals} \\
d_i & \quad \text{DIST(IND)} & \quad \text{distributions of individuals} \\
e & \quad \text{EVENT} & \quad \text{eventualities} \\
d_e & \quad \text{DIST(EVENT)} & \quad \text{distributions of events} \\
t & \quad \text{TIME} & \quad \text{temporal intervals} \\
p & \quad \text{PROP} & \quad \text{propositions} \\
b & \quad \text{TRUTH VAL} & \quad \text{truth values}
\end{align*}
\]

Our set of basic types is then

\[
\Theta = \{i, d_i, e, d_e, t, p, b\}
\]

which determines a unique minimal set

\[
\text{TYPE}(\Theta) = \Theta \cup \{\langle \alpha, \beta \rangle \mid \alpha, \beta \in \text{TYPE}(\Theta)\}
\]

of types. We build up higher order domains as functions defined over more primitive domains \(D_\tau\) for \(\tau \in \Theta\) in a strictly hierarchical fashion, so that the objects in a functional domain \(\langle \alpha, \beta \rangle\) are given by

\[
D_\langle \alpha, \beta \rangle = D_\beta^{D_\alpha}
\]

This means that an object of type \(\langle \alpha, \beta \rangle\) will be a function from objects of type \(\alpha\) to objects of type \(\beta\).

3.4.2 Relations and Propositional Types

In practice, we will not use all of the functional types that these frames provide. We will only employ the set of propositional types, which is defined to be the smallest set such that
(73)  i. \( p \) is a propositional type, and
  
  ii. \( \langle \alpha, \beta \rangle \) is a propositional type if \( \beta \) is a propositional type and \( \alpha \) is either a basic type in \( \Theta \setminus \{ b, d_1, d_2 \} \) or a propositional type.

Each propositional type is either a proposition or a function defined on propositional types or basic types that produces a propositional type when applied to an argument. We will call an element in the domain \( D_\tau \) a \textit{propositional function} when \( \tau \) is a propositional type. Note that we have not allowed distributions to occur in our propositional types, since we will define the result of applying a function to a distribution to be the conjunction of the application of the function to all of the elements of the distribution.

We will fix a set \( C_\tau \) of constants and \( V_\tau \) of variables for each propositional and basic type \( \tau \). This yields a set of well-formed \( \lambda \)-terms \( T_\tau \) for each propositional and basic type \( \tau \) in the usual way.

Our model will then be fixed when we fix a valuation function \( \nu \) mapping the constants in \( C_\tau \) into their values in \( D_\tau \). We will simply assume for the time being that this has been done, but in fact, a large part of the sequel will be devoted to introducing constants and determining what their values should be.

Suppose that we have some relation \( R \) which takes \( n \) arguments. We will require that a relation specify the type of each of its argument positions, say \( \tau_1, \ldots, \tau_n \) in the case of \( R \). When the relation is applied to a sequence \( o_1, \ldots, o_n \) of objects such that \( o_i \in D_{\tau_i} \), we get a proposition, which we will write as \( R(o_1, \ldots, o_n) \), thinking of \( R \) as an \( n \)-place function from objects to propositions. By a procedure which has come to be known as Schönfinkel's trick (1924) and also as \textit{currying} after Curry (Curry and Feys 1961), we can associate each such \( R \) with a constant \( \text{Curry}(R) \in C_{(\tau_1, (\tau_2, \ldots, (\tau_n, p))}) \) which can be given a denotation \( \nu(\text{Curry}(R)) \in D_{(\tau_1, (\tau_2, \ldots, (\tau_n, p))}) \) where we take

\[
(74) \quad \nu(\text{Curry}(R))(o_1)(o_2) \cdots (o_n) = R(o_1, o_2, \ldots, o_n)
\]

if \( o_i \in D_{\tau_i} \) for \( 1 \leq i \leq n \).

In this way, if we start out with some relations, we can simply map them into the corresponding object in the domain of some propositional type. Similarly,
if we have an object in $D_r$ for some propositional type we can simply run the procedure in reverse to recover a relation. Since we have restricted our attention to functional objects which have propositional types, each function in a non-basic domain will determine a relation, providing some motivation for our restriction to propositional types in the first place. It simply does not seem to be necessary to introduce non-propositional types, as no expressions in natural languages make use of them. We will freely mix curried and regular relations in what follows.

### 3.4.3 Individuating Relations

Now that we have introduced a set of relations, the question arises as to how they have been individuated. The answer is that a relation is determined solely in terms of the propositions it produces when applied to an appropriate number of actual objects in its argument domains. Two relations which always produce the same proposition for the same sequence of arguments will be identified. This means that relations can only be individuated as finely as the domain $D_p$ of propositions will allow. For instance, no restriction we have made so far stops the domain of propositions from being isomorphic to the domain of truth values, although we have said why such a choice of propositions would not be a good one. If this is the case, then there will be very few distinct relations indeed, since relations which are true of the same objects will be identified. Similarly, we have not ruled out the possibility of constructing propositions out of functions from possible worlds to truth values. If such a domain of propositions is chosen, then two relations which necessarily produce the same truth values, such as say *groundhog* and *woodchuck* will be identified (see Thomason (1980)). Of course, this is not the only possibility, but only the middle ground. At the far extreme, we could suppose that there is a distinct proposition for each non-logical relation and sequence of arguments. Again, this is a fairly radical, although not uncommon approach to the individuation of relations (see, for instance Cresswell (1985)). Moore (1988) takes this fine-grained individuation of relations even further and does not equate a relation with the propositional function it is associated with, building up a basic domain of relations and then constructing a domain of propositional functions
from it.

We will also want to enforce a version of Leibniz's principle of the identity of indiscernibles which states that two objects which have exactly the same properties must be identical. For every type \( \tau \), we say that two elements \( x_1, x_2 \in D_\tau \) are indiscernible, which we write \( x_1 \sim x_2 \), according to the definition

\[
(75) \quad x_1 \sim x_2 \quad \text{if } P(x_1) = P(x_2) \quad \text{for every } P \in D_{(\tau,p)}
\]

We can then enforce the identity of indiscernibles by requiring our models to obey the condition that for \( x_1, x_2 \in D_\tau \), we have \( x_1 \sim x_2 \) if and only if \( x_1 = x_2 \). Of course, this condition will be trivial if we have assumed that the identity relation for each type \( \tau \) is in our model. The converse, which is known as the indiscernibility of identicals is a consequence of our defining relations as propositional functions, since two functions always take identical values for identical objects.

3.5 Higher Order Logic

Now that we have cast our model in terms of the \( \lambda \)-calculus, we could think of all of our restrictions on the values of terms to be \( \lambda \)-axioms or meaning postulates concerning the value of terms in models. Meaning postulates were introduced into the study of semantics by Carnap (1952). We write \( \alpha \leftrightarrow \beta \) for the axiom stating that \( \alpha \) and \( \beta \) stand for the same object. We will be particularly concerned with meaning postulates of the form

\[
(76) \quad \mathcal{U}_p \leftrightarrow 1
\]

which we will simply write as \( p \). Meaning postulates of this form insure the truth of the proposition \( p \) in a model. We already have a notion of validity and satisfaction for propositions which results from the validity and satisfaction of the above meaning postulate. We will also say that a proposition is a tautology if its corresponding meaning postulate is valid and say that it is a contradiction if its negation is a tautology.

Given a set of meaning postulates \( \mathbf{P} \), we can define the \( \lambda \)-theory \( \Lambda + \mathbf{P} \) consisting of the rules of the \( \lambda \)-calculus and extended by \( \mathbf{P} \). This gives us the usual notion of \( \mathbf{P} \)-consequence and \( \mathbf{P} \)-validity for both propositions and \( \lambda \)-formulas.
For a full higher logic, we can extend our propositional operations to elements in the domains of other relational types. With the aid of propositional functions, we can introduce quantification into the framework.

3.5.1 Quantification

We suppose that there are two constants $V, \exists \in C_{(i,p),((i,p),p)}$ which are called the universal and existential quantifiers. We restrict their behaviour so that

\[ U\forall(p)(Q) = 1 \text{ iff } Up \subseteq UQ \]

and

\[ U\exists(p)(Q) = 1 \text{ iff } Up \cap UQ \neq \emptyset. \]

We will adopt the abbreviation $\exists_1(P)$ for the proposition $\exists(\lambda x.T)(P)$ and similarly for $\forall_1$.

This is not the only possibility for defining the quantifiers. We could have used a strategy common in infinitary logics and simply have set

\[ (\nu(\forall))(P)(Q) = \bigcap_{up(x)=1} Q(x) \]
\[ (\nu(\exists))(P)(Q) = \bigcup_{up(x)=1} Q(x) \]

The reason we do not adopt these definitions is that if we think of propositions as essentially being about things, then we lose the propositional content of the first argument to a quantifier with these definitions. Of course, this is still one possibility within the framework we have adopted, since these definitions are stronger and meet our conditions on the extension of quantifiers. Our definition allows for the possibility that the quantificational propositions are in fact different, although they would have to be extensionally equivalent to, these conjunctions and disjunctions.

3.5.2 Generalised Logical Operations

Working in this domain of propositional functions, our logical operations of conjunction and disjunction can simply be taken as distinguished constants of the propositional type $\langle p, \langle p, p \rangle \rangle$ with negation being of type $\langle p, p \rangle$. 

161
What is interesting to note is that we can extend their definitions to arbitrary propositional types pointwise in the standard way. Suppose that \( \tau = \langle \tau_1, \langle \tau_2, \ldots \langle \tau_n, p \rangle \ldots \rangle \rangle \) is some propositional type and \( \alpha, \beta \in \mathcal{T}_\tau \). We can extend our conjunction operation to objects of type \( \tau \) by setting

\[
(81) \quad \alpha \land \beta = \lambda z_1 \ldots \lambda z_n. (\alpha(z_1) \cdots (z_n) \land \beta(z_1) \cdots (z_n))
\]

This definition was used by Gazdar (1980), Keenan and Faltz (1985), Partee and Rooth (1983) and Partee (1987), in giving the polymorphic value of the conjunction operation. The higher order versions of the other binary operations can be defined in exactly the same way. For negation, we simply set

\[
(82) \quad \sim \alpha = \lambda x_1 \ldots \lambda x_n. (\sim \alpha)
\]

and similarly for the truth value function \( \mathcal{U} \) which we can extend to propositional functions by

\[
(83) \quad \mathcal{U} \alpha = \lambda x_1 \ldots \lambda x_n. (\mathcal{U} \alpha)
\]

We will call \( \mathcal{U} \alpha \) the extension of the propositional function \( \alpha \), and notice that it is of type \( \tau = \langle \tau_1, \langle \tau_2, \ldots \langle \tau_n, t \rangle \ldots \rangle \rangle \). We will also assume that there is propositional identity relation \( \approx \) of type \( \langle \tau, \langle \tau, p \rangle \rangle \) for every type \( \tau \). We then make the natural assumption for terms \( \alpha \) and \( \beta \) of type \( \tau \), that

\[
(84) \quad \mathcal{U} (\alpha \approx \beta) = 1 \text{ iff } \alpha = \beta.
\]
Chapter 4

Meaning and Interpretation

Now that we have laid our syntactic and semantic foundations, we will proceed to discuss the tight relationship between these two domains. It is this relationship between symbols and their meanings that we take to define language.

4.1 Compositionality

As we mentioned in Section 1.1.2, the principle of compositionality is just that the meaning of an expression can be determined from the meaning of its parts and the way in which they were combined. With only a pre-theoretic notion of "meaning" and "combination", the principle of compositionality should be taken as a relational restriction. That is, restricting the basic expressions and their meanings will restrict possible methods of combination. Conversely, restricting the methods of combination restricts the meanings assigned to basic expressions, but neither the possible modes of combination or possible meanings are themselves independently restricted by the requirement that compositionality be satisfied. This means that compositionality is not something that will be easy to verify as a property of language, since it does not place any restrictions on the relation between expressions and their meanings which is independent of the particular method chosen for constructing compound expressions out of other expressions. Instead, the compositionality constraint should be viewed more as a methodological principle to be applied to the task of grammar design, much as structured programming is a methodology applied to software design.
The principle of compositionality is stated so that the meaning of an expression is dependent on the meaning of its parts and the way that they are combined. Recall that the meaning of an expression is simply taken to be whatever information an expression carries across different utterances of it. Since we have not said what meanings are, we can simply take them to be whatever it is that can be built up compositionally. But meanings are not the whole picture, since we have supposed that every utterance of an expression determines a collection of possible interpretations, where the interpretations are derived from the meaning of the expression uttered and the context in which it was uttered. This introduces an additional degree of freedom into the grammar, since nothing requires the interpretation of an expression in a context to be recoverable from the interpretations of the utterances of its subexpressions.

4.1.1 Rule to Rule Compositionality

We will begin with a precise definition of compositionality due to Montague, and then look at some particular proposals for restricting the methods of combining expressions and their meanings to form larger expressions.

Probably the most well known version of compositionality is that put forth by Montague (1974c) in an attempt to define the nature of languages, both natural and artificial. For Montague, the syntax and semantics of a disambiguated language are simply taken to be algebras defined over the same signature, say \( \langle D, \xi_1, \ldots, \xi_k \rangle \) and \( \langle M, \mu_1, \ldots, \mu_k \rangle \), where each \( \xi_i \) is an operation over \( D \) of the same arity as the corresponding operation \( \mu_i \) over \( M \). The elements of the syntactic algebra \( D \) are taken to be disambiguated expressions, while the elements of the semantic algebra \( M \) are what we have called meanings. The operations \( \xi_i \) represent syntactic operations while the \( \mu_i \) represent semantic operations. Montague also makes the assumption that the syntactic algebra \( D \) is freely generated by a set \( B \) of basic expressions. This means that \( D \) is exactly the closure of the set \( B \) of basic expressions under the free application of the operations \( \xi_i \), so that if \( \xi_i \) is an \( n \)-place operation,

\[
(1) \quad \xi_i(d_1, \ldots, d_n) = \xi_j(d'_1, \ldots, d'_m) \iff i = j \text{ and } d_k = d'_k \text{ for } 1 \leq k \leq n = m
\]
It is also assumed that there is a function \( \phi : B \to M \) mapping basic expressions in \( B \) to their meanings. Finally, it is assumed that \( \hat{\phi} \) is the unique homomorphism extending \( \phi \) to \( D \) such that

\[
\begin{align*}
\text{(2)} & \quad \hat{\phi}(d) = \phi(d) \text{ if } d \in B \text{ and } \\
& \quad \hat{\phi}(\xi_1(d_1, \ldots, d_n)) = \mu_1(\hat{\phi}(d_1), \ldots, \hat{\phi}(d_n)).
\end{align*}
\]

The existence of \( \hat{\phi} \) is guaranteed by a general theorem of algebra stating that a homomorphism on an algebra is uniquely determined by its behaviour on any set of generators of the algebra (see Cohn (1965)). It can be easily verified for the definition under consideration here. Under this scheme, each disambiguated expression \( d \in D \) receives a single well-defined meaning \( \hat{\phi}(d) \). In general, we will write \([d]m\) if \( m \) is a possible meaning for the expression \( d \), so that in the case of our disambiguated expressions in \( D \), we have

\[
\text{(3) \quad } [d]m \text{ iff } \hat{\phi}(d) = m.
\]

Of course, nothing requires \( M \) to be a free algebra or requires \( \phi \) to be one-one, so that \( \hat{\phi} \) may be many to one, mapping two distinct expressions to the same meaning.

Since an expression in a natural language can usually have more than one meaning, there seems to be a problem with Montague's characterisation of expressions as having uniquely determined meanings. To get at this phenomenon, Montague introduced a second set of expressions \( E \), which can be chosen arbitrarily, and an ambiguous relation \( \sim a \subseteq D \times E \) between the unambiguous expressions in \( D \) and their ambiguous counterparts in \( E \). The idea is that an expression \( e \in E \) is really ambiguous and can have the same meaning as any of its disambiguations, where we say that \( d \in D \) is a disambiguation of \( e \in E \) if \( d \sim a e \). Again, we write \([e]m\) if the expression \( e \) can have the meaning \( m \), where we set

\[
\text{(4) \quad } [e]m \text{ iff there is some } d \sim a e \text{ such that } [d]m.
\]

If the disambiguating relation is not taken to be regular in some sense, then the whole point of compositionality is missed, so we will now turn to this topic.

Suppose we fix sets \( E \) and \( M \) of possibly ambiguous expressions and meanings. We will also choose an additional set \( \Delta \), whose elements are taken to be syntactic
categories. We will also fix a lexicon \( \Lambda \subseteq E \times \Delta \times M \) where we interpret \( \langle e, \delta, m \rangle \in \Lambda \) to mean that the expression \( e \) can be given the category \( \delta \) with meaning \( m \). An element of a lexicon is called a lexical entry.

A grammar rule is then any tuple of the form

\[
(5) \quad \gamma = \langle \delta_0, \langle \delta_1, \ldots, \delta_n \rangle, e, \mu \rangle
\]

where \( \delta_i \in \Delta \) and \( e \) is some \( n \)-place operation on \( E \) and \( \mu \) is an \( n \)-place operation on \( M \).

Suppose that we have fixed a set \( \Gamma \) of grammar rules. The interpretation of a grammar rule in \( \Gamma \) can be seen in the definition of the full grammar \( \mathcal{G} \), which is obtained by closing the lexicon under the grammatical rules by taking the smallest set \( \mathcal{G} \) such that

\[
(6) \quad \begin{align*}
\text{i. } & \Lambda \subseteq \mathcal{G} \\
\text{ii. } & \langle e(e_1, \ldots, e_n), \delta, \mu(m_1, \ldots, m_n) \rangle \in \mathcal{G} \text{ if } \\
& \langle \delta, \langle \delta_1, \ldots, \delta_n \rangle, e, \mu \rangle \in \Gamma \text{ and } \langle e_i, \delta_i, m_i \rangle \in \mathcal{G} \text{ for } 1 \leq i \leq n
\end{align*}
\]

We will introduce a few convenient notational conventions. First, let

\[
(7) \quad \mathcal{G}_\delta = \{ \langle e, m \rangle \mid \langle \delta, e, m \rangle \in \mathcal{G} \}
\]

be the set of expression-meaning pairs assigned to the category \( \delta \) in the grammar. Similarly, we let

\[
(8) \quad \mathcal{G}^e = \{ \langle \delta, m \rangle \mid \langle \delta, e, m \rangle \in \mathcal{G} \}
\]

be the set of category-meaning pairs associated with the expression \( e \). Finally, we let

\[
(9) \quad \mathcal{G}_\delta^e = \{ m \mid \langle \delta, e, m \rangle \in \mathcal{G} \}
\]

be the set of meanings of \( e \) when given category \( \delta \).

We can define a notion of parse tree locally, by assuming that a local tree of the form

\[
(10) \quad \frac{\langle e_1, \delta_1, m_1 \rangle \ldots \langle e_n, \delta_n, m_n \rangle}{\langle e(e_1, \ldots, e_n), \delta, \mu(m_1, \ldots, m_n) \rangle}
\]
is licensed by the rule \( \langle \delta, (\delta_1, \ldots, \delta_n), \varepsilon, \mu \rangle \in \Gamma \) and that any licensed local tree is admissible. We also assume that any element of \( \Lambda \) is also an admissible tree. This induces a definition of admissibility for arbitrary trees in the usual way, by assuming a tree is admissible if and only if all of its local trees are admissible. Consequently, our admissible trees will always have lexical entries as leaves and elements of the grammar as nodes.

We say that a string is ambiguous if there is more than one parse tree for the string. We can relativise this notion to parse trees producing a particular syntactic or semantic category. In categorial grammar, there is particular interest in so-called spurious ambiguity, which occurs when there are two distinct parse trees of an expression yielding the same meaning (see Moortgat (1987,1987b), Pareschi and Steedman (1987), and Wittenberg (1986)). An expression is said to contain lexical ambiguity if it has two parse trees with different leaves.

Suppose that we have the grammar rules \( \Gamma = \{ \gamma_1, \ldots, \gamma_k \} \), where \( \mu_i \) and \( \varepsilon_i \) are the semantic and syntactic operations associated with rule \( \gamma_i \). To set things up so that these grammars satisfy Montague’s homomorphic version of compositionality it is merely necessary to define the appropriate disambiguated language and ambiguating relation. We simply take the obvious meaning algebra \( M = (M, \mu_1, \ldots, \mu_k) \) consisting of the domain of meanings and the meaning operations from the rules in \( \Gamma \).

Our disambiguated expression algebra will then be \( D = (D, \xi_1, \ldots, \xi_k) \), where each \( \xi_i \) is of the same arity as \( \mu_i \) and \( \varepsilon_i \) and the disambiguated expressions \( D \) are freely generated from the lexicon \( B = \Lambda \), which corresponds to our basic expressions. We define the \( \xi_i \) and set of disambiguated expressions \( D \) inductively by taking

\[
\begin{align*}
(11) & \quad \text{i. } \Lambda = B \subseteq D \\
& \quad \text{ii. } \xi_i(d_1, \ldots, d_n) = \langle i, d_1, \ldots, d_n \rangle \in D \text{ if } \\
& \quad \quad \gamma_i = \langle \delta_0, (\delta_1, \ldots, \delta_n), \varepsilon, \mu \rangle \in \Gamma \text{ and } d_j \in D \text{ for } 1 \leq j \leq n.
\end{align*}
\]

We take the homomorphism \( \hat{\phi} \) from \( D \) to \( M \) to be the unique homomorphism \( \hat{\phi} \) generated from the mapping \( \phi \) taking every lexical entry to its meaning, so that
The ambiguating relation \( \sim_a \) is then taken to be the minimal relation such that

\[
\begin{align*}
\text{(i)} & \quad (e, \delta, m) \sim_a e \text{ if } (e, \delta, m) \in \Lambda \\
\text{(ii)} & \quad (i, d_1, \ldots, d_n) \sim_a e_i(e_1, \ldots, e_n) \text{ if } \gamma_i \in \Gamma \text{ and } d_j \sim_a e_j \text{ for } 1 \leq j \leq n.
\end{align*}
\]

What we have done is constructed a disambiguated algebra \( D \) which is isomorphic to the term algebra with function symbols for each operation in each rule and constants for the lexical entries. Our function \( \hat{\phi} \) can then be seen as the interpretation of this term algebra in the algebra of meanings, with \( \sim_a \) being the interpretation in the algebra of expressions.

As it turns out, basic expressions in \( B = \Lambda \) really consist of a possibly ambiguous expression in \( E \) along with its intended categorization and meaning, which can be thought of as eliminating the ambiguity in \( E \). The disambiguated expressions which are not in the lexicon correspond to the minimal information necessary to reconstruct a parse tree, and hence a unique meaning, since each parse tree has a unique meaning associated with the meaning of its root node. Note that we only need to record the index of a rule in our disambiguated expressions, since we can recover the expression and meaning at a node from those of the subexpressions, given the rule determined by the index.

Grammars designed in this way were dubbed \textit{rule to rule} compositional by Bach (1976), since the syntactic rules come paired with semantic rules. Of course, we have placed absolutely no restriction at all on the behaviour of the syntactic or semantic components of a rule. Montague and those following him in this strategy have taken many liberties with the notion of syntactic rule, allowing arbitrary surface transformations of expressions to be carried out by syntactic rules such as wrapping one expression around another, adding extra pieces onto an expression, deleting part or all of an expression and even more specific operations such as replacing one expression for the first pronoun in a given expression with the same subscript and marking all further pronouns with the same subscript with the same gender as the inserted expression, and so on.
It would be simple enough to restrict our attention to the case where all of the syntactic operations were simple concatenations, so that

\[(14) \ e(e_1, \ldots, e_n) = e_1 \cdots e_n\]

for all \(n\)-place operations \(e\). In this case, the syntactic components of the rule can be viewed as phrase-structure rules, while the meaning component of a rule is still an arbitrary operation on the meanings of its input expressions. To see this more clearly, we define a possible semantics for the phrase-structure grammars directly, by saying that an interpreted phrase-structure grammar is a tuple \(\Gamma = (B, \Delta, M, \Phi, \Lambda)\) where \(B\) is some set of basic expressions, \(\Delta\) a set of syntactic categories, \(M\) a domain of meanings and where an element of \(\Phi\) is a rule of the form

\[(15) \ (\delta_0 \rightarrow \delta_1 \cdots \delta_n, \mu)\]

where \(\mu : M^n \rightarrow M\) is its semantic component and each \(\delta_i \in \Delta\), and \(\Lambda \subseteq B \times \Delta \times M\) is a lexical assignment function, again associating each category with a set of expression-meaning pairs. We can then look at the set of expression-meaning pairs assigned to an arbitrary category by restricting our general definition of of the set \(G\) of expression category meaning triples associated with a grammar by taking the minimal set such that

\[(16) \ i. \ \Lambda \subseteq G\]

\[\text{ii. } \langle e_1 \cdots e_n, \delta, \mu(m_1, \ldots, m_n) \rangle \in G \text{ if } \langle \delta_0 \rightarrow \delta_1 \cdots \delta_n, \mu \rangle \in \Phi \]

\[\text{and } \langle e_i, \delta_i, m_i \rangle \in G \text{ for } 1 \leq i \leq n\]

In the case of interpreted phrase-structure grammars, we can simplify our trees somewhat, by assuming that a tree of the form

\[(17) \ \frac{\langle \delta_1, f_1 \rangle \cdots \langle \delta_n, f_n \rangle}{\langle \delta, \mu(f_1, \ldots, f_n) \rangle}\]

is admitted by the rule \(\langle \delta \rightarrow \delta_1 \cdots \delta_n, \mu \rangle \in \Gamma\). We will also assume that a tree of the form

\[(18) \ \frac{e}{\langle \delta, m \rangle}\]
is admissible if \( (e, \delta, m) \in \Lambda \) is a lexical entry, which leads to the usual definition of admissibility for arbitrary trees. The reason that we don't include the expression at each node in the tree is that the expression at any node can be read left to right from the leaves of the subtree rooted at that node.

Of course, there is nothing requiring two rules with identical syntactic operations to have identical semantic operations, since we could have two rules \( (\delta_0 \rightarrow \delta_1 \cdots \delta_n, \mu) \) and \( (\delta_0 \rightarrow \delta_1 \cdots \delta_n, \mu') \) with \( \mu \neq \mu' \). This means that as many meaning operations as necessary may be attached to each phrase-structure operation. This is exactly the slack which is used by Cooper (1983) in developing a rule to rule account of quantifier scope based on a syntactically natural phrase structure grammar.

4.1.2 Type-Driven Translation

Most of the theoretical work in linguistic semantics has employed models based on the typed \( \lambda \)-calculus. In this framework, some set of basic domains is fixed and then closed under the construction of total function spaces. This set of basic domains and functions is then identified with the domain \( M \) of meanings. Then each syntactic category is assigned to some type, with the idea being that expressions of that category will receive meanings of the assigned type. The semantic operation of a grammar rule will then be required to map meanings of the types of the daughters onto meanings of the type of the mother expression.

To make this more precise, suppose we restrict our attention to grammars satisfying rule to rule compositionality. We also fix a set \( \Theta \) of basic types along with a type function \( \theta : \Delta \rightarrow \text{Typ}(\Theta) \) supplying each syntactic category in \( \Delta \) with exactly one type, where, as usual, the set \( \text{Typ}(\Theta) \) is minimal such that

\[
\begin{align*}
(19) \quad & \text{i. } \text{Typ}(\Theta) \subseteq \Theta \\
& \text{ii. } \text{Typ}(\Theta) \subseteq \text{Typ}(\Theta) \times \text{Typ}(\Theta). 
\end{align*}
\]

Suppose we have also fixed a frame \( D \) for interpretation derived from a collection of domains \( D_\tau \) for each basic type \( \tau \in \Theta \). We then take our meaning domain \( M \) to be the frame \( \bigcup_{\tau \in \text{Typ}(\Theta)} D_\tau \) where \( D_{\theta(\tau)} = D_\tau^{D_{\theta(\tau)}} \). We require our lexical assignment function \( \Lambda \) to be such that \( \{ m \mid (e, \delta, m) \in \Lambda \} \subseteq D_{\theta(\delta)} \) so that every
category is lexically assigned meanings of the appropriate type. We further require that every rule \( \gamma = (\delta_0, (\delta_1, \ldots, \delta_n), \epsilon, \mu) \) is such that its semantic operation \( \mu \) is a typed function \( \mu : D_{\theta(\delta_1)} \times \cdots \times D_{\theta(\delta_n)} \to D_{\theta(\delta_0)} \) which, when curried, is a function of type \( \theta(\delta_1), \theta(\delta_2), \ldots, \theta(\delta_n), \theta(\delta_0) \)).

This is enough to insure that

\[ (20) \ m \in D_{\theta(\delta)} \text{ if } (\epsilon, \delta, m) \in \mathcal{G} \]

for every category \( \delta \), as can be proved by a simple induction.

With these definitions, we have departed from Montague's strict algebraic framework in that our set \( M \) of meanings no longer forms an algebra with total operations. The operations \( \mu \) are only required to be defined on inputs of the appropriate types. This is, in fact, another harmless move, since we could simply work in a domain \( M \cup \{*\} \), where we suppose that the result of applying a semantic operation to arguments of the wrong type yields \(*\), as is often done in treating partial functions over some domain as total functions over a domain extended to include an element standing for "undefined". We simply ignore this minor detail, as we will never have any reason to apply a semantic operation to inappropriate arguments.

Klein and Sag (1985) have developed a method for generating meaning functions for syntactic rules, called type-driven translation, which avoids the arbitrary stipulation of a semantic component for every rule in the grammar by deriving the semantic component of a grammar rule in a uniform way. Suppose we are working in a typed system of the above sort, with a set of syntactic categories \( \Delta \), set of basic types \( \Theta \) and with a given type function \( \theta : \Delta \to \text{Typ}(\Theta) \) supplying categories with their (possibly complex) types. We then suppose that we are given a set \( \Gamma \) of grammar rules, where a rule is similar to earlier rules, only without a place for a semantic operation, so that a rule \( \gamma \in \Gamma \) will be of the form \( \gamma = (\delta_0, (\delta_1, \ldots, \delta_n), \epsilon) \). We can then translate a grammar of this form into our earlier format by simply specifying which semantic operation \( \mu \) should be associated with each rule \( \gamma \in \Gamma \). In general, for a rule \( \gamma_i \) as above, take \( \mu_i \) to be the denotation of the \( \lambda \)-term \( \lambda x_1^{\theta(\delta_1)}, \ldots, \lambda x_n^{\theta(\delta_n)}, \alpha \) in a frame based on the given basic domains, where \( \alpha \) is a closed pure \( \lambda \)-term with exactly one occurrence of each variable \( x^i \) of type \( \theta(\delta_i) \) and no other variables or constants. First note that
since this term is closed it will not depend on an assignment, and since it is pure, its denotation will only depend on the domains assigned to basic types and not on any values assigned by the model to constants. More importantly, it should be noted that this term is not necessarily unique, as can be seen by considering a rule with three constituents with semantic types $\tau, (\tau, \tau), (\tau, \tau)$, which can be applied in two different orders, yielding the distinct $\lambda$-terms

$$
\lambda x^1_\tau. \lambda x^2_{(\tau, \tau)}. \lambda x^3_{(\tau, \tau)} x^2(x^1) \neq \lambda x^1_\tau. \lambda x^2_{(\tau, \tau)}. \lambda x^3_{(\tau, \tau)} x^2(x^1).
$$

There are two options at this point. The first is to rule out any syntactic rules which produce multiple semantic possibilities, and the second is to allow the non-determinism by permitting any of the semantic operations to be applied, producing a separate grammar rule for each semantic possibility. The former choice is taken in generalized phrase structure grammar (Gazdar et al. 1985) where each rule must have a unique translation, with the type-driven translation thus filtering out grammar rules which might be judged well-formed by other components of the grammar.

4.1.3 Application Driven Translation

In the tradition of categorial grammar, even stronger constraints are put on the compositional nature of the relation between syntactic form and meaning. This can all be carried out within the framework of type-driven translation, as it is really just a special case of it.

Suppose we have fixed a categorial grammar $\Gamma = (\Delta, \Sigma, \Lambda)$ with the basic expressions $\Delta$, basic categories $\Sigma$, categories $\text{CAT}(\Sigma)$, and lexical assignment $\Lambda \subseteq \Delta \times \text{CAT}(\Sigma)$. Suppose we have also fixed a frame derived from sets given for each basic type in $\Theta$. The crucial assumption of application driven translation is then that the typing function $\theta : \text{CAT}(\Sigma) \rightarrow \text{Typ}(\Theta)$ is defined so that it meets the restriction

$$
\theta(\alpha / \beta) = \theta(\alpha \setminus \beta) = (\theta(\beta), \theta(\alpha))
$$

if $\alpha, \beta \in \text{CAT}(\Sigma)$. This typing was originally introduced in the categorial grammars of Montague (1974c). We will often write $\mathcal{D}_\theta$ for $\mathcal{D}_{\theta(\alpha)}$ when $\theta$ is understood.
Note that we have not placed any restriction on the types of the basic categories, but once these are fixed, the types of the functional categories will be fully determined. This means that functional categories receive functional types based on the type of their argument and result categories.

Restricting our interpreted phrase-structure grammars to the type-driven categorial grammars, we say that an interpreted categorial grammar is a triple $\Gamma = (\Delta, \Sigma, \Lambda)$, where $\Delta$ is a set of basic expressions, $\Sigma$ a set of basic categories and $\Lambda$ is a set of triples of the form $(e, \alpha, m)$ where $\alpha \in \text{CAT}(\Sigma)$ is a category, $e$ is the expression of that category, and $m \in D_{\theta(\alpha)}$ is a meaning assigned to $e$.

Now consider the application rule schemata

\begin{enumerate}
  \item $\alpha \rightarrow \alpha / \beta$ \hspace{1cm} \beta
  \item $\alpha \rightarrow \beta \hspace{0.5cm} \alpha \setminus \beta$
\end{enumerate}

Assuming type-driven translation, there is only one possibility for the meaning of the mother category, and that is the functional daughter's meaning applied to that of its argument's meaning. This produces the two interpreted rule schemata

\begin{enumerate}
  \item $\langle \alpha \rightarrow \alpha / \beta \hspace{0.5cm} \beta, \text{[}\lambda x.\lambda y. x(y)\text{]} \rangle$
  \item $\langle \alpha \rightarrow \beta \hspace{0.5cm} \alpha \setminus \beta, \text{[}\lambda y.\lambda x. x(y)\text{]} \rangle$
\end{enumerate}

where $x$ is a variable of type $\theta(\alpha / \beta) = \theta(\alpha \setminus \beta) = \langle \theta(\beta), \theta(\alpha) \rangle$ and $y$ is a variable of type $\theta(\beta)$. This means that the semantics of the mother category with a syntactic category $\alpha$ will be of type $\theta(\alpha)$. Note that the $\lambda$-terms in question are closed and so will only depend on the frame and not on the values assigned to constants or variables.

Consider the interpreted categorial grammar which results if we take our simple English categorial grammar from Section 2.3.3 and assume that each expression is assigned not only a category, but an appropriately typed meaning. We will simply write this meaning as the expression in bold face. So, a lexical entry like $(john, np)$ from the simple context free grammar, turns into the lexical entry $(john, np, john)$, where it is assumed that $john \in D_{\theta(np)}$. Similarly, we have lexical entries like $(ran, s \setminus np, ran)$, where $ran \in D_{(np,s)}$, and $(in, n \setminus n / np, in)$, where $in \in D_{(np,(n,n))}$. For the time being, we will not consider the contents of
the basic semantic domains, but simply assume that they have been fixed. This interpreted categorial grammar admits the parse trees

(25) \[
\begin{array}{ccc}
\text{every} & \text{penguin} & \text{ran} \\
\langle np/ n, \text{every} \rangle & \langle n, \text{penguin} \rangle & \langle s \setminus np, \text{ran} \rangle \\
\langle np, \text{every(penguin)} \rangle \\
\langle s, \text{ran(every(penguin))} \rangle \\
\end{array}
\]

(26) \[
\begin{array}{ccc}
\text{tall} & \text{penguin} & \text{by} & \text{opus} \\
\langle n/ n, \text{tall} \rangle & \langle n, \text{penguin} \rangle & \langle n \setminus n/ np, \text{by} \rangle & \langle np, \text{opus} \rangle \\
\langle n \setminus n, \text{by(opus)} \rangle \\
\langle n, \text{by(opus)(penguin)} \rangle \\
\langle n, \text{tall(by(opus)(penguin))} \rangle \\
\end{array}
\]

(27) \[
\begin{array}{ccc}
\text{tall} & \text{penguin} & \text{by} & \text{opus} \\
\langle n/ n, \text{tall} \rangle & \langle n, \text{penguin} \rangle & \langle n \setminus n/ np, \text{by} \rangle & \langle np, \text{opus} \rangle \\
\langle n, \text{tall(penguin)} \rangle & \langle n \setminus n, \text{by(opus)} \rangle \\
\langle n, \text{by(opus)(tall(penguin))} \rangle \\
\end{array}
\]

(28) \[
\begin{array}{ccc}
\text{gave} & \text{binkley} & \text{a cat} \\
\langle s \setminus np/ np/ np, \text{gave} \rangle & \langle np, \text{binkley} \rangle & \langle np, \text{a(cat)} \rangle \\
\langle s \setminus np/ np, \text{gave(binkley)} \rangle \\
\langle s \setminus np, \text{gave(binkley)(a(cat))} \rangle \\
\end{array}
\]

(29) \[
\begin{array}{ccc}
\text{slowly} & \text{ran} & \text{by} & \text{opus} \\
\langle s \setminus np/ (s \setminus np), \text{slowly} \rangle & \langle s \setminus np, \text{ran} \rangle & \langle s \setminus np/ (s \setminus np), \text{by(opus)} \rangle \\
\langle s \setminus np, \text{by(opus)(ran)} \rangle \\
\end{array}
\]

In extended categorial grammars, a more general version of type-driven translation is often maintained to account for functional composition in unbounded dependency contexts, among other things. For investigations of the semantics of extended categorial grammars and their relation to implicational logics and deductive systems, see Steedman (1987,1988), Morrill (1988) and Morrill and Carpenter (forthcoming) and van Benthem (1986,1987).

4.2 Unification and Logical Form

In this section we explain how to relate the type-driven semantics of pure categorial grammars to the unification based categorial grammars introduced in Sec-
tion 2.3.4. We will show how to represent \(\lambda\)-terms by means of first-order terms and how this can be used to provide a unification grammar which derives the semantics of a phrase in the same way as its syntactic category.

4.2.1 Intermediate Translations

Consider the nature of the meaning domains employed so far. We have said nothing about them other than that they are members of some algebra \(M\) or typed functional domain \(M\) of meanings. The lexicon can be thought of as providing expressions with both categories and appropriate meanings drawn from \(M\). Each grammar rule contains a syntactic component and a corresponding operation on \(M\). Under type-driven translation, the meaning algebra \(M\) can simply be taken to be the objects in some frame for the \(\lambda\)-calculus, constructed from basic semantic domains and typed functions between (possibly functional) domains.

Consider what we did in the case of application-driven translation and type-driven translation. In both cases, the way in which we got a handle on the semantic function associated with each grammatical rule was by writing down \(\lambda\)-terms which have denotations in the algebra of meanings. For instance, the meaning operation associated with a rule under type-driven translation was determined by finding a \(\lambda\)-term which could be built out of variables of the appropriate types and then abstracting over the result. The reason for defining things this way in the first place is simply because \(\lambda\)-terms are easy to manipulate, being designed for just this purpose.

What we could have done instead, and what Montague (1974c,1974d) actually did in practice, was assume that the lexicon \(\Lambda\), instead of assigning pairs of expressions and meanings of the appropriate type to each category, could assign pairs of expressions and \(\lambda\)-terms of the appropriate type. Where before each grammatical rule carried a semantic operation over \(M\), we would now have a \(\lambda\)-term of the appropriate type. Where before the semantic operation component of a rule would be applied to the semantic components of the daughters, now a \(\lambda\)-term corresponding to the mother’s semantics would be constructed. In fact, this is more or less what we did implicitly in the previous section on application-driven
translation. Each node in the tree can either be thought of as a pair consisting of a category and member of $M$ or as a category and $\lambda$-term. Given some fixed model of the $\lambda$-calculus, a category-$\lambda$-term pair determines a category-meaning pair by taking the meaning to be the value of the $\lambda$-term in the model. More precisely, it can be shown that any homomorphism $\phi : A \rightarrow B$, where $A$ and $B$ are algebras defined over the signature $\Sigma$, can be decomposed into two homomorphisms $\psi : A \rightarrow T$ and $\rho : T \rightarrow B$, such that $\phi = \rho \circ \psi$, where $T$ is the term algebra defined over the signature $\Sigma$ and $\rho$ is the interpretation function defined over some fixed model and assignment to variables, which will always be a homomorphism (see Cohn (1965)).

Consider the properties that a homomorphism into the term algebra will have. First, it will necessitate a unique representation for each element of the domain, since homomorphisms are functions. Often, though, we would like to be able to substitute a term for an equivalent term in any admissible tree and still be left with an admissible tree. Treating our operations as operations over terms introduces the possibility of semantic operations that do not meet this condition. We will see an example of this in Section 5.1.2.

4.2.2 Interpreted Unification Categorial Grammars

In this section we present three different term unification grammars to represent a given interpreted categorial grammar. We use a first-order encoding of $\lambda$-terms due to Pereira and Shieber (1987) (which they claim is implicit in the work of Colmerauer (1982), Dahl (1981) and Pereira and Warren (1980).

First off, we fix a set $\Theta$ of basic syntactic types and a set of constants $C$, and variables $V$ for each type $\tau \in \text{TYP}(\Theta)$. This determines a set of $\lambda$-terms constructed from these constants and variables. Next, we will need to fix an interpreted categorial grammar $\Gamma = (\Delta, \Sigma, \Lambda)$, where $\Delta$ is a set of basic expressions, $\Sigma$ a set of basic categories and the lexicon $\Lambda$ is a set of triples of the form $(e, \delta, m)$ where $\delta \in \text{CAT}(\Sigma)$ is a category, $e$ is an expression of that category, and $m \in T_{\theta}(\delta)$ is a $\lambda$-term of type $\theta(\delta)$, which will determine a meaning for the expression $e$.

Next we consider the first-order functional signature $\Upsilon$ over which we will rep-
resent our interpreted categorial grammar. We assume that it contains the slash
constructors in $\mathcal{T}_2$ as well as constants in $\mathcal{T}_0$ corresponding to basic categories.
To represent $\lambda$-terms, we assume that $\mathcal{T}$ is such that

\begin{enumerate}
  \item $C \subseteq \mathcal{T}_0$
  \item $/, \backslash, :, ., \otimes \in \mathcal{T}_2$
\end{enumerate}

so that every constant $\lambda$-term is a 0-place function symbol. We use $\otimes$ to construct
terms corresponding to the pairing of a syntactic term with a semantic term.
We will use the other binary function symbols to construct representations of $\lambda$-
terms, where $t_1 \otimes t_2$ represents the application of the term represented by $t_1$ to
that represented by $t_2$. Similarly, we use $t_1 \cdot t_2$ to represent abstraction, as we will
see below. We need to include an explicit two-place function symbol to represent
functional application, since the term phrase-structure formalism does not allow
us to unify function symbols. In general, we could not use the $t_1(t_2)$ in place of
$t_1 \otimes t_2$, since there will be cases where $t_1$ is not a function symbol. Having said
this, we will still use the abbreviation $t_1(t_2)$ for $t_1 \otimes t_2$ where convenient.

System 1

We define a mapping $\phi$ from the $\lambda$-terms to terms in our signature, by taking

\begin{enumerate}
  \item $\phi(x) = X$ if $x$ is a variable
  \item $\phi(c) = c$ if $c$ is a constant
  \item $\phi(\alpha(\beta)) = \phi(\alpha) \otimes \phi(\beta)$
  \item $\phi(\lambda x.\alpha) = X.\phi(\alpha)$
\end{enumerate}

Of course, this does not insure that the encoding of two equivalent $\lambda$-terms will
be alphabetic variants of one another. First of all, consider the simple equivalence
$(\lambda x.x)(a) \leftrightarrow a$ in the $\lambda$-calculus. The encoding of $(\lambda x.x)(a)$ is given by $(X.X) \otimes a$, 
assuming that $a$ is a constant. This is not anything like the simple term $a$. Another
thing to note is that there is no notion of binding for unification terms, so that
$\lambda x.\lambda x.x$ which is equivalent to $\lambda y.\lambda x.x$, is encoded by $X.(X.X)$ which is not an
alphabetic variant of the encoding $Y.(X.X)$ and so under substitutions we get
$(\lambda x.\lambda x.x)[x/t] = \lambda x.\lambda x.x$, while $(X.(X.X))[X/t] = t.(t.t)$.
We assume that the term phrase-structure grammar has the lexicon

\[(32) \Lambda_1 = \{(e, \alpha : \phi(m)) \mid (e, \alpha, m) \in \Lambda\}.\]

Note that this lexicon pairs an expression and a term representing both its syntactic category and semantic function.

We then take the two term phrase-structure rules

\[\begin{align*}
(33) & \quad X : (F@G) \rightarrow X/Y : F \quad Y : G \\
(34) & \quad X : (F@G) \rightarrow Y : G \quad X \setminus Y : F
\end{align*}\]

Using this unification grammar, we can exactly mirror any derivation in the original interpreted categorial grammar, since we will have constants for lexical entries and simple application terms corresponding to the applications in the interpreted categorial grammar.

System 2

This is not the only way in which interpreted categorial grammars can be cast in a unification framework. Consider the following mapping of \(\lambda\)-terms into the set of unification terms

\[(35) \begin{array}{l}
\text{i. } \psi(x) = X \text{ if } x \text{ is a variable of a basic type} \\
\text{ii. } \psi(x) = \psi(y).\psi(z) \text{ if } x \text{ is a variable of type } (\sigma, \tau) \text{ and } y \text{ is a variable of type } \sigma \text{ and } z \text{ is a variable of type } \tau \\
\text{iii. } \psi(c) = c \text{ if } c \text{ is a constant of a basic type} \\
\text{iv. } \psi(c) = \psi(x_1)\cdots\psi(x_n).(c@\psi(x_1)\cdots@\psi(x_n)) \text{ if } c \text{ is a constant of type } (\tau_1, \tau_2, \ldots, \tau_n) \text{ and } x_i \text{ is a variable of type } \tau_i \\
\text{v. } \psi(\lambda x.\alpha) = \psi(x).\psi(\alpha) \\
\text{vi. } \psi(\alpha(\beta)) = t_2[\sigma] \text{ if } \psi(\alpha) = t_1.t_2, \psi(\beta) = t'_1 \text{ and } \sigma \text{ is a most general unifier of } t_1 \text{ and } t'_1
\end{array}\]

For instance, in the case of \textsc{slowly} of type \((\langle np, s \rangle, \langle np, s \rangle)\), we have

\[(36) \psi(\text{slowly}) = \psi(x_{(np,s)}).\psi(y_n).(\text{slowly}@\psi(x_{(np,s)})@\psi(y_n))\]
where we have

\[(37) \quad \psi(x_{(np,a)}) = \psi(x_{np}).\psi(y_a) = X.Y.\]

Now consider the case of a \(\lambda\)-term like \(f(g)\) where \(f\) is of type \((\alpha, \beta)\) and \(g\) is of type \(\beta\) for some atomic types \(\alpha\) and \(\beta\). We would then have \(\psi(f) = X.f@X\) and \(\psi(g) = g\), so that \(\psi(f(g)) = f@g\) after unifying \(g\) with \(X\), where we have taken \(t_2 = f@X\), \(t_1 = X\) and \(t'_1 = g\) in (35)vi.

As a consequence of these definitions, a lexical entry with a functional syntactic category will have a semantic component of the form \(t_1.t_2\), where \(t_1\) is the image under \(\psi\) of some variable. It should be noted that if \(x\) is a variable then \(\psi(x)\) will be entirely composed of variables, and will hence unify with the image of any other term of the same type as the variable, making it the most general term of its type. Since there are only countably many variables of countably many different types, we can assume without loss of generality that the image of any variable \(x\) under \(\psi\) is, in fact, composed of unique variables, so that we will never have to worry about variable clashes. This is enough to insure that the unification in the last clause will always exist and such a reduction can be carried out. The encodings \(\phi\) and \(\psi\) are very closely related, so that for any \(\lambda\)-term \(\alpha\), there is a term \(\alpha'\) such that \(\alpha \leftrightarrow \alpha'\) and \(\psi(\alpha) = \phi(\alpha')\). In fact, we will have \(\text{Rng}(\phi) \subseteq \text{Rng}(\psi)\), since equivalent \(\lambda\)-terms receive encodings which are alphabetic variants under \(\psi\), since we can view the translation procedure as first moving to the canonical form of a term, and then providing an abstraction semantics. The same is not true of \(\phi\), as can be seen in the case of the equivalent terms \((\lambda x.c(x))(a) \leftrightarrow c(a)\) which receive the encodings \(\phi((\lambda x.c(x))(a)) = (X.(c@X))@a\), and \(\phi(c(a)) = c@a\), respectively.

We construct the lexicon for the second system in the same manner as the first, taking

\[(38) \quad \Lambda_2 = \{(e,\alpha : \psi(m)) \mid (e,\alpha, m) \in \Lambda\}\]

While this lexicon would seem to be equivalent to the first in light of the results about \(\phi\) and \(\psi\), the fact that the \(\lambda\)-terms are canonically encoded by \(\psi\) as having abstraction semantics, allows us to adopt the unification rules

\[(39) \quad X : F \rightarrow X / Y : (G.F) \quad Y : G\]
These rules carry out the equivalent of $\beta$-conversion in our unification framework, since a straightforward induction carries the result from the lexicon to the full grammar, that if a category has a functional category it will have abstraction semantics.

Carrying out this conversion on the interpreted categorial grammar from our simple English fragment yields the lexicon

<table>
<thead>
<tr>
<th>Word</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>opus</td>
<td>np : opus</td>
</tr>
<tr>
<td>penguin</td>
<td>n : penguin</td>
</tr>
<tr>
<td>the</td>
<td>np / n : X.the@X</td>
</tr>
<tr>
<td>sneezed</td>
<td>s \ np : X.sneezed@X</td>
</tr>
<tr>
<td>loved</td>
<td>s \ np / np : Y.X.loved@Y@X</td>
</tr>
<tr>
<td>tall</td>
<td>n / n : X..tall@X</td>
</tr>
<tr>
<td>in</td>
<td>n \ n / np : W.X.in@W@X</td>
</tr>
<tr>
<td>slowly</td>
<td>s \ np / (s \ np) : (Y.X).Z.slowly@Y@X</td>
</tr>
</tbody>
</table>

Now consider the sentence the tall penguin loved opus. We can combine the entries for loved and opus by forward application, so that we get the category

\[(42) \quad s \ np : X.loved@opus@X\]

for loved opus. Working on the subject, we get the category

\[(43) \quad n : tall@penguin\]

for the nominal tall penguin by forward application. We can then combine the determiner with the nominal to produce the phrase the tall penguin with the category

\[(44) \quad np : the@(tall@penguin)\].

This noun phrase then combines with the verb phrase by backward application to produce the category

\[(45) \quad s : loved@opus@(the@(tall@penguin))\].
There is a standard problem with this encoding of terms that we will not encounter. Consider the application of a function to two different arguments, as in \( \lambda x.f(x)(a) \land \lambda x.f(x)(b) \), which is encoded by

\[(X.f \otimes X)a \land (X.f \otimes X)b.\]

To reduce this to standard form, it would be necessary to unify the variable \( X \) with both \( a \) and \( b \), which is not possible. Such a case will not arise in the semantic translations that we employ, as we can always assume that our basic terms have unique variables in their abstracts. This problem will also not arise in any simple categorial operations, since they always apply one function to one argument.

**System 3**

Now that we have represented our semantic terms in a canonical fashion, we can go even further and simplify our system down to that of our original term categorial grammar rules

\[(47) \quad X \rightarrow X / Y \ Y \]

\[(48) \quad X \rightarrow Y \ X \backslash Y.\]

The lexicon for this system is defined in terms of the last system, with

\[(49) \quad \Lambda_3 = \{(e, \rho(c)) \mid (e, c) \in \Lambda_2\}\]

where we set

\[(50) \quad \begin{align*}
\rho(\delta : t) &= \delta : t \text{ if } \delta \text{ is a basic category} \\
\rho(\alpha / \beta : t_\beta.t_\alpha) &= \rho(\alpha : t_\alpha) / \rho(\beta : t_\beta) \\
\rho(\alpha \backslash \beta : t_\beta.t_\alpha) &= \rho(\alpha : t_\alpha) \backslash \rho(\beta : t_\beta)
\end{align*}\]

As a consequence, we will get categories of a very different sort in this grammar. Categories are no longer pairs of a syntactic category and semantic term, but consist of either a basic syntactic category paired with a basic semantic term or a slash category built out of basic categories in the usual way. For instance, since \( \psi(\text{slowly}) = (X.Y).Z.\text{slowly} \otimes (X.Y) \otimes Z \) we will get the lexical entry
\[(51) \quad \langle \text{slowly}, \rho(s \backslash np \backslash s \backslash np : (X.Y).Z.\text{slowly}@(X.Y)@Z) \rangle \]

where

\[(52) \quad \rho(s \backslash np \backslash s \backslash np : (X.Y).Z.\text{slowly}@(X.Y)@Z) =
\rho(s \backslash np : Z.\text{slowly}@(X.Y)@Z) \backslash \rho(s \backslash np : X.Y) =
s : \text{slowly}@(X.Y)@Z \backslash np : Z \backslash (s : Y \backslash np : X)\]

Notice that the terms \(X\) and \(Y\) are split across the two subcategories of the argument category, but get reconnected in the semantics of the final result category.

The lexicon of our simple English fragment will then look like

\[(53) \quad \begin{array}{ll}
\text{Word} & \text{Category} \\
opus & np : \text{opus} \\
penguin & n : \text{penguin} \\
the & np : \text{the} \backslash X / n : X \\
sneezed & s : \text{sneezed} \backslash X \backslash np : X \\
loved & s : \text{loved} \backslash Y @ X \backslash np : X / np : Y \\
tall & n : \text{tall} \backslash X / n : X \\
in & n : \text{in} \backslash W \backslash X \backslash n : X / np : W \\
slowly & s : \text{slowly} \backslash (Y.X)@Z \backslash np : Z / (s : X \backslash np : Y) \\
\end{array} \]

### 4.3 Naïve Event Semantics

In this section, we take the first step toward a semantics for a categorial grammar using the domains introduced in Chapter 3. The naïveté of this fragment stems from the fact that we will not deal with any contextual information and make a number of other simplifying assumptions.

The grammar in this section will be presented as an interpreted categorial grammar.

#### 4.3.1 Simple Nominals

**Proper Names**

Proper names such as *opus, binkley, milo* and so on have the syntactic category *np*. We will assume that the meaning of any *np* is an individual in \(\text{IND}\) so that

\[(54) \quad D_{np} = \text{IND}.\]
All of the noun phrases in the simple English fragment are singular. We will not yet concern ourselves with the various readings we discussed for plural noun phrases. With these assumptions in hand, it follows that simple names must pick out unique individuals. We are led to lexical entries such as

\[(55) \ (opus, np, opus)\]

where \(opus \in \text{IND}\). Of course, nothing restricts us from having a number of lexical entries, one for each individual named \(opus\). For instance, if \(J \subseteq \text{IND}\) is the collection of individuals named John then we could assume that \((john, np, j) \in \Lambda\) for each \(j \in J\), so that the expression \(john\) would be lexically ambiguous as many ways as there are individuals named John.

**Common Nouns**

The next case is that of common nouns such as \(dog, cat, kid\) and so on. We will suppose in this naïve fragment that the meaning of a simple common noun is a property. Recall that properties are functions from the domain of individuals \(\text{IND}\) into the domain \(\text{PROP}\) of propositions. This means that we will take \(\theta(n) = (i, p)\), so that

\[(56) \ D_n = \text{PROP}^{\text{IND}}.\]

Note that nothing in our definitions requires that basic domains be given interpretations as unstructured sets, or that the domains \(D_{\tau}\) for \(\tau \in \text{TYP}\) be disjoint. For instance, Montague (1974c,1974d) gave nouns and verb phrases the semantic type \((i, p)\) as our nouns. Our lexical entries for common nouns will then look like

\[(57) \ (kid, n, kid)\]

where \(kid \in D_n\) is a property such that for \(x \in \text{IND}\), \(kid(x)\) is true if and only if \(x\) is a kid. As promised, we continue to ignore the actual nature of the property represented by the constant \(kid\), but simply assume that whatever property we use the expression \(kid\) to pick out is the denotation of the constant \(kid\).
Definite Determiners

To complete the analysis of simple nominals, we consider the definite determiner the, which has the syntactic category np / n. Since \( \theta(np / n) = (\theta(n), \theta(np)) \), the meaning of a determiner is a function from n meanings to np meanings, so that \( \theta(np / n) = ((i, p), i) \).

We assume that the, the meaning of the definite determiner the, is a function which maps a property to the unique element of IND which has the property. We can think of a partial function from objects of type \( \tau \) to objects of type \( \sigma \) as being a special kind of relation \( R \) of type \( \langle \tau, (\sigma, p) \rangle \) such that for every object \( x \) of type \( \tau \) there is at most one \( y \) of type \( \sigma \) such that \( R(x)(y) \). From such an \( R \), we can determine an element of \( \langle \sigma, \tau \rangle \) in the usual way. We will write the propositional function corresponding to the relation \( R \) simply as \( R \). No confusion should arise due to the fact that the objects are of different types. So, in particular, we will write \( R(x) \), as usual, for the unique \( y \) such that the proposition \( R(x)(y) \) is true.

Note that the propositional function \( R \) of type \( \langle \tau, (\sigma, p) \rangle \), just like all propositional functions, is intensional in that it takes a value which is a proposition rather than just a simple truth value. Thus, two partial functions which are extensionally equivalent may be distinguished in terms of their relational representations.

We will take the to be of type \( \langle (i, p), (i, p) \rangle \), where \( \text{the}(P)(x) \) holds if and only if \( x \) is the unique object with property \( P \). We will write \( \text{the}(P) \) for the \( x \) such that \( \text{the}(P) \). This means that we will notate the as if it were a function of type \( \langle (i, p), i \rangle \).

The lexical entry for the definite determiner will then be

\[
(58) \quad \langle \text{the}, np / n, \text{the} \rangle.
\]

To deal with the fact that the is only a partial function, we could suppose that there is a distinguished element of \( * \in \text{IND} \) in the range of the, which acts as the referent of non-referring noun phrases, such as the unicorn, when there are no unicorns, or the student, when there are many objects which have the property of being a student. Otherwise, we could suppose that such an expression is simply not well-formed if it does not have a well defined meaning. We make the latter assumption for concreteness, since it is somewhat unnatural to assume that
sentences containing non-referring definites have any meaning at all and introduc-
ing * into IND will produce a number of unwanted functions in the domains of
functional types. When we move beyond the naïve semantics, this problem will
disappear. Also note that the definite article the is the only determiner that can
be appropriately modeled as a function from properties into individuals. We take
up the task of general determiners such as every and no in Section 5.1.

4.3.2 Simple Verbs

Simple intransitive verbs are given the syntactic category $s \setminus np$, so that we have
$\theta(s \setminus np) = (i, \theta(s))$. This means that $D_{s \setminus np}$ will be determined once we have fixed
the domain $D_s$ assigned to sentences. The standard assumption in model theoretic
semantics seems to be that $V, = PROP$, so that the meaning of a sentence is a
proposition. Employing this basic type, we would find that $\theta(s \setminus np) = \theta(n) =
(i, p)$, so that nouns and verb phrases would take their meanings from the same
domain. This was the type assignment employed by Montague (1974c, 1974d).

As should be obvious from Section 3.2.2, we will assume that the semantics
of sentences is tied up in an essential way with events. More or less following
situation semantics (Barwise and Perry 1983, 1985), we assume that $\theta(s) = (e, p)$,
so that a sentence meaning will be a property of events, with

$$ D_s = PROP_{EVENT}. $$

It follows that $D_s \subseteq D_n$. This is because we assume that EVENT $\subseteq IND$ so that
every event is an individual. In this way, we take the meaning of a sentence to
determine a property of events in the same way as a noun determines a property
of individuals.

This means that a typical lexical entry for a verb such as ran will look like

$$ (\text{ran}, s \setminus np, \text{ran}). $$

Note that we are completely ignoring the treatment of tense, using a constant
ran corresponding to the finite verb ran. This is because we have no way of
representing the time of utterance in our naïve context independent semantics.
This gives us a meaning such as ran(opus) for a sentence such as opus ran. The
intended interpretation of ran is such that the proposition ran(x)(e) is true iff e is an event of x's running.

Most of those committed to an event based semantics make the assumption that the meaning (or at least the interpretation) of a finite sentence is a proposition. For Davidson (1980) and Parsons (1985), the meaning of a sentence such as opus ran is the proposition resulting from existentially quantifying over the property that is produced under our type assignment. That is, opus ran would be assigned the meaning

\[(61) \, \exists_1(\text{ran(opus))} \].

With our standard definition of the unrestricted existential quantifier \( \exists_1 \), this proposition will true if and only if there is an event e such that ran(opus)(e) is true. Of course, by giving a sentence a meaning which is a property of events, we can uniformly recover the proposition assigned by the systems Davidson and Parsons by simply quantifying as we have done here.

This means that our approach will be somewhat more general than theirs. The reason for this is that we will be able to recover their translations from ours, but they will not, in general, be able to recover the property from the existential quantification. But, with a fine enough individuation of existential quantifications, it would be possible to recover the value of \( P \) from the value of \( \exists_1(P) \). Such is the case for the structured meanings of Cresswell (1985) and in situation semantics (Barwise and Perry 1985, Barwise and Etchemendy 1987). In current versions of situation semantics (Barwise and Etchemendy 1987), a property of events, called the descriptive content, is produced by every sentence utterance. It is also assumed that the interpretation of a sentence is a proposition, where an additional described situation or event has been supplied by context, with the propositional content of an utterance of a sentence being the proposition that the described situation has the property supplied by the descriptive content. Situation semantics identifies two propositions only if they have the same relation and same arguments. This allows the descriptive content of an utterance to be recovered from its propositional content.

Parsons reconciles taking the interpretations of sentences to be propositions
with the view that what is being uncovered by an event-based semantics is the internal structure of the proposition corresponding to a sentence utterance. Having a property of events rather than a proposition for the meaning of a sentence comes in handy for the treatment of adverbial modifiers and other categories that take verbal complements as we will see in the next sections.

4.3.3 Modifiers

We say that any category of the form \(\alpha / \alpha\) or \(\alpha \backslash \alpha\) is a basic modifier. We define what it means for one category to be an applicative result of another category by assuming that any category is an applicative result of itself and that a category \(\gamma\) is an applicative result of \(\alpha / \beta\) or \(\alpha \backslash \beta\) if it is an applicative result of \(\alpha\). We define \(\text{AppRes}(\alpha)\) to be the set of applicative results of \(\alpha\), taking the smallest sets such that

\[\text{(62)} \quad \begin{align*}
    i. \quad & \alpha \in \text{AppRes}(\alpha) \\
    ii. \quad & \text{AppRes}(\alpha) \subseteq \text{AppRes}(\alpha / \beta) \\
    iii. \quad & \text{AppRes}(\alpha) \subseteq \text{AppRes}(\alpha \backslash \beta)
\end{align*}\]

All natural language modifiers will either be basic modifiers or have a basic modifier as an applicative result. In this section, we will be concerned with the case of nominal and verbal modifiers, called adnominals and adverbials, where \(\alpha = n\) or \(\alpha = s \backslash np\), respectively. It is only when we consider modification, that the choice of \(\mathcal{D}_n\) and \(\mathcal{D}_s\) begin to make sense.

Adnominals

We will start with the case of nominal modification. The simplest kind of adnominal is the pre-nominal adjective, such as tall, red, alleged. We call these adjectives pre-nominal, as they typically occur before the noun complex they modify. Since these all receive the syntactic category \(n / n\), they will receive meanings which are functions from properties to properties. A typical lexical entry would look like

\[\text{(63)} \quad (\text{fake}, n / n, \text{fake}).\]
Now consider the semantic behaviour of the three adjectives *red*, *tall* and *alleged*. The first two are what are called *restrictive* modifiers. A mapping $A$ from properties to properties is said to be restrictive if for every property $P$, the extension of $A(P)$ is a subset of the extension of $P$. As a meaning postulate, this becomes

\[(64) \quad \uplus A(P) \subseteq \uplus P. \quad \text{(restrictive).}\]

Non-restrictive modifiers such as *alleged* or *fake* are often called *intensional*. There is no restriction on the extensional behaviour of such a non-restrictive modifier.

The adjective *red* is also an example of what is called a *intersective* adjective. A mapping $A$ from properties to properties is intersective if there is some property $Q_A$ such that

\[(65) \quad \uplus A(P) = \uplus (Q_A \cap P), \quad \text{(intersective)}\]

Note that we are using the generalised conjunction operation over properties defined in Section 3.5. The reason for calling such a modifier intersective should be clear from the fact that an object has the property $A(P)$ if and only if the object has the properties $Q_A$ and $P$. Note that an intersective modifier $A$ is monotonic in that if $P$ and $Q$ are two properties where $\uplus P \subseteq \uplus Q$, then $A(P) \subseteq A(Q)$. This is not true for general restrictive modifiers such as *tall*. Simply consider the fact that even though every ant is an animal, and every tall ant is an ant, it does not follow that a tall ant is a tall animal.

There are now two options if we want to explicitly model the behaviour of intersective modifiers. The simplest is to restrict our attention to models where the meaning of every intersective adjective is in fact intersective. The standard method for doing this is by including meaning postulates or axioms that a model is required to satisfy. In this case, we would have a meaning postulate

\[(66) \quad \uplus A(P) \leftrightarrow \uplus (Q_A \cap P)\]

for some $Q_A \in V_{(i,p)}$, for every modifier $A$ we assume to be intersective. Note that this axiom is weaker than the axiom

\[(67) \quad A(P) \leftrightarrow (Q_A \cap P)\]
This latter axiom places a condition on the intensional behaviour of the properties in question which entails that they will be intersective extensionally.

Meaning postulates have fallen out of favour in linguistic semantics to some extent. This is because they appear to be more descriptive than explanatory. With meaning postulates, arbitrary relationships can simply be written down in logical notation as axioms which models are required to respect. It is usually thought to be more desirable to build these kinds of restrictions into the definition of models. In general, with the models of the typed $\lambda$-calculus we are working with, we will have no other recourse than meaning postulates to capture relationships that we are interested in, such as that between the interpretations of terms such as *bachelor* and *unmarried man*. But, in the case of intersective adnominals, we can get away without meaning postulates if we exploit the additional power of using an intermediate translation to explicitly represent the intersective nature of an adjective such as *red*, by using a lexical entry such as

\[(68) \ (\text{red}, n / n, \lambda P. (\text{red} \cap P))\]

We will simply assume that $\text{red}(x)$ is true if and only if $x$ is red. Of course, $\lambda P. (\text{red} \cap P)$ trivially meets the meaning postulate for intersective adjectives, no matter what property is assigned to the constant *red*, and in fact, meets the stronger condition on propositional equivalence.

Note that *tall* is not intersective. There can be no property $T$ which holds of “tall” things. This is because what it means to be tall will depend on a class of objects which the object in question is being compared with.

The simple post-nominal adnominals, such as *outside*, which have the syntactic category $n \setminus n$ are analysed in exactly the same way as the forward-looking adnominals. It is interesting to note that all of the post-nominal adnominals are intersective.

**Adverbials**

Adverbials are slightly more complicated than their adnominal counterparts. The reason for this is that they modify a more complex category $a = s \setminus np$. A manner adverb like *slowly* can appear before or after a verb phrase, as in
(69)  a. Opus chewed slowly.
      b. Opus slowly chewed.

To handle this fact with our encoding of directionality in our slashes, a manner adverb will be assigned to the two syntactic categories $s \setminus np \setminus (s \setminus np)$ and $s \setminus np / (s \setminus np)$. Of course, the two lexical entries this produces for slowly can have the same meaning, because

(70)  $\theta (s \setminus np \setminus (s \setminus np)) = \theta (s \setminus np / (s \setminus np))$.

The meaning of an adverb is then a function from what is essentially two-place relation between individuals and events to another such relation.

Luckily, the syntactic complexity is greatly reduced in the semantics. We assume the modification works over the meaning which would have been assigned to the sentence without the adverb, and simply carries along the subject argument position to feed to the meaning of the verb. This leads to lexical entries of the general form

(71)  $\langle \text{intentionally}, s \setminus np / (s \setminus np), \lambda V. \lambda x. \text{intentionally}(V(x)) \rangle$

where the constant intentionally is of type $\langle \theta(s), \theta(s) \rangle$, which is $\langle (e, p), (e, p) \rangle$.

Using this general format, the semantic constant used in the lexical entry for an adverb will act as a property modifier in exactly the same way as the constants corresponding to adjectives. This is because

(72)  $D_{\langle (e, p), (e, p) \rangle} \subseteq D_{\langle (i, p), (i, p) \rangle}$.

Consider the expression ran slowly, which is assigned the syntactic category $s \setminus np$, and given the content $\lambda x. \text{slowly}(\text{ran}(x))$, so that john ran slowly receives the meaning slowly(\text{ran}(\text{john})). Note that the subexpression \text{ran}(\text{john}) is of type $\langle e, p \rangle$ which makes it a property of events. We assume slowly(\text{ran}(\text{john}))(e) is a true proposition just in case $e$ is a slow event of John running. Of course, $e$ will be a slow event of John running if \text{ran}(\text{john})(e) is a true proposition.

The meaning of an adverb can be subcategorised as intersective, restrictive or intentional in exactly the same way as was carried out for nominal modifiers. Our intuitive gloss of the value of slowly implied that it was restrictive,
since \( \text{slowly}(P)(e) \) can only be true if \( P(e) \) is. On the other hand, the adverb \textit{intentionally} is intensional. Finally, an adverb like \textit{yesterday} has a meaning which is an intersective modifier, where we can assume that if \( T \) is a property of events, then \( \text{yesterday}(T)(e) \) holds iff \( e \) has the property \( T \) and \( e \) was yesterday. This means that we could use a second constant, say \textit{yest} and assume that

\[
\text{(73) } \text{yesterday}(T)(e) = T(e) \cap \text{yest}(e),
\]

where we take \( \text{yest}(e) \) to be true just in case the event \( e \) occurred yesterday. Note that \textit{yesterday} is only categorised as a post-verbal adverbial in our categorial grammar in Section 2.3.3. This is because it can not show up pre-verbally like a manner adverb. There are also adverbs such as \textit{probably} which only occur pre-verbally and these will also receive only one syntactic category.

\textbf{Prepositions}

Prepositional phrases such as \textit{in the house}, \textit{near Opus} and so on are assigned nominal and verbal modifier syntactic categories of the form \( n \setminus n \) and \( s \setminus np \setminus (s \setminus np) \). Thus they act semantically just like other modifiers. In fact, prepositional phrase meanings will all be intersective. We will not consider the possibly non-intersective nature of prepositional phrases such as \textit{in a dream} which led Montague (1974d) to treat the entire class of prepositions as intensional modifiers.

Prepositions are simply prepositional phrases lacking a complement, and thus of the syntactic form \( \alpha \setminus \alpha \setminus np \) where \( \alpha \) is \( n \) or \( s \setminus np \). Our nominal and verbal modifier entries for the typical preposition \textit{in} are given by

\[
\text{(74) } \langle in, n \setminus np, \lambda y. \lambda P.(P \cap \text{in}(y)) \rangle
\]

and

\[
\text{(75) } \langle in, s \setminus np \setminus (s \setminus np) / np, \lambda y. \lambda V. \lambda x.(V(x) \cap \text{in}(y)) \rangle.
\]

Notice that we have used the same constant \textit{in} for both the nominal and verbal modifier category. This is made possible by taking the domain of events to be a subdomain of the domain of individuals. In fact, we could define the semantics of the verbal entry in terms of the semantics of the nominal entry and conversely. We
interpret \( \text{in}(x)(y) \) to mean that the individual \( y \) is in the location \( x \). Our intuitive understanding of events is that they can be thus located in the same manner as other individuals. Of course, we still require distinct syntactic categories for every preposition that can act both as a verbal and nominal modifier.

4.4 Context and Discourse Referents

In this section we introduce our general theory of contextual information embodied in the relational categorial grammar framework. This will be the final grammar formalism that we introduce.

4.4.1 Relational Meaning

In this section, we modify the naïve semantics that we gave in the last section to incorporate contextual information. From now on, we will consider the meaning of an expression to be a relation between a context and \( \text{interpretation} \). For an expression \( e \), we will write \( \llbracket e \rrbracket \) for its meaning and suppose that \( c \llbracket e \rrbracket i \) if the expression \( e \) can have the interpretation \( i \) in the context \( c \). Thus, we think of the combination of the context \( c \) and the expression \( e \) as representing an utterance. Montague (1974e) represented the meaning of an expression as a function from a set of \( \text{indices} \) onto a set of interpretations. This followed the suggestion of Bar-Hillel's (1954) to define pragmatics as the study of the effect of context on interpretation or reference. This, in turn, follows Quine's (1961) idea to partition semantics into a theory of reference or interpretation and a theory of meaning. Montague's (1974a, 1974b, 1974c) mathematical development of meaning was based on the intensional logic of Kaplan (1964).

We will say that \( e \) can be interpreted as \( i \) if there is some context \( c \) in which \( i \) is the interpretation of \( e \). We intend this definition to range over our ambiguous expressions. Recall that the ambiguous expressions for us are just strings of basic expressions. In certain contexts, we will wish to restrict our attention to the meaning of a disambiguated expression \( d \). We will write \( \llbracket d \rrbracket \) for the meaning of the disambiguated expression \( d \), where our disambiguated expressions correspond
to skeletal parse trees. So, we have

$$\{e\} = \bigcup\{[d] \mid d \sim_a e\},$$

where $\sim_a$ is our usual ambiguating relation.

We can view Montague grammar in this framework with two additional assumptions. First, in Montague grammars the context $c$ is taken to be a so-called index, consisting of a function mapping contextual parameters to their values. The notion of index was introduced by C. S. Peirce in the last century, but was reintroduced by Bar-Hillel (1954) in his definition of pragmatics. Montague himself only considered two indices for the time and possible world at which the expression was uttered. Lewis (1970) extended the set of indices to include things such as the speaker, hearer and so on. In Montague's grammars, the effect of contextual information, such as the time and place of an utterance, is not separated from the distinction between sense and reference which comes into play in the distinction in meaning between groundhog and woodchuck. Kaplan (1977, 1979), in his theory of demonstratives, introduced the idea of a two-stage process, wherein the expression and context together determine a sense, which was essentially a proposition in the case of sentences. We follow the most general approach in allowing the context to be arbitrarily structured and allowing it to only determine the relevant sense of an expression rather than its reference.

Montague also required $[d]$ to be a total function from contexts to interpretations for any disambiguated expression $d$, so that fixing a context and expression will determine a unique interpretation. We will not make the assumption that $[d]$ is total or even that it is a function for a disambiguated expression $d$. Thus, we can think of an utterance as determining a (possibly empty) set of interpretations of the utterance. The generalisation to a relational, as opposed to a functional theory of context is due to Barwise and Perry (1981, 1983, 1985).

Another way in which our grammar will depart from traditional Montague grammar is in allowing additional structured objects into our context. Cresswell (1973) introduced the idea of allowing the contextual information to be arbitrarily structured in terms of properties. Roughly in line with situation semantics (Barwise and Perry 1983, 1985), we suppose that there are really two components
to context. The first contains the so-called **presuppositions** or **background conditions** of the utterance. The presuppositions of an utterance are taken to be the conditions that are presupposed by the speaker independently of the truth of the utterance. That is, the presuppositions of a sentence must be met if it is to have any interpretation at all. This definition of presupposition is due to Strawson (1950, 1952) in his analysis of Russell’s (1905) theory of definite determiners. We will analyse the presuppositional content of an utterance as a proposition that we will require to be true. In situation semantics (Barwise and Perry 1983), these background conditions are taken to be part of the so-called **utterance situation** and are modeled as a situation.

Besides the background conditions, the context will also be assumed to supply a collection of so-called **discourse referents** which are individuals in IND. Our discourse referents play the same role as the connections in situation semantics (Barwise and Perry 1983), the discourse referents of discourse representation theory (Kamp 1984), the file cards of Heim’s (1982) file card semantics or the pegs of Landman’s (1986) data semantics. That is, they are used to keep track of the objects that speakers and hearers assume that they are conversing about. Following Strawson (1950), they are called referents because the objects will have to exist in the world for the utterance to be interpretable. If an utterance’s presuppositions are not met then it will have an empty set of interpretations.

### 4.4.2 Relational Categorial Grammar

In this section, we will introduce the basics of yet another grammar formalism, **relational categorial grammar**, which we present in the form of a unification grammar with the additional operations of string concatenation and set union.

First off, we assume our usual set

\[(77) \text{BasCat} = \{s, n, np\}\]

of syntactic categories. Furthermore, we take the typing function $\theta$ from our naïve semantics, with
We next define the collection $\text{RELCAT}$ of relational categories to be terms of the form $(E, C, S, R, B)$, which we write

$\text{(79)}$ 

\begin{align*}
\text{EXP: } & E \\
\text{SYN: } & C \\
\text{CON: } & S \\
\text{REF: } & R \\
\text{BG: } & B
\end{align*}

where

$\text{(80)}$ 

\begin{enumerate}
  \item $E$ is an expression
  \item $C$ is an element of $\text{CAT(} \text{BAS CAT})$, the \textit{syntactic category}
  \item $S$ is the encoding of a $\lambda$-term of the appropriate type for $C$, the \textit{content}
  \item $R$ is a sequence of individual variables from $\mathcal{V}_i$, the \textit{reference markers}
  \item $B$ is a set of propositions, the \textit{background conditions or presuppositions}
\end{enumerate}

What we mean by saying that $S$ is of the appropriate type for $C$ is that it is the image of some term of type $\theta(C)$ under our second encoding $\psi$ of $\lambda$-terms by first-order terms from Section 4.2.2. We further require all of the free variables in the content $S$ and background conditions $B$ to be reference markers in $R$. Intuitively, the propositions in $B$ will be taken to restrict possibilities for values of the reference markers, while $S$ will determine an interpretation once the reference markers are assigned referents.

The expression $E$ is not really necessary, but will make our presentation go more smoothly. We simply take $E$ to be the expression of the category given. With such an expression in place, we can take a \textit{lexical assignment} to be a simple set of categories $\Lambda \subseteq \text{RELCAT}$, since each category determines its own expression.
This definition of category follows the definition of sign in head-driven phrase structure grammar (Pollard 1985, 1988, Pollard and Sag 1987) and unification categorial grammar (Zeevat, Klein and Calder 1987).

We use the following application rule schemata

\[
\begin{align*}
(81) & \quad \text{Exp: } E_1 E_2 \quad \rightarrow \quad \text{Exp: } E_1 & \quad \text{Exp: } E_2 \\
& \quad \text{Syn: } C_1 & \quad \text{Syn: } C_1 / C_2 & \quad \text{Syn: } C_2 \\
& \quad \text{Con: } F & \quad \text{Con: } X.F & \quad \text{Con: } X \\
& \quad \text{Ref: } R_1 \cdot R_2 & \quad \text{Ref: } R_1 & \quad \text{Ref: } R_2 \\
& \quad \text{BG: } B_1 \cup B_2 & \quad \text{BG: } B_1 & \quad \text{BG: } B_2
\end{align*}
\]

where \( E_1 E_2 \) is the concatenation of the expressions \( E_1 \) and \( E_2 \), \( R_1 \cdot R_2 \) is the concatenation of the lists \( R_1 \) and \( R_2 \), and \( B_1 \cup B_2 \) is the union of the sets \( B_1 \) and \( B_2 \). Notice that this matches our second system of unification categorial grammar rules from Section 4.2.2, which explains why we took our contents \( S \) to be images of our second encoding \( \psi \). We can then provide a definition of the set \( G \) of categories admitted by the grammar, by taking the least set such that

\[
\begin{align*}
(83) & \quad i. \ A \subseteq G \\
& \quad ii. \ \gamma_0 [\varphi] \in G \text{ if } \gamma_1, \gamma_2 \in G, \gamma_0 \rightarrow \gamma_1 \gamma_2 \text{ and } \varphi \text{ is the most general substitution unifying } \gamma_1 \text{ with } \gamma_1' \text{ and } \gamma_2 \text{ with } \gamma_2'
\end{align*}
\]

where we extend our notions of unification and substitution pointwise on our relational categories. Quite significantly, the assumption that we are working in our second system of unification grammars will be enough to insure that we can determine the possibility of an expression falling into a particular syntactic category by inspecting the syntactic components of its lexical entries. This is just saying that it will never be possible for categories matching the syntactic part of the application rules to fail to match the rest of the rule. Of course, the content is the only other thing that is forced to unify, since the reference list and expression are concatenated and the background conditions are unioned. This observation
means that parsing can take place using standard categorial grammar parsing algorithms, as in the restriction based systems introduced by Shieber (1985b).

We define the meaning $[E]_C$ of the expression $E$ when given category $C$ to be that relation such that

$$ (84) \quad B, R[E]_{C}I \iff \langle E, C, S, R, B \rangle \in G. $$

We can remove the relativisation to a syntactic category and write $B, R[E]I$ if $B, R[E]_{C}I$ for some $C \in \text{CAT}$. Notice that we have snuck in the assumption that the context of an utterance is a pair consisting of background conditions and a list of individuals, thus allowing us to think of $[E]$ as a three-place relation.

Definitions of this sort sometimes use an assignment function $\phi$ mapping variables to objects in place of our reference list as a component of context. We do not choose to do this because it would commit us to the fact that the meaning of an expression is a relation between assignment functions, background conditions and interpretations. Since we are trying to maintain a realistic approach to semantics, and since the variables we use as placeholders are not real objects in the metaphysical sense, we would not want our meanings to be relations which take such assignments as arguments. But, it is often argued by proponents of situation semantics (Barwise and Perry 1983) and discourse representation theory (Kamp 1984) that such an assignment function should be considered as real, and the variables that we used in our reference list should be considered to be real objects. Their reality would stem from the fact that they are simply uniformities over the placeholders in actual discourses that go on in the world. Of course, since a theory involving substitutions can be transformed into one using reference lists and conversely, nothing important hinges on our decision.

4.4.3 Utterance Events

An important part of the analysis of context is that of the utterance situation. For an utterance of the expression $\delta_1 \cdots \delta_n$, we will always assume that the reference list contains an element $E_i$ and $T_i$ for $1 \leq i \leq n$ where $E_i$ is meant to pick out the event of the speaker uttering the expression $\delta_i$ to the hearer at time $T_i$. We thus suppose that we have elements of the reference list for the speaker and hearer as

197
well, say $S_i$ and $H_i$. To insure that these conditions hold, we will assume that there is a proposition

$$\text{say}(\delta_i)(S_i)(H_i)(E_i)(T_i)$$

in the background conditions for $1 \leq i \leq n$, where $\text{say}$ has the obvious value in the model. But this is not the only condition on the utterance events. We also include propositions of the form

$$\text{before}(T_i)(T_{i+1})$$

for $1 \leq i < n$, since $\delta_i$ must be uttered before $\delta_{i+1}$ in an utterance of $\delta_1 \cdots \delta_n$. Furthermore, we require that the utterance of the smaller expressions are subevents of an utterance of a larger expression, so that

$$E_{\delta_1 \cdots \delta_n} \Downarrow E_{\delta_i}$$

is also included among the background conditions.

We could continue in this vein, but this gives us all that we need for what we do later. In fact, things are even worse than they may appear, since the number of ways a complex expression can be broken down in terms of its subexpressions is an exponential function of the length of the expression.

Working with such long reference lists and large sets of background conditions is tedious, to say the least. So, we will follow the lead of situation semantics and not explicitly include these referents and conditions in the categories we write down, but simply suppose that they are there along with their background conditions. We will use $E_{\delta}, T_{\delta}, S_{\delta}$ and $H_{\delta}$ for the referents introduced by an utterance of $\delta$. This move should not be seen as introducing some new kind of object into the theory, such as the roles or restricted parameters which do the same duty in situation semantics (Barwise and Perry 1983, 1985 and Barwise 1984).

There is another complication stemming from the fact that an expression $\delta$ may occur twice as a subexpression of a larger expression. In this case, we want to keep the two token expressions distinct. While this is not a problem with the full lists, since the distinct tokens will be uttered at different times, our simple abbreviatory convention hides this possibility, and is thus not truly faithful to our
definitions. Fortunately, this does not cause any problems in practice, since we will simply mark different occurrences of an expression with subscripts when necessary, again being sure not to interpret this as being some additional component of the theory, but merely a notational convenience.

4.5 English Relational Categorial Grammar

In this section we present our primary grammar fragment in the relational categorial grammar formalism introduced in the previous section.

4.5.1 Simple Noun Phrases

Indexicals

Consider the case of *indexical* expressions such as *i, you, now* and so on. The full lexical category for *i* would really look like

(88) \[
\begin{align*}
\text{EXP}: & \ i \\
\text{SYN}: & \ np \\
\text{CON}: & \ X \\
\text{REF}: & \ \langle X, T, E, H \rangle \\
\text{BG}: & \ \{ \text{say}(i)(X)(H)(T)(E) \}
\end{align*}
\]

With our abbreviatory conventions, this will be written as

(89) \[
\begin{align*}
\text{EXP}: & \ i \\
\text{SYN}: & \ np \\
\text{CON}: & \ S_i \\
\text{REF}: & \ \{S_i\} \\
\text{BG}: & \ \{\}
\end{align*}
\]

This means that the interpretation of an utterance of the expression *i* in some context must be the individual who actually uttered the expression in that context.

Exactly the same analysis will be used for the other indexicals, with *you* picking out the hearer and *now* picking out the time of utterance in its nominal lexical entry.
Names

Proper names such as *opus* are lexically assigned categories of the form

\[(90) \quad \text{EXP: } \text{opus} \]
\[\text{SYN: } np \]
\[\text{CON: } X \]
\[\text{REF: } \{X\} \]
\[\text{BG: } \{ \text{name(opus)}(X) \} \]

where *name* is of type \(\langle i, \langle i, p \rangle \rangle\). We will assume that \(\text{name}(n)(x)\) is true if and only if the individual \(x\) is named \(n\). It should be noted that \(n\) here is an expression, not some other kind of individual. Since we have assumed that abstract objects such as names can be individuals, we include such linguistic objects in our ontology. As we have set things up, a proper name will only receive interpretations which are individuals having that name.

Demonstratives

The so-called *demonstratives* such as *this*, *that* and *those*, whose meanings are picked out by context will get lexical entries such as the following for demonstrative *this*

\[(91) \quad \text{EXP: } this \]
\[\text{SYN: } np \]
\[\text{CON: } X \]
\[\text{REF: } \{X\} \]
\[\text{BG: } \{ \text{demonst}(E_{this})(X) \} \]

where we will not specify what exactly it means for an utterance event to demonstrate an individual, but simply assume that there is some kind of relationship *demonst* between the utterance situation for a demonstrative and the individual who is picked out. Note that if it is found to be necessary, we could add arguments for the speaker and hearer to the demonstrative background condition.

Deictic Pronouns

A pronoun such as *he*, *she*, *it* can be used in two ways. The simplest use of a pronoun is its *deictic* use, and we will postpone the complications of the anaphoric
uses of pronouns until Section 5.2. A pronoun used *deictically* is much like an impoverished name, with the following category for deictic he

(92) \begin{align*}
\text{Exp: } & \text{he} \\
\text{Syn: } & np \\
\text{Con: } & X \\
\text{Ref: } & \langle X \rangle \\
\text{BG: } & \{ \text{male}(X) \}
\end{align*}

With this category, the deictic utterance of the pronoun he could be interpreted as any individual which happens to be male.

4.5.2 Common Nouns

Common nouns such as *dog, penguin, kid* and so on will get very simple lexical entries of the form

(93) \begin{align*}
\text{Exp: } & \text{dog} \\
\text{Syn: } & n \\
\text{Con: } & X.\text{dog}(X) \\
\text{Ref: } & \langle \rangle \\
\text{BG: } & \langle \rangle
\end{align*}

where *dog* is the property of being a dog. Note that for our constant dog of type \(\langle i, p \rangle\) we have

(94) \[ \psi(\text{dog}) = X.\text{dog}(X) \]

under our canonical encoding \(\psi\) of \(\lambda\)-terms. A simple common noun does not, by itself, introduce any referents or any background conditions which need to be met. It simply contributes an interpretation which is a property.

4.5.3 Referential Determiners

The two so-called *referential determiners* *the* and a look for a noun category to their right to produce a category which looks very much like those given to simple noun phrases such as names and deictic determiners. The indefinite determiner *a* has the lexical entry
The background condition will assure us that the referent $X$ is interpreted as an object which has the property $X.P$ given by the nominal argument. Thus, an utterance of a *penguin* can have all and only the penguins as possible interpretations.

Similarly, the definite determiner *the* is assigned the lexical category

(96) \[
\begin{align*}
\text{EXP: } & \text{the} \\
\text{SYN: } & np/n \\
\text{CON: } & (X.P).Y \\
\text{REF: } & (Y) \\
\text{BG: } & \{ \text{the}(X.P)(Y) \}
\end{align*}
\]

where the is just as in the naïve fragment with $\text{the}(P)(x)$ holding just in case $x$ is the unique individual having the property $P$. This means that the difference between the definite and indefinite is simply a uniqueness condition, since $P(x)$ is true whenever $\text{the}(P)(x)$ is according to our definition of the. With our treatment of background conditions, we are left with a treatment of definite determiners identical to that of Strawson (1950).

An interpretation of an utterance of *the kid* will thus be unique if one exists, while there may be many different interpretations of an utterance of a *kid*. Both determiners require the interpretation to be an individual which has a property specified by the interpretation of the noun supplied as the argument to the determiner interpretation.

### 4.5.4 Resource Situations and Speaker Intentions

The question of speaker intentions arises most naturally in the case of the referential noun phrases we have just introduced. Consider a deictic use of the pronoun *he*. We might be tempted to include a proposition such as $\text{referring}(S_{he})(X)$ in the background conditions to account for the fact that the speaker $S_{he}$ usually
has a particular individual in mind for the interpretation of an utterance of he. Such a proposition could be included in the background conditions of the lexical category for proper names and demonstratives, as well as deictic pronouns.

We prefer to separate out speaker intentions from the context of utterance. Such intentions are not available to the hearer and could not play a direct role in the hearer's construction of a possible interpretation. Rather, we think of speaker intentions as playing a part in the speaker's choice of expression to utter. Of course, a listener trying to discover the intended interpretation among the possible interpretations of an utterance will employ facts concerning the intentions and knowledge of the speaker. Again, nothing really crucial hinges on this decision in the theory, and such reference facts could be uniformly included along with the other background conditions. If this is done, it becomes evident how often speaker intentions could be included in the meaning of an expression, so that they would have to be treated like the utterance situation, utterance time and other items that automatically get included with the utterance of every expression.

In situation semantics, the context of an utterance of a singular noun phrase must always supply a so-called resource situation, whose purpose is to further restrict the possibilities for reference (Barwise and Perry 1983). The reason for this is that sentences such as

(97)  a. The student worked hard.
       b. Every student worked hard.

when uttered, will not contribute the simple property of studenthood, but will have some additional restriction in mind, such as students in the class of the speaker. Thus the first sentence could be uttered truthfully even if there was not a unique student, and similarly, the second could be uttered without every student having worked hard, but only a particular subset of all of the students.

In our framework, we will assume that a restriction can be placed on the interpretation of any noun by means of a resource property, so that the lexical entry for penguin would be
The category for a penguin would then be

\[
\text{Exp: penguin} \\
\text{SYN: n} \\
\text{CON: } X.(\text{penguin}(X) \cap R(X)) \\
\text{REF: } \langle R_{\text{the}} \rangle \\
\text{BG: } \{ \text{resource}(E_{\text{penguin}})(R_{\text{penguin}}) \}
\]

where we interpret \text{resource}(E)(R) to mean that the utterance event \( E \) supplies the resource property \( R \) which is then used as a restriction on the property supplied by the noun, which when used in the noun phrase, restricts the interpretations to those with the property \( R \). Unfortunately, we do not have anything more to say about the relation \text{resource}, so we leave open the question of how a resource restriction is actually supplied by an utterance event. In fact, it may be that the resource constant is just playing the same role as referring would, and would be better left out of the background conditions.

Since resource restrictions come along for the ride with every noun, we will again simply assume that they are there and not explicitly represent their presence unless the resource interacts with some other aspect of meaning. Also, this may only be the tip of the iceberg in that every expression may require some kind of contextual restriction to capture its intended rather than literal meaning. In this case, some sort of general mechanism seems necessary, which may not even play a role at this level of analysis. For discussion of this issue in the context of situation semantics, see Levinson (1988).

### 4.5.5 Simple Verbs

In this section, we provide a more in-depth analysis of the syntactic and semantic structure of the verb phrase. As a start, we will assume that our basic category \( s \)
is subcategorised into the five subcategories

\[ s(bse), \ s(fin), \ s(perf), \ s(pred), \ s(inf) \]

with the following basic expressions

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s(bse) \ \text{\textbackslash{} np} )</td>
<td>sing, eat, talk, perform, sneeze</td>
</tr>
<tr>
<td>( s(fin) \ \text{\textbackslash{} np} )</td>
<td>sang, ate, talked, sneezed, performed</td>
</tr>
<tr>
<td>( s(perf) \ \text{\textbackslash{} np} )</td>
<td>sung, eaten, talked, sneezed, performed</td>
</tr>
<tr>
<td>( s(pred) \ \text{\textbackslash{} np} )</td>
<td>singing, eating, talking, sneezing, performing</td>
</tr>
</tbody>
</table>

with expressions such as \textit{to eat} and \textit{to sing} being of category \( s(inf) \ \text{\textbackslash{} np} \). Instead of treating infinitives like \textit{to sing} and \textit{to eat} as basic expressions, we adopt the treatment of generalized phrase structure grammar (Gazdar, Pullum and Sag 1982) as will be seen in Section 4.5.6.

We will furthermore assume that

\[ \theta(s(v)) = \theta(s) = (e, p) \]

for every \( v \in VFORM = \{ bse, fin, perf, pred, inf \} \). In particular, we do not assume that the finite forms are of a different arity than the base forms, which might be assumed to leave an argument position in non-finite forms for the time at which the event is presumed to have taken place. This will simply not be necessary on our approach, since we can determine the time of an event uniquely from the event itself with our temporal trace function. Consequently, we are spared the trouble of introducing multiple categorisations for things like adverbials, since adverbials can take as arguments verbs of any of the five verb forms.

To carry out this analysis within our unification format, we take \( s \) to be a one-place function symbol in \( \Upsilon_1 \) and each of the verb forms to be constant symbols in \( \Upsilon_0 \).

Notice that there is no additional verb form marking on verb phrases. The verb form of a verb phrase is determined by inspecting the final applicative result. In the rest of this section we will detail each of the basic verb forms.
Base Form

Verbs are entered into the lexicon in their base or uninflected forms. While we will not consider the effect inflection has on either orthographic or phonological form, we will see that the meanings of all of the other forms of a verb can be determined in a uniform way from that of the base form.

The base form entry for a simple intransitive verb like *sing* is

(103) \[ \text{Exp: } \text{sing} \]
\[ \text{SYN: } s(bse) \setminus np \]
\[ \text{CON: } X.E.\text{sing}(X)(E) \]
\[ \text{REF: } () \]
\[ \text{BG: } {} \]

where we interpret sing\(x)(e)\) to mean that \(e\) is an event of \(x\) singing. Notice that there is no mention of time in the base entry for a verb.

Similarly, a transitive verb like *love* and ditransitive verb like *give* get the following lexical entries in their base forms

(104) \[ \text{Exp: } \text{love} \]
\[ \text{SYN: } s(bse) \setminus np/np \]
\[ \text{CON: } Y.X.E.\text{love}(Y)(X)(E) \]
\[ \text{REF: } () \]
\[ \text{BG: } {} \]

(105) \[ \text{Exp: } \text{give} \]
\[ \text{SYN: } s(bse) \setminus np/np/np \]
\[ \text{CON: } Z.Y.X.E.\text{give}(Z)(Y)(X)(E) \]
\[ \text{REF: } () \]
\[ \text{BG: } {} \]

Finite Form

There are two finite forms for each singular verb, the present and the past. In its present finite form, the verb *sing* has the category

(106) \[ \text{Exp: } \text{sang} \]
\[ \text{SYN: } s(fin) \setminus np \]
\[ \text{CON: } X.E.(\text{sing}(X)(E) \cap \text{at}(E)(T)) \]
\[ \text{REF: } (T) \]
\[ \text{BG: } \{ \text{before}(T)(T_{sang}) \} \]
We assume that
\[(107) \forall t \forall (e) (at(t)(e) \leftrightarrow \tau(e) = t)\]
so that \(at(t)(e)\) is true just in case the temporal trace of \(e\), which we think of as the time at which \(e\) occurred, is \(t\). For \textbf{before} and its relatives \textbf{during} and \textbf{after}, we have the following meaning postulates
\[(108) \qquad \begin{array}{l}
i. \quad \forall t \forall (t') \quad \text{before}(t)(t') \leftrightarrow (t < t') \\
    \quad \text{ii.} \quad \forall t \forall (t') \quad \text{during}(t)(t') \leftrightarrow (t \leq t') \\
    \quad \text{iii.} \quad \forall t \forall (t') \quad \text{after}(t)(t') \leftrightarrow (t > t')
\end{array}\]
so that they correspond to \(t\) being before \(t'\), \(t\) overlapping \(t'\) and \(t\) being after \(t'\).

This leads to the following entry for \textit{sings}, which is the singular present tense form of the verb \textit{sing},
\[(109) \quad \begin{array}{l}
\text{Exp: } \textit{sings} \\
\text{Syn: } s(fin) \setminus np \\
\text{Con: } X.E.(\text{sing}(X)(E) \cap \text{at}(E)(T)) \\
\text{Ref: } \langle T \rangle \\
\text{Bg: } \{ \text{during}(T)(T_{sings}) \}
\end{array}\]

It is rare to describe an event which is presently happening, so the present tense is not very common. Since sports announcers typically describe the events that are currently going on around them, this is why the present is often referred to as the “sportscaster” tense.

These entries for finite verbs allow us to analyse a finite sentence such as \textit{opus ran} as having the category
\[(110) \quad \begin{array}{l}
\text{Exp: } \textit{opus ran} \\
\text{Syn: } s(fin) \\
\text{Con: } E.(\text{run}(X)(E) \cap \text{at}(T)(E)) \\
\text{Ref: } \langle X, T \rangle \\
\text{Bg: } \{ \text{name}(opus)(X) \}
\end{array}\]

This means that the property \(P\) of events can be an interpretation of an utterance of the sentence \textit{opus ran} just in case \(P\) is a property of someone named Opus running at some time which is before the time of the utterance.
We will say that an utterance of a finite sentence can be interpreted *truthfully* in the context $c$ if it has an interpretation $P$ in $c$ as a property of events such that $\exists_1(P)$ is a true proposition. This means that an utterance of a sentence $e$ can be truthful just in case it can have the interpretation $P$ as a property of events, and there is an actual event in the world that has the property, since $\exists_1(P)$ is true just in case there is an event $e$ such that $P(e)$ is true. We will say that a sentence can be true if it has a truthful interpretation in some context.

It is interesting to note that $\exists_1(P)$ is what Barwise and Etchennedy called a *Russellian proposition*. Most of their book is devoted to arguing against the Russellian analysis of sentence interpretations. Their preferred alternative is to interpret a sentence as what they call an *Austinian proposition*. An Austinian proposition is of the form $P(e)$ for some real $e$, which is determined by the context and a property of events $P$. Note that $P(e)$ is what was called the propositional content of an utterance in Section 4.3.2. Of course, our type assignment allows the direct recovery of the Russellian proposition corresponding to a sentence utterance. With the simple addition of another discourse referent, we could just as easily represent the Austinian proposition corresponding to a sentence utterance.

**Perfect Form**

We assume the following lexical category for the expression *sung*, which is the perfect form of the base form *sing*

\[(111)\]

\[
\begin{align*}
\text{Exp: } & \text{sung} \\
\text{Syn: } & s(\text{perf}) \setminus np \\
\text{Con: } & E2.X.\text{perf}(E.\text{sing}(X)(E))(E2) \\
\text{Ref: } & \emptyset \\
\text{Bg: } & \emptyset
\end{align*}
\]

We interpret the constant \text{perf} in such a way as to insure that $\text{perf}(P)(e)$ is true just in case $e$ is an event corresponding to the completion of an event which has the property $P$. We can assume, following Kamp (1979), that such an $e$ will correspond to the state of the events having been completed, and will thus be located after some event with the property $P$, so that we would have the axiom

\[(112)\] $\text{perf}(P)(e) \supset \exists_1(P \cap \text{after}(e))$.
With this constraint, an event $e$ can only have the property $\text{perf}(P)$ if it is after some event with the property $P$, thus enforcing the constraint of Reichenbach (1947) on the interpretation of the time of a perfective. See Moens (1987) for a more detailed treatment of the actual value of $\text{perf}$ compatible with the semantics presented here.

**Progressive Form**

We assume the following lexical category for the expression *singing*, which is the progressive form of the base verb *sing*

\[
\begin{align*}
\text{EXP: } & \text{singing} \\
\text{SYN: } & s(\text{pred}) \setminus np \\
\text{CON: } & E2.X.\text{prog}(E.\text{sing}(X)(E))(E2) \\
\text{REF: } & () \\
\text{BG: } & \{
\end{align*}
\]

We interpret the constant $\text{prog}$ to be such that $\text{prog}(P)(e)$ is true just in case $e$ is an event corresponding to the progression of an event which has property $P$. Like our analysis of the perfect, this will hardly be satisfactory until it is understood exactly what it means for an event to be a “progression”. While intentionality may seem to play a role, consider

\[
\begin{align*}
\text{a. } & \text{John was crossing the street when he was run over.} \\
\text{b. } & \text{The ball was falling off the roof when it was caught.}
\end{align*}
\]

where the ball may be falling off the roof without even being an agent that could have intentions. There is a vast literature on the semantics of the progressive, references to which can be found in Dowty (1979), Vlach (1981) and Moens (1987).

While we have only given the inflected form of the intransitive verbs, all of the other verbs will inflect in exactly the same manner. For details of an inflectional and lexical system compatible with that presented here, see Carpenter (1987).

**4.5.6 Auxiliary Verbs**

Auxiliary verbs are the so-called *helping* verbs, that take verb phrase complements, such as *be, is, have, do, will* and so on. In a theory of the auxiliary, we would
like to be able to account for the ordering of auxiliaries, so that they occur in the right order before the verb. Auxiliary sequences such as will have been eating are not at all uncommon, and can only be well formed with this exact ordering.

In a categorial framework, this pattern is straightforward to capture. We will take an auxiliary to syntactically map verb phrases of one form into verb phrases of another form. They will then have to be sequenced in an order in which the subexpressions have the correct categories for the application rule to be applied. This is basically the treatment found in Bach's (1983b) extended categorial grammar. It is also similar to the analysis of generalized phrase structure grammar (Gazdar, Pullum and Sag 1982, Gazdar et al. 1985), which involves complex feature structures and specific phrase-structure rules of the form

\[(115)\text{VP[INF, +AUX]} \rightarrow H[12], \text{VP[BSE]}.\]

This rule says that an infinitive verb phrase can be rewritten as the verb to which is the only expression of category H[12] and a base form verb (Gazdar et al. 1985). Separate linear precedence principles are then employed to make sure each auxiliary occurs before its verb phrase. Of course, under this analysis, we need a different phrase-structure rule for each type of auxiliary. Instead, we will have a different lexical entry for each of the base form auxiliaries. The binary analysis of the auxiliary sequence is due to Ross (1967).

**Copula**

The copula be is the prototypical auxiliary, and has a lexical entry

\[(116)\text{EXP: be} \]

\[\text{SYN: } s(bse) \mathbin{\unicode{x7c}} np / (s(pred) \mathbin{\unicode{x7c}} np)\]

\[\text{CON: } (X.E.P).X.E.P\]

\[\text{REF: } \{\}\]

\[\text{BG: } \{\}\]

which has the identity function as content and a category looking for a progressive verb phrase to its right to produce a base form verb phrase as a result. This means that be singing will get the category
which only differs from the category of *singing* in its verb form.

The copula can be inflected for verb form just like any other verb, yielding the following category for the past tense *was*

(118) **EXP:** was  
**SYN:** s(fin) \ np \ (s(pred) \ np)  
**CON:** (X.E.P).X.E.(P \at\ (E)(T))  
**REF:** (T)  
**BG:** \{ before(T)(Twas) \}

With this category, the finite sentence *opus was singing* receives the category

(119) **EXP:** opus was singing  
**SYN:** s(fin)  
**CON:** E2.(prog(E.sing(X)(E))(E2) \at\ (E2)(T))  
**REF:** (X,T)  
**BG:** \{ name(opus)(X) \ 
\{ before(T)(Twas) \} \}

We interpret this content as requiring *E2* to be an event corresponding to a particular progression of an Opus singing event which occurred before the time of utterance of the sentence. A similar entry will be included for the present tense copula *is*, which is just like that for the past tense, only with *at* in place of *before*.

Similarly, the copula takes its own perfective and predicative forms, so that we have the following category for the perfect copula *been*

(120) **EXP:** been  
**SYN:** s(perf) \ np \ (s(pred) \ np)  
**CON:** (X.E.P).X.E2.perf(E.P)(E2)  
**REF:** ()  
**BG:** ()

with an entry for *being* which only differs from the perfect in that it has a final applicative result category *s(pred)* and the constant *prog* in place of *perf*.  

211
Base Auxiliary

The two auxiliaries does and did do not have any base forms. Their only function is to map a base form verb phrase onto a finite verb phrase. The intuitive motivation is that a compound verb phrase such as does run should have the same possible interpretations as runs and did run should behave like ran. The base auxiliary comes in handy when we need to modify an event description which does not come with a temporal location. This is useful for negation (see below), among other things, and is simply not possible with a finite verb which is entered in the lexicon in its tensed form. The lexical entry for does is

(121) EXP: does  
SYN: \(s(fin)\backslash np/(s(bse)\backslash np)\)  
CON: \((X.E.P).X.E.(P \cap at(E)(T))\)  
REF: \(T\)  
BG: \{ during(T)(T_{does}) \}

with a similar entry for did with before in place of during, as usual. It can then be verified that the category for does sing is almost identical to that of sings, with the minimal difference being the expression which introduces the utterance time.

Perfect Auxiliary

The various forms of have all take arguments which are perfective verb phrases. The lexical category of the base form have is

(122) EXP: have  
SYN: \(s(bse)\backslash np/(s(perf)\backslash np)\)  
CON: \((X.E.P).X.E.P\)  
REF: \(\{}\)  
BG: \{\}

which only differs from be in the syntactic category of its argument. This base form entry can then be used to determine the past form had, present form has, and finally the progressive having, while there is no perfect form of had.

Modal Auxiliaries

The so-called modal auxiliaries all have the syntactic category of a finite verb phrase subcategorised for a base form verb phrase. The semantic effect of a modal
auxiliary can be more wide-ranging than the other auxiliaries, as it can be used to do more than just introduce tense and change syntactic category. The simple modal auxiliary *will* acts as a future tense marker in the same way as *did* acts as a past tense marker. An auxiliary is necessary to mark an event as occurring in the future, since English does not inflect verbs for future tense. The lexical entry for *will* is then

\[(123) \text{EXP: will} \]
\[
\text{SYN: } s(fin) \backslash np / (s(bse) \backslash np) \\
\text{CON: } (X.E.P).X.E.(P \cap at(E)(T)) \\
\text{REF: } (T) \\
\text{BG: } \{ \text{after}(T)(T_{will}) \}
\]

which is quite similar to the entry for *does*. But consider a modal auxiliary like *should* which actually has modal content and requires the lexical entry

\[(124) \text{EXP: should} \]
\[
\text{SYN: } s(fin) \backslash np / (s(bse) \backslash np) \\
\text{CON: } (X.E.P).X.E2.(should(X)(E.P)(E2) \cap at(E2)(T)) \\
\text{REF: } (T) \\
\text{BG: } \{ \text{during}(T)(T_{should}) \}
\]

where we assume *should(x)(P)(e)* is true just in case the individual *e* is the stative eventuality corresponding to the fact that *x* should bring about an event which has the property *P*. Other modal auxiliaries such as *might, must, may, can* and so on will all be analysed along the same line as *should*, only varying in the semantic constant used in place of *should*.

**Infinitve Auxiliary**

As promised, we follow generalized phrase structure grammar in assuming that the expression *to* can act as an auxiliary turning base form verb phrases into infinitive form verb phrases. Since we assume that *to* does not add anything to the semantics of its argument, we get a lexical entry for *to* similar to that for *be*, given by
(125) \[\begin{align*}
\text{EXP: } & \text{to} \\
\text{SYN: } & s(\text{inf}) \setminus np / (s(bse) \setminus np) \\
\text{CON: } & (X.E.P).X.E.P \\
\text{REF: } & \emptyset \\
\text{BG: } & \emptyset
\end{align*}\]

The significance of this lexical entry will only become apparent when we consider complex verbs which themselves take infinitive verbs as their arguments.

Negation

We include the negation operator along with the auxiliaries for lack of a better place to present it. It differs from other auxiliaries in not changing the verb form of its argument, and in this way patterns like a modifier. On the other hand, adverbial modifiers can attach to verbs of any form, including finite form verbs, whereas \textit{not} only occurs before a non-finite verb. Thus we have a lexical entry

(126) \[\begin{align*}
\text{EXP: } & \text{not} \\
\text{SYN: } & s(v) \setminus np / (s(v) \setminus np) \\
\text{CON: } & (X.E.P).X.E2.\text{not}(E.P)(E2) \\
\text{REF: } & \emptyset \\
\text{BG: } & \emptyset
\end{align*}\]

for each \(v \in \{\text{bse, perf, pred, inf}\}\). Note that this is really four different lexical entries, one for each of the verb forms except the finite form. This is exactly the situation in which it would be helpful to have disjunctive feature specifications, so that the verb form of \textit{not} could be rendered as

(127) \(\text{bse} \sqcap \text{perf} \sqcap \text{pred} \sqcap \text{inf}\)

which would be shared by both the argument and result, to insure that they were the same. But this is the only place in which we would have to resort to such disjunctive features in our presentation, and it is easier to just assume four distinct lexical entries. Nothing theoretically significant hinges on the distinction, though processing will be influenced depending on the representation used (see Kasper (1987)). With this entry for \textit{not}, we can analyse \textit{opus did not sing} as having the category
(128) EXP: opus did not sing  
SYN: s(fin)  
CON: C2.(not(E.sing(X))(E))(E2) ∧ at(E2)(T))  
REF: (X)  
BG: \{ \begin{align*}  & \text{name(opus)(X)} \\ & \text{before(T)(T\_did)} \end{align*} \}

There are a number of possible options for the value of not which reflect varying intuitions as to the function of negation in an event based semantics. The obvious candidate would have not(P)(e) be true of any event e which does not have the property P. This would lead to the axiom

(129) \( \mathcal{U} \text{not}(P)(e) \iff \neg(\mathcal{U}P(e)) \)

governing the behaviour of not. The interpretation of an utterance of opus not sing with this semantics for the negative would be a property which holds of all events which are not singings of someone named Opus. This gives very liberal truth conditions to an utterance of opus not sing, since there will presumably be a large number of events which were not singings of someone named Opus. But, we do have the leeway of fixing some of the contextual parameters, which would account for the possibility of Partee's (1984) example

(130) I did not turn off the stove.

being interpreted as a property of events which happened at a specified time. For arguments as to why this is the correct thing to do, see Partee (1984), Hinrichs (1986), Moens (1987), Moens and Steedman (1987) and Webber (1987). In fact, this is the only option we have so far for interpreting such an utterance.

The other possibility for not would have it that not(P)(e) is true just in case e is the stative eventuality of there being no events of type P. We could then replace the previous weak axiom with the stronger

(131) \( \mathcal{U} \text{not}(P)(e) \iff \mathcal{U}e_{\forall}(\neg P) \)

where we again apply the higher order negation operator to a property. Under this strategy, the truth of not(P)(e) would imply that there are no eventualities with the property P. We find this approach more attractive in general, but note that either axiom would be consistent with our general program.
Finally, it should be noted that in any case, what is being located at a time is the actual event corresponding to the negation rather than the event described by the complement to not. This is the only analysis possible with our compositional framework, because we can not recover the property of events $P$ from the property $\text{not}(P)$ without additional assumptions. We might then make some kind of relativisation to (131) to account for the times of events in question. For instance, we might use

\[(132) \forall \text{not}(P)(e) \leftrightarrow \forall \text{before}(e)(\sim P).\]

With this axiom, *opus did not sing* would mean that there were no singing events with Opus as their agent before the time parameter fixed by the finite auxiliary verb *did*. This time parameter is itself restricted to be prior to the time of utterance by the background conditions in the lexical entry for *did*.

### 4.5.7 Simple Modifiers

#### Nominal Modifiers

The simple nominal modifiers of the naïve fragment can be carried over to the relational fragment by simply giving them an empty list of discourse markers and an empty set of presuppositions.

This means we will take the entry

\[(133) \begin{align*}
\text{EXP: } & \text{red} \\
\text{SYN: } & n/n \\
\text{CON: } & (X.P).X.(P \cap \text{red}(X)) \\
\text{REF: } & \emptyset \\
\text{BG: } & \emptyset
\end{align*}\]

for the intersective adjective *red*. Notice that here we do not need to explicitly apply $P$ to the $X$ argument, since it will already be a part of $P$, as can be verified by checking the entry for nominals, so that *red ball* would get the category

\[(134) \begin{align*}
\text{EXP: } & \text{red ball} \\
\text{SYN: } & n \\
\text{CON: } & X.(\text{ball}(X) \cap \text{red}(X)) \\
\text{REF: } & \emptyset \\
\text{BG: } & \emptyset
\end{align*}\]
derived from the category

(135) \textbf{Exp:} \textit{ball}  \\
\textbf{Syn:} \textit{n}  \\
\textbf{Con:} \textit{Y}.\textit{ball}(Y)  \\
\textbf{Ref:} \{\}  \\
\textbf{BG:} \{\}

since \( P \) will unify with \textit{ball}(Y) and \( X \) with \( Y \) to produce the final result.

We will also have the entry

(136) \textbf{Exp:} \textit{very}  \\
\textbf{Syn:} \textit{n/n / (n/n)}  \\
\textbf{Con:} \((X.P).(Y.Q)).(Z.R).W.\textit{very}((X.P).(Y.Q))(Z.R)(W)  \\
\textbf{Ref:} \{\}  \\
\textbf{BG:} \{\}

for the intensifier \textit{very}, so that \textit{very red ball} would be analysed as

(137) \textbf{Exp:} \textit{very red ball}  \\
\textbf{Syn:} \textit{n}  \\
\textbf{Con:} \textit{W.very}((X.P).(X.(P \texttt{fl} red (X))))(Z.\textit{ball}(Z))(W)  \\
\textbf{Ref:} \{\}  \\
\textbf{BG:} \{\}

where we take \( \text{very}(A)(P)(x) \) to be true just in case \( x \) is an individual of type \( P \) which is very \( A \), where we leave open exactly what it means for an individual to have very much of an adjectival property. The intensifier \textit{very} modifies its adnominal argument and then takes this property as characterising the individual which also happens to be of the type of the noun argument. Thus, it needs to be of high enough order to take as arguments the contents assigned to its complements. The constant \textit{very} assigned in our lexical entry for \textit{very} has the minimal type necessary to take the appropriately typed arguments.

**Verbal Modifiers**

The simple verbal modifiers of the naïve fragment can also be carried over directly to the relational fragment, with the additional assumption that they preserve the verb form of their argument. This can be carried out by unification as in our
term categorial grammar, so that an adverb like *slowly* which occurs before a
verb phrase, gets the lexical entry

(138) \[\text{Exp: slowly} \]
\[\text{Syn: } s(V) \backslash np \rightarrow (s(V) \backslash np) \]
\[\text{Con: } (X.E.P).X.E2.slowly(E.P)(E2) \]
\[\text{Ref: } () \]
\[\text{BG: } {} \]

This will insure that *slowly run* will get the syntactic category \(s(bse) \backslash np\) and
that *slowly ran* gets the category \(s(fin) \backslash np\). Again, we carry out application by
unification rather than explicitly, with *slowly run* having the category

(139) \[\text{Exp: slowly run} \]
\[\text{Syn: } s(bse) \backslash np \]
\[\text{Con: } X.E2.slowly(E.run(X)(E))(E2) \]
\[\text{Ref: } () \]
\[\text{BG: } {} \]

The other verbal modifiers can be carried over from the naïve fragment in exactly
the same way.

Prepositional Modifiers

The prepositions in the relational fragment will be just like those of the naïve
fragment, since they will not introduce any additional discourse referents or pre-
suppositions. We will still need both a verbal and a nominal entry for every
preposition, with the following entries for the nominal and verbal forms of *in*

(140) \[\text{Exp: in} \]
\[\text{Syn: } n \backslash n \rightarrow np \]
\[\text{Con: } X.(Y.P).Y.(P \cap \text{in}(X)(Y)) \]
\[\text{Ref: } () \]
\[\text{BG: } {} \]

(141) \[\text{Exp: in} \]
\[\text{Syn: } s(V) \backslash np \rightarrow (s(V) \backslash np) \rightarrow np \]
\[\text{Con: } X.(Y.E.P).Y.E.(P \cap \text{in}(X)(E)) \]
\[\text{Ref: } () \]
\[\text{BG: } {} \]
4.5.8 Complex Verbs

In this section we will consider verbs which subcategorise for arguments other than noun phrases.

Simple Attitude Verbs

After a noun phrase, the next simplest thing for which a verb may subcategorise is a finite sentence. Such verbs are often used to report the attitude of an agent, as in

(142) a. Opus thinks Binkley ran.
    b. Binkley believes Milo ate.
    c. Milo knows Opus sang.

All of these finite sentences involve proper constituents which are themselves finite sentences. For the base form think, we use the category

(143) EXP: think
    SYN: s(bse) \ np / s(fin)
    CON: (E.P).X.E2.think(X)(E.P)(E2)
    REF: {}
    BG: {}

where we assume that think(x)(P)(e) is true just in case e is an event of x thinking that there is an actual instance of the property of events P. This allows us to analyse the sentence opus think binkley ran as the category

(144) EXP: opus think binkley ran
    SYN: s(bse)
    CON: E2.think(X)(E.(run(Y)(E) \ at(E)(T)))(E2)
    REF: (X,Y,T)
    BG: \{ name(opus)(X) \}
        \{ name(binkley)(Y) \}
        before(T)(Tran)

The entries for the other simple attitude verbs follow the same pattern. Now consider the case of know. An utterance of (142)c can only be true if Opus did
in fact sing. If a verb can only be part of a truthful utterance if its complement is truthful, then it is said to be veridical.

We can model this behaviour with a simple axiom

\[(145) \, ^u \text{know}(x)(P) \supset ^u \exists_1(P)\]

which means that if \(x\) knows that \(P\) is instantiated, then \(P\) must really be instantiated. The other attitude verbs do not involve such a condition, since it is possible to believe that the earth is flat without things actually being that way, but it is not possible to know that the earth is flat without the world actually being flat. Another possible analysis for such verbs which is much more in line with that of situation semantics as presented in Barwise and Perry (1983) would be to take the category

\[(146) \, \text{Exp: } \text{know} \]

\[\text{Syn: } s(bse) \setminus np \setminus s(fin)\]

\[\text{Con: } (E.P).X.E2.\text{know}(X)(E)(E2)\]

\[\text{Ref: } \{E\}\]

\[\text{BG: } \{P\}\]

for the attitude verb \text{know}. This treats the sentential complement as an indefinite, thus requiring the existence of an event which has the property specified by the complement, so that, for instance, we would have

\[(147) \, \text{Exp: } \text{opus know binkley ran} \]

\[\text{Syn: } s(bse)\]

\[\text{Con: } E2.\text{know}(X)(E)(E2)\]

\[\text{Ref: } \langle X, E, Y, T \rangle\]

\[\text{BG: } \left\{ \begin{array}{l}
\text{name(opus)(X)} \\
\text{(run(Y)(E) \cap at(E)(T))} \\
\text{name(binkley)(Y)} \\
\text{before(T)(Tran)}
\end{array} \right\}\]

Notice that the order of the discourse markers corresponds to the order of the lexical entries that introduce them.

Most of the simple attitude verbs also take complements in the form of complementised sentences. For instance, consider the sentences in
Control Verbs

Consider the following sentences involving so-called control phenomena

(149) a. Opus seemed to swim.
    b. Opus promised Binkley to swim.
    c. Opus knew Binkley to swim.
    d. Opus persuaded Binkley to swim.

In the first two cases, the subject of the infinitival complement to swim is taken to be the subject of the main clause, while in the second two cases it is taken to be the direct object. The former is referred to as subject control, with the latter being termed object control. In the first two cases, it is Opus who is the understood agent of the swimming in question, while in the third and fourth cases it is Binkley.

Such control phenomena are quite easy to encode within a unification based framework, where we have the lexical entries

(150) EXP: seem
    SYN: s(bse) \ np / (s(inf) \ np)
    CON: (X.E.P).X.E2.seem(E.P)(E2)
    REF: {} 
    BG: {}
The difference between the subject control and object control cases is represented by the unification of the infinitival subject with either the subject or object of the main clause. In the first two cases it is the subject of the sentence, while in the last two it is the object content. The base forms of the sentences in (149) will then be analysed as

(151) **Exp:** promise  
**SYN:** $s(bse) \backslash np / (s(inf) \backslash np) / np$  
**REF:** {}  
**BG:** {}  

(152) **Exp:** know  
**SYN:** $s(bse) \backslash np / (s(inf) \backslash np) / np$  
**CON:** $Y.(Y.E.P).X.E2.know(E.P)(X)(E2)$  
**REF:** {}  
**BG:** {}  

(153) **Exp:** persuade  
**SYN:** $s(bse) \backslash np / (s(inf) \backslash np) / np$  
**REF:** {}  
**BG:** {}  

(154) **Exp:** opus seem to swim  
**SYN:** $s(bse)$  
**CON:** $E2.\text{seem}(E.\text{swim}(X)(E))(E2)$  
**REF:** {}  
**BG:** \{ name(X)(opus) \}  

(155) **Exp:** opus promise binkley to swim  
**SYN:** $s(bse)$  
**CON:** $E2.\text{promise}(Y)(E.\text{swim}(X)(E))(X)(E2)$  
**REF:** {}  
**BG:** \{ name(X)(opus) \}  
\{ name(Y)(binkley) \}
We have not specified the temporal relations between the main verb and its infinitival complement. Since the infinitival complement will be a property of events not sensitive to time, the semantic content of the constant associated with a control verb can freely dictate its own temporal relations. That is, if john wants to run really means that John wants to run at the time of the utterance, we can simply require want(x)(P)(e) to be true if and only if e is an event of x wanting to take part in an event of type P at the same time as the event e, which will itself be restricted by the tense of the main verb. Any regularities in the time restrictions on properties of events introduced by infinitival complements will thus have to be dealt with by meaning postulates.

There is another dimension of contrast between the examples in (149). The verbs seem and know are often called raising verbs, while promise and persuade are called equi verbs, due to transformational terminology describing how they were derived. In our theory, as in lexical-functional grammar (Bresnan 1982) and head-driven phrase structure grammar (Pollard and Sag 1987), the difference between equi and raising verbs is reflected in their contents. In the case of the raising verbs, there is no argument position for the implicit subject, while in the case of the equi verbs, the implicit subject shows up not only as an argument to the complement sentence, but also to the equi verb itself. While this will not have any significant impact on the rest of the theory that we will present, it will affect other components of the grammar as detailed in Pollard and Sag (1987). Most
notably, it affects the distributions of reflexive anaphors such as *themselves* and the possibility for passivisation and detransitivisation in the lexicon (see Carpenter (1987)).

Note that this analysis of control is purely descriptive in that it does not explain why the control phenomena are the way they are. Control simply shows up in the lexical entries of certain verbs, and that is as deeply as we analyse things here. For accounts of various aspects of lexical information more or less compatible with what is described here, see Flickinger, Pollard and Wasow (1985), Pollard and Sag (1987) Bresnan (1982, 1982b), Levin (1987) and Dowty (1978, 1979).

We have also left off any mention of *for-to clauses* such as *for mary to run* as they appear in

\[(158)\]

\[\begin{align*}
\quad & a. \text{ John waited (for Mary) to run. } \\
\quad & b. \text{ (For Mary) to run would bother John. } \\
\quad & c. \text{ John bought some shoes (for Mary) to wear. }
\end{align*}\]

In all of these cases the *for* portion of the construction is optional as is indicated by the parentheses. When the subject marked by *for* is absent, and even in some cases when it is present, the infinitive must be controlled from somewhere else. For an argument as to why this "control" must be pragmatic in nature, see Ladusaw and Dowty (1986).

There is also a great deal of interest in the uses of the dummy noun phrases *it* and *there* and their interaction with control phenomena. Consider the cases of

\[(159)\]

\[\begin{align*}
\quad & a. \text{ It is raining. } \\
\quad & b. \text{ It seems to be raining. } \\
\quad & c. \text{ It appears that Mary ran. } \\
\quad & d. \text{ It is unlikely to rain. }
\end{align*}\]

\[(160)\]

\[\begin{align*}
\quad & a. \text{ There is nothing here. } \\
\quad & b. \text{ There seems to be nothing here. } \\
\quad & c. \text{ There was running and jumping. }
\end{align*}\]
which are based on examples in Gazdar et al. (1985). The analysis of dummy *it* and *there* is quite straightforward, and follows that presented in generalized phrase structure grammar (Gazdar et al. 1985). The dummy noun phrases themselves can be given the following lexical entries

(161) **EXP:** *it*

    **SYN:** np(*it*)
    **CON:** *it*
    **REF:** ()
    **BG:** {}

(162) **EXP:** *there*

    **SYN:** np(*there*)
    **CON:** *there*
    **REF:** ()
    **BG:** {}

We have assumed that noun phrases are broken down into subcategories in the same way as verb phrases, as was the case for our grammar in Section 2.3.5. We will simply take *it* and *there* to be two alternatives to the normal case markings of subj and obj. That is, every other noun phrase besides *it* and *there* will get a different value for case. This allows verbs to specify whether they take arguments which are normal or dummies. All of the verbs we have mentioned so far only take normal complements. We then only need to assume that there are some verbs which lexically subcategorise for dummy subjects, as is the case with

(163) **EXP:** *rain*

    **SYN:** s(bse) \ np(*it*)
    **CON:** \ X.E.rain(E)
    **REF:** ()
    **BG:** {}

and

(164) **EXP:** *be*

    **SYN:** s(bse) \ np(*there*) / np
    **CON:** Y.X.E.exists(Y)(E)
    **REF:** ()
    **BG:** {}
Note that the dummy argument does not show up in the semantics. Also notice that only a dummy noun phrase can show up as the subject of either of these verbs. This allows us to analyse the first sentences in the above examples just like for any other verb. The proper interaction with control is then guaranteed by supposing that the control verbs carry through the case of their infinitival complements to the case of the controlled element. For instance, the lexical syntactic category for seem would be $s(bse) \backslash np(C) / (s(inf) \backslash np(C))$ and the entry for the infinitival auxiliary to would be $s(inf) \backslash np(C) / (s(bse) \backslash np(C))$. This will allow the case marking on the infinitival complement to percolate through the analysis, giving us the category $s(inf) \backslash np(it)$ for to be raining and the category $s(fin) \backslash np(it)$ for seemed to be raining. Dummy there functions in exactly the same way. A similar comment would go for number, as well, which would account for number agreement under control. We will not deal with the case of extraposition here, but see Morrill (1987,1988) for an analysis of the long-distance dependency facts and Carpenter (1987) for the lexical details, both following the analysis of generalized phrase structure grammar (Gazdar et al. 1985).

Perception Verbs

One of the most convincing aspects of situation semantics is its treatment of the so-called naked infinitive perceptual reports (Barwise 1981, Barwise and Perry 1983). These occur in utterances of sentences such as

(165) a. binkley saw opus eat.
   b. binkley saw opus eating.

We can incorporate the insight of situation semantics directly into our translations by taking the base form of the verb see to have the lexical category

(166) EXP: see
   SYN: $s(bse) \backslash np / (s(bse) \backslash np) / np$
   REF: $(E)$
   BG: $\{P\}$
involving object control. The reason we use the control construction is for compatibility with a theory of long-distance dependencies and lexical rules in which the subject of the complement may be extracted (for details, see Gazdar et al. (1985), Morrill (1987, 1988) and Carpenter (1987)). This use of background conditions and the reference list leads to the following analysis of *opus see binkley walk*

(167) **EXP:** *opus see binkley walk*  
**SYN:** \( s(\text{bse}) \)  
**CON:** \( E_2.\text{see}(E)(X)(E_2) \)  
**REF:** \( (X, Y, E) \)  
**BG:**  
\[
\begin{align*}
\text{name}(\text{opus})(X) \\
\text{name}(\text{binkley})(Y) \\
\text{walk}(Y)(E)
\end{align*}
\]

Notice that this is identical to the alternative treatment we proposed for veridical attitude verbs such as *know*. It differs from the standard Montagovian approach which analyses the content of a sentence as a proposition. Under that account, there is no event which can be recovered to be the object perceived. Here there are three discourse markers, one for each of the nominals and an additional marker for the event corresponding to the embedded clause. We assume that \( \text{see}(e)(x)(e') \) is true just in case \( e' \) is an event of \( x \) seeing the event \( e \). In the case at hand, we can interpret *opus see binkley walk* truthfully just in case there is some event of Binkley walking and Opus happened to see that event. This corresponds to our intuitive view of events as limited parts of the world that intelligent agents such as human beings can perceive. Of course, this leads to a veridical account of perception, since if an event of a certain type was seen, then it certainly must exist, since only real objects stand in relations.

There are also perceptual report verbs which take predicative verb form complements, but in other respects behave just like those taking base form complements. For instance, in the case of *see*, we also have the lexical categorisation
which only varies from the previous entry in the form of the verb for which it is subcategorised. This means that the category for opus see binkley running will be similar to that opus see binkley run, with the only difference being in the progressive meaning of the complement running as opposed to run.

4.5.9 Complex Modifiers

There are a number of adverbial modifiers which take clausal complements in addition to the verb phrases they modify. For example, consider the clausal adverbials in

(169) a. Opus eat after Binkley spoke.
    b. Opus eat while speaking.
    c. Opus eat with Binkley speaking.
    d. Opus eat to be loved.

The adverbial after takes a finite sentential complement, with the lexical entry

(170) EXP: after  
SYN: s(V) \ np \ (s(V) \ np) / s(fin)  
CON: (E.P).X.E2.(X.E2.P2).(P2 \ while(E)(E2))  
REF: (E)  
BG: \{ P \}

giving our example opus eat after binkley ate the category

(171) EXP: opus eat after binkley spoke  
SYN: s(bse)  
CON: E2.(eat(X)(E2) \ after(E)(E2))  
REF: (X, E, Y, T)  
BG: \{ name(opus)(X) 
    speak(Y)(E) \ at(E)(T) 
    name(binkley)(Y) 
    before(T)(T_eat) \}
As a notational convention, we will allow an event $e$ to occur in places normally occupied by a temporal interval, with the understanding that it picks out its temporal trace, so that we interpret $\text{after}(e)(e')$ as $\text{after}(\tau(e))(\tau(e'))$. Notice that we have used a veridical account of the complement clause, in virtue of its event variable showing up as a reference marker.

The following lexical entry accounts for *while*

(172) **EXP:** while  
**SYN:** $s(V) \setminus np \setminus (s(V) \setminus np) / (s(pred) \setminus np)$  
**CON:** $(X.E.P).(X.E2.P2).X.E2.(P2 \cap \text{during}(E)(E2))$  
**REF:** $\langle E \rangle$  
**BG:** $\{ P \}$

giving our example *opus eat while speaking* the category

(173) **EXP:** opus eat while speaking  
**SYN:** $s(bse)$  
**CON:** $E2.(eat(X)(E2) \cap \text{during}(E)(E2))$  
**REF:** $\langle X, E \rangle$  
**BG:** $\{ \text{name}(opus)(X), \text{prog}(E3.speak(X)(E3))(E) \}$

where we again interpret *during* temporally to mean that its two argument events have temporal traces which overlap. This means that an utterance of this sentence must be interpreted as a property of events which are eating events by someone named Opus which also overlap some progressing event of Opus speaking. Notice the control that is exerted between the subject of the adverbial and the subject of the sentence. This is carried out by using the argument $X$ three times in the contents of the complements of *while*.

The other complex adverbials receive similar categorisations, with the following entry for *with*

(174) **EXP:** with  
**SYN:** $s(V) \setminus np \setminus (s(V) \setminus np) / (s(pred) \setminus np) / np$  
**CON:** $Y.(Y.E.P).(X.E2.P2).X.E2.(P2 \cap \text{with}(E)(E2))$  
**REF:** $\langle E \rangle$  
**BG:** $\{ P \}$

229
where we leave open whether or not with is to be interpreted temporally or as
some more complex relation between events. Notice that in this adverbial, control
of the complement verb phrase goes to the complement noun phrase rather than
the main clause subject.

Finally, to can be given the category

(175) EXP: to
SYN: \( s(V) \setminus np \rightarrow (s(V) \setminus np)/(s(bse) \setminus np) \)
CON: \( (X.E.P).(X.E2.P2).X.E2.(P2 \land to(E.P)(X)(E2)) \)
REF: {}
BG: {}

where we take an “in-order-to” reading of to, so that \( to(P)(e)(x) \) is true just in
case \( e \) is an event that \( x \) carried out in order to bring about an event of type \( P \).
Notice that we do not make any assumptions of veridicality for to, since it seems
that it is possible to exercise to get fit, but not ever reach a state of being fit.
Chapter 5

Scope and Dependency

In this chapter, we extend our relational categorial grammar to noun phrases involving quantification and anaphoric dependencies.

5.1 Quantification

All of the noun phrases introduced in the previous section were singular referring noun phrases. What this means is that an utterance in which they occur can only be true if there is an individual in the world with certain properties. For instance, the sentences

(1)  a. Opus ate.
    
        b. A penguin ate.
    
        c. The penguin ate.

     can only be uttered truthfully if there turns out to be some individual in the world who ate. The first sentence requires the individual eating to be named Opus, the second requires the individual to be a penguin, while the last requires the individual who ate to be the unique penguin (modulo resource constraints as in Section 4.5.4). But so far we have no semantic analyses for sentences such as

(2)  a. No penguin ate.
    
        b. Some penguin loved every herring.
    
        c. Some penguin on every iceberg ate many herring.
and others like them containing general noun phrases. A general noun phrase consists of a general determiner such as every, no, some, most and so on followed by a nominal expression. Consider the truth conditions of sentences like those in (2) involving general noun phrases. Unlike previous cases, there is no one to one correspondence between noun phrases and individuals which must exist for the sentences to be used truthfully. For instance, (2)a can be uttered truthfully only if there is not a penguin who ate, which certainly does not entail the existence of any particular penguin.

But this is not the only semantic complication introduced by general noun phrases. Consider an utterance of (2)b above, which could have two distinct interpretations. Under the first interpretation, the utterance could be truthful if there was a single herring which was loved by every penguin. The second interpretation would allow for the possibility that every penguin does not love the same herring, but simply that for every penguin there is some herring which it loves. In the first reading we say that every herring takes wide scope and some penguin takes narrow scope, and conversely for the second reading.

A grammar that is meant to model general noun phrases must provide some method for generating quantifier scopings. In a compositional framework, this will require a corresponding syntactic difference between expressions taking different scopes. This is because we can only have semantic ambiguity where there is syntactic ambiguity. Ideally, we would like to maintain the application-driven version of compositionality that we started with, but unfortunately this is simply not possible. Many techniques have been explored for generating quantifier scopes in a rule to rule fashion, and we will consider a number of these in turn. First, we introduce Montague's treatment of quantification by means of term insertion. Then we cover a popular method of quantifier scoping from the computational linguistics community. Finally, we settle with some amendments to our relational categorial grammar employing a technique similar to Cooper storage and show how this relates to the other two approaches.
5.1.1 PTQ

Montague, in the ambitiously titled 'The Proper Treatment of Quantification in Ordinary English' (1974c), which we call PTQ, developed a grammar within his universal grammar paradigm that was capable of generating all of the possible scopings of an expression containing general noun phrases. But, Montague's analysis relies to a large degree on the extremely generous definition of expression and general algebraic framework for combining expressions syntactically and computing their meanings. In this section, we present a trimmed down version of the PTQ grammar modified to use the categorial syntax we have already introduced.

For simplicity, in this section and the next, we will follow Montague in assuming that

\[ \mathcal{D}_s = \text{Prop} \]

This means that an interpretation of an utterance of a sentence will be a proposition. Recall that we can always think of such a proposition as coming from the existential quantification of some property of events determined by the interpretation of the sentence in our event semantics. Furthermore, we will forget about contextual information for the time being, as it is really not important in the current setting. This would not be the case if we were assuming that there is an element of context that fixes the quantifier scope of expressions uttered in it as in Gawron and Peter's version of situation semantics (Gawron and Peters forthcoming). Rather than fix the quantifier scope by context, Montague allowed the syntactic derivation tree to determine the scope. Remember that the derivation trees are the disambiguated expressions of the grammar. Since each disambiguated expression must have a unique meaning, there was a different disambiguated expression for each possible scoping. Of course, the alternative is to allow the meaning of an expression containing quantifiers to be neutral with respect to scope information. This is the case with Gawron and Peter's grammar and the one we develop in Section 5.1.2 following Cooper (1983).

We will also introduce another basic category \( q \) for quantified noun phrases such as every penguin. We assume that the type of \( q \) is given by

\[ \theta(q) = ((i,p),p) \]
Now consider the following simple lexicon associating expressions with syntactic categories and corresponding λ-terms of the appropriate type.

(5) | Expression | Category | Interpretation |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>his</td>
<td>np</td>
<td>(x_n)</td>
</tr>
<tr>
<td>man</td>
<td>n</td>
<td>man</td>
</tr>
<tr>
<td>every</td>
<td>(q/n)</td>
<td>(\lambda P.\lambda Q.\forall(P)(Q))</td>
</tr>
<tr>
<td>in</td>
<td>(n \setminus n/np)</td>
<td>(\lambda y.\lambda P.\lambda x.P(x) \cap \text{in}(x)(y))</td>
</tr>
<tr>
<td>ran</td>
<td>(s \setminus np)</td>
<td>ran</td>
</tr>
<tr>
<td>loved</td>
<td>(s \setminus np/np)</td>
<td>loved</td>
</tr>
<tr>
<td>yesterday</td>
<td>(s \setminus np \setminus (s \setminus np))</td>
<td>yesterday</td>
</tr>
<tr>
<td>in</td>
<td>(s \setminus np \setminus (s \setminus np)/np)</td>
<td>(\text{in}_e)</td>
</tr>
</tbody>
</table>

We are assuming here that \(x_n\) is the \(n\)th individual variable of type \(i\). We will not try to spell out here what the possible value for \(\text{in}_e\) could be under this type strategy, but simply note that it must be different than the value for the nominal prepositional phrase, despite the identical semantic types assigned to the categories \(s \setminus np\) and \(n\) for verb phrases and nouns.

Notice that with this type assignment, we already have the type equivalences

(6) \(\theta(q) = \theta(s / (s/np)) = \theta(s / (s \setminus np)) = \theta(s \setminus (s/np)) = \theta(s \setminus (s \setminus np))\).

The reason that we will not just use these already existing categories in place of \(q\) is that we will not be concerned with the directionality of a quantifier. In fact, we never want a quantifier expression to be a syntactic functor, but only partake in one special rule created to resolve quantification scope ambiguity. This was not the case with Montague's (1974d) original PTQ grammar, which allowed quantifier expressions to combine by functional application as well as by the term insertion rule we present below. Hendriks (1987) develops an extended categorial grammar formalism that employs the categorisations in (6) to deal with scope ambiguity.

We take the usual directional application schemata, presented here in their full rule to rule format as the admissible local tree schemata

(7) \(\langle \delta_1, \alpha/\beta, f \rangle \langle \delta_2, \beta, g \rangle \rightarrow \langle \delta_1 \delta_2, \alpha, f(g) \rangle\)

234
Recall that the interpretation of the first rule means that an expression $\delta_1$ of category $\alpha / \beta$ with semantics $f$ can combine with an expression $\delta_2$ of category $\beta$ and semantics $g$ to form the expression $\delta_1 \delta_2$ of category $\alpha$ and semantics $f(g)$.

In the grammar generated from our lexical assignment and just these rules, we get the analyses

(9) \( \langle \text{every man}, q, \lambda P. \forall (\text{man})(P) \rangle \)

(10) \( \langle h e_0 \text{ ran}, s, \text{ ran}(x_0) \rangle \)

but will not be able to parse every man ran. Montague (1974d) employed a rule schema of term insertion to analyse such a quantificational sentence. The term insertion schema translates into our current system as the schematic rule

(11) \( \frac{\langle \delta_1, q, Q \rangle \langle \delta_2, s, S \rangle}{\langle \delta_2[he_k/\delta_1], s, Q(\lambda x_k.S) \rangle} \)

where $he_k$ is a subexpression occurring exactly once in $\delta_2$. As usual, we take $\delta_2[he_k/\delta_1]$ to be the result of replacing the occurrence of $he_k$ in $\delta_2$ with $\delta_1$, where $\delta_1$ in this case will be an expression categorised as a general noun phrase. This schema depends crucially on the value $k$ of the pronoun's subscript, and thus corresponds to a number of rules, one for each subscript. The tree

(12) \( \langle \text{every penguin}, q, \forall (\text{penguin}) \rangle \langle h e_0 \text{ ate}, s, \text{ ate}(x_0) \rangle \)

\( \langle \text{every penguin ate}, s, \forall (\text{penguin})(\lambda x_0.\text{ ate}(x)) \rangle \)

is then admitted by the term insertion rule, where we have

(13) \( (he_0 \text{ ate})[he_0/(\text{every penguin})] = \text{every penguin ate} \).

There are a number of things worth noting about this grammar. The first is that we have expressions like $he_0$ which never show up in well-formed sentences and have as their interpretation a function from assignments to values. It is by allowing the meaning algebra to contain terms with free variables that Montague is able to bind the quantifier to the right variable when it finally gets inserted. Also notice that the operation on the expression algebra corresponding to the
A term insertion rule is reasonably complex in that it converts an expression with an occurrence of $he_0$ into one that is just like the original only with a quantifier expression in place of $he_0$.

In the case of sentences with two quantifiers, the analysis will require first generating a sentence with two subscripted pronouns and then applying term insertion twice. The fact that term insertions can be applied in any order leads to the two readings of a sentence such as _some penguin loved every herring_, as can be seen by considering the following elements of the grammar, where we will display a triple $(e, c, s)$ in the grammar as

(14) \[
\begin{align*}
\text{Exp: } & e \\
\text{Syn: } & c \\
\text{Con: } & s
\end{align*}
\]
when it is too large to fit on one line, just as in our relational grammar.

(15) \[
\begin{align*}
\text{Exp: } & \text{some penguin} \\
\text{Syn: } & q \\
\text{Con: } & \lambda P.\exists(penguin)(P)
\end{align*}
\]

(16) \[
\begin{align*}
\text{Exp: } & \text{every herring} \\
\text{Syn: } & q \\
\text{Con: } & \lambda P.\forall(herring)(P)
\end{align*}
\]

(17) \[
\begin{align*}
\text{Exp: } & \text{he}_0 \text{ loved } \text{he}_1 \\
\text{Syn: } & s \\
\text{Con: } & \text{loved}(x_1)(x_0)
\end{align*}
\]

(18) \[
\begin{align*}
\text{Exp: } & \text{some penguin loved } \text{he}_1 \\
\text{Syn: } & s \\
\text{Con: } & \exists \ (\text{penguin}) \\
& \ (\lambda x_0.\text{loved}(x_1)(x_0))
\end{align*}
\]

(19) \[
\begin{align*}
\text{Exp: } & \text{some penguin loved every herring} \\
\text{Syn: } & s \\
\text{Con: } & \forall (\herring) \\
& \ (\lambda x_1.\exists \ (\text{penguin}) \ \\
& \quad \ (\lambda x_0.\text{loved}(x_1)(x_0))
\end{align*}
\]
(20) EXP: $he_0$ loved every herring  
SYN: $s$  
CON: $\forall (\text{herring})$  
$$ (\lambda x_1.\text{loved}(x_1)(x_0))$$

(21) EXP: some penguin loved every herring  
SYN: $s$  
CON: $\exists (\text{penguin})$  
$$ (\lambda x_0.\forall (\text{herring})$  
$$ (\lambda x_1.\text{loved}(x_1)(x_0))$$

In the case of verbal prepositional phrases, we can derive

(22) EXP: $he_0$ ate in $he_1$  
SYN: $s$  
CON: $\text{in}_e(x_1)(\text{ate})(x_0)$

where the two place holders $he_1$ can be inserted for just as in the case of some penguin loved every herring.

The distinctions between $q$ and $np$ that we have introduced in the grammar are syntactically unmotivated. There is nothing in the syntactic distribution of quantified noun phrases to separate them from regular noun phrases. That is, any ordinary noun phrase in an expression may be replaced by a general noun phrase without loss of well-formedness, and conversely. The term insertion rule is included solely to get the semantic details correct.

To provide a uniform treatment of the syntax and semantics of noun phrases, Montague (1974d) assumed that all noun phrases were inserted as category $q$. The ordinary $np$ category was then reserved for the dummy pronouns $he_k$ which do not show up in well formed sentences of the language. To go along with the syntactic type raising, there had to be a semantic type-raising as well. The strategy here was to take an ordinary noun phrase content $x$ and derive a generalised quantifier $Q_x$ from it. The definition assumed that

(23) $Q_x \leftrightarrow \lambda P. P(x)$

for every individual $x$ and property $P$. The process of converting an individual into a quantifier is called type-raising. By $\beta$-conversion, for every property $P$ and individual $x$ we have
as a theorem. This means that there is a one-one mapping of the individuals into the generalised quantifiers. Consequently, Montague would have a lexical entry such as

\[ Q_x(P) \leftrightarrow P(x) \]

(25) \[ \text{EXP: opus} \]
\[ \text{SYN: } q \]
\[ \text{CON: } \lambda P.P(\text{opus}) \]

for a proper name like opus. Note that we are back to using an individual constant opus, as in our naïve semantics in Section 4.3.1, but this is not what is at issue here. We can then see that putting this entry in place of the entry for every penguin in the above derivation would lead to the following instance of our general theorem

\[ (\lambda P.P(\text{opus}))(\text{run}) \leftrightarrow \text{run(opus)}. \]

We can even type-raise pronouns, giving he_0 the syntactic category q and semantics \( \lambda P.P(x_n) \), but pronouns would still require their basic np category unless every np complement in the grammar was replaced with a q complement. This latter move is actually closer to the spirit of Montague's (1974d) original grammar, but does not affect the possible scope readings of generalised quantifiers with the lexical entries we have discussed so far. In Section 5.1.7, we discuss some semantic evidence for subcategorising a verb for a type-raised noun phrase.

Within this framework, there will be a great deal of spurious ambiguity. Not only will we be able to generate a general noun phrase as many ways as there are subscripted pronoun placeholders, but we will get more serious structural ambiguity in the derivation of such simple sentences as john loved every woman and even john loved mary, depending on which order term insertion was applied.

5.1.2 Post-syntactic Processing

A number of algorithms have been developed in the computational linguistics literature for the lazy evaluation of quantifiers, where quantificational scope is not resolved until after the syntactic analysis is complete. This is the strategy...
adopted in the systems of Woods (1978), Bobrow and Webber (1980), Halvorsen (1983), Schubert and Pelletier (1984), and Hobbs and Shieber (1987). There appears to be no other reason besides Montague’s conception of compositionality to clutter up the syntax of a grammar with all kinds of quantificational facts, since the scope of quantifiers is not a purely syntactic phenomena (but see May (1985) for a government-binding theory account of syntax and semantics which involves a single set of principles operating over both). There is no morphological marking in any language to determine quantificational scope in cases where it might be ambiguous. Syntactic contributions to quantifier scope are reflected in the surface structure of an expression (see, for instance, May (1985) or Cooper (1983)). Of course, in an interleaved model of processing, such as the theoretical model of Fenstad, Halvorsen, Langholm and van Benthem (1987), where syntactic, semantic and pragmatic analyses are tackled concurrently, it may be desirable to allow partial resolution to take place before a syntactic analysis is completed.

In the post-syntactic approach, the “logical form” of an expression is usually taken to include so-called **quantifier terms**, which are of the form $GQ(Q, x, R)$ where

(27)   
\begin{enumerate}
  \item $x$ is an individual variable, the **focus**
  \item $Q$ is a term of type $\langle(i, p),\langle(i, p),p\rangle\rangle$, the **quantifier**
  \item $R$ is a propositional term with a free occurrence of $x$, the **restriction**.
\end{enumerate}

The set of well-formed terms is then extended to include the quantifier terms among the individual terms, making them of type $i$ for the sake of term formation.

The procedure is then to build up a logical term with all of the quantifiers inside of the terms that they will scope over. Then, the embedded quantifiers can be pulled out of the position in which they occur to scope over some formula which contains them. We can mimic the basic algorithm in our compositional grammar. For the sake of simplicity, we assume the same lexicon as we used for PTQ, with the only change coming in the entry for quantificational determiners, with entries like

239
for the quantifiers.

Under this strategy, we will be able to build up a number of semantic terms with embedded quantifier terms. These terms with quantifier terms as constituents will not have a well defined value in the semantic model we have presented, and will therefore live a second-class existence. Their sole purpose is to act as place-holders for quantifiers which have not yet had their scope resolved. The problem with this, of course, is that expressions which are assigned terms containing quantifier terms as meanings will not have any interpretations in a model. Of course, we could always define the meaning of an expression with a quantifier term as being the collection of all of its possible scopings, although this would still not be enough due to the fact that quantified noun phrases such as every penguin would not be interpretable by themselves, since there are no rules that can apply to reduce the quantification term it introduces.

We keep the application rules we introduced in the last section, but include the unary rule schema

\[(29) \quad \frac{\langle \delta, c, S \rangle}{\langle \delta, c, S' \rangle} \]

whenever \( S' \in \text{Pull}(S) \) holds, where the intuitive idea is that \( \text{Pull}(S) \) is the set of all possible quantifier scopings generated from the term \( S \) by pulling quantifier terms out of their embedded positions and resolving their scopes. The definition of \( \text{Pull} \) is based on that of Hobbs and Shieber (1987). \( \text{Pull} \) is defined inductively to be the minimal function such that

\[(30) \quad i. \ S \in \text{Pull}(S) \]

\[\quad \text{ii. } Q(\lambda x.R)(\lambda x.S) \in \text{Pull}(S[x/\text{GQ}(Q,x,R)]) \text{ if } S \text{ is of type } p \]

and \( x \in \text{FreeVar}(R) \subseteq \text{FreeVar}(S) \)

\[\quad \text{iii. } T[x/S'] \in \text{Pull}(T[x/S]) \text{ if } S' \in \text{Pull}(S) \]

The first clause of the definition is straightforward, while the third has the meaning that if \( T \) is a term with \( S \) inside of it and \( S \) can be reduced to \( S' \) then \( T \) can be
reduced to $T[S/S']$ where $S'$ is substituted for $S$ in $T$. The intuitive meaning of the second clause is that a term with a quantifier term inside of it can be reduced to a term corresponding to the application of the quantifier to the appropriate abstraction of the original semantics.

For instance, consider the sentence

(31) Every man loved some woman.

In the present grammar, this is assigned the content

(32) love $$(GQ(some, y, woman(y)))$

$$ (GQ(every, x, man(x)))$$

The result of $Pull$ applied to this formula contains the following contents

(33) i. love $$(GQ(some, y, woman(y)))$

$$ (GQ(every, x, man(x)))$$

ii. some $(\lambda y. woman(y))$

$$ (\lambda y. love(y)(GQ(every, x, man(x))))$$

iii. every $(\lambda x. man(x))$

$$ (\lambda x. love(GQ(some, y, woman(y)))(x))$$

This demonstrates the iterative nature of $Pull$, which is only able to pull one constituent out at a time. Now consider the sentence

(34) Opus saw every herring swim.

This receives the content

(35) saw(swim(GQ(every, x, herring(x))))(opus).

In this case, applying $Pull$ gives us a set of contents containing the items

(36) i. saw(swim(GQ(every, x, herring(x))))(opus)

ii. every $(\lambda x. herring(x))$

$$ (\lambda x. (see(swim(x))(opus)))$$

iii. see(every $(\lambda x. herring(x)) )(opus)$

$$ (\lambda x. swim(x))$$

241
The element (36)iii demonstrates the recursive clause of the definition of \textit{Pull} in (30)iii.

The restriction that $x \in \text{Free Var}(R) \subseteq \text{Free Var}(S)$ in (30)iii will insure that quantifiers never get pulled out in the wrong place so that the final result has unbound variables. If the first condition that $x \in \text{Free Var}(R)$ were not met, then the quantifier would be vacuous, and if the second condition were not met, it would be possible to pull out a quantifier with a restriction containing free variables not in $S$. Then, when scoping the result after the pulling has been done, we would be left with free variables. We will see an application of this for the content (58) in Section 5.1.3 below.

It should also be pointed out that this system of rules does not require all of the scoping to be resolved once the entire expression has been generated. In fact, quantification can be carried out on the fly, with a quantifier being able to scope as soon as it enters the derivation as an argument. We could define notions of early and late canonical derivations, where an \textit{industrious} derivation would be the derivation where all of the quantifiers are scoped as soon as possible, with a \textit{lazy} derivation taking quantifiers to apply as late as possible. It should be obvious that a lazy canonical derivation will always have all the quantifier reductions after all of the other rule applications. The lazy processing strategy is to do just this and wait until all of the other syntactic information is assembled before any of the scoping takes place.

5.1.3 Quantification and Relations

So far, we have deliberately avoided the issue of quantifiers whose scopes are resolved wholly within a nominal element. Consider the nominal

\begin{equation}
(37) \text{penguin with every herring}
\end{equation}

which may be interpreted as the property

\begin{equation}
(38) \lambda x. (\text{penguin}(x) \land \forall \text{herring}(\lambda y. \text{with}(y)(x))).
\end{equation}

This property holds of an individual just in case the individual is a penguin and is with each of the herring.
It is not possible to generate this reading with the term insertion rule as we have presented it. As it is, term insertion has only been defined to range over sentential categories. To allow term insertion within nominals would require the additional term insertion schema

\[(\delta_1, q, Q) \quad \delta_2, n, P \quad (\delta_2[he \delta_1], n, \lambda y.Q(\lambda x_k.P(y)))\]

which would take the grammar entry

(40) EXP: *penguin with he* 
    SYM: n 
    CON: \(\lambda y.(penguin(x) \cap with(x_0)(y))\)

along with a usual quantifier entry

(41) EXP: *every herring* 
    SYM: q 
    CON: \(\lambda P.\forall(herring)(P)\)

to produce

(42) EXP: *penguin with every herring* 
    SYM: n 
    CON: \(\lambda y.\forall(herring)\) 
        \((\lambda x_0.penguin(y) \cap with(x_0)(y))\)

This is not quite what we had in (38). Here the term *penguin*(y) falls in the immediate scope of the universal quantifier. But, the truth conditions of the two clauses will be the same in this case, because the variable \(y\) is only bound outside of the universal quantifier. This will not be the case with the negative quantifier *no* which would produce

(43) EXP: *no herring* 
    SYM: q 
    CON: \(\lambda P.\exists(herring)(P)\)

from which we could derive

(44) EXP: *penguin with no herring* 
    SYM: n 
    CON: \(\lambda y.\exists(herring)\) 
        \((\lambda x_0.penguin(y) \cap with(x_0)(y))\)
An object could have this property without even being a penguin. The only requirement made by this property is that the object be such that there is no herring which it is a penguin with. Note that this is quite different than the analogue of (38),

(45) EXP: *penguin with no herring*

        SYN: $n$

        CON: $\lambda y. penguin(y) \cap \sim \exists (\text{herring})$

        $(\lambda x_0. \text{with}(x_0)(y))$

where an object will have the property given by the content of this entry only if it is *a penguin* and also not with any herring.

There have been many studies of the lexical semantics of the various quantifiers. For instance, see Barwise and Cooper (1981), van Benthem (1983, 1984) Keenan and Stavi (forthcoming) and Westerståhl (1986). The study of generalised quantifiers usually revolves around their extensional behaviour. As it turns out, the truth or falsity of a proposition formed by applying a generalised quantifier to two properties is determined by the extensional behaviour of the argument properties. For instance, this would mean that the truth value $^u \forall (P)(Q)$ of the universal quantifier restricted to the property $P$ with the scope of $Q$ would be fully determined by the sets of objects $^u P$ and $^u Q$. With our encoding of sets, $x \in ^u P$ if and only if $P(x)$ is true. Of course, the intensional behaviour of the quantifiers is necessary due to distinctions in meaning between

(46) a. Opus thought every unicorn flew.

b. Opus thought every vampire flew.

This meaning difference must be attributed to the meanings of the nominals *unicorn* and *vampire*. This is because with the principle of compositionality, the only way two expressions can have different meanings is if they have subexpressions with different meanings. Similarly, *every unicorn* and *every vampire* must differ in meaning, as well, if the clausal complements to the above sentences are to have different meanings. Consequently, the meaning of every is sensitive not only to the extension of its restriction, but also to its intension. In Montague's own models for PTQ, it looks like a quantifier is being applied to two extensional terms,
but Montague has an intension-forming operator that will allow the intension of terms to be recovered when calculating the value of the entire proposition.

Now we will consider what happens when nominals containing reduced quantifiers occur inside larger expressions. First of all, we can apply a determiner to the nominal in (42) to get

(47) \( \text{Exp: } \text{some penguin with every herring} \)
\[ \text{SYN: } q \]
\[ \text{CON: } \exists (\lambda y. \forall (\text{herring}) (\lambda x_0. \text{penguin}(y) \cap \text{with}(x_0)(y))) \]

which can be inserted into

(48) \( \text{Exp: } \text{he}_2 \text{ ate} \)
\[ \text{SYN: } s \]
\[ \text{CON: } \text{ate}(x_2) \]

to yield

(49) \( \text{Exp: } \text{some penguin with every herring ate} \)
\[ \text{SYN: } q \]
\[ \text{CON: } \exists (\lambda x_2. \forall (\text{herring}) (\lambda x_0. \text{penguin}(y) \cap \text{with}(x_0)(y))) (\lambda x_2. \text{ate}(x_2)) \]

There is another derivation for this string, which is generated from

(50) \( \text{Exp: } \text{some penguin with he}_0 \text{ ate} \)
\[ \text{SYN: } s \]
\[ \text{CON: } \exists (\lambda y. \text{penguin}(y) \cap \text{with}(x_0)(y)) (\lambda y. \text{ate}(y)) \]

which, after the term insertion of every herring from (41), produces

(51) \( \text{Exp: } \text{some penguin with every herring ate} \)
\[ \text{SYN: } s \]
\[ \text{CON: } \forall (\text{herring}) (\lambda x_0. \exists (\lambda y. \text{penguin}(y) \cap \text{with}(x_0)(y)) (\lambda y. \text{ate}(y)) \]

These are the only two derivations possible for this expression. One of the scopings has the quantifier resolved within the nominal, and the other gives it wide scope.
There is no reading where every herring takes narrow scope with respect to some penguin and remains outside of the nominal.

Now consider the post-syntactic method of generating quantifier scopings. Here, we do not need to make any additional assumptions to resolve quantifiers within nominal categories. We can derive the following semantics for some penguin with every herring ate

\[
\text{ate}(GQ(\text{some}, x, (\text{penguin}(x) \land \text{with}(GQ(\text{every}, y, \text{herring}(y))))(x)))
\]
to which we can apply our rule for pulling once to produce

\[
\text{some } (\lambda z. (\text{penguin}(x) \land \text{with}(GQ(\text{every}, y, \text{herring}(y))))(x)) \quad (\lambda z. \text{ate}(x))
\]
This in turn, will reduce to

\[
\text{every } (\lambda y. \text{herring}(y)) \quad (\lambda y. \text{some } (\lambda x. \text{penguin}(x) \land \text{with}(y)(x)))
\]
Alternatively, we could reduce the interior quantifier first to give

\[
\text{ate}(GQ(\text{some}, x, \text{every } (\lambda y. \text{herring}(y)))) \quad (\lambda y. (\text{penguin}(x) \land \text{with}(y)(x)))
\]
which then reduces to

\[
\text{some } (\lambda x. \text{every } (\lambda y. \text{herring}(y))) \quad (\lambda y. (\text{with}(y)(x) \land \text{penguin}(x))) \quad (\lambda x. \text{ate}(x))
\]

It is in just this case when our condition on free variables in a term's restriction become important. For instance, we can get the partial scoping

\[
\text{every } (\lambda y. \text{herring}(y)) \quad (\lambda y. \text{ate}(GQ(\text{some}, x, \text{penguin}(x) \land \text{with}(y)(x))))
\]
but when we consider the restriction \((\text{penguin}(x) \land \text{with}(y)(x))\) of the remaining quantifier term, we see that it has a free variable \(y\) which is not a free variable of the expression in which it occurs, since \(y\) is bound there. Hence we can not reduce this term any further. In fact, the definition effectively blocks any unwanted derivations with free variables occurring in the fully scoped content.
Another complication arises in the post-syntactic method for the same reason as we are able to treat intensional verbs like *sought* in terms of compound logical forms (see Section 5.1.7). Thus, we will also be able to reduce our original term with quantifiers in place to

\[
\text{ate}(\text{GQ(some, } x, (\text{penguin}(x) \land \text{every } (\lambda y \cdot \text{herring}(y))))
\]
\[
(\lambda y \cdot \text{with}(y)(x))
\]

While this gets the intuitively correct results in the case of the negative quantifier *no*, it will also give our original result, thus making the expression ambiguous in three ways instead of two, also allowing the intuitively incorrect reading of the negative quantifier. Of course, we could assume that the lexical entry for every expression was a constant. For instance, we could give the nominal preposition *with* the semantic content \(\text{with}_i\). We could then take a meaning postulate such as

\[
\text{with}_i = \lambda y \cdot \lambda P \cdot \lambda x.(P(x) \land \text{in}(x)(y))
\]

to capture the value of \(\text{with}_i\). With this lexical entry we would derive

\[
\text{EXP: } \text{penguin with every herring}
\]
\[
\text{SYN: } n
\]
\[
\text{CON: } \lambda y \cdot \text{with}(\text{GQ(every, } x, \text{herring}(x)))(\text{penguin})(y)
\]

which could then be reduced only to

\[
\text{EXP: } \text{penguin with every herring}
\]
\[
\text{SYN: } n
\]
\[
\text{CON: } \lambda y \cdot \text{every } (\text{herring})
\]
\[
(\lambda x \cdot \text{with}(x)(\text{penguin})(y))
\]

Note that this is just the case where the actual form of the \(\lambda\)-term in the content is significant. With the meaning postulate (59), the two contents must have the same value. Montague's grammars were designed so that the semantic translation was insensitive to the form of the \(\lambda\)-term representation of an object in the meaning domain.
5.1.4 Quantification and Events

With our strategy of interpreting utterances of sentences as properties of events, there are a number of possible options for the semantics of generalised quantifiers. The added complication stems from the fact that the interpretation of an expression of category $n$ will be of type $(i, p)$, while an expression of category $s \setminus np$ will be of type $(i, (e, p))$. Our intuition is that a quantifier should take two arguments, one which has the type of a noun and the other which has the type of a verb phrase, and produces a result which has the type of a sentence. This is a key point of Barwise and Cooper's (1981) treatment of generalised quantifiers, even though they used Montague's type assignment for sentences. But, as we have it so far, all of our quantifiers are of type $(\langle i, p \rangle, \langle \langle i, p \rangle, p \rangle)$, so that they take two properties to form a proposition, rather than of type $(\langle i, p \rangle, \langle e, \langle i, p \rangle \rangle, \langle e, p \rangle))$, which seems necessary if we are going to preserve our event based semantics.

Suppose that $\text{every}_e$ is a constant of type $(\langle i, p \rangle, \langle e, \langle i, p \rangle \rangle, \langle e, p \rangle))$ that we will take as the content for the general determiner $\text{every}$. We would then have a content for $\text{every penguin eat}$ which looked like

\[(62) \lambda e. \text{every}_e (\lambda x. \text{penguin}(x)) . (\lambda x. \lambda e'. \text{eat}(x)(e')) (e)\]

Here we have two arguments to $\text{every}_e$, one of which is a noun content and one of which is a verb phrase content. The final $e$ abstracted is there just to highlight the fact that we have a property of events. Note that in $\lambda$-calculus, we have $Q_e(P)(Q) \leftrightarrow \lambda e. Q_e(P)(Q)(e)$ as a specific instance of $\eta$-conversion. We spend the rest of this section examining possible values for the constant $\text{every}_e$. We will continue to use the $e$ subscript to distinguish our event-based quantifiers from the previous ones which we will now write as constants with an $i$ subscript, so that $\forall$ becomes $\text{every}_i$.

Consider the simple relation eat introduced by the intransitive verb eat. eat is a constant of type $(i, (e, p))$ in our event based semantics. We think of eat$(x)(e)$ as being true just in case $e$ is an event of $x$ eating. The question arises as to whether there could be an event $e$ such that eat$(x)(e)$ and eat$(y)(e)$ hold for some $x \neq y$. If this were the case, then $e$ would be both an event of $y$ eating
and an event of $x$ eating. If this is a plausible way of looking at events, we might assume the following axiom for $\text{every}_e$

$$\text{every}_e(P)(V)(e) \leftrightarrow \text{every}_i(P)(V(e))$$

so that the content of $\text{every penguin eat}$ would come out as

$$\lambda e.\text{every}_i(\text{penguin}) (\lambda x.\text{eat}(x)(e))$$

An event $e$ will have this property just in case every individual $x$ such that $\text{penguin}(x)$ holds is also such that $\text{eat}(x)(e)$ holds. Consequently, a single event $e$ could have the property corresponding to the interpretation of both $\text{opus ate}$ and $\text{every penguin ate}$. While this is a very flat way of modeling the complex event in which every penguin is eating, it is the method most often used within situation semantics (see Barwise (1986), Gawron and Peters (forthcoming)).

The next alternative we consider is where we take $\text{every}_e(P)(V)(e)$ to hold just in case $e$ is the stative eventuality corresponding to every individual with the property $P$ participating in some event with the property $V(x)$. This is much like the case for the stative analysis of negation, where we would think of $\text{not}(P)(e)$ holding just in case $e$ is the state of there not being any events of type $P$. Of course, this leaves the actual value of $\text{every}_e$ open, but we would at least constrain it with the axiom

$$\forall \text{every}_e(P)(V)(e) > \forall \forall(P)(\lambda x.\exists(V(x)))$$

Recall that $>$ is just the standard material conditional. We would have $\forall p > \forall q$ just in case the truth of $p$ implies the truth of $q$. Also recall that an axiom in the form of an expression whose value is a truth value is taken to assert the truth of that expression, which is really just shorthand which saying its value is equivalent to the truth value 1. Under this analysis, there would be a state of every penguin eating just in case for every penguin there was some event of its eating.

Negative quantifiers such as $\text{no}$ serve as strong evidence that the interpretation of a sentence containing generalised quantifiers should hold of states corresponding to quantificational facts. Consider an utterance of

$$\text{No penguin ran.}$$
which must be given an interpretation as some property of events to correspond to our basic type assignment. There seems to be no other way to interpret such an utterance as making some kind of positive statement about events that exist in the world, which is necessary for our definition of truthful utterances. This is because if the interpretation of an utterance is the property \( P \) of events, then the utterance is truthful just in case \( \exists_1(P) \) which means that there must be some real \( e \) such that \( P(e) \) is true. This alternative may not seem so implausible if we were to follow Barwise and Etchemendy (1987) in making the interpretation of a sentence an Austinian proposition (see Section 4.5.5).

In particular, we could not have something like

\[
(67) \quad \lambda e.\forall(penguin)(\lambda x.\sim\exists_1(\lambda e'.(\text{eat}(x)(e') \cap e' \Downarrow e)))
\]

The reason why this is disallowed is that it is a property that holds of every event \( e \) which does not contain any subevents of penguins eating. Clearly, we do not want this generous definition of truth for utterances of negatively quantified sentences. Such an utterance would always be true as long as there were real events not containing sub-events of penguins eating.

Our final consideration will be the meaning postulate

\[
(68) \quad \text{\textasciitilde}\text{every}_e(P)(V)(e) \leftrightarrow \text{\textasciitilde}\text{every}_i(P)(\lambda x.\exists(V(x))(\lambda e'.e \Downarrow e'))
\]

In this case, the interpretation of \textit{every penguin eat} would be truth-conditionally equivalent to

\[
(69) \quad \lambda e.\text{\textasciitilde}\text{every}_i(penguin)
\]

\[
(\lambda x.\exists(\lambda e'.\text{eat}(x)(e')))
\]

\[
(\lambda e'.e \Downarrow e')
\]

An event would have this property just in case it was a group event which was composed of a number of individual events corresponding to the event of each penguin eating. Presumably, such a group event would not also have the property of being an eating of opus, but this is not ruled out by the assumption that it is a group event, because we have not required the properties of groups to be related to the properties of their constituents in any uniform way. Similarly, the group event may have properties that none of its constituents share (see Section 3.2.1). Note
that this strategy shares with the others the assumption that a truthful utterance of *every penguin ran* requires there to be some eating event for each penguin.

Consider the sentence

(70) Every penguin ate slowly.

There seem to be two possible interpretations of an utterance of this sentence (see, for instance, Thomason and Stalnaker (1973) and Richards (1976) for Montagovian analyses of these interpretations). The first would be where there were a number of penguins, each of which ate slowly. The second would be where there was a group event of each penguin eating which was itself slow, even though some of the penguins might have eaten quickly. If this latter reading were possible it would strongly suggest that something like the group reading of quantifiers was the correct one. It would only make sense where a dynamic event rather than a static one can have the property given by an interpretation of an utterance of *every penguin ate*, because *slowly*, like only modifies verb phrases which can be interpreted as properties of particular sorts of events. For instance, *slowly walk* is fine, but *slowly knew bill* sounds really odd due to the nature of knowing events. See Moens and Steedman (1987) for an analysis of the selectional restrictions placed on adverb and verb interpretations by aspectual considerations such as whether the event is dynamic like most runnings, or static like most knowings.

An alternative which is superficially similar to the last one would be to give a quantificational sentence an interpretation in terms of distributions rather than groups. In this case, the interpretation of an utterance of *every penguin ran* could be a property which holds of distributions of events containing the individual events of each penguin eating. For an utterance of *every penguin ran* to be true under this sort of interpretation, it is merely necessary for the event of each penguin eating individually to exist. this will guarantee that the distribution of the events will exist, since the domain of individuals, and hence of events, is closed under arbitrary distribution formation. We do not use this method, as it is not really in step with the type assignment to sentences or our general semantic system. We take utterances of sentences to be interpreted as properties of events in EVENT, not as properties of distributions of events. The real drawback

251
to this kind of schema is that our interpretation of distributions is downwardly monotonic. That is, we take a property to hold of a distribution if and only if it holds of all of the members of the distribution. This means that if \( P \) is property and \( P(d) \) holds for some distribution \( d \), then \( P(d') \) holds for every distribution \( d' \subseteq d \). This means that we can not actually define properties of events which are not downwardly monotonic, so that we could not have an interpretation \( P \) of every penguin ran which is true of the distribution containing an event of eating for every penguin, say \( d \) that is not also true of every subdistribution of \( d \). This obviously gives the wrong truth conditions for sentences, as no events would ever have to exist. Properties of events are just always true of the empty distribution.

5.1.5 Cooper Storage

The approach to quantification that we will take is based on the technique of Cooper storage (see Cooper (1982,1983), Bach and Partee (1980) and Partee and Bach (1984)). The principle behind Cooper storage is very much like that of our reference list. The meaning of an expression is extended to include not only a term for the interpretation, but also an additional list of quantifiers, called the store. Since we already have a reference list in our relational categorial grammar framework, we will simply extend this mechanism to encompass quantifier terms as well as discourse markers.

More precisely, we will suppose that a relational category can contain quantifier terms of the sort used in the post-syntactic system as well as discourse markers on its reference list. We then use the following relational category as a lexical entry for the quantifier every

\[
(71) \quad \text{EXP: every} \\
\text{SYN: np/n} \\
\text{CON: P.X} \\
\text{REF: } \langle \text{GQ(every, X, P)} \rangle \\
\text{BG: } \{\}
\]

We will use the following two reduction schemata for popping quantifiers off the reference list and applying them
These rules are necessarily schematic in that they encode a unification rule for each possible location of the quantifier term in the reference list. Finding such an embedded term cannot be carried out by simple unification using a list encoding such as the one we have here. Luckily, we will always be searching only a finite list in practice, so that this move to rule schemata does not introduce any disastrous computational complexity into derivations. The reason for using the reference list for two purposes should become clear in Section 5.2 when we introduce anaphora into the picture. For now, we will simply present some derivations in this grammar.

We start with the simple case of

(74) Every penguin love some herring.

for which we have the category

(75) | EXP: every penguin love some herring |
| Syn: s |
| Con: E.love(X)(Y)(E) |
| Ref: (GQ(every, Y, penguin(Y)), GQ(some, X, herring(X))) |
| BG: {} |

This category matches the left-hand side of the quantifier reduction rule with $R_1 = \emptyset$ and $R_2 = \langle GQ(some, X, herring(X)) \rangle$. This allows us to rewrite this category as the new category
Again, this category matches the quantifier reduction rule, this time with \( R_1 = R_2 = \{ \} \). Popping the quantifier term off the reference list according to the reduction rule then yields the category

(77) \[ \text{Exp: every penguin love some herring} \]
\[ \text{SYN: } s \]
\[ \text{CON: } E3.\text{some}_e \ (X.\text{herring}(X)) \]
\[ (X.E2.\text{every} \ (Y.\text{penguin}(Y))) \]
\[ (Y.E.\text{love}(X)(Y)(E)) \]
\[ (E2) \]
\[ (E3) \]
\[ \{ \} \]

The alternative scoping, with every penguin taking wide scope is generated by reducing the quantifiers in the opposite order.

With these quantifier reduction rules, we have not committed ourselves to any of the interpretations of the event quantifiers. Instead, we have taken the most general representation possible, which is compatible with all three approaches. As a general research strategy, we prefer to work at this level of generality so that the quantifier reduction mechanism will not stand or fall based on the lexical semantics we might assume for the quantifiers themselves. It should also be noted that our general reduction strategy would be compatible with a more traditional non-event based semantics for sentences such as that presented in the sections on term insertion and post-syntactic processing.

We now consider quantification inside of a nominal element, where the nominal

(78) \[ \text{penguin with every herring} \]

gets the category

254
(79) \[\text{EXP: } \textit{penguin with every herring} \]
\[\text{SYN: } n\]
\[\text{CON: } \forall X. (\text{penguin}(X) \land \text{with}(Y)(X))\]
\[\text{REF: } \langle \textit{GQ}(\text{every}, Y, \text{herring}(Y)) \rangle \]
\[\text{BG: } \{\}\]

This matches the reduction rule for nominal quantification, again with the remaining reference lists \(R_1\) and \(R_2\) being null. We can thus rewrite our category as

(80) \[\text{EXP: } \textit{penguin with every herring} \]
\[\text{SYN: } n\]
\[\text{CON: } \forall X. \textit{every}, (Y. \text{herring}(Y)) \]
\[\quad \quad (Y. (\text{penguin}(X) \land \text{with}(Y)(X)))\]
\[\text{REF: } \{\}\]
\[\text{BG: } \{\}\]

In this case, we have used the nominal quantifier, which we assume is just the standard universal quantifier so that, for instance \(\textit{every}, = \forall\). Similarly, we take \(\textit{some}, = \exists\) and \(\textit{no}, = \sim \exists\). We can then use the resulting reduced nominal category for an analysis of

(81) Some penguin with every herring eat.

by allowing the derivation

(82) \[\text{EXP: } \textit{some penguin with every herring eat} \]
\[\text{SYN: } s\]
\[\text{CON: } \exists E. \text{eat}(X)(E)\]
\[\text{REF: } \langle \textit{GQ}(\text{some}, X, \textit{every}, (Y. \text{herring}(Y)) \quad (Y. (\text{penguin}(X) \land \text{with}(Y)(X))) \rangle \]
\[\text{BG: } \{\}\]

Note that the already reduced nominal shows up inside the restriction of the subject quantifier. By reducing this category with our sentential quantifier rule, we are left with
Alternatively, we could postpone reducing the quantifier within the nominal element and derive the category

(84)  EXP: some penguin with every herring eat
SYN: s
CON: E.eat(X)(E)
REF: (GQ(some, X, (penguin(X) \cap with(Y)(X))), )
GQ(every, Y, herring(X))
BG: {}

from which we can pop the first quantifier to yield

(85)  EXP: some penguin with every herring eat
SYN: s
CON: E2.some_e (X.penguin(X) \cap with(Y)(X))
(\text{X.E.eat}(X)(E))
(E2)
REF: (GQ(every, Y, herring(X)))
BG: {}

so that by popping the second quantifier, we can derive

(86)  EXP: some penguin with every herring eat
SYN: s
CON: E3.every_e (Y.herring(Y))
(Y.E2.some_e (X.(penguin(X) \cap with(Y)(X))))
(X.E.eat(X)(E))
(E2)
(E3)
REF: {}
BG: {}

If we had tried to reduce the quantifiers in the opposite order, we would have been left with the category
which is a bit of an embarrassment. The occurrence of $Y$ in the restriction of some is not bound, because the variable $Y$ is bound in the restriction of some. The post-syntactic account eliminated this reading by placing a condition on the free variables of a term to be pulled and the scope that it would take if it were applied. Using our Cooper storage mechanism we can remove this analysis by restricting the definition of relational categories. Before we introduced quantification, we required the free variables of the content and background conditions of a category to be a subset of the list of discourse markers on the reference list. We extend this requirement to quantifier terms by requiring the free variables of the content and background conditions to be a subset of the reference markers and set of foci of quantifier terms, where the focus of a quantifier term $GQ(Q, X, R)$ is $X$. Note that (87) violates this condition because $Y$ is a free variable not occurring in the empty reference list. This condition can be easily checked before allowing a reduction to be carried out.

5.1.6 Quantification in Unsaturated Clauses

In the current framework, we have not allowed quantifiers to reduce, except inside of saturated sentences or nominals. In this section we will consider cases where such reductions are necessary to achieve the proper contents. Consider a sentence with an adverbial and a quantified subject, such as

(88) Every penguin probably ran.

With the lexical entries we have introduced, the only available analysis is as the category
The reason for this is that we must combine the adverb and the verb before the noun phrase can be attached. The only possible reduction of this category is to

Now consider the post-syntactic generation system, where we assume the following lexical entry

for probably. The constant prob is thus taken to be of type $(p, p)$. We then have the analysis

for every penguin probably ran. There are now two options for pulling the quantifier term out of this content. The first leaves it within the scope of probably, yielding

while the second pulls it outside of the whole term to give

(94) every $(X.\text{penguin}(X))$
    \( (X.\text{prob}(\text{ran}(X))) \)
which corresponds closely to the only scoping we can derive for the quantifier in the relational categorial grammar system. The first reduction corresponds to the interpretation where it is probable that all of the penguins ran. The second reduction leaves us with an interpretation where for every penguin, it is probable that it ran. While there may not seem like much intuitive difference between these readings, consider what would happen with negation in place of the adverbial in a sentence such as

(95) Every penguin did not eat.

While we have not given enough categories to derive this sentence within the post-syntactic framework, it is likely to be given an unreduced content such as

(96) \text{not}(\text{eat}(\text{QQ}(\text{every}, X, \text{penguin}(X))))).

Note that we have ignored the contribution of the auxiliary \textit{did}. This is because we did not introduce any mechanism for representing tense in our original fragment. Assuming this content, we get the two reductions

(97) \text{not}(\text{every} (X.\text{penguin}(X)) )
    
    (X.\text{eat}(X))

(98) \text{every} (X.\text{penguin}(X))
    
    (X.\text{not}(\text{eat}(X)))

The first reduction corresponds to the interpretation where it is simply not the case that every penguin ate. This could be true if some of the penguins did eat, as long as there was one which did not. The second reduction represents the interpretation where it is the case for every penguin that it did not eat. Notice that the truth of the second interpretation will imply the truth of the first.

In our relational categorial grammar, the only reading we get approximates the stronger reading in the second reduction. The situation for the term insertion system we have presented is the same and the reasons are similar. More specifically, an intermediate expression such as he \textit{probably ran} must be produced before term insertion can be applied to every penguin. At this point it is already too late to get the quantifier inside the scope of the adverbial.

There is a similar difficulty with the expression
Opus quickly ate every herring.

but for a different reason. Again, there is a possible interpretation of an utterance of this sentence where what was quick was the eating of every herring, rather than the eating of each herring. This could be the case if Opus had a very large supply of herring, say enough to last him a week, and he ate them all in one day. While he may not have eaten any one of the herring quickly, his slow but sure progress through all of the herring allows us to derive a truthful interpretation of the sentence. Of course, there is no problem with the reading that we can already get where the eating of each individual herring was quick.

Ignoring the tense and working with the base form verb eat, we can derive the verb phrase

\begin{align*}
\text{(100) EXP: } & \text{ eat every herring} \\ & \text{ SYN: } s \backslash \text{np} \\ & \text{ CON: } X.E.\text{eat}(Y)(X)(E) \\ & \text{ REF: } \{\text{cq(\text{every},Y,\text{herring}(Y))}\} \\ & \text{ BG: } \{\} 
\end{align*}

which we would like to be able to reduce to

\begin{align*}
\text{(101) EXP: } & \text{ eat every herring} \\ & \text{ SYN: } s \backslash \text{np} \\ & \text{ CON: } X.E2.\text{every}_e \ (Y.\text{herring}(Y)) \\
& \quad (Y.E.\text{eat}(Y)(X)(E)) \\
& \quad (E2) \\ & \text{ REF: } \{\} \\ & \text{ BG: } \{\} 
\end{align*}

This analysis is not possible because we can only pop something off of the quantifier stack in a category that is syntactically an s or an n.

To get around this problem, we introduce the pair of unary recursive rule schemata.
We can think of these recursive schemata in a number of different ways. The first is to treat them as inference rules in addition to the ordinary schemata which can be treated as axioms. In this way, we can maintain our usual deductive system interpretation of phrase structure under a natural generalisation from axiom schemata to rule schemata. Thus we read an inference rule of the form

\[(104) \frac{\gamma_1, \ldots, \gamma_n}{\gamma}\]

as stating that we can prove that \(\gamma\) is a valid rewriting whenever \(\gamma_1, \ldots, \gamma_n\) all are.

This is the approach to complex rules taken by Lambek (1958, 1961) and those following him such as Moortgat (1987, 1987b, 1987c, 1988), van Benthem (1986b, 1987) and Morrill and Carpenter (forthcoming).

Within the phrase structure tradition, another way of looking at such recursive rules has become popular. Within the GPSG framework pioneered by Gazdar (1981b), recursive rules are called metarules. Instead of looking at these recursive
rules as logical deductions, they are viewed as a recursive specification of an object grammar. Starting with a core grammar (our axioms) we can generate an enlarged set of rules (theorems) from the set of metarules (our recursive rules). If we look at these rules as logical statements, we see that the two approaches will be formally equivalent, in that the grammar closed under the metarules will simply be the set of provable statements under the deductive system interpretation. Of course, either way of looking at the recursive rules yields the same grammatical results. For a study of the generative power of systems involving metarules which meet various conditions, see Uszkoreit and Peters (1985).

Besides the recursive rules, quantifier rules are the only other unary rules allowing a category to be rewritten as a single other category. This means that the recursive rules will only interact with themselves and the quantifier reduction rules. But, such recursive rules are often augmented with rules of type-raising and other recursive rules to deal with grammatical phenomena such as coordination unbounded dependency constructions in questions and relative clauses, as can be found in Geach (1972), Moortgat (1987, 1987b, 1987c, 1988), van Benthem (1986b, 1987) and Morrill (1987, 1987b). Steedman (1987, 1988) and Dowty (1988), on the other hand, only make use of a finite number of instances of the more general recursive combination schemata.

Before giving some derivations with these rules, it should be pointed out that the antecedents of the recursive rules are simpler than their consequents in terms of the number of slashes that occur. This property is enough to show that a grammar augmented with these recursive rules will be decidable. The reason is that if we try to show that one category rewrites as another and try to apply a recursive rule, we are left with a strictly easier problem, where easiness is defined in terms of the number of slashes. For a related proof and grammar system in a binary rule setting, see Lambek (1958). In our case, this guarantees the eventual termination of any such search for derivations, leaving us with the result that any category can only be rewritten as finitely many different categories. For the rest of this section we will present a number of derivations which are made possible by this added recursive power.

First consider the example sentence (99) with the verb phrase derivation
By simply adding the subject and adverbial, we can derive the category

(106) Exp: opus quickly eat every herring
SYN: s
CON: E2.quick(E.eat(Y)(X)(E))(E2)
REF: (GQ(every,Y,herring(Y)))
BG: \{ name(opus)(X) \}

With this in hand, we can reduce the quantifier to produce the reading

(107) Exp: opus quickly eat every herring
SYN: s
CON: E3.every_e (Y.herring(Y))
\quad \quad (Y.E2.quick(E.eat(Y)(X)(E))(E2))
\quad \quad (E3)
REF: \{ \}
BG: \{ name(opus)(X) \}

This reduction captures the interpretation where each of the herring was eaten quickly. To produce the second interpretation, we must use the recursive rule. Matching the input verb phrase against the left hand side of the consequent, we see that we can derive

(108) Exp: eat every herring
SYN: s\ np
CON: X.Z
REF: R_2
BG: B_2

if we can find a rewriting rule of the form

(109) Exp: eat every herring  \rightarrow  Exp: eat every herring
SYN: s
CON: Z
REF: R_2
BG: B_2

\quad \quad Exp: eat every herring
SYN: s
CON: E.eat(Y)(X)(E)
REF: (GQ(every,Y,herring(Y)))
BG: \{ \}
In this case, we know that

\[(10)\text{ EXP: eat every herring} \]
\[\text{ SYN: s} \]
\[\text{ CON: } E2.\text{every}_e (Y.\text{herring}) \]
\[ (Y.E.\text{eat}(Y)(X)(E)) \]
\[ (E2) \]
\[\text{ REF: } \emptyset \]
\[\text{ BG: } \emptyset \]

rewrites as

\[(11)\text{ EXP: eat every herring} \]
\[\text{ SYN: s} \]
\[\text{ CON: } E.\text{eat}(Y)(X)(E) \]
\[\text{ REF: } \langle GQ(\text{every}, Y, \text{herring}(Y)) \rangle \]
\[\text{ BG: } \emptyset \]

so that we have

\[(12)\text{ EXP: eat every herring} \]
\[\text{ SYN: s np} \]
\[\text{ CON: } X.E.\text{eat}(Y)(X)(E) \]
\[\text{ REF: } \langle GQ(\text{every}, Y, \text{herring}(Y)) \rangle \]
\[\text{ BG: } \emptyset \]

analysed as

\[(13)\text{ EXP: eat every herring} \]
\[\text{ SYN: s np} \]
\[\text{ CON: } X.E2.\text{every}_e (Y.\text{herring}) \]
\[ (Y.E.\text{eat}(Y)(X)(E)) \]
\[ (E2) \]
\[\text{ REF: } \emptyset \]
\[\text{ BG: } \emptyset \]

This one reduction of the object of a transitive verb is enough to illustrate a number of different situations in which recursive reduction is desirable. First of all, it should be mentioned that exactly the same construction works for any number of pre-verbal modifiers. Consider the sentence

\[(14)\text{ Opus will have probably been eating every herring.} \]
We can now derive six independent meanings for this sentence depending on where in the string of $s \ np / (s \ np)$ categories the quantifier is reduced. By symmetry we can also reduce quantifiers at any stage in post-verbal modification, allowing three distinct readings of

(115) Opus ate every herring quickly yesterday.

The same reduction even works for mixed cases, giving us a grand total of six distinct readings for the sentence

(116) Opus probably ate every herring yesterday.

More readings are produced in this case due to the added ambiguity of which adverbial is first attached to the verb.

The next example that we consider is

(117) Opus wanted to eat every herring.

which is a subject controlled equi construction. The intuition here is that the sentence can either mean that Opus had a desire to eat an entire collection of herring or that for each herring it was the case that Opus wanted to eat it. The first case is where we would have the derivation

(118) \( \text{EXP: opus want to eat every herring} \)
    \( \text{SYN: s} \)
    \( \text{CON: E.want(E2.eat(Y)(X)(E2))(X)(E)} \)
    \( \text{REF: (GQ(every, X, herring(X)))} \)
    \( \text{BG: \{ name(opus)(X) \}} \)

It is then a simple matter to pop the quantifier off the reference list to produce

(119) \( \text{EXP: opus want to eat every herring} \)
    \( \text{SYN: s} \)
    \( \text{CON: E3.every}_e (Y.herring(Y)) \)
    \( \quad (Y.E.want(E2.eat(Y)(X)(E2))(X)(E)) \)
    \( \quad (E3) \)
    \( \text{REF: \{ \} } \)
    \( \text{BG: \{ name(opus)(X) \}} \)

But, if we start with the verb phrase analysis with the quantifier already reduced, we can produce the category
This represents the second possible interpretation. Similar analyses will work for other cases of control where there is a quantifier in the infinitival complement, such as

(121) a. Binkley persuaded Opus to eat every herring.
    b. Opus promised Binkley to eat every herring.
    c. Binkley believed Opus to have eaten every herring.

Now consider what happens with a quantifier in the subject or direct object position in control constructions, as seen in

(122) a. Binkley persuaded every candidate to withdraw.
    b. Bill promised every candidate to withdraw.
    c. Binkley believed every penguin to swim.

It can be verified that with the lexical entries provided, none of these constructions allows anything but wide scope for the quantifier over the entire content. While this meshes quite nicely with our intuitions in the first two cases of equi constructions, in the case of the raising verb believe we might think that the quantifier could reduce inside of the complement, with a reading where Binkley had a belief that every herring swam without having a belief about each of the individual herring that exist. Note that we get this sort of reading with the sentence

(123) Opus believed every penguin swam.

In this case, ignoring tense, we can produce the category
(124) EXP: every penguin swam
    SYN: s
    CON: E2.every_{e} (X.penguin(X))
         (X.E.swim(X)(E))
         (E2)
    REF: ()
    BG: {} for the complement. Incorporating this as an argument to believe produces the
category

(125) EXP: opus believe every penguin swam
    SYN: s
    CON: E3.believe (E2.every_{e} (X.penguin(X))
         (X.E.swim(X)(E))
         (E2)
         (Y)
         (E3)
    REF: ()
    BG: \{ name(opus)(Y) \}

While such a reading is not possible in the case of the raising category for believe, we could get this result with the lexical entry

(126) EXP: believe
    SYN: s(bse) \ np / s(inf)
    CON: (E.P).X.E2.believe(X)(E.P)(E2)
    REF: ()
    BG: {} The key to this solution is to take a verb which was originally looking for a
noun phrase followed by an infinitival verb phrase and give it an entry looking
for an infinitival sentence. In this case, the complement to believe would be
the entire clause every penguin to swim which can be analysed as a sentence.
Unfortunately, there are sound reasons in terms of the structure of the lexicon and
unbounded dependency constructions to prefer our original analysis (see Bresnan
(1982,1982b), Gazdar et al. (1985) and Carpenter (forthcoming)).

Our final consideration will be quantification within complex adverbial modi-
fiers, as seen in
(127) a. Opus ran to find every herring.

b. Opus ate while watching every show.

Take the first example. On its first reading, Opus would have had to run to find each of the herring. The second reading could be truthful if Opus had only run once in a quest to find every herring. Similarly, the second example can be interpreted as having Opus watching the shows one at a time and eating during each, with another reading where Opus watched all of the shows simultaneously and ate while he did it. The phrases find every herring and watching every show can be analysed in exactly the same manner as eat every herring, with the quantifiers either reduced or on the stack. Leaving the quantifiers on the stack, we get the analysis

(128) EXP: opus run to find every herring
    SYN: s
    CON: E2.(run(X)(E2) \cap to(E.find(Y)(E))(X)(E2))
    REF: (GQ(every,Y,herring(Y)))
    BG: { name(opus)(X) }

This, in turn, reduces to

(129) EXP: opus run to find every herring
    SYN: s
    CON: E3.every_e(Y.herring(Y))
        (Y.E2.(run(X)(E2) \cap to (E.find(Y)(E)) ))
        (X)
        (E2)
    REF: ()
    BG: { name(opus)(X) }

which is the first reading we expected. Using the reduced form of the verb phrase, we get the second reading
Of course, the second example (127)(b) has an identical analysis.

5.1.7 De Dicto Nominals

Consider the sentences

(131) a. Opus sought some unicorn.

b. Opus sought some vampire.

The meanings of these two sentences are quite different. One may be true while the other one is false. As we said in Section 5.1.3, this difference must be attributed to a difference in meaning between the noun phrase some unicorn and some vampire. In this section we will consider possible treatments of the verb sought and others like it such as visualise, worship and so on.

We will first deal a possible approach in the post-syntactic treatment. Suppose that we have a lexical entry

(132) EXP: sought
SYN: s \np / np
CON: λy.λx.try(find(y)(x))(x)

for the verb sought, where we assume that find is of type \(i, (i, p)\) and that try is of type \(i, (p, p)\).

Intuitively, try(x)(p) will be true just in case the individual x tries to make p a true proposition. With standard entries for unicorn and some we will be able to derive

(133) EXP: some unicorn
SYN: np
CON: gQ(some, x, unicorn(x))

269
and thus be able to get

\[ \text{EXP: opus seeks some unicorn} \]
\[ \text{SYN: } s \]
\[ \text{CON: try (opus)} \]
\[ (\text{find (opus)}) \]
\[ (GQ(some,y,unicorn(y))) \]

We can take this result and apply our rules for pulling in two ways, yielding

\[ \text{(135) some (unicorn)} \]
\[ (\lambda y. \text{try(opus)(find(opus)(y)))} \]

and

\[ \text{(136) try(opus)(some (unicorn))} \]
\[ (\lambda y. \text{find(opus)(y))} \]

The first of these is called the \textit{de re} reading of the object noun phrase and the second is called the \textit{de dicto} reading. The first interpretation is only true if there is some unicorn in the world which Opus tries to find. The second can be true if Opus tries to make the more complex proposition that there will be some unicorn that he will find true. This can happen without his having any particular unicorn in mind. That is, we can think of Opus as being content with finding anything with the same attributes as a unicorn, but not caring which particular instance he finds. Of course, since there are no unicorns, only the latter \textit{de dicto} reading stands a chance of being true.

Our lexical entry explicitly incorporates Quine's (1960) definition of the meaning of \textit{look for} in terms of the meaning of construction \textit{endeavour to find}. In this way, a uniform treatment can be provided for the sensitivity of \textit{seek} to the intensional value of its direct object. That is, the \textit{de dicto} reading is triggered by the propositional context of the semantic argument to the verbs meaning. A similar strategy was adopted as the basic strategy of generative semantics. See, for instance McCawley (1970,1973), Lakoff (1972) and Bach (1968). Also see Dowty (1979) for a critical survey of generative semantics. The basic problem of this sort of analysis is that it is not so clear that it will work in general for other transitive verbs such as \textit{worship} and \textit{imagine} in which it is not so easy to see how to analyse them in a sentential format (see Bennett (1974) and Partee (1974)).
Quine's analysis is not possible with Montague's grammar, as the arguments of sought will be inaccessible for the kind of manipulation necessary to resolve scope within in an embedded context if it occurs lexically. The reason we can not get inside the interpretation of sought is that Montague makes the assumption that the behaviour of the meaning component of a rule will be invariant over equivalent terms so that the logical form will be of no import. That is, any analysis with the complex logical form assigned to sought would be equivalent to one employing a lexical entry

\[(137)\] EXP: sought  
SYN: \(s \downarrow np/np\)  
CON: \(\lambda y.\lambda x.\text{seek}(y)(x)\)

with the meaning postulate

\[(138)\] \(\text{seek}(y)(x) \iff \text{try}(\text{find}(y)(x))(x).\)

Obviously, this alone will not be adequate to capture the de dicto readings within the term insertion system we have presented. There will still only be one possible scoping for a quantifier in the object position of a sentence such as opus sought some unicorn.

Montague's analysis of the de dicto readings of verbs like seek was quite different than the solution of the post-syntactic system. Instead of using a compound logical form, Montague (1974d) introduced a constant sought which is interpreted as a relation between individuals and generalised quantifiers. In our term insertion system, this translates into a lexical entry such as

\[(139)\] EXP: sought  
SYN: \(s \downarrow np/q\)  
CON: \(\lambda Q.\lambda x.\text{sought}(Q)(x)\)

The type of sought must then be \((\langle i, p \rangle, p, \langle i, p \rangle)\). With the compositionally motivated assumption that general noun phrases like some unicorn and some vampire receive different meanings, we have enough fine-grainedness to distinguish between the de dicto meanings of the sentences in (131). The meanings of the two sentences in (131) are then
(140) i. sought(opus)(\lambda P . \exists(unicorn)(P))
    ii. sought(opus)(\lambda P . \exists(vampire)(P))

These propositions are not necessarily identical, even if we assume that there is no property that any unicorn or vampire has in virtue of there being no unicorns or vampires.

This solution requires some transitive verbs like sought to be of a higher type than the entry we presented earlier for love. Again, Montague's goal of a uniform type assignment was served by raising the type of all of the transitive verbs to the type (((i,p),p), (i,p)) of sought. This move results in a lexical entry such as

(141) EXP: love
    SYN: s \ np / q
    CON: \lambda P.\lambda x.P(\lambda y.loved(x)(y))

for the garden variety transitive verbs, where the constant loved is of type \langle i, \langle i, p \rangle \rangle just as before. Just as with type-raised noun phrases, the procedure mapping a transitive verb to its type-raised equivalent is easily definable. Of course, this is usually implemented in terms of a simple constant love' for the lexical entry and a meaning postulate such as

(142) love(y)(x) \leftrightarrow love'(\lambda P.P(x))(y)

Note that this is necessary due to our type raising of simple individuals x to the property of properties \lambda P.P(x). Note that this meaning postulate does not restrict the behaviour of love' on generalised quantifiers not generated by individuals in this way.

Unfortunately, it remains an open problem to incorporate de dicto readings of the complements to intensional transitive verbs such as seek into the relational categorial grammar system presented here.

5.1.8 Attributive Nominals

Consider the sentences

(143) a. Opus hopes to find a penguin.
    b. Opus hopes Binkley finds a penguin.
In the first case, Opus will be satisfied with any penguin he finds, and in the second case, he will be content with any penguin that Binkley finds. These readings have been dubbed *attributive* by Donnellan (1966), as opposed to the earlier *referential* uses we introduced in Section 4.5. It seems that what is needed is to allow both the describing conditions of the noun phrase and the individual it describes to enter into the meaning of an expression. Strawson (1950) allowed only the descriptive conditions and Russell (1905) allowed only the individual described as the meanings of noun phrases. Fodor and Sag (1981) have recently argued for an approach similar to the one adopted here.

Attributive readings can not yet be generated by our grammar, since the individuals introduced by definites or indefinites are placed immediately on the reference list, which forces a wide-scope existential reading on them. Incorporating these readings requires a simple addition to the set of objects that can occur on the reference list as well as minor changes in the lexical entries for the previously referential determiners. Carrying out this change will not actually invalidate any of our old derivations, but simply add new ones to the stock of possibilities.

We begin by assuming that the reference list can also contain *descriptive terms* of the form $\text{DT}(X, R)$ where

(144) i. $X$ is an individual variable, the *focus*

ii. $R$ is a propositional term with a free occurrence of $X$, the *restriction*.

Thus the descriptive terms look like quantifier terms without the generalised quantifier relation. We think of a descriptive term in much the same way as a property of the form $X.R$, which we can think of as a description of a class of objects. With this additional kind of entry for the referent list we will need to extend our well-formedness conditions on categories. We allow the foci of descriptive terms to count among the variable binders just as the foci of quantifier terms. Our condition on well-formedness will now require every free variable in the content or background conditions to occur as either a reference marker or as a foci of a quantifier term or a descriptive term.
Next, we modify the lexical entries for the two referential determiners, giving them the new entries

\[
(145) \begin{align*}
\text{Exp: } a \\
\text{SYN: } np/n \\
\text{CON: } (X.P).X \\
\text{REF: } \langle \text{DT}(X,P) \rangle \\
\text{BG: } \{\}
\end{align*}
\]

and

\[
(146) \begin{align*}
\text{Exp: } the \\
\text{SYN: } np/n \\
\text{CON: } (X.P).Y \\
\text{REF: } \langle \text{DT}(Y,\text{the}(X.P)(Y)) \rangle \\
\text{BG: } \{\}
\end{align*}
\]

These are very similar to their original entries, with the difference coming in the location of the property introduced by the nominal complement. In our original category, this information went directly into the background conditions, but now it is being put into the restriction of a descriptive term on the reference list.

To allow for our original referential reading of noun phrases involving these referential quantifiers, we allow descriptive terms to simply contribute their conditions to the background and disappear. Thus, we need a rule schema

\[
(147) \begin{align*}
\text{Exp: } E & \rightarrow \text{Exp: } E \\
\text{SYN: } S & \rightarrow \text{SYN: } S \\
\text{CON: } C & \rightarrow \text{CON: } C \\
\text{REF: } R_1 \cdot R_2 & \rightarrow \text{REF: } R_1 \cdot \text{DT}(Y,R) \cdot R_2 \\
\text{BG: } \{R\} \cup B & \rightarrow \text{BG: } B
\end{align*}
\]

Most notably, this rule will allow us to derive our original categories for the referential determiners from the new ones, thus allowing us to derive

\[
(148) \begin{align*}
\text{Exp: } a \\
\text{SYN: } np/n \\
\text{CON: } (X.P).X \\
\text{REF: } \langle X \rangle \\
\text{BG: } \{P\}
\end{align*}
\]
Of course, this means that all of our old derivations involving these quantifiers remain valid.

To derive the attributive readings of these noun phrases, we will follow Fodor and Sag (1981) in invoking existential quantification. We allow the attributive term to turn into a full-fledged existential quantifier, using the rule schema

\[ \text{Exp: } E \quad \rightarrow \quad \text{Exp: } E \]

\[ \text{SYN: } S \quad \quad \text{SYN: } S \]

\[ \text{CON: } C \quad \quad \text{CON: } C \]

\[ \text{REF: } R_1 \cdot \text{GQ(some, } Y, R) \cdot R_2 \quad \text{REF: } R_1 \cdot \text{DT}(Y, R) \cdot R_2 \]

\[ \text{BG: } B \quad \quad \text{BG: } B \]

Both of these rules are necessarily schematic in that they allow the reduction to take place on descriptive terms at any location in the reference list. This latter rule allows us to transform our referential determiners into generalised quantifiers, admitting the category

\[ \text{Exp: } a \]

\[ \text{SYN: } np/n \]

\[ \text{CON: } (X.P).X \]

\[ \text{REF: } \langle \text{GQ(some, } X, P) \rangle \]

\[ \text{BG: } \{ \} \]

This category will then allow the first sentence in

1. Opus thinks a penguin disappeared.
2. Opus thinks some penguin disappeared.

to have all of the readings that the second one could have. In particular, it will have a reading structurally similar to that in (125).

Rather than going through the intermediate stage of a descriptive term, we could have simply assumed that the referential determiners were lexically ambiguous between their original meanings and the quantified entries that are produced by this rule. The reason we choose to go through the intermediate descriptive term is that it is a more general approach and can be easily modified in the light of additional uses for descriptions.

Other places in which we can use descriptive terms are in the times introduced with occurrences of verbs. The only reading we get for
(152) Opus thought Binkley swam.

is where Opus has a belief that a swimming took place at a particular time. There is a discussion of this particular reading and its relevance to discourse representation theory in Partee (1984). The reason we only get this reading is because the temporal marker gets introduced as a condition on the content of the complement and has its own condition entered into the background. Verbs could have lexical entries like

(153) \textbf{EXP}: sang  \
\textbf{SYN}: s(fin) \setminus np  \
\textbf{CON}: X.E.(\text{sing}(X)(E) \cap \text{at}(E)(T))  \
\textbf{REF}: \langle DT(T, \text{before}(T)(T_{sang})) \rangle  \
\textbf{BG}: \{\}

which would then map into our original entry

(154) \textbf{EXP}: sang  \
\textbf{SYN}: s(fin) \setminus np  \
\textbf{CON}: X.E.(\text{sing}(X)(E) \cap \text{at}(E)(T))  \
\textbf{REF}: \langle T \rangle  \
\textbf{BG}: \{ \text{before}(T)(T_{sang}) \}\}

or the attributive version

(155) \textbf{EXP}: sang  \
\textbf{SYN}: s(fin) \setminus np  \
\textbf{CON}: X.E.(\text{sing}(X)(E) \cap \text{at}(E)(T))  \
\textbf{REF}: \langle \text{GQ(some, } T, \text{before}(T)(T_{sang})) \rangle  \
\textbf{BG}: \{\}

The temporal quantifier would then behave just like any other category with a quantifier in it. In particular, it will allow quantifier scope ambiguities with times. This means that the sentence

(156) Every penguin sang.

can now be analysed as the category
(157) EXP: every penguin sang
SYN: s(fin)
CON: E.(sing(X)(E) \( \cap \) at(E)(T))
REF: \{ GQ(every,X,penguin(X)), GQ(some,T,before(T)(Tsang)) \}
BG: {} 

Of course, this category can be reduced in two ways, depending on the order in which the quantifier terms are popped from the reference list. Popping the temporal quantifier first gives it narrow scope and corresponds to the reading where there might be a different time at which every penguin ran. The reverse order would require every penguin to have run at the same time, but would still have an attributive reading of the tense. This comes in handy in cases such as (152) where the sentence shows up as a complement to an attitude verb. We will now be able to get a reading of Opus believes every penguin sang which allows Opus to not have beliefs about the times at which penguins sang, except that they were before the present.

We can use this same technique to provide *de dicto* readings for the events in the complex adverbials discussed in Section 4.5.9. For instance, we could use the entry

(158) EXP: while
SYN: s(V) \( \setminus \) np \( \setminus \) (s(V) \( \setminus \) np) / (s(pred) \( \setminus \) np)
CON: (X.E.P).(X.E2.P2).X.E2.(P2 \( \cap \) during(E)(E2))
REF: \{ dt(E,P) \}
BG: {} 

for *while*. With this entry, a sentence like

(159) Binkley believes Opus ate while speaking.

could have a reading where Binkley does not have a belief about a particular speaking event, but just believes that there was some speaking event which overlapped some eating event.
5.2 Dependency

In this section we will extend our relational categorial grammar to account for simple dependencies between the interpretations of phrases in a larger expression. The simplest kind of dependency is that of anaphoric dependency, where a pronoun is taken to refer to an individual introduced by another noun phrase. Consider the sentences

(160) a. Steve loved \textit{himself}.
   
   b. Opus believed \textit{he} swam.

Both of these examples can be interpreted as requiring the italicised pronoun to be the same thing as the subject of the main clause. Of course, these interpretations were already possible with deictic uses of pronouns. We could have just used two discourse markers for the same individual, one introduced by the subject and the other by the pronoun. But now consider a case with quantification, as in

(161) a. Every lawyer loves \textit{himself}.
   
   b. Every penguin believes \textit{it} swam.

There is an interpretation of the first sentence where what is being asserted is that for every lawyer, that particular relation stands in a loving relation with himself. Similarly, the second example can mean that every penguin is such that that penguin believes that it swam. These examples are paradigm cases of dependency, where the interpretation of one element of an expression, in this case a pronoun, depends on the interpretation of another expression. So far we have no mechanism in our grammar capable of capturing such interpretations. This section is intended to remedy this situation. Along the way we will introduce a large number of other sentences whose semantics necessitate some representation of dependency.

Before beginning, we should point out that we will only be dealing with cases of intra-sentential anaphora and not inter-sentential cases such as

(162) Opus ate the herring. He thought it was tasty.
In these cases we have two anaphoric pronouns, *he* and *it*, referring back to individuals introduced in the previous sentence. We restrict ourselves to dependencies within sentences due to practical constraints of space and energy. It should be fairly straightforward to extend our account to cases of dependency across sentences, although a number of other phenomena will have to be dealt with such as narrative structure.

### 5.2.1 PTQ

First off, we will consider the treatment of dependency in Montague's PTQ system. Montague handled dependencies with exactly the same mechanism as quantification, namely term insertion. As we originally stated the term insertion rules in (11) and (39), they only operated over expressions with exactly one occurrence of a dummy pronoun with the appropriate subscript. This dummy pronoun was then replaced by a noun phrase in the expression component of the term insertion rule. To handle the simplest kinds of anaphora in PTQ we need merely relax this restriction to allow for one or more occurrences of a subscripted pronoun, where the first occurrence will be replaced with the noun phrase expression and subsequent occurrences will be replaced with a pronoun of the appropriate form.

More precisely, we use the term insertion rule

\[
\frac{\langle \delta_1, q, Q \rangle \langle \delta_2, s, S \rangle}{(F(\delta_1, \delta_2, k), s, Q(\lambda x_k.S))}
\]

where \( he_k \) is a subexpression of \( \delta_2 \) and where \( F(\delta_1, \delta_2, k) \) is the result of replacing the first occurrence of \( he_k \) in \( \delta_2 \) with \( \delta_1 \) and each subsequent occurrence of \( \delta_2 \) with the "appropriate" pronoun. For instance, some cases of the behaviour of \( F \) are

\[
\begin{align*}
\text{i. } & F(\text{every penguin}, he_0 \text{ loves } he_0, 0) \\
& = \text{every penguin loves himself} \\
\text{ii. } & F(\text{every penguin}, he_{10} \text{ believes } he_{10} \text{ swam}, 10) \\
& = \text{every penguin believes it swam} \\
\text{iii. } & F(\text{every penguin}, he_{17} \text{ loves } he_{16}, 17) \\
& = \text{every penguin loves } he_{16}
\end{align*}
\]
Of course, deciding which English pronoun to substitute for a given expression is a non-trivial problem. For instance, the gender of the pronoun is not determined by the meaning of the noun phrase to which it is bound, but by the particular interpretation of that noun phrase in a context. Simply consider the pronouns in

(165) a. Some packrat saved her notes.
   
b. Some packrat stored its food.

It is also non-trivial to determine the distribution of reflexive pronouns. Consider the choice between the reflexive and the ordinary pronoun in

(166) a. Every lawyer loves \{ \{ \text{himself} \}, \{ \ast \text{him} \} \}
   
b. Every lawyer believes \{ \{ \text{he} \}, \{ \ast \text{himself} \} \} is important.

This selection is one of the roles of the binding theory of the government-binding paradigm (see Reinhart (1976, 1983), Chomsky (1980, 1981, 1982) and Aoun (1985). Reflexive distribution has also been studied thoroughly in other grammar formalisms such as Montague grammar (Bach and Partee 1980, Partee and Bach 1984), generalized phrase structure grammar (Pollard and Sag 1983), head-driven phrase structure grammar (Pollard and Sag 1987, Sag and Pollard 1987) unification categorial grammar (Popowich 1987) and lexical-functional grammar (Bresnan 1982, Mohanan 1983).

Note that the ungrammaticality marking here is for the case where the pronoun is dependent on the subject. In other semantic contexts, these pronouns could be acceptable. Finally, there is the matter of case, where we have

(167) a. \{ \{ \text{He} \}, \{ \ast \text{Him} \} \} loves \{ \{ \text{she} \}, \{ \ast \text{her} \} \}.
   
b. He wants her to love him.

The pronouns he and she show up in subject position, while him and her only show up in object position.

We now turn to the semantics associated with our extended rule of term insertion. Consider the two analyses
(168) EXP: he\textsubscript{0} loves he\textsubscript{0}
SYN: s
CON: love(x\textsubscript{0})(x\textsubscript{0})

(169) EXP: he\textsubscript{17} believes he\textsubscript{17} swam
SYN: s
CON: believe(swam(x\textsubscript{17}))(x\textsubscript{17})

Note that we have assumed that believe is of category \(s \setminus np/s\), with an associated constant believe of type \((p,(i,p))\). We can apply term insertion to both of these categories to produce

(170) EXP: every lawyer loves himself
SYN: s
CON: \(\forall\text{(lawyer)}(\lambda x_0.\text{love}(x_0)(x_0))\)

(171) EXP: every penguin believes it swam
SYN: s
CON: \(\forall\text{(penguin)}(\lambda x_{17}.\text{believe}(x_{17})(\text{swam}(x_{17})))\)

It is quite convenient to have the same mechanism for the treatment of quantification as anaphora. The complication arises in the complex notion of operation over expressions and the surface form of expressions. We will see in the next sections some dependency phenomena that simple term insertion is not powerful enough to deal with, and propose a method for treating these in the relational categorial grammar fragment. We will not detail any treatment of anaphora in the post-syntactic framework.

5.2.2 Cooper Storage

In this section we will expand our framework of relational categorial grammar to account for the same range of phenomena as that of term insertion. We will proceed in a number of stages.

To begin, we assume that in addition to discourse markers and quantifier terms, our reference list will also be able to hold anaphor terms of the form \(\text{AN}(X,R)\) where
(172) i. $X$ is an individual variable, the *focus*

ii. $R$ is a propositional term with a free occurrence of $X$, the *restriction*.

Thus the dependent terms look like quantifier terms without the generalised quantifier relation. We will again have to extend our well-formedness condition on variables in categories to allow for the fact that free variables in the content or background conditions may appear as the foci of anaphor terms. Thus, our final well-formedness condition requires the set of free variables in the content and background conditions to be a subset of the set of discourse markers and foci of the descriptive, anaphor and quantifier terms.

We next extend our lexicon by including terms for the dependent pronouns, using entries such as

(173) EXP: *he*
    SYN: np(subj)
    CON: $X$
    REF: $\langle\text{AN}(X,\text{male}(X))\rangle$
    BG: {}

(174) EXP: *her*
    SYN: np(obj)
    CON: $X$
    REF: $\langle\text{AN}(X,\text{female}(X))\rangle$
    BG: {}

(175) EXP: *it*
    SYN: np(C)
    CON: $X$
    REF: $\langle\text{AN}(X,\text{true})\rangle$
    BG: {}

Recall the simple English agreement categorial grammar introduced in Section 2.3.5. All of the noun phrases were marked not only for number but also for case. Ignoring number, this gives us the two fully instantiated noun phrase possibilities np(subj) and np(obj). With pronouns given syntactic categories of the above form, it is a simple matter to encode case agreement on other categories. For
instance, we assume a transitive verb such as love has the syntactic category 
$s \backslash np(subj) / np(obj)$. This requires its object to be of the category $np(obj)$ and its subject to be of the category $np(subj)$. Of course, a pronoun such as it or a quantified noun phrase such as every penguin with category $np(C)$ will be able to fill either position. While we have not given the case markings for all of our complex verbs, the general strategy of marking all of the subject position noun phrases as $np(subj)$ and all of the other noun phrases as $np(obj)$ is adequate for English. In fact, all of the noun phrases that are looked for in the forward direction are objects, and all of those looked for backward are subjects, but this is just a coincidence of the subject-verb-object order of English. Other languages that employ free word order have much more elaborate systems of case agreement, but still pose no problem for a unification based approach. A number of examples drawn from languages with interesting case systems are given in Pollard and Sag (1988) and Barlow (1988, 1988b). For the sake of brevity we will omit all of the case-marking information from now on, as it is not really relevant to the semantic task at hand.

To complete the analysis of simple nominal anaphoric dependencies, we need additional grammar rules to bind the dependents to their antecedents. To this end, we introduce the four similar rules

(176)  
\[
\text{EXP: } E \\
\text{SYN: } S \\
\text{CON: } C \\
\text{REF: } R_1 \cdot Y \cdot R_2 \cdot R_3 \\
\text{BG: } B \\
\rightarrow \text{EXP: } E \\
\text{SYN: } S \\
\text{CON: } C \\
\text{REF: } R_1 \cdot Y \cdot R_2 \cdot AN(Y, Q) \cdot R_3 \\
\text{BG: } B
\]
To get a feel for what is going on, consider the simplest form of dependency displayed in the sentence

(180) Every lawyer loves himself.

We will simply ignore the differences between the ordinary pronoun *him* and the reflexive *himself*, treating them as variant spellings of the same category, which we
write as *him*(self). Of course, as we noted at the end of Section 5.2.1, the selection of reflexive as opposed to non-reflexive pronouns is a far from trivial matter, but it is not our main concern here.

For material on the distribution of reflexive pronouns compatible with what is presented here, see Pollard and Sag (1987) and Popowich (1987). Our new lexical entry allows us to derive the category

(181) \[ \text{Exp: } \textit{every lawyer love him}(self) \]
\[ \text{Syn: } s \]
\[ \text{Con: } E.\text{love}(Y)(X)(E) \]
\[ \text{Ref: } \{ \text{GQ}(\text{every}, X, \text{lawyer}(X)), \text{AN}(Y, \text{male}(Y)) \} \]
\[ \text{BG: } \{ \} \]

Applying the anaphora reduction schema for quantifiers, we can then derive

(182) \[ \text{Exp: } \textit{every lawyer love him}(self) \]
\[ \text{Syn: } s \]
\[ \text{Con: } E.\text{love}(X)(X)(E) \]
\[ \text{Ref: } \{ \text{GQ}(\text{every}, X, \text{lawyer}(X)) \} \]
\[ \text{BG: } \{ \} \]

by unifying the foci \( X \) and \( Y \). This is the import of using the same variables in the foci of the rule schema. From this, we simply use our normal quantifier schema to derive

(183) \[ \text{Exp: } \textit{every lawyer love him}(self) \]
\[ \text{Syn: } s \]
\[ \text{Con: } E2.\text{every}_e (X.\text{lawyer}(X)) \]
\[ (X.\text{E.love}(X)(X)(E)) \]
\[ (E2) \]
\[ \text{Ref: } \{ \} \]
\[ \text{BG: } \{ \} \]

Next, consider the sentence

(184) Opus loved a herring with him\((self)\).

In this case, we can derive the category
(185) \[ \text{EXP: opus love a herring with him(self)} \]
\[ \text{SYN: s} \]
\[ \text{CON: E.love}(Y)(X)(E) \]
\[ \text{REF: } \langle X, \text{DT}(Y, \text{herring}(Y) \cap \text{with}(Z)(Y)), \text{AN}(Z, \text{male}(Z)) \rangle \]
\[ \text{BG: } \{ \text{name}(opus)(X) \} \]

By reducing the descriptive term, we get

(186) \[ \text{EXP: opus love a herring with him(self)} \]
\[ \text{SYN: s} \]
\[ \text{CON: E.love}(Y)(X)(E) \]
\[ \text{REF: } \langle X, Y, \text{AN}(Z, \text{male}(Z)) \rangle \]
\[ \text{BG: } \{ \text{herring}(Y) \cap \text{with}(Z)(Y), \text{name}(opus)(X) \} \]

We can then resolve the anaphor term to produce

(187) \[ \text{EXP: opus love a herring with him(self)} \]
\[ \text{SYN: s} \]
\[ \text{CON: E.love}(Y)(X)(E) \]
\[ \text{REF: } \langle X, Y \rangle \]
\[ \text{BG: } \{ \text{herring}(Y) \cap \text{with}(X)(Y), \text{name}(opus)(X) \} \]

by unifying \( X \) and \( Z \). This time we had a choice as to which term to unify with the focus of the anaphor term. Also, had we carried out the anaphora resolution first and then done the description reduction, we would have been left with the same result. Notice that in our rule, the condition on the anaphor term's focus disappears once it is unified with its antecedent. We use this analysis because we view the conditions on anaphor foci as discourse conditions rather than interpretative conditions. That is, we assume, following Landman (1986), Barlow (1988,1988b), and Pollard and Sag (1988), that the purpose of the marking on an anaphor is to help the hearer choose an appropriate antecedent. In this case, with the property of being male on the anaphor, we would be able to choose the appropriate antecedent. Such a choice of antecedent is not something we wish to deal with here, so we will leave off any discussion of procedures for carrying out such anaphora resolution (but see Winograd (1972), Mellish (1981,1985), Haddock (1987,1987b), Hobbs (1978,1979), Webber (1979,1983), Grosz (1977,1986), Sidner (1983), and Grosz and Sidner (1986)). We are only interested in the logical possibilities, and those seem to be unconstrained by the ontological status of the objects in question (see Barlow (1988,1988b)).
Nothing changes in the case of adverbials, where the sentence

(188) Opus ran with himself.

will produce the category

(189) EXP: opus run with him(self)
SYN: $s$
CON: $E. (run(X)(E) \cap with(Y)(E))$
REF: $(X, \text{AN}(Y, \text{male}(Y)))$
BG: \{ \text{name}(opus)(X) \}

We can then resolve the anaphora by unifying $X$ and $Y$, leaving us with

(190) EXP: opus run with him(self)
SYN: $s$
CON: $E. (run(X)(E) \cap with(X)(E))$
REF: $(X)$
BG: \{ \text{name}(opus)(X) \}

In more complicated cases of control, things are still the same, with the sentence

(191) binkley persuade opus to love him(self).

being assigned the category

(192) EXP: binkley persuade opus to love him(self)
SYN: $s$
CON: $E2. \text{persuade}(Y)(E. \text{love}(Y)(Z))(X)(E2)$
REF: $(X, Y, \text{AN}(Z, \text{male}(Z)))$
BG: \{ \text{name}(binkley)(X), \text{name}(opus)(Y) \}

This can then reduce to either

(193) EXP: binkley persuade opus to love him(self)
SYN: $s$
CON: $E. \text{persuade}(Y)(E2. \text{love}(Y)(X)(E2))(X)(E)$
REF: $(X, Y)$
BG: \{ \text{name}(binkley)(X), \text{name}(opus)(Y) \}

or

287
depending on which antecedent the anaphor is unified with.

Things begin to get slightly more interesting in the case of

(195) some student believed every lawyer loves him(self).

In this case, we can analyse the complement every lawyer loves him(self) just as in
(183), which can then contribute to the overall sentence in allowing the derivation

(196) EXP: some student believe every lawyer loves him(self)
SYN: s
CON: E.believe (E2.every (X.lawyer(X))
(X.E.(love(X)(X)(E) ∩ at(E)(T)))
(E2)
(Y)
(E)
REF: (GQ(some, Y, student(Y)))
BG: { before(T)(E\text{loved}) }

We can then pop the quantifier off the reference list to produce the category

(197) EXP: some student believe every lawyer loved him(self)
SYN: s
CON: E3.some (Y.student(Y))(Y.E.\phi)(E3)
REF: {}
BG: { before(T)(E\text{loved}) }

where

(198) \phi = believe (E2.every (X.lawyer(X))
(X.E.(love(X)(X)(E) ∩ at(E)(T)))
(E2)
(Y)
(E)

Another alternative would be to reduce the quantifier within the complement sentence, but use the main subject quantifier term as the antecedent of the pronoun. We could also leave the quantifiers and anaphor unresolved until after the entire constituent structure is built up to derive the category
(199) **EXP:** some student believe every lawyer loved him(self)
**SYN:** s
**CON:** \[ E2.\text{believe}(E.\text{love}(X)(Z)(E))(Y)(E2) \]
**REF:** \( \{ \text{GQ}(\text{some},Y,\text{student}(Y)), \text{GQ}(\text{every},X,\text{lawyer}(X)), \text{AN}(Z,\text{male}(Z)) \} \)
**BG:** \( \{ \text{before}(T)(E\text{loved}) \} \)

From here, the antecedent can be either of the generalised quantifiers and the quantifiers can reduce in either order, for a total of four readings where the quantifiers take widest possible scope. This gives a total of six different readings for this sentence.

5.2.3 Polyadic Quantification

We will now turn to the analysis of so-called “donkey sentences” such as

(200) a. every student who buys a book reads it.

b. every student with a book read it.

which were introduced by Geach (1962). One of the outstanding features of Discourse Representation Theory is its ability to produce the right analysis of these sentences (see Kamp (1984)). The puzzle is how to give the indefinite noun phrase universal import. Consider the second sentence not involving a relative clause, which is the one we will analyse here. It can be uttered truthfully only if for every student and every book the student is with, it is the case that the student read the book. Somehow, the existential force of the indefinite becomes universal when it falls inside the scope of a universal quantifier.

Our analysis hinges on a polyadic treatment of universal quantifiers (see Fensstad et al. (1987)) for a similar treatment). The content we wish to derive is

(201) \[ E.\text{every}_e ((X,Y).((\text{student}(X) \cap \text{book}(Y) \cap \text{with}(Y)(X)))) \]

\[ ((X,Y).E2.\text{read}(Y)(X)(E2)) \]

\[ (E) \]

The intended semantics of every\(_e\) in this binary case is the obvious generalisation of the unary semantics. More precisely, we just treat tuples as objects which are being quantified over. We then have
if and only if \( E \) is the event corresponding to every \( X \) and \( Y \) such that \( \phi(X)(Y) \) being such that there is an event \( E_2 \) such that \( \psi(X)(Y)(E_2) \) (see Section 5.1.4 for further details).

To account for this kind of dependency, we introduce another unary reduction schema

\[
\text{(203) Exp: } E \\
\text{Syn: } S \\
\text{Con: } C \\
\text{Ref: } R_1 \cdot \text{gq}(Q, (Y, X), (P \cap Q)) \cdot R_2 \cdot R_3 \\
\text{B: } B
\]

In the case of (200), we can derive the category

\[
\text{(204) Exp: } \text{every student with a book read it} \\
\text{Syn: } s \\
\text{Con: } E.\text{read}(Z)(X)(E) \\
\text{Ref: } \{ \text{gq(every, } X, (\text{student}(X) \cap \text{with}(Y)(X))), \} \\
\quad \text{dt}(Y, \text{book}(Y)), \\
\quad \text{an}(Z, \text{neuter}(Z)) \\
\text{B: } \{\}
\]

We can resolve the anaphora to produce the category

\[
\text{(205) Exp: } \text{every student with a book read it} \\
\text{Syn: } s \\
\text{Con: } E.\text{read}(Y)(X)(E) \\
\text{Ref: } \{ \text{gq(every, } X, (\text{student}(X) \cap \text{with}(Y)(X))), \text{dt}(Y, \text{book}(Y)) \} \\
\text{B: } \{\}
\]

This matches the left hand side of our new rule schema, which allows us to derive
(206) EXP: every student with a book read it  
SYN: s  
CON: E.read(Y)(X)(E)  
REF: \( \langle GQ(every, (X, Y), ((student(X) \land with(Y)(X)) \land book(Y))) \rangle \)  
BG: {}  
This matches our original quantifier rule, as nothing will preclude a tuple from unifying as the focus of a quantifier term. Our final result will then be the category (207) EXP: every student with a book read it  
SYN: s  
CON: E2.every (X, Y).((student(X) \land with(Y)(X)) \land book(Y))  
\( \langle X, Y \rangle .E.read(Y)(X)(E) \)  
\( (E2) \)  
REF: {}  
BG: {}  
This has exactly the content that we were after.

Binary quantification alone is not sufficient to handle all possibilities. The sentence (208) Every teacher near a boy with a book gave him it.

is enough to show that at least ternary quantification is necessary, and we see no reason to stop there. As it is, our system will handle this sentence in exactly the same way as the binary case. First, the anaphors are resolved and then the descriptive terms introduced by a book and a boy will be added to the quantifier term.

We will also need the generality to allow a descriptive term with a definite noun phrase to become bound by a quantifier. Consider the sentence (209) The chef in every restaurant cooked.

which has a reading where each restaurant has its own unique cook. We can derive the category (210) EXP: the chef in every restaurant cooked  
SYN: s  
CON: E.cook(X)(E)  
REF: \{ DT(X, the(X)(Y.chef(Y) \land in(Y)(X))), \}  
\( GQ(every, Y, restaurant(Y)) \)  
BG: {}
Unfortunately, our polyadic quantification schema only works in one direction so far, so we need to add the rule schema

\[
\begin{align*}
(211) \quad \text{EXP: } & E \\
\text{SYN: } & S \\
\text{CON: } & C \\
\text{REF: } & R_1 \cdot R_2 \cdot \text{GQ}(Q, (Y, X), (P \cap Q)) \cdot R_3 \\
\text{BG: } & B \\
\rightarrow \quad \text{EXP: } & E \\
\text{SYN: } & S \\
\text{CON: } & C \\
\text{REF: } & R_1 \cdot \text{DT}(X, Q) \cdot R_2 \cdot \text{GQ}(Q, Y, P) \cdot R_3 \\
\text{BG: } & B
\end{align*}
\]

to allow us to incorporate descriptive terms that precede quantifiers. This schema then lets us reduce our category to

\[
\begin{align*}
(212) \quad \text{EXP: } & \text{the chef in every restaurant cooked} \\
\text{SYN: } & s \\
\text{CON: } & E.\text{cook}(X)(E) \\
\text{REF: } & \langle \text{GQ(every, (Y, X), (restaurant(Y) \cap \text{the(X)}(Y.(chef(Y) \cap \text{in(Y)(X)))))} \rangle \\
\text{BG: } & \{\}
\end{align*}
\]

Finally, we can pop the quantifier from this category to get the desired reading.

5.2.4 Sloppy Anaphora

Our final topic will be the cases of so-called "sloppy" anaphora, where the pronoun used is often called a pronoun of "laziness" (this latter term is also due to Geach (1962)). A typical case of sloppy anaphora is

\[
(213) \quad \text{Only John believes he won.}
\]

There are two possible interpretations for an utterance of this sentence. The traditional "strict" reading is where only John had the belief that John won. Our mechanisms can already account for this kind of reading. The reading we are interested in here is the sloppy reading in which John was the only one who was not such that he thought he himself won. In this case, everyone but John could
believe that John won, but it would have to be the case that Fred did not believe
Fred won, Jane did not believe Jane won and so on.

For cases of sloppy anaphora, we will slightly amend the unary recursive rules
(102) and (103). Our new recursive rules are

\begin{align*}
(214) \quad \text{Exp: } D & \rightarrow \text{Exp: } D \\
\text{Syn: } C_1 & \rightarrow \text{Syn: } C_1 \\
\text{Con: } Y & \rightarrow \text{Con: } Z \\
\text{Ref: } (X) \cdot R_1 & \rightarrow \text{Ref: } (X) \cdot R_2 \\
\text{BG: } B_1 & \rightarrow \text{BG: } B_2
\end{align*}

\begin{align*}
(215) \quad \text{Exp: } D & \rightarrow \text{Exp: } D \\
\text{Syn: } C_1 & \rightarrow \text{Syn: } C_1 \\
\text{Con: } Y & \rightarrow \text{Con: } Z \\
\text{Ref: } (X) \cdot R_1 & \rightarrow \text{Ref: } (X) \cdot R_2 \\
\text{BG: } B_1 & \rightarrow \text{BG: } B_2
\end{align*}

Again, these new recursive schemata allow a strict superset of the original derivations to be constructed. None of our earlier derivations are invalidated by these new schemata. We can always just ignore the new item on the reference list.

The significance of this rule is that it allows what Gawron and Peters (forthcoming) call role-linking anaphora. That is, any argument content will be available as a possible antecedent. Consider the derivable category

\begin{align*}
(216) \quad \text{Exp: } & \text{believe he won} \\
\text{Syn: } & s \backslash np \\
\text{Con: } & X.E.\text{believe}(E2.\text{win}(Y)(E2))(X)(E) \\
\text{Ref: } & \langle \text{AN}(Y, \text{male}(Y)) \rangle \\
\text{BG: } & \{\}
\end{align*}
in which we have ignored the contribution of the tense marker in won. This matches the input to our revised unary rules. That is, being able to reduce

(217) EXP: _believe he won_  
SYN: s  
CON: E.believe(E2.win(Y)(E2))(X)(E)  
REF: (X, AN(Y, male(Y)))  
BG: {}  

to

(218) EXP: _believe he won_  
SYN: s  
CON: E.believe(E2.win(X)(E2))(X)(E)  
REF: (X)  
BG: {}  

allows us to reduce the verb phrase category (216) to

(219) EXP: _believe he won_  
SYN: s \ np  
CON: X.E.believe(E2.win(X)(E2))(X)(E)  
REF: {}  
BG: {}  

In effect, the anaphor term has unified with the argument X in the original content.

To complete our analysis, we need a category for _only_. We will use

(220) EXP: _only_  
SYN: np(sing) / np(sing)  
CON: X.Y  
REF: (GQ(only,Y,X ≈ Y))  
BG: {}  

Recall that ≈ is our intensional equality relation. This category allows us to derive the subject noun phrase _only john_ as

(221) EXP: _only john_  
SYN: np  
CON: Y  
REF: (GQ(only,Y,X ≈ Y),X)  
BG: { name(john)(X) }
Attaching this noun phrase as the subject of our newly derived verb phrase gives us the category

(222) Exp: only john believe he won  
     Syn: s  
     Con: E.believe(E2.win(Y)(E2))(Y)(E)  
     Ref: (gq(only,Y,X \approx Y),X)  
     Bg: \{ \text{name(john)}(X) \}  

We can then pop the quantifier in the ordinary fashion to get the desired reading

(223) Exp: only john believe he won  
     Syn: s  
     Con: E3.only (Y.X \approx Y)  
     \quad (Y.E.believe(E2.win(Y)(E2))(Y)(E))(E3)  
     Ref: (X)  
     Bg: \{ \text{name(john)}(X) \}  

This is exactly the content we want in the sloppy anaphora case. The scope of the quantifier is the property of being a Y such that Y believes Y will win. The quantifier then claims that there is an individual X named John and he is the only one with this property. We will not present the derivation for the strict case, since it does not make use of the extended recursive rule.

5.2.5 Other Dependents

While we have only discussed pronominal anaphora so far, it is a simple matter to incorporate categories for other dependent elements into our existing grammar. We consider a number of these in turn.

Possessive Determiners

The possessive determiners his, hers and its introduce anaphoric terms in the same manner as their corresponding determiners. For his, we assume the category

(224) Exp: his  
     Syn: np / n  
     Con: (X,P).X  
     Ref: (\text{AN}(Y,\text{male}(Y)),\text{DT}(X,(P \cap \text{possess}(Y)(X))))  
     Bg: {}
Thus a possessive will introduce two items into the reference list. The first is an anaphor term for the possessor and it can be resolved just like any other anaphor term. The second is a descriptive term for the item possessed. We have built in the assumption that possessives work like indefinites, but it would have been just as easy to replace the indefinite descriptive term with a definite one.

Reciprocals

Dependent terms can show up in adjectives, as well as in determiners and noun phrases. Consider the case of adjectives such as other and similar. We can give other the category

(225) Exp: other
    Syn: n/n
    Con: (X.P).(X.(P ∩ ~ (X ≈ Y)))
    Ref: (AN(Y, true))
    BG: {}

Note that we have simply included a dummy restriction on the anaphor term where true is some proposition that is always true. This is because other places no restriction on the gender of its antecedents. For instance, the sentence

(226) Opus likes every other penguin.

will receive the category

(227) Exp: opus like every other penguin
    Syn: s
    Con: E.like(Y)(X)(E)
    Ref: (X, GQ(every, Y, (penguin(Y) ∩ not(Y ≈ Z))), AN(Z, true))
    BG: { name(opus)(X) }

We can use the referent for Opus as the antecedent for the anaphor term to produce

(228) Exp: opus like every other penguin
    Syn: s
    Con: E.like(Y)(X)(E)
    Ref: (X, GQ(every, Y, (penguin(Y) ∩ not(Y ≈ Z) ≈ Z))))
    BG: { name(opus)(X) }
The quantifier can then be popped from the reference list in the ordinary way to give us the appropriate content.
Appendix A

English Relational Categorial Grammar

A.1 Grammar Rules

A.1.1 Categorial Application Rules

Forward Application

(1) \( \text{Exp: } E_1 E_2 \rightarrow \text{Exp: } E_1 \quad \text{Exp: } E_2 \)

\begin{align*}
\text{Syn: } & C_1 & & C_1 / C_2 & & C_2 \\
\text{Con: } & F & & X . F & & X \\
\text{Ref: } & R_1 \cdot R_2 & & R_1 & & R_2 \\
\text{BG: } & B_1 \cup B_2 & & B_1 & & B_2
\end{align*}

Backward Application

(2) \( \text{Exp: } E_1 E_2 \rightarrow \text{Exp: } E_1 \quad \text{Exp: } E_2 \)

\begin{align*}
\text{Syn: } & C_1 & & C_2 & & C_1 \setminus C_2 \\
\text{Con: } & F & & X & & X . F \\
\text{Ref: } & R_1 \cdot R_2 & & R_1 & & R_2 \\
\text{BG: } & B_1 \cup B_2 & & B_1 & & B_2
\end{align*}
A.1.2 Unary Recursive Abstraction Rules

Unary Forward Abstraction

(3) \[ \begin{array}{ll}
\text{Exp: } D & \rightarrow \text{Exp: } D \\
\text{Syn: } C_1 & \text{Syn: } C_1 \\
\text{Con: } Y & \text{Con: } Z \\
\text{Ref: } R_1 & \text{Ref: } R_2 \\
\text{BG: } B_1 & \text{BG: } B_2 \\
\end{array} \]

\[ \begin{array}{ll}
\text{Exp: } D & \rightarrow \text{Exp: } D \\
\text{Syn: } C_1 / C_2 & \text{Syn: } C_1 / C_2 \\
\text{Con: } X.Y & \text{Con: } X.Z \\
\text{Ref: } R_1 & \text{Ref: } R_2 \\
\text{BG: } B_1 & \text{BG: } B_2 \\
\end{array} \]

Unary Forward Abstraction

(4) \[ \begin{array}{ll}
\text{Exp: } D & \rightarrow \text{Exp: } D \\
\text{Syn: } C_1 & \text{Syn: } C_1 \\
\text{Con: } Y & \text{Con: } Z \\
\text{Ref: } R_1 & \text{Ref: } R_2 \\
\text{BG: } B_1 & \text{BG: } B_2 \\
\end{array} \]

\[ \begin{array}{ll}
\text{Exp: } D & \rightarrow \text{Exp: } D \\
\text{Syn: } C_1 \setminus C_2 & \text{Syn: } C_1 \setminus C_2 \\
\text{Con: } X.Y & \text{Con: } X.Z \\
\text{Ref: } R_1 & \text{Ref: } R_2 \\
\text{BG: } B_1 & \text{BG: } B_2 \\
\end{array} \]

A.1.3 Quantifier Term Reduction Rules

Sentential Quantifier Reduction

(5) \[ \begin{array}{ll}
\text{Exp: } D & \rightarrow \text{Exp: } D \\
\text{Syn: } s & \text{Syn: } s \\
\text{Con: } E2.Q_e(X.R)(X.E.P)(E2) & \text{Con: } E.P \\
\text{Ref: } R_1 \cdot R_2 & \text{Ref: } R_1 \cdot GQ(Q, X, R) \cdot R_2 \\
\text{BG: } B & \text{BG: } B \\
\end{array} \]
Nominal Quantifier Reduction

(6) \[ \text{Exp: } D \quad \rightarrow \quad \text{Exp: } D \]
\[ \text{Syn: } n \quad \quad \text{Syn: } n \]
\[ \text{Con: } Y.Q,(X.R)(X.P) \quad \text{Con: } Y.P \]
\[ \text{Ref: } R_1 \cdot R_2 \quad \text{Ref: } R_1 \cdot \text{GQ}(Q,X,R) \cdot R_2 \]
\[ \text{BG: } B \quad \text{BG: } B \]

A.1.4 Descriptive Term Reduction Rules

Descriptive Term Backgrounding

(7) \[ \text{Exp: } E \quad \rightarrow \quad \text{Exp: } E \]
\[ \text{Syn: } S \quad \quad \text{Syn: } S \]
\[ \text{Con: } C \quad \quad \text{Con: } C \]
\[ \text{Ref: } R_1 \cdot R_2 \quad \text{Ref: } R_1 \cdot \text{DT}(Y,R) \cdot R_2 \]
\[ \text{BG: } \{R\} \cup B \quad \text{BG: } B \]

Descriptive Term Quantifying

(8) \[ \text{Exp: } E \quad \rightarrow \quad \text{Exp: } E \]
\[ \text{Syn: } S \quad \quad \text{Syn: } S \]
\[ \text{Con: } C \quad \quad \text{Con: } C \]
\[ \text{Ref: } R_1 \cdot \text{GQ}(\text{some},Y,R) \cdot R_2 \quad \text{Ref: } R_1 \cdot \text{DT}(Y,R) \cdot R_2 \]
\[ \text{BG: } B \quad \text{BG: } B \]

A.1.5 Anaphora Term Reduction Rules

Referent Anaphora

(9) \[ \text{Exp: } E \]
\[ \text{Syn: } S \]
\[ \text{Con: } C \]
\[ \text{Ref: } R_1 \cdot Y \cdot R_2 \cdot R_3 \]
\[ \text{BG: } B \]
\[ \rightarrow \quad \text{Exp: } E \]
\[ \text{Syn: } S \]
\[ \text{Con: } C \]
\[ \text{Ref: } R_1 \cdot Y \cdot R_2 \cdot \text{AN}(Y,Q) \cdot R_3 \]
\[ \text{BG: } B \]
Description Anaphora

(10) \( \text{Exp: } E \quad \text{Syn: } S \quad \text{Con: } C \quad \text{Ref: } R_1 \cdot \text{DT}(Y,P) \cdot R_2 \cdot R_3 \quad \text{BG: } B \)

\[ \rightarrow \quad \text{Exp: } E \quad \text{Syn: } S \quad \text{Con: } C \quad \text{Ref: } R_1 \cdot \text{DT}(Y,P) \cdot R_2 \cdot \text{AN}(Y,Q) \cdot R_3 \quad \text{BG: } B \]

Quantifier Anaphora

(11) \( \text{Exp: } E \quad \text{Syn: } S \quad \text{Con: } C \quad \text{Ref: } R_1 \cdot \text{GQ}(G,Y,P) \cdot R_2 \cdot R_3 \quad \text{BG: } B \)

\[ \rightarrow \quad \text{Exp: } E \quad \text{Syn: } S \quad \text{Con: } C \quad \text{Ref: } R_1 \cdot \text{GQ}(G,Y,P) \cdot R_2 \cdot \text{AN}(Y,Q) \cdot R_3 \quad \text{BG: } B \]

Anaphor Anaphora

(12) \( \text{Exp: } E \quad \text{Syn: } S \quad \text{Con: } C \quad \text{Ref: } R_1 \cdot \text{AN}(Y,P) \cdot R_2 \cdot R_3 \quad \text{BG: } B \)

\[ \rightarrow \quad \text{Exp: } E \quad \text{Syn: } S \quad \text{Con: } C \quad \text{Ref: } R_1 \cdot \text{AN}(Y,P) \cdot R_2 \cdot \text{AN}(Y,Q) \cdot R_3 \quad \text{BG: } B \]
A.2 Lexical Entries

A.2.1 Simple Noun Phrases

Indexicals

(13) \textbf{Exp}: \textit{i}  \\
\textbf{Syn}: \textit{np}  \\
\textbf{Con}: \textit{S}_i  \\
\textbf{Ref}: \langle \textit{S}_i \rangle  \\
\textbf{BG}: \{\}

Names

(14) \textbf{Exp}: \textit{opus}  \\
\textbf{Syn}: \textit{np}  \\
\textbf{Con}: \textit{X}  \\
\textbf{Ref}: \langle \textit{X} \rangle  \\
\textbf{BG}: \{ \text{name}(\textit{opus})(\textit{X}) \}

Demonstratives

(15) \textbf{Exp}: \textit{this}  \\
\textbf{Syn}: \textit{np}  \\
\textbf{Con}: \textit{X}  \\
\textbf{Ref}: \langle \textit{X} \rangle  \\
\textbf{BG}: \{ \text{demonst}(\textit{E}_{this})(\textit{X}) \}

Pronouns

(16) \textbf{Exp}: \textit{he}  \\
\textbf{Syn}: \textit{np(subj)}  \\
\textbf{Con}: \textit{X}  \\
\textbf{Ref}: \langle \text{AN}(\textit{X}, \text{male}(\textit{X})) \rangle  \\
\textbf{BG}: \{\}

(17) \textbf{Exp}: \textit{him(self)}  \\
\textbf{Syn}: \textit{np(obj)}  \\
\textbf{Con}: \textit{X}  \\
\textbf{Ref}: \langle \text{AN}(\textit{X}, \text{male}(\textit{X})) \rangle  \\
\textbf{BG}: \{\}
Expletive Nouns

(18) \( \text{EXP: } \textit{it} \)  
SYN: np(it)  
CON: it  
REF: ()  
BG: {}

(19) \( \text{EXP: } \textit{there} \)  
SYN: np(there)  
CON: there  
REF: ()  
BG: {}

A.2.2 Common Nouns

(20) \( \text{EXP: } \textit{dog} \)  
SYN: n  
CON: X.dog(X)  
REF: ()  
BG: {}

A.2.3 Determiners

Referential Determiners

(21) \( \text{EXP: } \textit{a} \)  
SYN: np/n  
CON: (X.P).X  
REF: (dt(X,P))  
BG: {}

(22) \( \text{EXP: } \textit{the} \)  
SYN: np/n  
CON: (X.P).Y  
REF: (dt(Y, the(X.P)(Y)))  
BG: {}
Possessive Determiners

(23) **EXP:** his  
**SYN:** np/n  
**CON:** (X.P).X  
**REF:** (AN(Y,male(Y)), DT(X, (P \ni possess(Y)(X))))  
**BG:** {}  

Generalized Quantifiers

(24) **EXP:** every  
**SYN:** np/n  
**CON:** P.X  
**REF:** (GQ(every, X, P))  
**BG:** {}  

A.2.4 Nominal Modifiers

Adjectives

(25) **EXP:** red  
**SYN:** n/n  
**CON:** (X.P).X.((P \ni red(X))  
**REF:** ()  
**BG:** {}  

(26) **EXP:** fake  
**SYN:** n/n  
**CON:** (X.P).Y.fake(Y, X.P)  
**REF:** ()  
**BG:** {}  

Reciprocals

(27) **EXP:** other  
**SYN:** n/n  
**CON:** (X.P).((X.P \ni \sim (X \approx Y))  
**REF:** (AN(Y, P))  
**BG:** {}
Intensifiers

(28) \text{EXP: very}  
\text{SYN: } n/n/(n/n)  
\text{CON: } (X.P).(Y.Q).((Z.R).W.very((X.P).(Y.Q)))(Z.R)(W)  
\text{REF: } \emptyset  
\text{BG: } \emptyset

A.2.5 Simple Verbs

Intransitive Verbs

(29) \text{EXP: sing}  
\text{SYN: } s(bse) \setminus np  
\text{CON: } X.E.sing(X)(E)  
\text{REF: } \emptyset  
\text{BG: } \emptyset

(30) \text{EXP: sang}  
\text{SYN: } s(fin) \setminus np  
\text{CON: } X.E.(\text{sing}(X)(E) \cap \text{at}(E)(T))  
\text{REF: } (T)  
\text{BG: } \{ \text{before}(T)(T_{sang}) \}

(31) \text{EXP: sings}  
\text{SYN: } s(fin) \setminus np  
\text{CON: } X.E.(\text{sing}(X)(E) \cap \text{at}(E)(T))  
\text{REF: } (T)  
\text{BG: } \{ \text{during}(T)(T_{sings}) \}

(32) \text{EXP: sung}  
\text{SYN: } s(perf) \setminus np  
\text{CON: } E2.X.\text{perf}(E.\text{sing}(X)(E))(E2)  
\text{REF: } \emptyset  
\text{BG: } \emptyset

(33) \text{EXP: singing}  
\text{SYN: } s(pred) \setminus np  
\text{CON: } E2.X.\text{prog}(E.\text{sing}(X)(E))(E2)  
\text{REF: } \emptyset  
\text{BG: } \emptyset

305
Transitive Verbs

(34) EXP: love
SYN: s(bse) \ np / np
CON: Y.X.E.love(Y)(X)(E)
REF: {}
BG: {}

Ditransitive Verbs

(35) EXP: give
SYN: s(bse) \ np / np / np
CON: Z.Y.X.E.give(Z)(Y)(X)(E)
REF: {}
BG: {}

A.2.6 Auxiliary Verbs

Copula

(36) EXP: be
SYN: s(bse) \ np / (s(pred) \ np)
CON: (X.E.P).X.E.P
REF: {}
BG: {}

Base Auxiliary

(37) EXP: does
SYN: s(fin) \ np / (s(bse) \ np)
CON: (X.E.P).X.E.(P \ at(E)(T))
REF: {T}
BG: { during(T)(T_{does}) }

Perfect Auxiliary

(38) EXP: have
SYN: s(bse) \ np / (s(perf) \ np)
CON: (X.E.P).X.E.P
REF: {}
BG: { }
Modal Auxiliaries

(39) EXP: will
SYN: s(fin) \ np / (s(bse) \ np)
CON: (X.E.P).X.E.(P \at(E)(T))
REF: (T)
BG: \{ after(T)(T_{will}) \}

(40) EXP: should
SYN: s(fin) \ np / (s(bse) \ np)
CON: (X.E.P).X.E2.(should(X)(E.P)(E2) \at(E2)(T))
REF: (T)
BG: \{ during(T)(T_{should}) \}

Infinitive Auxiliary

(41) EXP: to
SYN: s(inf) \ np / (s(bse) \ np)
CON: (X.E.P).X.E.P
REF: ()
BG: {} 

Negation

(42) EXP: not
SYN: s(v) \ np / (s(v) \ np) for v \in \{bse, perf, pred, inf\}
CON: (X.E.P).X.E2.not(E.P)(E2)
REF: ()
BG: {} 

A.2.7 Attitude Verbs

(43) EXP: think
SYN: s(bse) \ np / s(fin)
CON: (E.P).X.E2.think(X)(E.P)(E2)
REF: ()
BG: {}
A.2.8 Perception Verbs

(44) EXP: see
    SYN: s(bse)\np / (s(bse)\np) / np
    REF: {E}
    BG: \{P\}

A.2.9 Control Verbs

Subject Raising Verbs

(45) EXP: seem
    SYN: s(bse)\np / (s(inf)\np)
    CON: (X.E.P).X.E2.seem(E.P)(E2)
    REF: \{}
    BG: \{}

Object Raising Verbs

(46) EXP: know
    SYN: s(bse)\np / (s(inf)\np) / np
    REF: \{}
    BG: \{}

Subject Equi Verbs

(47) EXP: promise
    SYN: s(bse)\np / (s(inf)\np) / np
    REF: \{}
    BG: \{}

Object Equi Verbs

(48) EXP: persuade
    SYN: s(bse)\np / (s(inf)\np) / np
    REF: \{}
    BG: \{\}
A.2.10 Expletive Verbs

(49) EXP: rain
SYN: s(bse) \ np(it)
CON: X.E.rain(E)
REF: {}
BG: {}

(50) EXP: be
SYN: s(bse) \ np(there) / np
CON: Y.X.E.exists(Y)(E)
REF: {}
BG: {}

A.2.11 Verbal Modifiers

Simple Adverbs

(51) EXP: probably
SYN: s(V) \ np / (s(V) \ np)
CON: (X.E.P).X.E2.probably(E.P)(E2)
REF: {}
BG: {}

(52) EXP: slowly
SYN: s(V) \ np / (s(V) \ np)
CON: (X.E.P).X.E2.slowly(E.P)(E2)
REF: {}
BG: {}

Complemented Adverbs

(53) EXP: after
SYN: s(V) \ np \ (s(V) \ np) / s(fin)
CON: (E.P).(X.E2.P2).X.E2.(P2 \ \ while(E)(E2))
REF: {E}
BG: { P }

(54) EXP: while
SYN: s(V) \ np \ (s(V) \ np) / (s(pred) \ np)
CON: (X.E.P).(X.E2.P2).X.E2.(P2 \ \ during(E)(E2))
REF: {E}
BG: { P }

309
(55) **EXP:** with  
**SYN:** \(s(V) \setminus np \setminus (s(V) \setminus np) / (s(pred) \setminus np) / np\)  
**CON:** \(Y.(Y.E.P).(X.E2.P2).X.E2.(P2 \cap with(E)(E2))\)  
**REF:** \(\langle E \rangle\)  
**BG:** \{ P \}

(56) **EXP:** to  
**SYN:** \(s(V) \setminus np \setminus (s(V) \setminus np) / (s(bse) \setminus np)\)  
**CON:** \((X.E.P).(X.E2.P2).X.E2.(P2 \cap to(E.P)(X)(E2))\)  
**REF:** \(\{\}\)  
**BG:** \{\}

### A.2.12 Prepositions

#### Nominal Prepositions

(57) **EXP:** in  
**SYN:** \(n \setminus n / np\)  
**CON:** \(X.(Y.P).Y.(P \cap in(X)(Y))\)  
**REF:** \(\{\}\)  
**BG:** \{\}

#### Verbal Prepositions

(58) **EXP:** in  
**SYN:** \(s(V) \setminus np \setminus (s(V) \setminus np) / np\)  
**CON:** \(X.(Y.E.P).YE.(P \cap in(X)(E))\)  
**REF:** \(\{\}\)  
**BG:** \{\}
Bibliography


314


Kaplan, D., 1977. Demonstratives: an essay on the semantics, logic, metaphysics and epistemology of demonstratives and other indexicals. manuscript, University of California, Los Angeles.


Ladusaw, W. and D. R. Dowty, 1986. Toward a non-grammatical account of thematic roles. manuscript, University of California, Santa Cruz and Ohio State University.


Landman, F., 1987. Groups. manuscript, University of Massachusetts, Amherst.


Levinson, S., 1988. Pragmatics and situation semantics. manuscript, Center for the Study of Language and Information, Stanford University.


325


329
In Hanson, P., editor, Information, Language and Cognition, University of British Columbia Press, Vancouver.


Rounds, W., 1988. Set values for unification-based grammar formalisms and logic programming. manuscript, Xerox PARC.


