A Corpus-Trained Parser for Systemic-Functional Syntax

David Clive Souter

Submitted in accordance with the requirements for the degree of Doctor of Philosophy.

The University of Leeds
School of Computer Studies

August 1996

The candidate confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to the work of others.

Abstract

This thesis presents a language engineering approach to the development of a tool for the parsing of relatively unrestricted English text, as found in spoken natural language corpora.

Parsing unrestricted English requires large-scale lexical and grammatical resources, and an algorithm for combining the two to assign syntactic structures to utterances of the language. The grammatical theory adopted for this purpose is systemic functional grammar (SFG), despite the fact that it is traditionally used for natural language generation. The parser will use a probabilistic systemic functional syntax (Fawcett 1981, Souter 1990), which was originally employed to hand-parse the Polytechnic of Wales corpus (Fawcett and Perkins 1980, Souter 1989), a 65,000 word transcribed corpus of children's spoken English. Although SFG contains mechanisms for representing semantic as well as syntactic choice in NL generation, the work presented here focuses on the parallel task of obtaining syntactic structures for sentences, and not on retrieving a full semantic interpretation.

The syntactic language model can be extracted automatically from the Polytechnic of Wales corpus in a number of formalisms, including 2,800 simple context-free rules (Souter and Atwell 1992). This constitutes a very large formal syntax language, but still contains gaps in its coverage. Some of these are accounted for by a mechanism for expanding the potential for co-ordination and subordination beyond that observed in the corpus. However, at the same time the set of syntax rules can be reduced in size by allowing optionality in the rules. Alongside the context-free rules (which capture the largely horizontal relationships between the mother and daughter constituents in a tree), a vertical trigram model is extracted from the corpus, controlling the vertical relationships between possible grandmothers, mothers and daughters in the parse tree, which represent the alternating layers of elements of structure and syntactic units in SFG. Together, these two models constitute a quasi-context-sensitive syntax.

A probabilistic lexicon also extracted from the POW corpus proved inadequate for unrestricted English, so two alternative part-of-speech tagging approaches were investigated. Firstly, the CELEX lexical database was used to provide a large-scale word tagging facility. To make the lexical database compatible with the corpus-based grammar, a hand-crafted mapping was applied to the lexicon's theory neutral grammatical description. This transformed the lexical tags into systemic functional grammar labels, providing a harmonised probabilistic lexicon and grammar. Using the CELEX lexicon, the parser has to do the work of lexical disambiguation. This overhead can be removed with the second approach: The Brill tagger trained on the POW corpus can be used to assign unambiguous labels (with over 92% success rate) to the words to be parsed. While tagging errors do compromise the success rate of the parser, these are outweighed by the search time saved by introducing only one tag per word.

A probabilistic chart parsing program which integrated the reduced context-free syntax, the vertical trigram model, with either the SFG lexicon or the POW trained Brill tagger was implemented and tested on a sample of the corpus. Without the vertical trigram model and using CELEX lexical look-up, results were extremely poor, with combinatorial explosion in the syntax preventing any analyses being found for sentences longer than five words within a practical time span. The seemingly unlimited potential for vertical recursion in a context-free rule model of systemic functional syntax is a severe problem for a standard chart parser. However, with addition of the Brill tagger and vertical trigram model, the performance is markedly improved. The parser achieves a reasonably creditable success rate of 76%, if the criteria for success are liberally set at at least one legitimate SF syntax tree in the first six produced for the given test data. While the resulting parser is not suitable for real-time applications, it demonstrates the potential for the use of corpus-derived probabilistic syntactic data in parsing relatively unrestricted natural language, including utterances with ellipted elements, unfinished constituents, and constituents without a syntactic head. With very large syntax models of this kind, the problem of multiple solutions is common, and the modified chart parser presented here is able to produce correct or nearly correct parses in the first few it finds.

Apart from the implementation of a parser for systemic functional syntax, the re-usable method by which the lexical look-up, syntactic and parsing resources were obtained is a significant contribution to the field of computational linguistics.

Contents

Abstract	ii
Contents	iii
Tables and Illustrations	vi
Acknowledgments	vii
Preamble	viii
Chapter 1. Introduction	**1**
1.1 Aims.	1
1.2 Parsing Natural Language: An Example.	1
1.3 Setting the Parsing Scene.	3
1.4 Definitions.	7
1.4.1 Natural Language.	7
1.4.2 The Lexicon.	7
1.4.3 Competence and Performance Grammar.	8
1.4.4 The Grammatical Description.	8
1.4.5 The Grammatical Formalism.	9
1.4.6 Systemic Functional Grammar.	10
1.4.7 Language Corpora.	11
1.4.8 Parsing.	11
1.5 Application and Domain.	13
1.6 Scope.	14
1.6.1 Input Format.	15
1.6.2 Output Format.	15
1.7 Original Contributions to the Field.	16
1.8 The Structure of the Thesis.	18
Chapter 2. Background to the Thesis.	**20**
2.1 Corpus-Based Computational Linguistics.	20
2.1.1 The Intuition-Based Approach to Developing a Grammar.	21
2.1.2 The Corpus-Based Approach to Developing a Grammar.	22
2.1.3 Selecting a Parsed Corpus.	24
2.1.4 The Polytechnic of Wales Corpus.	25
2.1.4.1 Transcription.	26
2.1.4.2 Syntactic Analysis.	26
2.1.4.3 Corpus Format.	26
2.1.5 The Edited Polytechnic of Wales Corpus.	28
2.2 Systemic Functional Grammatical Description and Formalism.	29
2.2.1 Systemic Grammar in NL Generation.	32
2.2.2 Systemic Grammar in NL Parsing.	34
2.2.3 Problems for a SFG Parser.	37
2.2.4 Systemic-Functional Grammar in the POW Corpus.	38
2.2.5 Choosing a Grammar Formalism and its Effect on Parsing.	44
2.3 Lexical Resources for Corpus-Based Parsing.	47
2.3.1 Corpus-Based Tag Assignment.	48
2.3.2 Dictionaries and Morphological Analysers.	51
2.3.3 Lexical Databases.	52
2.3.4 Choice of Lexical Resource.	54
2.4 Parsing Techniques.	55
2.4.1 Rule-Based Parsing.	57
2.4.1.1 Shift-Reduce Parsing.	58

2.4.1.2 Chart Parsing. 58
2.4.1.3 Probabilistic Chart Parsing. 62
2.4.2 Probabilistic Parsing. 65
2.4.2.1 Probabilistic Language Models. 65
2.4.2.2 Search Techniques. 67
2.4.3 Selection of a Parsing Technique. 69

Chapter 3. Developing the Resources for Parsing. 71
3.1 Developing a Probabilistic Systemic Functional Syntax. 71
3.1.1 Inconsistencies in the Corpus. 73
3.1.2 Distribution and Frequency of the Rules. 73
3.1.3 Coverage of the Rules. 76
3.1.4 Editing the Corpus. 76
3.1.5 Collapsing a Syntax Model using Optionality. 77
3.1.6 Expanding a Syntax Model using Co-ordination. 81
3.2 Developing a Probabilistic Lexicon for Systemic Functional Grammar. 85
3.2.1 Inadequacies of the POW Corpus Word List. 86
3.2.2 Lexical Aims and Policy Decisions. 87
3.2.3 Selection of Lexical Resources. 88
3.2.4 The CELEX English Database. 88
3.2.5 Converting the CELEX Lexicon. 90
3.2.6 Testing the Lexicon. 93
3.2.7 Rapid Lexical Lookup. 95
3.2.8 Lexical Probabilities 96
3.3 Training the Brill Tagger 96
3.4 Conclusions. 97

Chapter 4. A Probabilistic Chart Parser for Systemic Functional Syntax. 99
4.1 Preliminary Test using Parser Version 5. 101
4.2 An Improved Parsing Algorithm: Version 6. 105
4.2.1 Incorporating Rules with Optional Daughters. 105
4.2.2 Incorporating Rules with Co-ordinating Daughters. 107
4.3 Version 7: Combining Context-Free and Vertical-Trigram Rules. 110
4.3.1 The Vertical Trigram Model. 111
4.3.2 Limiting Tree Depth. 112
4.3.3 Stopping the Parser. 112

Chapter 5. Parser Testing and Evaluation. 114
5.1. The Test Data. 114
5.2 Test 1: Prototype context-free parser with CELEX look-up. 116
5.2.1 Lexical Look-Up. 118
5.2.2 Limitations of the Syntax Formalism. 120
5.2.3 Near Misses. 122
5.3 Test 2: Brill tagging and a context-sensitive chart parser. 125
5.3.1 Test 2: Parser Efficiency. 128
5.3 2 Test 2: Lexical Tagging Results. 129
5.4 Formal Evaluation. 131
5.5 Conclusions. 135

Chapter 6. Improvements to the Parser. 137
6.1 Improving the Lexicon. 137
6.1.1 Disambiguated Tag Probabilities. 137
6.1.2 Refining the CELEX to POW Syntax Mapping. 138

6.1.3 Handling Recurrent Word Combinations. 138
6.1.4 Improving the Brill tagger. 139
6.2 Improving the Grammatical Formalism. 140
6.2.1 A Probability Matrix for Optional Rules. 140
6.2.2 Grammatical Coverage. 140
6.3 Improving the Probabilistic Chart Parser. 141
6.3.1 Efficiency Checks. 141
6.3.2 Applying Multi-Word Edges in a Chart. 142
6.3.3 Restricting Unnecessary Rule Application. 143
6.3.4 Controlling the Agenda. 143
6.3.5 Feature-Based Parsing. 145
6.3.6 Semantic Solution Pruning. 146
6.4 Conclusion. 147

Chapter 7. Conclusions. 148
7.1 Lexical Resources. 148
7.2 Grammatical Resources. 149
7.3 Parser Implementation and Testing. 150
7.4 The Research Method. 151
7.5 The Last Word. 153

References . 155

Appendices . 165
Appendix 1. Sample Fragments of Parsed Corpora. 166
Appendix 1.1 Lancaster/Leeds Treebank. 166
Appendix 1.2 Nijmegen Corpus (CCPP). 167
Appendix 1.3 Polytechnic of Wales Corpus. 168
Appendix 1.4 Susanne Corpus. 169
Appendix 1.5 IBM/Lancaster Spoken English Corpus. 170
Appendix 2. A Brief Description of the POW Corpus. 171
Appendix 3. Systemic-Functional Syntax Categories in the POW Corpus. 172
Appendix 4. A Mapping from LDOCE to POW SF Syntax Tags. 175
Appendix 5. A Fragment of a Context-Free SF Syntax Maintaining the Distinction between Filling and Componence. 179
Appendix 6. A Fragment of a Context-Free SF Syntax Ignoring the Distinction between Filling and Componence. 180
Appendix 7. A Fragment of a Vertical Trigram Model from the POW Corpus. 181
Appendix 8. Rule and Word-Wordtag Frequency Distribution in the POW Corpus. 182
Appendix 9. A Prototype Competence Systemic Functional Syntax. 183
Appendix 10. Brill Tagger Context Rules Learned from POW 186
Appendix 11. General lexical tagging rules used by the Brill tagger for untrained words. 187
Appendix 12. The Reduced EPOW Filling Grammar. 188
Appendix 13. The 100 Most Frequent Word-Wordtag Pairs in the EPOW Lexicon. 189
Appendix 14. Pocock and Atwell's Weight-Driven Chart Parser. 190
Appendix 15. Parser Version 7: Test Results. 191

Tables and Illustrations

Figure 1. Sample of transcribed speech from the Polytechnic of Wales Corpus. 1
Figure 2. Parse Trees for the Utterances in Figure 1. 3
Figure 3. A Graphical Representation of a Parse Tree. 12
Figure 4. A Sample Section of a POW Corpus File. 27
Figure 5: A fragment of a system network for MOOD in English. 30
Figure 6. A Sample Section of GENESYS Output. 33
Figure 7. Section of Lispified LDOCE. 51
Figure 8. Building a Chart. 60
Figure 9. The 20 Most Frequent Word-Wordtag Pairs in the POW Corpus. 63
Figure 10. The 20 Most Frequent Context-Free Rules in the POW Corpus. 63
Figure 11. A Fragment of Probabilistic RTN from EPOW. 66
Figure 12. Ambiguous Analyses of the Sentence *Fight on to save art*. 68
Figure 13. Rules which occur only once (6047 out of 8522). 74
Figure 14. Syntax Rule Reduction Using Optionality. 77
Figure 15. A Probabilistic QQGP Syntax. 80
Figure 16. A Reduced Probabilistic QQGP Syntax. 80
Figure 17. Co-ordination Likelihood in the EPOW Corpus. 82
Figure 18. Variants of the Word-Stem *brick* in the POW Lexicon. 85
Figure 19. Singleton Word/Word-tag Co-occurrences in the POW Corpus Lexicon. 86
Figure 20. A Fragment of our CELEX English Lexicon. 89
Figure 21. Structure of our CELEX Lexicon according to Grammatical Category. 90
Figure 22. A Fragment of the Reformatted CELEX SFG Lexicon. 93
Figure 23. Testing the CELEX SFG Lexicon with the EPOW Word List. 93
Figure 24. A Trivial Test of Parser Version 5. 103
Figure 25. Trivial Output from a Weight-Driven Chart Parser (version 6). 108
Figure 26. Tree Depth in the POW Corpus. 113
Figure 27. The Test Samples. 116
Figure 28. Test 1: Results. 118
Figure 29. Data from a Suspended Six-Word Parse. 121
Figure 30. Near Misses. 122
Figure 31. Test 2: Results. 127
Figure 32: Evaluation of test results on 19 ‘seen’ and 6 unseen test utterances. 133

Acknowledgments

I would like to thank my supervisor, Eric Atwell for his generous support, insight and direction throughout my period of part-time study. Without him I would never have become a corpus linguist, and still been a John-likes-Marian.

I am also indebted to my wife Clair, who has only ever known me as a Ph.D. student, and my mother, for their continuing encouragement.

I would also like to thank my erstwhile colleagues on the COMMUNAL project at Cardiff, and Robin and Mick for collecting the POW corpus in the first place.

I am most grateful to the School of Computer Studies and its staff, who persuaded me to embark on a part-time Ph.D. in the first place, and have never let me forget it. Special thanks go to my CCALAS colleagues over the years, particularly Eric Atwell, Tim O'Donoghue, Rob Pocock, John Hughes, George Demetriou and Uwe Jost.

Finally, I would like to thank my external examiners, Robin Fawcett and Willem Meijs for their useful comments and criticisms, which resulted in this revised thesis.

Preamble

This project falls within the domain of language engineering, a relatively young discipline in which (usually large-scale) linguistic resources are harnessed using artificial intelligence and computer science techniques in order to provide intelligent tools allowing humans to interact with and exploit computers by natural language. As such, it will not consist of a great new insight into linguistic theory, nor will it necessarily advance the theoretical bounds of computer science. In contrast, its originality will lie in a combination of elements; the selection and development of linguistic resources (a natural language corpus and a lexical database of English), the extraction of lexical and syntactic knowledge from these resources into machine-tractable formalisms, their harmonisation under one grammatical description (systemic functional grammar), and finally the integration of the lexical and syntactic mode with a parsing algorithm. Of particular importance is the unrestricted nature of the spoken language in the corpus, which makes the language model very large, including the ability to handle the common features of speech, such as ellipted items, false starts, replacement items and unfinished sentences. This sets the present enterprise apart from parsing work based on competence grammars of the language, which are usually manually developed using expert linguistic intuition, and often rely on the notion of a syntactic constituent containing a linguistic head. It is only when computational linguists try to accommodate the full range of relatively unrestricted natural language as it is performed, that the potential of NL systems will finally be commercially realised.

The chosen grammatical description is systemic functional grammar (SFG), and specifically the version of systemic functional syntax used in the annotation of the Polytechnic of Wales corpus. SFG has been used quite widely as a model of natural language generation within the broader context of social interaction. Some researchers have worked explicitly on the task of systemic-functional syntax parsing within the confines of the syntactic coverage of a related NL generation system, such as that of the Penman project (Kasper 1988) and in the COMMUNAL project, O'Donoghue's Vertical Strip Parser (1993) and Weerasinghe's Probabilistic On-line Parser (1994) [1]. The syntactic coverage of such parsers has been deliberately constrained in these cases by learning the syntax model from what the generator could generate. The parser's output was intended to be the input of a semantic interpreter,

[1] Although Weerasinghe does draw on the POW corpus for his probabilisitic model, the syntax itself is derived from the NL generator. O'Donoghue produced an earlier Simulated Annealing Parser based on the POW corpus (see Atwell et all 1988, Souter and O'Donoghue 1991), but his thesis work focuses on compatibility with the COMMUNAL generator GENESYS.

which once implemented, would produce semantic representations compatible with those in the generator, opening the way to a complete SFG-based NL interface. With such an application in mind, there is little point expanding the coverage of a syntactic parser beyond the potential of the generator (unless the interpreter was able to collapse semantically related syntactic analyses onto just the semantic representations that the generator could produce). The present project, however, does not so constrain itself, and can therefore analyse a wider range of structures. The corpus-trained syntax models which are presented here should provide useful source material to the developers of systemic-functional (and other) NL generation systems. One disadvantage of using the corpus-based syntax though, is that it is not directly compatible with that being used in the COMMUNAL NL generator, (even though they were both authored by Robin Fawcett); the syntactic node labels are slightly different, and the corpus version does not include participant roles. In this project, therefore, I am not attempting to accommodate the syntax of the related NL generator. I am working with a wider, arguably still richer model.

It should also have become clear that I consider the process of semantic interpretation to be beyond the scope of my project. When I refer to parsing, I will mean purely syntactic parsing with respect to the corpus-trained model, and not include the process of semantic interpretation (as some systemicists do). SFG provides a description and formalism at both the syntactic and semantic level. Therefore the use of the term *SFG* can be seen to encompass more than just a syntactic grammar, putting it at odds with linguists who would use the word *grammar* more exclusively for a purely syntactic model, such as a set of context-free rules. I will therefore (try to remember to) refer to my corpus-trained 'grammar' as a SF syntax, and reserve the term SFG for when I especially wish to refer to the full syntactic and semantic description.

The ethos of the work presented here is to allow the method to be re-used for other corpora and syntactic descriptions, rather than be tied exclusively to the one description. There are some areas where specific POW-corpus SF syntax modules have inevitably been included, but these have deliberately been kept to a minimum. There are many competing grammatical theories and descriptions, both in the field of corpus linguistics, and more general theoretical linguistics, none of which can be said to have universal support as yet. I will present a generally re-usable method, but implement it for one particularly rich (and therefore quite tricky) description.

Chapter 1. Introduction.

1.1 Aims.

It is the aim of this thesis to produce a reliable method for assigning syntactic structure to sentences of relatively unrestricted English, as found in spoken corpora of the language. This process, which is called *parsing*, is one of the key components in many computer applications which require natural language processing (NLP) of some kind. Apart from building a particular implementation of a parser for the syntax of systemic functional grammar (SFG: Fawcett 1981) as found in the Polytechnic of Wales corpus (Fawcett and Perkins 1980, Souter 1989b), I will argue that, given adequate lexical and corpus resources, the same method can be adopted to develop parsers for other grammatical descriptions.

1.2 Parsing Natural Language: An Example.

By way of introduction to the process of parsing, Figures 1 and 2 show the sort of input and output a parser for relatively unrestricted English might be expected to handle. The input would be transcribed spoken text, which ideally (but not invariably) would consist of separate sentences. The sample in Figure 1 is part of a spoken text produced by a twelve year old boy (PG) in conversation with two others while building some LEGO, and was chosen at random from a corpus of 65,000 words of spoken English called the Polytechnic of Wales (POW) Corpus.[1]

Figure 1. Sample of transcribed speech from the Polytechnic of Wales Corpus.

(1) PG: WHY
(2) PG: WHAT 'S THE POINT
(3) PG: YOU PUT THESE ON FOR WINDOWS
(4) AW: you don't have to
(5) SM: won't be long
(6) PG: IT 'S EASIEST MIND

[1] The samples are taken from the corpus file 12abpspg.

(7) AW: I know something easy. Build a garage.

(8) PG: FANTASTIC

(9) SM: or something like a skyscraper

(10) PG: THIS WORKED OUT IT WON'T FIT

(11) SM: Go on. We can always move it along can't we.

(12) PG: WILL THAT ONE FIT IN BY-THERE

(13) PG: COME ON LETS GET GOING

(14) PG: I CAN'T EVEN

The text in Figure 1 has been orthographically transcribed from recordings of the spoken data. While this fragment of speech hardly represents a typical interaction one might imagine between a human and a computer, it does contain a range of utterance types we would want a computer to be able to deal with; queries (1,2,11b,13), statements (3-7a,10), exclamations (8), and commands (7b,11a,13). It also includes examples of syntactic phenomena we would want our parser to be able to handle; juxtaposed sentences[2] (10,13), ellipted (missing) words (5,7,9) and unfinished sentences (4,14), which are a particular problem for many parsing programs. I have selected this spoken text as an example because it illustrates such difficulties. By doing so I do not mean to preclude the parser from working on written corpora or adult English, which contain their own different problems of more complex grammatical structure and longer utterances, but these language varieties will not be the primary material the parser should handle.

Each sentence in the POW Corpus has been syntactically analysed manually, so that each word is labelled with a syntactic category (appearing immediately to the word's left), and the words grouped into phrases and clauses, forming a tree structure of nested labelled brackets (see Figure 2).

---

[2] Juxtaposed sentences are those with no explicit separator between them. It is a matter of linguistic debate whether (10) is a single sentence consisting of two clauses, or two separate sentences. In a spoken corpus, such separation is hopefully marked by prosodic features such as pauses and intonation contours. Briscoe (1994; 97-8) refers to this as the *chunking* problem.

Figure 2. Parse Trees for the Utterances in Figure 1.

(1) [Z [CL [AWH [QQGP [AXWH WHY]]]]]

(2) [Z [CL [CWH [NGP [HWH WHAT]]] [OM 'S] [S [NGP [DD THE] [H POINT]]]]]

(3) [Z [CL [S [NGP [HP YOU]]] [M PUT] [C [NGP [DD THESE]]] [CM [QQGP [AX ON]]] [A [PGP [P FOR] [CV [NGP [H WINDOWS]]]]]]]

(6) [Z [CL [S [NGP [HP IT]]] [OM 'S] [C [QQGP [AX EASIEST]]] [AF MIND]]]

(8) [Z [CL [EX FANTASTIC]]]

(10) [Z [CL [S [NGP [DD THIS]]] [M WORKED] [CM [QQGP [AX OUT]]]] [CL [S [NGP [HP IT]]] [OMN WON'T] [M FIT]]]

(12) [Z [CL [OM WILL] [S [NGP [DD THAT] [HP ONE]]] [M FIT] [CM [QQGP [AX IN]]] [C [QQGP [AX BY-THERE]]]]]

(13) [Z [CL [M COME] [CM [QQGP [AX ON]]]] [CL [O LETS] [M GET] [C [CL [M GOING]]]]]

(14) [Z [CLUN [S [NGP [HP I]]] [OMN CAN'T] [AI EVEN]]]

These manually parsed trees might be treated as the desired solutions a parser should find and produce as output (provided the experts who produced the trees agree that they are correct). However, in arriving at these analyses, the manual annotators may have had access to the original recordings (or information derived from them), which would have provided intonation and other contextual cues leading the annotators to select one analysis from perhaps many other readings. It is typical in natural language for a sentence to be syntactically ambiguous, but the hearer will (usually unconsciously) use prosodic, semantic and pragmatic information, as well as knowledge of the world, to select the most likely interpretation. The parser being developed here will not have access to such information. It will therefore be the parser's job to provide the permissible syntactic structures (with respect to a particular syntactic model) and no more, for subsequent semantic and pragmatic pruning by further modules of a NLP system, or by a human post-editor. Consequently, each parse tree in Figure 2 should be viewed as one of perhaps many legitimate analyses which a syntactic parser should produce.

1.3 Setting the Parsing Scene.

Why is it so important to be able to accurately assign syntactic structure to a sentence? A reasonable analogy within the computing world would be to ask why is it important to be able to compile a program? The compiler parses the program from some programmer's file, interpreting

it as a set of instructions to perform on some given data. It is only by parsing the program that it can then successfully perform the instructions.

In the context of natural language input to a computer, the language utterance is analysed syntactically either because that is the goal of the computer system, or more commonly because the utterance is to be interpreted as a command, a query or a statement, and the meaning of the utterance is to be determined. In the latter case, the form of the utterance is found by the parser, and the form will dictate what the computer then does with its semantic content: add it to a database, perform some query on a database, or perform some other action such as opening a file. This kind of interaction presumes that the human is using natural language to access a data or knowledge base stored on computer. The advantages to be gained from using language in this way are that the user does not need to learn a specific database query language or operating system to make the computer act. Furthermore, if the language input mode is spoken, rather than written, it saves the user from needing to type, or click on a mouse, and hence will generally be quicker, and leave the user's hands free for other tasks.

There are, however, many other applications in which natural language is parsed, which promise to revolutionise the way we interact with computers and each other. Speech recognition devices, in which the user employs ordinary spoken language which is transformed into written text by the computer, commonly use a parser to help decide between the alternative possible mappings from acoustic signal to lists of words. If a speech recogniser has to choose between the following two mappings from signal to written text, the syntactic analysis will hopefully force it to select the first:

I'm hoarse because I scream
I'm horse because ice cream

One of the anticipated NLP applications which has so far proved elusive is a general purpose machine translation or interpreting system between any pair of natural languages. One part of such a system might be to take the input sentence and parse it, passing the output of the parser to a semantic interpreter. It has long been assumed by many logicians that “the meaning of a complex expression should be a function of the meaning of its parts” (Allwood et al 1983; 130), the compositionality principle which is often attributed to Gottlob Frege (see for example Frege

1952). The function referred to by which the meanings of the parts of a sentence are combined to form the meaning of the whole is often taken to be the syntactic structure. The meaning of, say, a noun phrase is first determined from the phrase's subparts according to the structure of the noun phrase, and then combined with the meaning of other phrases in the same clause, and finally other clauses, to produce a meaning of the sentence. This method is commonly adopted when arriving at a semantic representation for a sentence (see for example Dowty et al 1981, Gazdar et al 1985), and used not only in machine translation, but in other applications such as text abstracting and summarising, and NL interfaces (as described above).

Other applications in which a syntactic analysis module is presumed are grammar and style checkers, where the purpose of the program is to point out to an author (perhaps non-native) that the construct they have used is, according to the system's grammar, unacceptable, or at least idiosyncratic.

Although parsers are usually associated with the analysis and interpretation of text, there is a further application in the domain of text-to-speech generation. Text-to-speech synthesizers are already commercially available. In early products the speech produced was quite intelligible, but sounded unnatural, primarily because of the lack of varying intonation and other prosodic features such as pauses for breath at the end of a tone unit. Recent systems have achieved more natural sounding speech by parsing the text, and applying an intonation and stress pattern to the generated speech according to the structure of the sentence.

Although many of these much heralded applications for natural language processing are beginning to be realised, many are doing so by restricting the complexity of the problem. For example, a speech recognition device might require the speaker to train the system to recognise his individual speech, and to leave a slight gap between each word (Dragon Dictate is one example). A machine translation system might be developed for just a limited domain between two languages, for instance translating weather forecasts between English and French (METEO, TAUM Montreal).

It is fair to say that researchers in NLP have for a long time confined their efforts to producing relatively modest working prototypes with restricted lexical and grammatical capabilities. One of the reasons for their limited success is the fact that they have imposed somewhat tight boundaries

on the range of words and constructs the user can produce. With 'toy' lexicons and grammars of probably fewer than a hundred words and rules, fast, efficient parsers have been developed using simple algorithms. Of course, such parsers fail frequently when they come across a word or syntactic structure not described in the lexicon or grammar, but perform admirably within the chosen sublanguage. It is only when speakers (or writers) are given a free rein with their linguistic creativity that NLP systems will graduate from prototypes to a robust and mature technology, and go on to have the profound impact on society, commerce and industry which has long been envisaged. It is in this context that the so-called language industry is emerging.

In the last eight to ten years, however, this problem of handling realistic unrestricted English text has begun to be addressed by more than just by a limited number of interested research groups (Sampson 1990). Several of the theories and techniques that have been developed for the small prototype systems are coming under close scrutiny by researchers faced with all the complexity and size of the English language, as observed in the large collections of a variety of English texts which I am referring to as *corpora*.

Several major questions arise when considering the parsing of unrestricted English:

What sort of lexicon and grammatical formalism is capable of describing the complex syntactic relations of a natural language like English? Do existing grammatical theories capture many or all of the phenomena found in real use of spoken and written natural language, such as ellipsis, unfinished sentences, discontinuous, repeated and replacement elements? What sort of parsing techniques are suitable for the very large grammars and lexicons that describe unrestricted English? Can the rule-based techniques used for competence grammars be adapted easily, or are purely probabilistic methods the only realistic alternative? Is it possible to produce a parser which combines the simplicity and efficiency of rule-based parsing with the robustness of the probabilistic counterparts? Some researchers are also questioning the abstract or so-called rational approach to grammar and lexicon development, where rules are written intuitively using expert linguistic knowledge, using what Chomsky called *competence* in the language, to an empirical approach guided by the actual *performance* of the language (Lyons 1970; 38-39). A grammar which stems from one of these two different approaches may be called either a competence grammar or a performance grammar.

The research presented here attempts to answer some of these questions, and produce a solution to the problem of wide-coverage parsing of unrestricted English. This solution falls squarely within the performance rather than the competence paradigm of grammatical development.

1.4 Definitions.

In the preceding sections, I have tried to give a general introduction to the parsing problem, without providing any formal definition of technical terms. I will now assume the following definitions:

1.4.1 Natural Language.

Parsing occurs with respect to a particular *natural language*. A natural language is distinguished from mathematical, logical or programming languages by being articulated by native human speakers, and typically containing ambiguity and vagueness. A natural language is itself defined by a *lexicon* and *grammar* for that language. The lexicon and grammar may be referred to together as the lexico-grammar. The grammar may be divided into components specifying the structure of words (morphology) and sentences (syntax), and the meanings of words and sentences (semantics).

1.4.2 The Lexicon.

The lexicon theoretically lists all the wordforms (words) in the language, but in practice usually contains a more or less large subset of the morphemes in the language. Morphemes are the smallest semantically significant parts of words, and are referred to as stems and affixes. For instance, the word *trains* can be split into two morphemes: *train* and *s*. The suffix *s* is so called because it follows the stem. Since the morpheme *train* is syntactically ambiguous (it can be a noun or a verb), the suffix here could either be the plural ending for a noun, or the third person singular ending for a verb. A word is broken down into its component morphemes by a process called morphological analysis, and a program which does this is called a *morphological analyser*. A lexicon which consists of only morphemes is usually supplemented by a set of regular rules of morphology, which allow wordforms to be constructed from or broken down into their morphemes. If a wordform cannot be derived from the regular morphological rules for the

language, it is deemed to be irregular and stored in the lexicon as a separate morpheme. A typical product of lexical look-up and/or morphological analysis for a parser is the part-of-speech (syntactic tag/category) for a word (or more rarely a group of words treated as one item). Where a word is syntactically ambiguous, a lexicon should provide all the alternative parts-of-speech, but a syntactic tagging program (*tagger*) usually chooses the most likely single tag with respect to the surrounding words (and their tags).

1.4.3 Competence and Performance Grammar.

The grammar (as has been suggested in section 1.3) can be defined in two different ways: In one view, a competence grammar describes how the wordforms of the language can be combined to produce grammatical sentences acceptable to the native speakers of the language. A competence grammar is an idealised encapsulation of what native speakers consider to be acceptable and meaningful word combinations in the language, and is usually created by introspection on the part of the speaker.

In the second view, a performance grammar describes all the ways in which the wordforms in the language can be combined by a speaker when uttering the language. A speaker usually produces an utterance for some communicative purpose, such as conveying some meaning. An utterance may be spoken or written, and may consist of one or more sentences, or only part of a sentence.

The relationship between performance and competence grammars is that the performance grammar reflects the way the language is used, and so must account for performance features such as interruptions, repetitions, slips of the tongue and mistakes, whereas the competence grammar should reflect what the speaker would ideally produce without these ‘imperfections’.

1.4.4 The Grammatical Description.

A grammar contains a unique *description* of the language. The description includes labels for wordforms which behave similarly, classifying them into groups. The labels are commonly called parts of speech, such as noun, verb, preposition and adjective, but will be referred to here as word tags, and also as terminal categories (since they are the labels on the ends of the branches in a parse tree, and are attached to the wordforms themselves). The grammar description also

specifies recurring combinations of word tags, called constituents, which, in an individual grammar may be referred to as clusters, units, groups or phrases. These constituents themselves may be further combined into higher level groupings called clauses, which finally combine to make up a sentence. The labels for such constituents are called non-terminal categories. The label for the sentence constituent itself is called the root category. A description for unrestricted natural language is likely to contain many tens or even hundreds of grammatical categories depending on the level of delicacy the writer of the grammar wishes to identify. The finest-grained description would have a separate label for each wordform, which has led Halliday (1961; 267) to refer to the lexicon (lexis) as the most delicate grammar. In practice, grammatical descriptions tend to vary between a few dozen and upto around 200 categories. In some descriptions, the categories themselves are not treated as indivisible atomic items, but as combinations of syntactic *features*.

The constituent structure describes the *form* of the sentence, but a second level of description capturing the *function* of different parts of the sentence is employed in some grammars. The form of the string of words (15) is a noun phrase, but it may function differently depending on its position and relationship with other words in the sentence. In (16) the noun phrase acts as subject of the main verb. In (17) it acts as the verb's object, or complement, and in (18) it acts as the possessor in an enclosing noun phrase.

(15) *Two fat ladies*
(16) *Two fat ladies wanted a photo*
(17) *I photographed two fat ladies*
(18) *I took the two fat ladies' photo*

1.4.5 The Grammatical Formalism.

The relationship between the categories is captured by a particular grammatical *formalism*. Several formalisms can theoretically be employed for any one description, although it is typical for one formalism to be associated with one description (and be referred to simply as a grammar), usually because the description will have been created for a particular purpose, such as teaching the language structure to native or non-native learners, linguistic research, automatic parsing of sentences, or generation of sentences. Examples of formalisms that have been employed to

capture grammatical relations in sentence analysis are finite-state rules, context-free and context-sensitive phrase structure rules, and unrestricted rewrite rules. Each of these formalisms varies in its generative power according to a hierarchy commonly referred to as the Chomsky hierarchy, and as the power increases, generally the complexity of the formalism does so too. Not all sentences permitted by a context-free grammar would be permitted by a finite-state grammar, for example. For a more detailed introduction see for example (Lyons 1970; 47-82) or (Chomsky 1957). It has long been the goal of linguists to discover the least powerful formalism which would still adequately cover the sentence structures of a natural language. In models of sentence generation other formalisms are adopted. For instance, systemic grammar employs system networks to specify the semantic and syntactic options a speaker has. Each choice may be associated with a realisation rule, which partially specifies the sentence structure or a lexical element.

Although I have used the general term *grammar* in my distinctions between competence and performance, and between description and fomalism, in my development of a grammatical model for use in parsing, I will focus on the syntactic component of such grammars.

1.4.6 Systemic Functional Grammar.

If one's goal is to be able to parse unrestricted English, it is necessary either to create one's own grammatical description or select one from those ‘on offer’. The broad description chosen for this project is systemic functional grammar as defined by Fawcett (1981), and specifically its syntactic description as exemplified in the POW corpus. Systemic functional grammar (SFG) has developed from the linguistic traditions of J. R. Firth and Michael Halliday, and is primarily a grammar of language generation. That is to say it specifies how to generate acceptable sentences of the language from a set of semantic choices (technically referred to as *systems*). This might seem a bizarre choice in the light of this project’s aims in language analysis. However, a variant of the syntactic component of SFG has been applied in the manual analysis of the Polytechnic of Wales Corpus (Fawcett and Perkins 1980), and it is seen as desirable precisely because variants of the same description have been used for both analysis and generation. This decision is explained further in sections 1.5 and 2.2.

1.4.7 Language Corpora.

A set of utterances which have been produced by the speakers of a language may be collected as a *corpus*. A corpus is a strategic collection of texts, spoken or written, which attempt to represent the language as a whole. Corpora may also be collected to represent restricted usage of the language (sublanguages), such as that produced by children, or that used in legal, commercial or business environments, for example. A corpus is distinguished from an archive of texts by the fact that it has been strategically collected as a representative sample, rather than randomly assembled. We have already seen an example taken from a spoken corpus, in which the recorded speech was transcribed orthographically into a written form. Written corpora or orthographically transcribed spoken corpora which have received no annotation will be referred to as *raw* corpora. In some cases, however, the wordforms in a corpus will have been tagged with terminal grammatical categories, and will then be referred to as a *tagged corpus*. When a tagged corpus has further been annotated with full grammatical analyses for each sentence, it is called a *parsed* or *analysed corpus* or *treebank*.

1.4.8 Parsing.

Parsing is the assigning of one or more syntactic analyses to an utterance of the language, and a program which achieves this is called a parser. An analysis can be represented as a parse tree, which unambiguously captures one particular structure for an utterance (for examples see Figure 2). An utterance may possibly be assigned more than one analysis, in which case the utterance is considered to be syntactically ambiguous with respect to the lexico-grammar.

In the case where the utterance contains words or constructs not described in the lexico-grammar, the utterance cannot be assigned a well-formed parse tree, and is termed ungrammatical. However, we should distinguish here between an utterance which is ungrammatical with respect to a theoretically complete lexico-grammar, and one which is ‘ungrammatical’ because it cannot be analysed according to one implementation of a lexicon and grammar. Such an implementation will inevitably be incomplete because it perhaps does not contain a rare word or construct. In practice then, an ‘ungrammatical’ utterance may indeed be perfectly acceptable. A parser for unrestricted English will minimise the possibility that an utterance is incorrectly classified as ungrammatical either because it contains wordforms not in the lexicon, or combinations of

syntactic categories not described by the syntactic component of the grammar. The identification of semantically anomalous sentences is not seen as part of the parser's work, but is part of semantic interpretation.

Parse trees can be represented by nested labelled brackets, as shown in Figure 2. The relationship between a constituent and its subparts may instead be represented numerically, as example (2) in the original numerical format of the parsed POW corpus shows. (The bracketed form is repeated here for comparison).

(2a - bracketed)

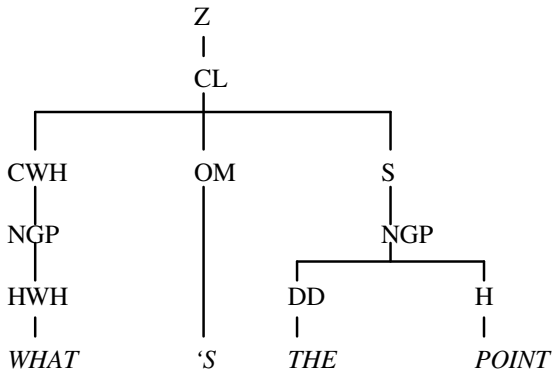
[Z [CL [CWH [NGP [HWH WHAT]]] [OM 'S] [S [NGP [DD THE] [H POINT]]]]]

(2b - numerical)

Z CL 1 CWH NGP HWH WHAT 1 OM 'S 1 S NGP 2 DD THE 2 H POINT

The structure is represented graphically in Figure 3.

Figure 3. A Graphical Representation of a Parse Tree.



In example (2), the category label *Z* is the root, representing the sentence itself. The root label contains a single clause, labelled *CL* and we refer to the relation between the two as that of *mother* and *daughter*. A mother is said to *dominate* its daughter, or daughters. The daughters of the clause in example (2) are *CWH*, *OM* and *S*. The reason for the numerical representation being used in the POW Corpus is to capture discontinuities in the daughters, i.e. cases where the

daughters do not directly follow each other in the sentence, but are interrupted by another constituent. Discontinuities can be represented more elegantly using the numerical rather than the bracketed format, and would require crossing lines between the categories in a graphical representation.

Having introduced some of the terminology of parsing, the context in which the parser is expected to work will now be described.

1.5 Application and Domain.

It is intended that the parser is designed as a general purpose tool, rather than having any single potential application in mind. However, during the early period of this research, I was working on phase 1 of the COMMUNAL Project (*CO*nvivial *M*an *M*achine *U*nderstanding through *NA*tural *L*anguage)[3] , directed by Eric Atwell at Leeds and Robin Fawcett at Cardiff, on the development of a parser which is expected to interact with a semantic interpreter, belief system and NL generator, as an interface to a knowledge-based system. As a consequence, this will be assumed to be a potential application for the parser, although I will not be attempting to explicitly link the parser to the generator's syntactic description, nor to limit its coverage to that displayed by the generator[4].

A further project in which the parser may yet find a home is AMALGAM (Automatic Mapping Among Lexico-Grammatical Annotation Models)[5], being conducted by Eric Atwell, Clive Souter and John Hughes at Leeds University. One of AMALGAM's aims is to produce a *multi-treebank* (a single corpus parsed according to several grammatical descriptions), and consequently there is a need for a general-purpose parser which can be adapted to work with such a variety of

---

[3] The COMMUNAL Project phase 1 was jointly sponsored by the Defence Research Agency's Speech Research Unit (then RSRE Malvern), ICL and Longman (UK). See, for examples of Leeds work, (Atwell and Souter 1988b, Atwell et al 1988, Souter and Atwell 1988a, 1988b, Souter and O'Donoghue 1991).

[4] Note that this approach to COMMUNAL as a possible application differs from the perspective of O'Donoghue (1993) and Weerasinghe (1994), who were developing parsers directly tied to a specific version of the COMMUNAL grammar. Their approach was essentially to assume the COMMUNAL grammar constituted a well-defined competence model of language, and to parse this well-bounded model.

[5] AMALGAM is sponsored by the UK government's Engineering and Physical Science Research Council (EPSRC) grant number GR/J53508. See for example (Atwell et al 1994).

grammatical descriptions. The use of the parser being developed here in the AMALGAM project will depend on its adaptability, efficiency and accuracy.

The domain of the parser (as distinguished from its application) specifies whether it is aiming to analyse a particular sublanguage of English. Perhaps rather ambitiously, at the outset, no restrictions are being placed on either lexicon or syntactic coverage. An alternative interpretation of the previous statement would be to say that the specific variety of English we hope to handle will be general, in as much as the lexical resources the parser will use will not be from a limited technical domain, and the range of constructs the syntactic grammar will describe will be extremely wide, as found in the POW corpus.

1.6 Scope.

As has been suggested in sections 1.2 and 1.3, producing parse trees may not be the final goal of an individual NLP system, (although there are some applications whose goal is primarily to produce such analyses, such as that of the AMALGAM project). The input to the parser may have come via other processes, such as speech recognition, and the output may be passed on to, say, a semantic interpreter or machine translation system. I will therefore delineate just where I expect the parser's work to begin and end, and specify what kind of text may be handled as input.

The parser will not be specially adapted to handle lattices of potential strings of words that might be produced by a speech recogniser. It will parse one utterance at a time, but will assign a score representing the likelihood of the analysis, so that alternative analyses can be ordered and compared. The parser will not be explicitly linked to a semantic interpreter for SFG or any other grammatical description. O'Donoghue (1990, 1991a, 1991e and 1994) has demonstrated a method for incorporating parse trees (virtually in the form to be produced here)[6] into a systemic semantic interpreter called REVELATION, which is still that being assumed by the COMMUNAL project.

---

[6] O'Donoghue's interpreter expects that the parse trees will include labels for SFG's participant roles, whereas the output of the present parser will not contain these. It is being assumed that the process of assigning such roles could be achieved automatically with a suitably large annotated lexicon for main verbs, developed according to Fawcett and Tucker's (1987) principles.

1.6.1 Input Format.

The input to the parser will be English language plain ASCII text in a string of words separated by blank spaces. It will not need to have been tagged grammatically with parts of speech, as the parser will incorporate a lexical look up phase. Like many others before me, I will however assume that the input has been preprocessed, into a format with the following characteristics:

- Characters are in lower case except for the word *I* and the first letter of any proper nouns: Utterances should not begin with upper case letters, unless they fall into the aforementioned categories.
- Sentence punctuation will have been removed, leaving only apostrophes and hyphens within words. Apostrophes in the morphemes *'s*, *'d*, *'ll* etc. representing the enclitic forms of the verbs *be*, *have* and some modals as well as those indicating possession (eg. *the boy's*, *the boys'*) should be separated from the noun phrase they are attached to by a blank space. Other apostrophes, such as those in the shortened negative morphemes of *isn't*, *don't*, *won't* and past tense endings *baa'd, ski'd* etc. should remain unseparated, as should those in monomorphemic wordforms such as *e'en*, *ne'er* and *ma'am*.
- Numbers appearing in the input should be written alphabetically, as say, *three hundred* rather than *300*.

1.6.2 Output Format.

The output resulting from the parser being applied to a single utterance will be a (possibly empty) list of parse trees ordered most likely first, in which the sentence structure will be captured by nested brackets. The syntactic categories in the parse trees will be those used in the systemic functional grammar found in the Edited Polytechnic of Wales Corpus (see section 2.1), and so will include both units and elements of structure (roughly corresponding to formal and functional labels respectively), but not participant roles. Although the structure of such trees is quite difficult for the human reader to read without close inspection (and bracket counting!), it does have several advantages. This format is amenable to direct display by the POPLOG programming environment's library program *showtree*, which produces more easily interpreted graphical tree representations. Such a facility would be important for a human selector/post-editor in the context of the AMALGAM project preparing a multi-treebank. Bracketed trees have the advantage that they are compact for storage purposes, and most easily utilised as list

structures in AI programming languages. Bracketed trees are also the starting point to storing a parsed corpus in the Nijmegen Linguistic DataBase, which provides parsed corpus browsing and search facilities (van Halteren and van den Heuvel 1990, Souter 1992). Finally, they are also probably the nearest to a standard tree representation (see Souter 1993).

1.7 Original Contributions to the Field.

Before describing the original contribution of this work to the field of computational linguistics, I would like to make clear the relationship it bears with that of colleagues who have been working alongside me over the past seven or eight years. The initial 18 month period of this research was spent working with the COMMUNAL team at Leeds and Cardiff, on a workplan devised by Robin Fawcett and Eric Atwell. In such a position, is quite difficult to separate out one's own research path from that of the project as a whole, and the early attempts at extracting a systemic functional grammar from the Polytechnic of Wales Corpus come under the auspices of COMMUNAL, as do investigations into morphological analysis and parsing using simulated annealing.

It has also been a problem (albeit a pleasant one) to be working part-time for my Ph.D. while Tim O'Donoghue was doing so full-time in the same subject area, often in close collaboration (see for example Souter and O'Donoghue 1991). Similarly, Rob Pocock was for 18 months a full-time research associate on the Leeds Speech Oriented Probabilistic Parsing (SOPP) Project. Rather than reinventing the wheel, the present research has, with due acknowledgment, built on some of their findings. I am grateful in this respect for Eric Atwell's guidance in keeping his research students working on separate but related paths.

After initially working quite closely with colleagues at Cardiff, the last six years of research have been conducted more or less independently of Robin Fawcett and his team, with only occasional communication, usually for clarification of some point in the grammar or corpus. Fawcett's research student Ruvan Weerasinghe has during this period also been developing a systemic-functional syntactic parser, which resulted in the publication of his thesis (Weerasinghe 1994) within a few weeks of my first submission. My own research had up to that point been conducted independently of his. However, in the last 18 months, I have benefitted from the insights and

experience he gained working primarily with the more up-to-date versions of the COMMUNAL syntax, as well as in capturing systemic-functional syntax dependencies in a probabilistic model.

Whereas at the outset of this project followers of the empirical or performance paradigm were still very much in the minority, it has become increasingly common in the last five years or so for parsers to take account of the likelihood of a syntactic structure. Simply producing a working probabilistic parser is no longer unusual. However, the parser developed here is original in its language engineering approach, having the following combination of elements:

It will produce trees annotated with a rich grammatical description, rather than the skeletal coarse-grained grammar which has been characteristic of much other corpus-based statistical parsing work (Lari and Young 1990, de Marcken 1990, Magerman and Marcus 1991, Perreira and Schabes 1992, Bod 1995)

Rather than using a limited hand-fashioned lexicon, or even a corpus word list, two alternative lexical resources will be developed and employed: (i) a large-scale, 60,000 wordform English lexicon (CELEX) has been transformed into a format compatible with a corpus-based grammar, and (ii) a POW-corpus trained version of Eric Brill's tagger (Brill 1992, 93, 94), both of which give the parser very good lexical tagging support.

Since Winograd's early work (1972), there has been a significant increase in attempts to use a systemic framework for syntactic analysis, instead of the more standard NL generation (Atwell et al (1988), Kasper (1988, 1989)[7], O'Donoghue (1993), Weerasinghe (1994) and O'Donnell (1994)). With the exception of the Leeds COMMUNAL work described in (Atwell et al 1988 and Souter and O'Donoghue 1991), each may potentially be linked to a corresponding SFG-based NL generator, achieving the desirable aim of using the same grammatical description in both language interpretation and generation. Kasper's parser however required the manual creation of some phrase-structure rules and the transformation of the *Nigel* generator grammar into a different formalism, Functional Unification Grammar (FUG: see Kay 1985). O'Donnell's work is similarly tied to the Nigel grammar. The syntactic models for O'Donoghue and

[7] Kasper was then working at the University of Southern California Information Science Institute (ISI) team developing a parser for their Nigel Grammar.

Weerasinghe's parsers are extracted automatically from a large sample of NL generator output[8], which limits their lexical and grammatical coverage to that of the generator. None of these has been developed using the syntactic coverage of an unrestricted language corpus such as POW, or incorporated any significant lexical resources. In each of these cases, the grammar will therefore only contain the structures which have been intuitively designed into the generators by their authors, i.e. they will be competence grammars.

Instead, the current parser contains a performance grammar, derived from a large genuine sample of unrestricted spoken English (the POW Corpus), including the grammatically problematic features of ellipsis, repetition and unfinished sentences. More importantly, the probabilities taken from such a corpus will be realistic, rather than artificial. Indeed, the range of grammatical constructs and their frequencies can be used to inspire the production of a generator's grammar, since it (currently, at least) goes beyond what the generator can handle.

Consequently, the parser being developed here is original in that it is derived from unrestricted English (albeit of children aged six to twelve), and in its very wide lexical and grammatical coverage.

1.8 The Structure of the Thesis.

The remainder of the thesis will be organised in the following manner: (Where relevant work of mine has already appeared in published form, I provide a reference).

Chapter 2 will present the background to the use of corpora in computational linguistics (Souter 1989b, 1990c, Souter and Atwell 1992, 1993), the chosen systemic functional syntax model (Souter 1990a), lexical look-up in NLP systems (Souter and Atwell 1988b), and parsing algorithms (Atwell et al 1988, Souter and O'Donoghue 1991).

Chapter 3 will explain how the grammatical and lexical resources were developed for the parser (Atwell and Souter 1988a, Souter 1990a,b, Souter 1993).

---

[8] The COMMUNAL NL generator, GENESYS (Fawcett and Tucker 1989), so called because it generates systemically, can be set to generate sentences randomly, to produce an artificial corpus or *ark* (Souter 1990a).

Chapter 4 discusses the chosen parsing algorithm, and how it was modified and improved to accommodate a systemic-functional syntactic model.

Chapter 5 presents an evaluation of the performance of two versions of the parser when tested with seen and unseen data, and discusses the nature of the parser's failures, its (in)efficiency and restrictions imposed by current hardware.

Chapter 6 describes improvements which might be made to the parser with respect to the development method, the linguistic data, and its efficiency.

Finally, in chapter 7 conclusions are drawn on the parser development process, how generally transferable it is to other types of grammar and corpora, and whether the original aims have been met successfully.

Chapter 2. Background to the Thesis.

Comments on related work will be divided into sections for each of the key elements contributing to parsing method: (i) corpus-based computational linguistics, (ii) syntactic description and formalism (which will itself depend on the availability of fully analysed corpora), (iii) associated lexical resources for the syntactic model, and (iv) suitable parsing techniques.

2.1 Corpus-Based Computational Linguistics.

In chapter 1, the distinction between competence and performance grammar was introduced. This section will explain why the choice was made to adopt a performance syntactic model for parsing unrestricted English, and to derive such a model from a corpus.

There are essentially two options for the grammarian or computer scientist faced with the task of developing a large-scale grammar for a natural language;

1. Use native-speaker linguistic intuition to create the rules of the grammar;
2. Study parsed corpora as the inspiration for common and uncommon grammatical structures which occur in the language.

Although they will be described separately below, in practice the competence and performance approaches do not diverge as widely as has just been suggested. When trying to build a large competence grammar, the grammarian will tend to look for inspiration in example sentences. Whereas when annotating a corpus, a prototype competence grammar or a handbook of case law examples will tend to be used. The main difference in the end result tends to be the size of the grammar, and its formalism. A competence grammar will be expressed in one particular formalism, whereas the parsed corpus will yield syntactic information for several formalisms. Typically the formalisms in which it is possible to extract a corpus-based grammar are less powerful than that found in a competence grammar. The former will normally contain atomic categories related through finite-state or context-free phrase structure grammars, but the latter may include a variety of enhancements to context-free grammar, such as categories as sets of

features which are combined by unification, feature percolation constraints, metarules, and transformations.

2.1.1 The Intuition-Based Approach to Developing a Grammar.

Chomsky's Transformational Grammar (Akmajian and Heny 1975, Radford 1981), which was prevalent in the 1960's and 70's, employed a phrase-structure rule component and a set of powerful transformational rules. More recently, grammarians have been concerned that the formalism should be psychologically plausible, and be able to be processed easily by computer programs. The result has been a whole host of feature augmented context-free grammars. In the mid to late 1980's, the grammar most in vogue was Generalised Phrase Structure Grammar (GPSG: Gazdar et al 1985), which replaced atomic category labels with sets of syntactic features, supplemented phrase structure rules with metarules, and, in common with a number of other grammars, had recourse to unification of syntactic features. The UK government's Alvey Programme sponsored a large scale project to produce a natural language toolkit (ANLT) for English containing a lexicon, morphological analyser, grammar and parser (Grover et al 1989, Pulman et al 1989, Philips and Thompson 1987). A close variant of GPSG was the chosen grammar for their project. The ANLT represents probably the largest and most advanced computer implementation of a competence grammar which is currently publicly available in the UK[1]. When expanded to its object grammar, it contains over 1,000 phrase-structure rules. Despite the obvious achievements which have been made in the ANLT project, the authors of its grammar are aware of some structures it does not describe, and assume that there are still more of which they are unaware (Grover et al 1989; 44).

This is a serious problem with the competence approach. Unless the language being covered is very restricted, the grammarian's intuition is likely to prove inadequate. It will be inadequate because it is idiolectic: being based on only one speaker's linguistic experience, and because it is highly likely that many structures in the language will be omitted, simply because they didn't come to the speaker's mind. The advantage of competence grammars is that the grammatical

[1] Other large-scale competence grammars exist, eg. COMMUNAL, SRI's Core Language Engine, but are not publicly available with user support provided.

relations are usually made explicit and written in a formal fashion, making them computationally tractable.

2.1.2 The Corpus-Based Approach to Developing a Grammar.

The second option, using corpora, stands a better chance of providing a comprehensive inventory of language structures, because it can be collected from several speakers over a period of time performing the language in different contexts. The problem with corpora, though, is that it is easy to assume they are exhaustive, when they are only as exhaustive as the corpus sample is representative of the language as a whole. A second problem with corpora comes precisely from their 'performance' nature. They contain language in all its natural beauty, warts and all! The mistakes and short cuts we make when using the language are all to be found in corpora, as well as some errors introduced by the process of collection itself, such as transcription, typing or optical character recognition errors. A further problem with the grammatical information in raw corpora is that it is only implicit. Fortunately, several corpora have already been annotated grammatically with word tags, and some even with full parse trees. This has been done either entirely manually, or with an automatic program based on a limited competence grammar or a manually annotated subsection of the corpus. In both cases, extensive proof-reading of the annotation is necessary to try to eliminate errors, and inevitably even the proof reading can be imperfect.

However the end product, a parsed corpus, contains a large sample of the language analysed according to a specific grammatical description, whose grammatical coverage usually far exceeds that possible using the competence approach.

Because of the enormous effort involved in annotating and post-editing text, only a handful of parsed corpora of English have been created at research sites in the UK and elsewhere. Those which are publicly available are:

1. The Gothenburg corpus (Ellegård 1978): 128,000 words of written American English, from the Brown corpus, analysed using a form of dependency grammar, including function labels;
2. The Nijmegen (CCPP) corpus (Keulen 1986): 130,000 words of written and spoken British English, including fiction and non-fiction texts, and 10,000 words of transcribed tennis commentaries, analysed using a context-free grammar derived from (Quirk et al 1972);

3. The Lancaster/Leeds treebank (Sampson 1987a): 45,000 words of written British English, from the LOB corpus, analysed using a specially devised surface-level phrase-structure grammar compatible with the CLAWS word-tagging scheme (Garside 1987);
4. The LOB corpus Treebank (Leech and Garside 1991): 144,000 words of written British English, a subset of the LOB corpus which was automatically parsed and manually post-edited using a parsing scheme slightly more coarse-grained than the Lancaster/Leeds Treebank;
5. The Polytechnic of Wales corpus (Fawcett and Perkins 1980, Souter 1989): 65,000 words of children's spoken British English, hand-parsed using Fawcett's systemic functional grammar (Fawcett 1981) which includes formal and functional labels;
6. The Penn treebank (Marcus and Santorini 1991, Marcus et al 1993): 3 million words of written American English, automatically parsed using Don Hindle's Fidditch parser (Hindle 1983), with a simple skeletal scheme consisting of 36 terminals, 12 punctuation markers and 14 non-terminals. This scheme is currently being redesigned to allow a more delicate annotation;
7. The Susanne corpus (Sampson 1994): 128,000 words of written American English, a reworking of the Gothenburg corpus into a more accessible and usable resource, analysed using an enhanced version of the Lancaster/Leeds treebank scheme, by a team at Leeds University.

The first five of these are described by Sampson (1992) in his consumer guide to the analysed corpora of English. In addition, the following two parsed corpora are used in academic research at Lancaster (and Leeds) and Nijmegen, but not publicly available:

1. The IBM/Lancaster Spoken English Corpus (Knowles and Lawrence 1987): 50,000 words of spoken British English recorded from BBC radio broadcasts, transcribed orthographically, phonetically and prosodically, and parsed using a scheme referred to as skeletal parsing. The scheme is so called because it is marginally less delicate than that used in the Lancaster/Leeds treebank and involves building a skeleton tree structure, to be labelled with a fairly basic list of categories. The SEC forms part of a larger private enterprise in corpus annotation between IBM and Lancaster University, which has produced a parsed corpus of 3 million words, called The Skeleton Treebank (Leech and Garside 1991).

2. The TOSCA corpus (Oostdijk 1991): 1.5 million words of written British English, of which at least 250,000 words have been semi-automatically parsed using extended affix grammar, which includes formal and functional labels.

The uses of parsed corpora are reviewed in (Souter and Atwell 1994), and a selection of samples from some of the aforementioned corpora (no's 1, 3, 4, 5 and 6) is included in Appendix 1. Parsed corpora tend to be small, only in the region of 50,000 to 150,000 words. compared to the vast raw corpora which are now being collected, many of which contain several tens of millions of words (for example ICE, British National Corpus, Bank of English, ACL Data Collection Initiative, European Corpus Initiative). These larger collections are driven by the respective aims of collecting international varieties of English, providing an inventory for lexicographic development, and assembling archives for computational linguistic research. Despite their relative poverty from the viewpoint of lexical coverage, parsed corpora are (by definition) grammatically rich, providing the best available resource for a parser whose scope is unrestricted English.

2.1.3 Selecting a Parsed Corpus.

Large, fully annotated corpora are still relatively few and far between, because of the tremendous manual effort required to perform the grammatical annotation. The annotators' response to the size of the task has been to follow one of two paths; either to use a skeletal manual parsing scheme which achieves more rapid progress and ultimately a larger unrefined corpus, or to use a very detailed scheme and be resigned to ending up with a small but highly refined annotated corpus. In the former category are the IBM/Lancaster treebank and the ACL Data Collection Initiative's Penn treebank, part of which is available on CD-ROM. In the latter category are the Lancaster-Oslo/Bergen (LOB) corpus treebank, the Polytechnic of Wales (POW) corpus, the Nijmegen corpus, and the Gothenburg and Susanne corpora (both annotated portions of the Brown corpus). Since the grammatical description of the corpora in the former category is only coarse-grained, these will be eschewed in favour of a richer description. In contrast to the other parsed corpora, at the outset of this project, the POW corpus had not been developed or ‘adopted’ by a research team investigating corpus-based parsing. It was originally collected as a resource for the study of child language development. Consequently, it has been chosen as the basis for an integrated lexicon, grammar and parser. Further reasons for this choice of corpus and

grammatical description are my familiarity with SFG from work on the COMMUNAL project, the fact that the grammar has been extended to handle features of unrestricted spoken English, because it contains both formal and functional labels, and because it has the potential to be reversible between NL interpretation and generation.

2.1.4 The Polytechnic of Wales Corpus.

This section will introduce the background to the POW corpus, including an explanation of its notation and format. A brief summary of this information was published in the Lancaster Survey of English Machine-Readable Corpora (Taylor et al 1991) and is reproduced in Appendix 2. A short handbook was written to accompany the distributed version of the corpus (Souter 1989), both of which are available from the International Computer Archive of Modern English (ICAME) at Bergen[2]. The corpus was originally collected between 1978-84 for a child language development project to study the use of various syntactico-semantic constructs in children between the ages of six and twelve. A sample of approximately 120 children in this age range from the Pontypridd area in South Wales was selected, and divided into four cohorts of 30, each within three months of the ages 6, 8, 10, and 12. These cohorts were subdivided by sex (B,G) and socio-economic class (A,B,C,D). The latter was achieved using details of

- 'highest' occupation of both the parents of the child, or one of them in single-parent families
- educational level of the parents.

The children were selected in order to minimise any Welsh or other second language influence. The above subdivision resulted in small homogeneous cells of three children. Recordings were made of a play session with a Lego brick building task for each cell, and of an individual interview with the same 'friendly' adult for each child, in which the child's favourite games or TV programmes were discussed.

2.1.4.1 Transcription.

The first ten minutes of each play session commencing at a point where normal peer group interaction began (i.e.: when the microphone was ignored) were transcribed by 15 trained

[2] The International Computer Archive of Modern English, Norwegian Computing Centre for the Humanities, P.O. Box 53, Universitetet, N-5027 Bergen, Norway, or e-mail icame@hd.uib.no for more information.

transcribers. Likewise for the interviews. Transcription conventions were adopted from those used in the Survey of Modern English Usage at University College London, and a similar project at Bristol. Intonation contours were added by a phonetician to produce a hard copy version, and the resulting transcripts published in four volumes (Fawcett and Perkins 1980). A short report on the project was also published (Fawcett 1980).

2.1.4.2 Syntactic Analysis.

Again ten trained analysts were employed to manually parse the transcribed texts, using Fawcett's version of Systemic-Functional Grammar (SFG). The SF syntax used in the analysis handles phenomena such as raising, dummy subject clauses and ellipsis. Despite thorough checking, some inconsistencies remain in the text owing to several people working on different parts of the corpus. SFG in general, and the particular SF syntax found in the POW corpus are described in more detail in section 2.2. The parsed version is available in machine readable form but does not contain any of the prosodic information included in the paper version.

2.1.4.3 Corpus Format.

The resulting parsed corpus consists of approximately 65,000 words[3], in 11,396 (sometimes very long) lines, each containing a parse tree. The corpus of parse trees fills 1.1 Mb. There are 184 files, each with a reference header which identifies the age, sex and social class of the child, and whether the text is from a play session or an interview. The corpus is also available in wrap-round form with a maximum line length of 80 characters, where one parse tree may take up several lines, but this makes it difficult to distinguish between numbers used for sentence reference and those which specify syntactic structure. The four-volume transcripts can be supplied by the British Library Inter-Library Loans System.

A short portion of the corpus in its original form is included in Figure 4.

---

[3] Earlier papers quote the size of the corpus as being approximately 100,000 words. The latest automatic extraction of a wordlist from the machine readable corpus shows it to be just over 65,000 words, but this figure can only be approximate. Noise in the original typing of the corpus in the form of omissions of category labels, or of the spaces between such labels and the words in the text, makes it difficult to give an accurate figure. The difference between the two totals is almost certainly the difference between the total for the recorded spoken texts, and the total for those which have been hand-parsed.

Figure 4. A Sample Section of a POW Corpus File.

**** 58 1 1 1 0 59
6ABICJ *(filename)*
1) [FS:Y...] **Z** 1 **CL** F YEAH 1 **CL** 2 S NGP 3 DD THAT 3 HP ONE 2 OM 'S 2 C NGP 4 DQ A 4 H RACING-CAR
2) Z CL 1 S NGP 2 DD THAT 2 HP ONE 1 OM 'S 1 C NGP 3 DQ A 3 MO QQGP AX LITTLE 3 H TRUCK
3) [HZ:WELL] Z 1 CL 2 S NGP HP I [RP:I] 2 AI JUST 2 HAD 2 C NGP 3 DQ A 3 MO QQGP AX LITTLE 3 H THINK 1 CL 4 & THEN 4 S NGP HP I 4 M THOUGHT 4 C CL 5 BM OF 5 M MAKING 5 C NGP 6 DD THIS 6 HP ONE
4) Z 1 CL 2 S NGP HP I 2 AI JUST 2 M FINISHED 2 C NGP 3 DD THAT 3 HP ONE 1 CL 4 & AND 4 S NGP HN FRANCIS 4 M HAD 4 C NGP 5 DD THE 5 H IDEA 5 Q CL 6 BM OF 6 M MAKING 6 C NGP 7 DQ A 7 RACING-CAR
5) [FS:THEN-I] Z CL 1 & THO 1 S NGP HP I 1 M MADE 1 C NGP DD THIS
6) Z CL 1 & THEN 1 S NGP HP FRANCIS 1 OX WAS 1 AI JUST 1 X GOING-TO 1 M MAKE 1 C NGP HP ONE 1 A CL 2 B WHEN 2 S NGP H YOU 2 M CAME 2 CM QQGP AX BACK 2 CM QQGP AX IN
7) [NV:MM] Z 1 CL F NO [FS:FRAN...] 1 CL 2 S NGP HP WE 2 M HAD 2 C NGP 3 DQ AN 3 H IDEA 3 Q CL 4 BM OF 4 M MAKING 4 C NGP 5 DQ FOUR 5 H THINGS
8) Z 1 CL F YEAH 1 CL 2 S NGP HP I 2 M PLAYED 2 C PGP 3 P WITH 3 CV NGP HP IT 2 A PGP 4 P AT 4 CV NGP H HOME
9) Z CL F YEAH

The tree notation employs numbers rather than the more traditional bracketed form to define mother-daughter relationships, in order to capture discontinuous units. The number directly preceding a group of symbols refers to their mother. The mother is itself found immediately preceding the *first* occurrence of that number in the tree. In Figure 4, the first tree shows a sentence (*Z*) consisting of two daughter clauses (*CL*), as each clause is preceded by the number 1. The long lines have been folded manually here for ease of reading. The first number in each tree is a sentence reference; after which I have inserted a closing bracket, ")", for ease of reading. These do not appear in the corpus itself. All alphabetic characters are in upper case. The only lower case alphabetical characters are in the sentence references, which have occasionally been subdivided into 24a, 24b etc., where what was initially analysed as one sentence was, on checking, re-analysed as two (or more).

Occasionally when the correct analysis for a structure is uncertain, the one given is followed by a question mark. Cases where unclear recordings have made word identification difficult are

treated similarly. Apart from the syntactic categories and the words themselves, the only other symbols in the tree are three types of bracketing:

- square [NV...], [UN...], [RP...], [FS...], for non-verbal, unclear/unfinished, repetition, false start, pragmatic element etc.
- round (...) for ellipsis of items recoverable from previous text.
- angle <...> for ellipsis of items not so recoverable, eg: in rapid speech.

Filenames indicate precisely which age (6,8,10,12), social class (A,B,C,D), sex (B,G) and recording situation (play-session (PS) or interview (I)) is involved, followed by the child's initials. Hence, the text sample in Figure 4 is from file 6ABICJ, involving a six year old, of social class A, who is a boy, in an interview, with initials CJ.

2.1.5 The Edited Polytechnic of Wales Corpus.

Because the original POW corpus contains various typographical and syntax errors which still remain after extensive proof reading, the automatic extraction of its grammar and lexicon is hampered, as described further in chapter 3. Consequently, Tim O'Donoghue has created an edited version of the corpus, which has become known as EPOW (O'Donoghue 1991b, 1991c). O'Donoghue used semi-automatic post-editing programs which searched for word and category patterns which broke the rules for a legitimate parse tree, and then amended them. He also ran a spelling checker over the words in the corpus, to find examples of typographical errors. The end product of his work is a much ‘cleaner’ resource with which to work, and is used as the training data for the parser under development here.

2.2 Systemic Functional Grammatical Description and Formalism.

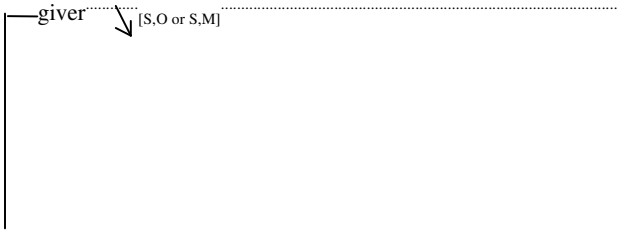
The aim of parsing relatively unrestricted English has necessitated a corpus-based, performance approach to grammar development, rather than the intuition-based competence approach. The choice of a parsed corpus comes hand in hand with the choice of a grammatical description, so must be made together, unless a new corpus parsing venture is to be embarked upon. I will now consider what effect on parsing the choice of systemic functional syntax will have, and how this

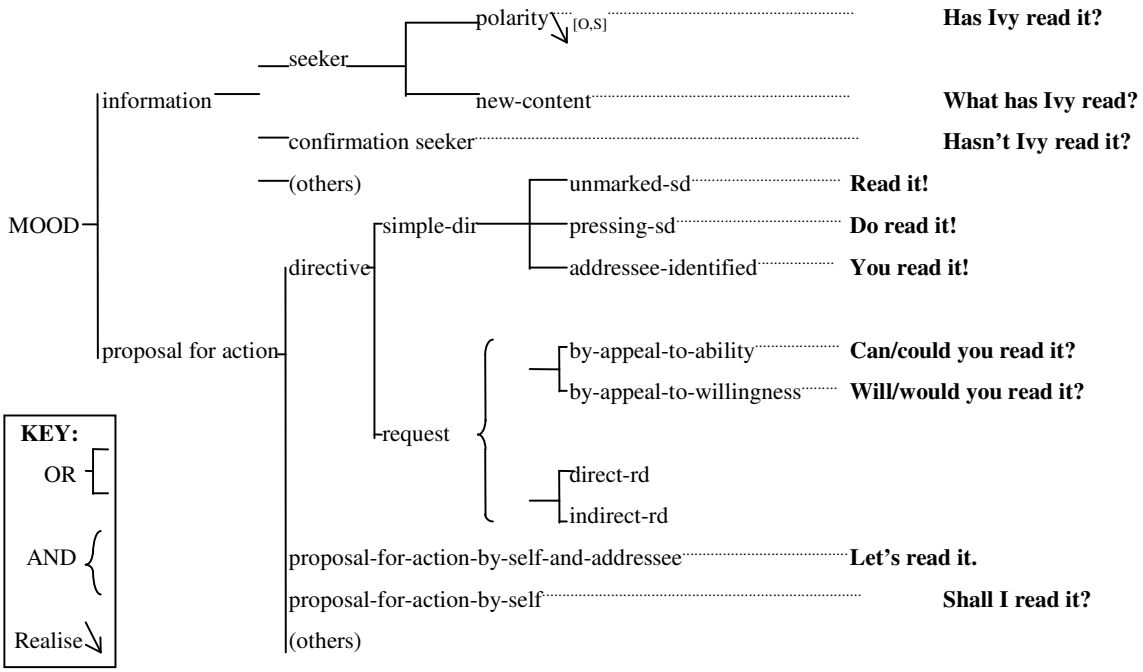
compares with other grammatical descriptions and formalisms. I will begin, however, by describing systemic functional grammar itself.

Systemic functional grammar differs markedly from many other language models in that it is a grammar which models the way language is produced, rather than the way language is interpreted. The syntactic structures with which we shall be dealing in virtually all of the rest of the thesis are not the centre of the language model at all, but are a product (via a process called *realisation*) of a series of semantic choices which are formalised in a *system network*, which is the heart of the model and is intended to represent the meaning potential of the language. The language model focuses on the choice of what sentence to generate, rather than how to understand a sentence that someone has already produced. Useful introductions to SFG are found in (Fawcett 1980, 1981, 1984, Halliday 1985, Butler 1985, Fawcett et al 1993, O'Donoghue 1993, Weerasinghe 1994). The SFG model of language we shall be dealing with here is that developed by Fawcett from the earlier work of Halliday, and has provided both the syntactic description found in the POW corpus, and the large grammar being built into the NL generator of the COMMUNAL project, called GENESYS (described in more detail in section 2.2.1).

SFG is heavily focused on semantic choices (called *systems*) made in generation, rather than the surface syntactic representations needed for parsing. When generating a sentence using systemic grammar, a number of choices of features are made in a system network. These vary from broad semantic and syntactic choices such as whether a sentence will be positive or negative, active or passive, to highly specific choices which select individual lexical items. Hence the grammar is often referred to as a lexico-grammar, since, in generation, the lexical, syntactic and semantic choices are all linked together. The choices may be linked together by disjunctive or conjunctive logic (choice of one feature may exclude or require the choice of others). Progress through the large networks may necessitate disjunctive or conjunctive entry (the prior selection of one or more features before the next can be chosen). A small fragment of a (simplified) system network for *mood* in English is illustrated in figure 5, taken from (Weerasinghe and Fawcett 1993). This example illustrates conjunctive and disjunctive systems (see the key), but not conjunctive and disjunctive entry conditions.

Figure 5: A fragment of a system network for MOOD in English.



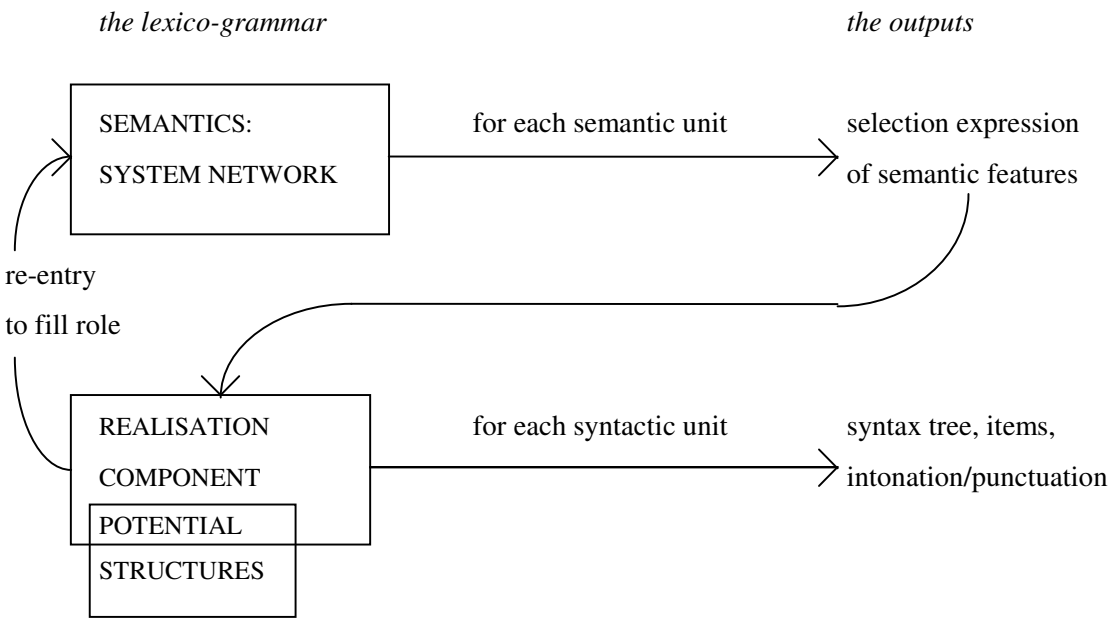


Some of the choices cause *realisation rules* to become active, indicated by a downwards pointing arrow in figure 5. The realisation rules are normally numerically referenced and stored separately. In figure 5 the overall effect of just two rules has been informally shown, and reflects the placing of the Operator (O) before the subject (S) (with features [information], [seeker] and [polarity] chosen) or after (with [information] and [giver] selected). In the GENESYS implementation of this and other networks, each choice point has a probability attached to it, expressing the linguist's intuition of the relative frequency of the feature, and allowing the generator to be set to produce sentences randomly. (It is here where the evidence from a parsed corpus such as POW can be useful in guiding the probabilities to be set, for systems directly affecting syntax and lexis, at least. It is the ability of the generator to work randomly which has enabled researchers such as O'Donoghue (1993) and Weerasinghe (1994) to generate a large sample of output; an artificial parsed corpus, or *ark*.) Realisation rules can have several effects, including

1. Causing the network to be re-entered after the current pass through has finished. This typically occurs to specify a functional role (realised as a (sub)constituent of a clause), after the clause itself has been created;
2. Causing some part of the structure of the sentence to be created, for example by specifying the ordering of the subject nominal group and the verb operator (auxiliary);

3. Causing a particular syntactic category to be expounded as a particular lexical item.

These components of the GENESYS lexico-grammar can be illustrated as follows (Fawcett et al 1993; 121):



The output of such a lexicogrammar is a parse tree, the structure of which is then deleted to leave the sentence itself. The leaves (words) are finally processed by the generation system into either a punctuated sentence or an utterance with an intonation pattern. Since the focus of this thesis is on parsing and not generation, we will not give a full blown account of this generation process, nor other examples of the many systems in the network, or their associated realisation rules. Suffice it to say that SFGs such as that used in GENESYS represent some of the largest competence grammars yet to have been constructed by computational linguists (Fawcett et al 1993; 119). A good example following the process of generation through system networks and realisation rules for English personal pronouns is given in (Fawcett 1988b).

Whereas in language interpretation the grammar and lexicon tend to be stored as separate entities, in systemic generation they are stored together in the system network. However the syntactic description can be viewed independently of this formalism in the trees the generator produces, or indeed in the Polytechnic of Wales corpus. The syntactic part of (an early variant

of) the description is also informally presented in (Fawcett 1981), which acts as a case law of example sentences and their structures, which would have been invaluable in hand parsing the corpus. Were I to be explicitly trying to link the output of my parser with the COMMUNAL generator then

“the generative model that is the core component of (the) SFG (would have to be) the source of the information that (would be) built into the parser” (Fawcett 1994; 398, my parentheses).

However, instead I will consider the syntactic evidence obtainable from the annotated POW corpus to be the linguistic data (in their own right) on which the parser should depend, since they represent a performance rather than a competence model.

2.2.1 Systemic Grammar in NL Generation.

Some well known implementations of systemic grammar generators have been produced: The first was Davey's *Proteus* (Davey 1974, 1978), which generated discourse for the game of noughts and crosses, and drew inspiration from earlier work by Winograd (1972). The second, which contains the *Nigel* grammar has been created by the Penman Project at the University of Southern California's Information Sciences Institute, has been used to generate both English and Japanese (Mann 1983, Matthiessen and Bateman 1991). A further generator developed at Edinburgh and derived from Nigel is Patten’s SLANG (Patten 1988). Houghton and Isard (1987) employed a systemic functional model for their FRED and DORIS system, in which two robots plan how to be together in the same room! *GENESYS*, a separate implementation of a NL generator containing a very large systemic grammar, has been produced by the COMMUNAL team at Cardiff (Fawcett and Tucker 1990, Fawcett 1990, Fawcett, Tucker and Lin 1993). A sample of sentences produced by GENESYS (prototype 1.5) is shown in Figure 6. (The latest ‘midi’ and ‘maxi’ versions produce a much greater variety of sentences and structures.) The figure shows the syntactic structures along with the sentences generated. The leaves of the trees (the words themselves) are also shown separately to aid the reader.

Figure 6. A Sample Section of GENESYS Output.

[Z [Cl [S/Af [ngp [dq some] [vq of] [dd that] [h paper]]][Xc isn't] [Xp be+ing] [M cook+ed] [e .]]]

[Z [Cl [S/Af [ngp [h what]]] [Xf is] [G going_to] [Xc be] [M die+ing] [e ?]]]

[Z [Cl [S/Ca [ngp [dq some] [vq of] [ds [qqgp [dd the] [a best]]] [vs of] [meme [qqgp [a happy]]] [h question]]] [Xr hasn't] [M glow+ed] [Ati [ngp [h today]]] [e .]]]

[Z [Cl [Xr have] [S/Ag [ngp [ds [qqgp [dd the] [a best]]]]] [M damage+ed] [C2/Af [ngp [h itself]]] [Ama [qqgp [a how]]] [e ?]]]

[Z [Cl [O don't] [M be] [C2/At [qqgp [a easy]]] [e !]]]

[Z [Cl [Xf are] [S/Ag [ngp [ds [qqgp [dd the] [a sad+est]]] [vs of] [h that]]] [G going_to] [Xc be] [M stand+ing] [Cm2 about] [e ?]]]

[Z [Cl [S/Af [ngp [h [genclr [g its]]]]] [M kiss+s] [Ama [qqgp [dd the] [a best]]] [e .]]]

[Z [Cl [Xr have] [S/Ca [ngp [h the_Chancellor_of_the_Exchequer]]] [M been] [C2/At [qqgp [a good]]] [e ?]]]

[Z [Cl [S/Af [ngp [ds [qqgp [dd the] [a best]]] [vs of] [h it]]] [Xc was] [Xp be+ing] [M broken] [C2 [pgp [p by] [cv/Ag [ngp [h who]]]]] [Ama [qqgp [a fast]]] [e .]]]

some of that paper isn't be+ing cook+ed .
what is going_to be die+ing ?
some of the best of happy question hasn't glow+ed today .
have the best damage+ed itself how ?
don't be easy !
are the sad+est of that going_to be stand+ing about ?
its kiss+s the best .
have the_Chancellor_of_the_Exchequer been good ?
the best of it was be+ing broken by who fast ?

These early examples of GENESYS generator output illustrate some of the key differences between the parse trees produced by the generator and those found in the POW corpus. Firstly, these have yet to be passed through a process converting them into the finished spoken or written output, so morphological rules have not been applied to *damage + ed* to produce *damaged*, for example[4]. Secondly, on the face of it, some of this output appears semantically anomalous, in the same way as Chomsky's famous *colourless green ideas sleep furiously*. The generator, in normal use, would produce output in response to a previous utterance in line with some overall conversational plan, and would be subject to higher semantic constraints. Some of these examples are also syntactically imperfect, needing the application of subject verb agreement control, for example. More significantly, though, from the point of view of training material for a parser, the tree structures contain an extra level of labelling, called participant roles, which is conflated onto the nodes for elements of structure within the clause or lower down the tree. In the tree

[Z [Cl [S/**Af** [ngp [ds [qqgp [dd the] [a best]]] [vs of] [h it]]] [Xc was] [Xp be+ing] [M broken] [C2 [pgp [p by] [cv/**Ag** [ngp [h who]]]]] [Ama [qqgp [a fast]]] [e .]]]

[4] It so happens that all of these examples result from the [written] feature having been chosen in the spoken/written system of the network.

the labels *S/Af* and *cv/Ag* indicate a subject acting as the *affected* element of the main verb, and a completive being the *agent* causing the action, in this case of something being broken. Such participant roles were not included in the labelling for the POW corpus, so will not be able to be automatically extracted in a syntactic formalism trained on the corpus. The trees also contain a less delicate syntactic description than that found in the corpus, (although obviously this difference is becoming less significant as the generator grammar grows). Nevertheless, the GENESYS trees also contain one further important feature - they are all well-formed with respect to *headedness*, a point we shall return to in section 2.2.4, where we see that the same feature certainly isn't guaranteed for POW corpus trees.

2.2.2 Systemic Grammar in NL Parsing.

We focus here on the use of SFG in parsing. Non-SFG approaches are described in section 2.4.

Parsing in SFG should ideally be part of a wider process of interpretation - taking a sentence and finding the set of semantic features which would have been chosen to generate it. Theoretically at least, it may be possible to derive the semantic features directly from the words in the sentence to be interpreted, given a mechanism for searching the system networks and realisation rules. However, most researchers have assumed that it is sensible to first find one or more syntactic structures for a sentence before mapping those structures onto the relevant semantic choices. Systemic functional syntax is normally represented in the form of realisation rules associated with choices in the system network of semantic features. Both of these formalisms have to be hand-crafted in a SFG model, and are not immediately amenable to standard parsing techniques. One might therefore be tempted to conclude that the goal of unrestricted NL parsing and the use of SFG were in some way at odds with each other, since one could only interpret as much as one could generate. The range of syntactic structures in a corpus of unrestricted English is likely to be considerably broader than that hand-crafted into the system network. The solution to this dilemma in the present work is to allow wider syntax than, say, the COMMUNAL generator can currently handle, and assume that the generator builders will attempt the task of expanding the lexico-grammar to approach that of a corpus by becoming more and more comprehensive, as Fawcett (1992; 23) proposes.

Despite its clear focus towards language generation, several computational implementations of systemic grammar for interpretation do exist. The earliest and best known of these is probably Winograd's SHRDLU (Winograd 1972), which could understand commands and queries in a simple blocks world. More recently, Kasper has developed a parser for the Nigel grammar which uses the Functional Unification Grammar formalism (Kasper 1988, 1989). The Nigel grammar has also been central to the systemic interpretation work of Patten (1988) and O'Donnell (1994). These researchers tend to focus on the difficult problem of semantic interpretation using system networks and realisation rules in reverse, as does O'Donoghue (1994). If we separate out the process of syntactic parsing from semantic interpretation, Fawcett and Tucker's COMMUNAL SF syntax has been used in the syntactic parsers of O'Donoghue (1991d, 1993) and Weerasinghe (1994).

O'Donoghue's parser uses a vertical strip grammar, which captures relations between a leaf (word) and its successive parent nodes working through the tree to the root (sentence) node, instead of traditional phrase structure rules, which relate an ordered horizontal set of daughters to one mother. The grammar is automatically extracted from an artificial corpus of English produced by GENESYS, The parser associated with the grammar, the vertical strip parser, achieves correct results 81% of the time on 1,000 test sentences which were generated by GENESYS (like those contained in the trees in Figure 6), but not used for the grammar extraction. The parser's success is limited to sentences which contain structures of the same depth (in terms of vertical strips) as those in the training material, and employs a lexicon of 940 items extracted from GENESYS PG 1.5 output.

Weerasinghe's probabilistic on-line parser (POP) has been carefully developed (like O'Donoghue's) to act as a robust syntactic parser to keep pace with the grammar used in the COMMUNAL generator. His results are very good (85% exact match success on a test comparable to O'Donoghue's). His parser uses a modified chart parsing algorithm, with a combined probabilistic model derived from both the POW corpus and GENESYS output. The syntactic model, however, is obtained solely from GENESYS output. This level of success is achieved by a combination of modifications to a standard chart parser using a context-free grammar. Firstly, the syntactic model consists of a horizontal component capturing probability of transition from one element to another (akin to the linear precedence model in GPSG, but with added probabilities). There is a vertical component capturing the likelihood of a particular

mother for any daughter (akin to a probabilistic immediate dominance model in GPSG). The lexicon consists of only 355 items extracted from the GENESYS 'maxi' lexico-grammar. The parsing algorithm itself has been modified to control explosion in the agenda caused by large-scale grammars. This has been achieved by a one-word look ahead before proposing a new chart hypothesis, by ordering the agenda according to likelihood, but favouring wider-spanning edges over those containing fewer words, and (crucially) by relying on the appearance of a head for each constituent before such a constituent can be built (Weerasinghe 1994; 110-11). These very promising results have yet to be replicated with large-scale lexicons and unconstrained conversational dialogue input.

At Leeds, a series of other parsing attempts based on a genuine corpus-trained syntax model taken from the POW corpus began in 1988, when Atwell and Souter (1988a) attempted to load a POW Definite Clause Grammar into POPLOG Prolog, but found that memory limitations prevented the built-in parser-generator function from being able to be applied to such a large grammar. Then, in the first phase of the COMMUNAL project (in which Leeds were partners with the Cardiff team), a simulated annealing parser called the RAP was developed, using a stochastic finite state automaton derived from POW as an evaluation function for the putative trees proposed by annealing (Atwell et al 1988, Souter 1989a, b, Souter and O'Donoghue 1991). The resulting parser was never fully tested, since it was extremely slow, and unreliable on sentences containing long-distance dependencies. However, O'Donoghue later trained the annealing parser (by then re-named the Dynamic Annealing Parser, DAP) on some GENESYS output, and achieved exact match success rates of around 30%, on tests for 'unseen' generator output (O'Donoghue 1993; 114). This forms the background to the present parser development. It is important to remember that the purpose of parsing need not solely be to progress towards a more robust systemic interpreter - parsers have many other uses in which a systemic description may be profitable. We now look at some of the difficulties a rich systemic-functional grammar presents for parsing programs.

2.2.3 Problems for a SFG Parser.

From the point of view of parsing, several problematic features of SFG should be highlighted. Firstly, the grammatical description includes not only formal categories (units), but also functional labels (elements of structure) and a further set of labels for the participant roles which

are attached to the main verb in a clause, such as the *agent* who performs an action, and its *affected* entity, or the *carrier* of an attribute, or the *location* of an entity or action. When a constituent of a sentence is given a label specifying its form, it may also therefore be given a functional label and a participant role (depending on the particular constituent). As we have seen, the examples in Figure 6 include participant role labels *Ag*, *Af* and *Ca*, which are conflated onto node labels for elements of structure. The designer of a parsing program for SFG has to decide where to derive the grammatical model from, and whether to consider applying these different layers of labelling on separate nodes or have them conflated onto one level.

The fact that the grammar was developed for generation may have an adverse affect on parsing. For instance, a lexical item may for the purpose of generation be labelled differently according to the larger construct it is part of. Such multiple labellings exacerbate the problem of ambiguity in parsing. In recent versions of SFG different labels would be given to the wordform *of* in examples (1) to (3).

(1) *Some of the men*
(2) *Part of the cake*
(3) *A picture of John*

The clear distinction between terminal and non-terminal categories in the grammar is not maintained in SFG. A non-terminal category which most frequently labels some higher level constituent in the sentence can exceptionally be employed as a terminal category if the lexical item it labels cannot be productively combined with other items within that constituent. For example, the label *AM*, (modal adjunct), would normally be given to each of examples (4) to (6).

(4) *Quite possibly*
(5) *Almost certainly*
(6) *Maybe*

(4) and (5) would be given a constituent structure in (7), whereas (6) would have the simple structure of a terminal category (8), since the word *maybe* cannot be productively combined in a modal adjunct. Ideally, a large-scale lexicon should reflect this fact, but most lexical resources

(other than a lexicon extracted from the POW corpus) fail to capture these distinctions, so alternative solutions have to be devised.

(7a) *[AM [T quite] [AX possibly]]*

(7b) *[AM [T almost] [AX certainly]]*

(8) *[AM maybe]*

2.2.4 Systemic-Functional Grammar in the POW Corpus.

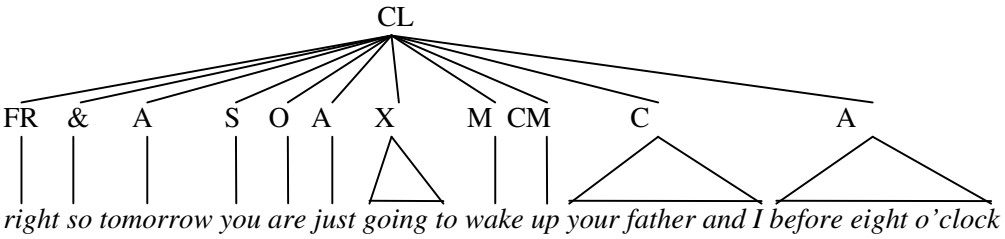
From the point of view of parsing, the most comprehensive and explicit description of SF syntax is found in the POW corpus. The version of SF syntax used in analysing the POW corpus was produced in the early 1980's, and was therefore a forerunner of that used for the COMMUNAL project's generator. The latter is constantly being amended and refined by Robin Fawcett and Gordon Tucker. The corpus grammar is preferred to that contained in the generator because it is stable, offers the possibility of extracting an authentic probabilistic grammar and lexicon, and as yet contains a wider range of constructs than the generator can produce. However the POW corpus does not contain participant roles.

As a reader not familiar with systemic linguistics may have already discovered, the terminology of SFG is quite different to that of generative grammars such as Transformational Grammar or Generalised Phrase Structure Grammar. In the corpus, a syntax tree is characterised by having two alternating types of category labels. The first are called *elements of structure*, such as Subject (S), Complement (C), Adjunct (A), head (h), modifier (mo) and qualifier (q). Note that, in a hand-analysis, capital letters are used for elements of *clause structure*, and lower case letters for elements of *group* (and *cluster*) structure. In the machine-readable version of the corpus, capitals are used throughout. Elements of structure are *filled* by the second type of category, i.e.: *units*; elements of clause structure are filled by either subordinate clauses, *groups*, (cf. phrases in TG or GPSG) such as nominal group (ngp), prepositional group (pgp) and quantity-quality group (qqgp), or *clusters* such as genitive cluster (gc). Terminal elements of structure are *expounded* by lexical items. The top-level symbol is *Z* (sigma in the hand-written form) and is invariably filled by one or more clauses (Cl). Trees tend to be fairly flat, but richly labelled, immediately below the clause level, notably because of the absence of a predicate or verb phrase constituent. (This has a direct effect on the size and shape of the formal grammar which can be extracted from the parsed corpus, as illustrated in chapter 3). Some areas have a very elaborate description, eg: there

are 15 types of adjuncts, six types of modifiers, nine different determiners, and ten auxiliaries. Other categories are relatively coarse, eg: main-verb (M), head (h), and apex (ax). (The apex occurs in a quantity-quality group, and is typically expounded by an adverb or adjective). A comprehensive list of all the categories used in the hand parsing of the POW corpus is given in Appendix 3, with details of whether the symbol is used as a non-terminal or terminal category (or both), and some example lexical items which expound the terminal categories. The main componence relationships found in the corpus will now be exemplified in turn.

The Clause

The clause displays the greatest range of structural variation in the corpus. The declarative main clause *right so tomorrow you are just going to wake up your father and I before eight o'clock* might be labelled with following constituents:



where *FR* = frame, *&* = linker, *A* = adjunct, *S* = subject, *O* = operator, *X* = one or more auxiliary, *M* = main verb, *CM* = main verb completing complement, and *C* = complement. This by no means exhausts the potential lists of daughters in a clause, as Appendix 3 shows. Similar variations are possible for other clauses for imperatives, interrogatives and subordinate clauses, with most of the constituents being optional, some being able to be repeated at several positions, and some being mutually exclusive. Weerasinghe (1994; 111) states that the head of the clause in SF syntax is the main-verb (M), and uses this to build well-formed clause edges in his parser. The main verb is certainly the semantic head (being responsible for the pattern of participant roles in a clause) and is certainly responsible for some syntactic patterns such as potential complementation, but one may argue that the operator, as the first auxiliary, performs the job of syntactic head (being the element which agrees with the subject in person and number), or indeed the complete set of auxiliaries and modals, since they show the tense and mood of the clause. If we explore the evidence found in the POW corpus, and generously take the operator, main verb

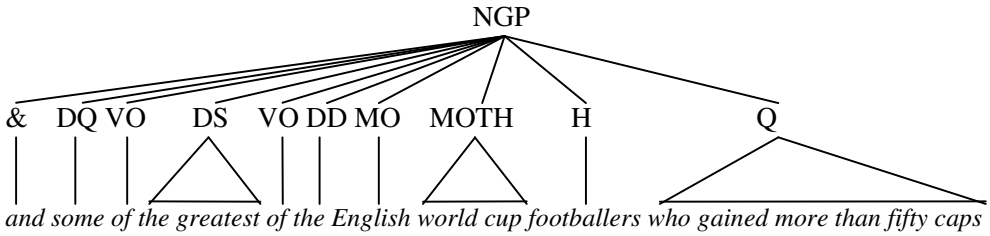
or any auxiliary to be a possible head daughter for the clause, we obtain some interesting figures for headless clauses found in the corpus.

Mother	**Alternative Heads**
CL	*M, O, OM, OMN, ON, OX, OXN, X, XM, XMN, XN*

Of all the componence rules for clauses found in the corpus, 28.6 % of the tokens are found to be without such a head. Several of these will describe the structure of a clause containing only a formula, such as *yes* or *no*. Some others will be the result of ellipted verbs or auxiliaries, whilst others belong to neither of these categories. Most of the headless clauses are those containing only a subject, a complement, an adjunct or even just a conjunction.

The Nominal Group

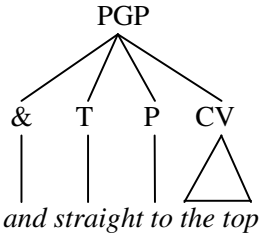
A (particularly productive) nominal group *and some of the greatest of the English world cup footballers who gained more than fifty caps* might consist of the following daughters:



where *&* = linker, *DQ* = quantifying determiner, *VO* = *of*, *DS* = superlative determiner, *DD* = deictic determiner, *MO* = one or more modifiers, *MOTH* = one or more thing-modifiers, *H* = head, and *Q* = one or more qualifier. Again the variation is much greater than shown here, permitting pronouns and namelike heads of the nominal group. If we again look at the corpus, using the set of possible NGP heads to be {*H, HN, HP, HPN, HSIT, HWH*}, we find that 9.6 % of componence rule tokens appear without a head. The headless nominal groups tend to be just determiners, modifiers, or both.

The Preposition Group

The other units in Fawcett's SF syntax for POW show somewhat less variation. The preposition group *and straight to the top* maximally involves four possible daughters:

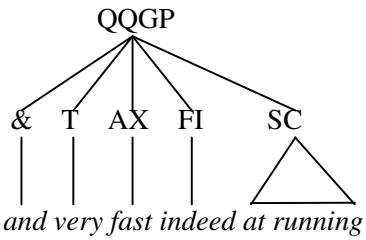


where *&* = linker, *T* = temperer, *P* = preposition and *CV* = completive. The first two of these elements are optional, and there is an alternative label for the head (*PM*) where it is a main verb completing preposition (for prepositional verbs). One might assume that both the preposition and completive were compulsory, and certainly the preposition itself, but taking the head of a *PGP* to be either *P* or *PM*, we find 3.9% of componence rules for PGP occurring without such a head in the corpus. The headless preposition groups contain just completive NGPs (i.e. the prepositional complement on its own)

The Quantity-Quality Group

This unit covers two constituents often dealt with separately in other grammatical descriptions, those of adjective group and adverbial groups, since the potential structure of the two is the same, despite their different functions as modifiers, qualifiers and adjuncts, for example. The structure of

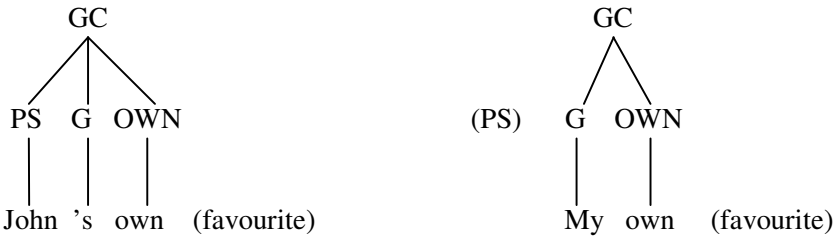
and very fast indeed at running is superficially the same as would be obtained by replacing the word *fast* (adj) with *quickly* (adv):



Here, the labels are & = linker, T = temperer, AX = apex, FI = finisher an SC = scope. The use of AX to label heads of (quantity quality groups filling) modifiers, qualifiers and adjuncts introduces a great deal of syntactic ambiguity into the parser, which could be avoided if the tag labelling had been distinct in each of these cases. However in SFG the emphasis is on classifying functional roles in generation (at the expense of ease of syntactic analysis, in this case). The apex (AX) is the head of the QQGP, with alternative forms for superlative/comparative (AXT) and wh-variants (AXWH). If we treat all three of these as possible heads, when exploring the corpus, we find only 0.4% of QQGP componence rule tokens are lacking a head. The headless quantity-quality groups contain temperers, scope and finishers each on their own or in combination.

The Genitive Cluster

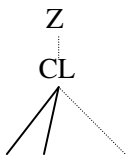
The genitive cluster is used in SFG to express belonging or possession, which, in English is found with the possessor being a full nominal group, or being pronominal, such as

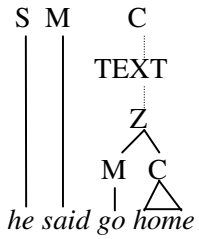


Here the labels are *PS* = possessor, *G* = genitive element and *OWN* = owner. Note that in the case of a pronominal, the possessing nominal group is subsumed within the genitive element. In the case of nested possessors, the *PS* element is filled by a further *GC*. The genitive element is the head of the cluster.

The Text Unit

The final unit found in the POW corpus is the text unit, which labels citations, such as *he said 'shut up'*. There is little variation in this rare structure, with the *TEXT* label consisting of a sub-sentence label, *Z*. Notice that this structure automatically licenses an alternative analysis (the top of which is shown by the broken lines in the example below) for almost any sentence, in which it is being uttered as a response to a question such as *What did he say*?





To conclude this description of the SF syntax found in the POW corpus, and in particular the data on headless units, we will estimate the proportion of the whole corpus which consists of utterances containing at least one constituent unit without a head. The percentage figures I have given are obtained by searching through all the componence rules extracted from the corpus (for more on this syntactic formalism see section 3.1). Each rule has an observed frequency, (which can be represented as a percentage), and by adding together the percentages for all clause rules without heads, we get an overall figure of 28.6%. It is not straightforward to associate this result directly with the number of sentences in the corpus which are 'ill-formed' in this way. Every tree would have to be checked individually, to see if it matched the list of headless rules. We can assume though that there will be some examples of sentences which contain more than one headless
constituent (co-ordinated clauses for example). Consequently, we might estimate that
about 25% of all sentences have at least one headless clause. However, some of the headed clauses
would also contain headless subconstituents, so a figure between 25-30% of all sentences is a fair estimate of the number of sentences missing at least one head in some part of their substructure. This finding has significant repercussions for any method of parsing which relies on the notion of syntactic head, including the Alvey Natural Language Tools Parser, and that developed by Weerasinghe.

2.2.5 Choosing a Grammar Formalism and its Effect on Parsing.

Several different grammatical descriptions and formalisms have been devised by linguists, logicians and computer scientists. As yet, there is no consensus as to which most adequately and elegantly captures all the complexity of unrestricted natural language. It is likely that this situation will persist, as grammars tend to be developed for varying purposes, most notably either

for language interpretation or language generation. However, two endeavours which are attempting to introduce an element of competition into grammar and parser development are the US DARPA sponsored Message Understanding Conference, and the Limerick Workshop on Industrial Parsing of Software Manuals. In the former conference, research teams are asked to automatically parse a set of unseen randomly chosen texts, to extract pre-defined fields of semantic content, and their results are graded and published. In the latter, the focus has been more on syntactic rather than semantic content. One of the main findings of such endeavours has been that it is very difficult to compare different parsing schemes in an objective manner, and that there is still little agreement as to what a parser should produce.

Having chosen the SFG contained in the POW corpus as a grammatical description, there remains the selection of a formalism for the grammar. Bound up in this decision is the choice of parsing algorithm, since different parsing algorithms work with different formalisms.

The grammar description can be extracted from the parsed corpus automatically in any formalism which is compatible with the way the parse trees in the corpus have been represented. Typically this will be limited to using a finite-state grammar, context-free grammar, or perhaps a vertical strip grammar (examples of each are given in chapter 3). None of the parsed corpora mentioned in section 2.1 have been annotated with category labels which are complexes of features, rather than being atomic (although many of the atomic labels are constructed from two or three letters which include the grammatical information some features contain). As a consequence, the corpus-based grammar will very possibly be less powerful (in terms of the Chomsky hierarchy) than its counterparts in the competence paradigm.

Geoffrey Sampson, who was involved in both the Leeds/Lancaster Treebank and the Susanne corpus-annotation projects, has argued that evidence from such corpora suggests that the number of rules needed to describe just the noun phrases in the Leeds/Lancaster treebank is open-ended (Sampson 1987b). Certainly, if the number of unique rules extracted from hand-parsed corpora are in a simple context-free phrase-structure rule formalism, they number several thousand (Atwell and Souter 1988a, Souter 1990) and are by no means exhaustive. Taylor, Grover and Briscoe (1989), in response to Sampson, argue that it is the very simple nature of the formalism which causes such open-endedness. They claim that, given rules which capture generalisations such as recursion, and categories as sets of features rather than atomic labels, the number of rules

needed to describe English noun phrases (and we are given to assume, English grammar in general) can be reduced to a much smaller, finite set. One example in the POW corpus where this is not the case is in the handling of agreement (between subject and main verb, for example). The POW corpus parse trees do not explicitly mark person and number agreement between categories, so introducing a feature which enabled this to happen would, although desirable from the parsing viewpoint, not reduce the categories in the grammar.

If we are to automatically take advantage of the grammatical information contained in the POW corpus (or any other parsed corpus), we will not be able to use non-atomic category labels, as these were not included in the original analysis. Taylor et al.'s grammar, derived from the Alvey Natural Language Toolkit (Grover et al. 1987), was manually constructed and successively modified in the light of the LOB corpus data. Unfortunately, no such formal grammar was created during the hand parsing of the POW corpus, and certainly not one which allowed for recursion in the rules, and non-atomic category labels. This is because the aim of the POW corpus compilation was for the study of child language development, and not natural language processing. Nevertheless, we are able to extract very large syntactic formalisms which are amenable to parsing, in the form of finite state models, sets of context-free rules, and dominance rules such as vertical trigrams and vertical strips. However, it is not possible to directly extract the principal SFG formalism which is amenable to NL generation, a system network, directly from the corpus text. The syntactic formalisms which are amenable to parsing are not necessarily at odds with the SFG model for generation, they merely have a different focus. They focus purely on the formal (and some functional) aspects of the SFG model, without addressing the semantic basis of SFG.

I am somewhat cautious as to the ultimate value of manually building a large rule-based syntax model, as there will always be some new sentences which contain structures not catered for in the grammar, so any parser using such a grammar will hardly be robust. Defenders of the competence paradigm would argue that corpora have gaps in too, which is a valid point. But the gaps in corpora will almost certainly be fewer. Furthermore, the endeavours of the TOSCA group working under the direction of Jan Aarts at Nijmegen University, Holland have been to incrementally build such a rule set using Extended Affix Grammar (EAG: Aarts and Oostdijk 1988), with reference to their TOSCA corpus. Their grammar now consists of several thousand rules, and parsing frequently results in several tens or even hundreds of ambiguous analyses. The

choice as to which analyses are the "right" ones must appeal to syntactic, semantic and pragmatic levels of information. Taylor et al. (1989: 258) also came across this kind of large-scale ambiguity problem. Rather than manually search through all the ambiguous parses for the semantically "correct" one, they decided only to manually apply the rules in the grammar to check that the semantically correct analysis could potentially be found. They also assume that spurious analyses may be filtered out by some sort of semantic component.

The scale of this multiple ambiguity problem lends weight to a final reason for adopting a corpus-based formalism. Corpus-based grammars provide the opportunity for recording the *frequency* of a wordform or construct. Such frequencies can be used to modify the search strategy of the parsing algorithm chosen for the grammar, and consequently order by likelihood the ambiguous analyses a large-scale grammar produces. If experiments show that (one of) the most likely parse(s) is the correct one, then it will not be necessary to have the parser produce all possible solutions. One implementation of a parser which adopts this approach using the POW corpus SFG exists: the Realistic Annealing Parser (see Atwell et al 1989, Souter and O'Donoghue 1991), which is described in section 2.4.

So far in this chapter I have argued in favour of a corpus rather than intuition-based approach to computational linguistics, and in particular for the use of parsed corpora as a source of grammatical information for parsing. As the primary source of linguistic data I have chosen the Polytechnic of Wales corpus, and the systemic functional description it contains, since there is more *truth* in the corpus than in an artificial corpus of generator output. Next, I will consider the lexical facilities that a wide-coverage corpus-based parser might need.

2.3 Lexical Resources for Corpus-Based Parsing.

The development of lexicons for natural language processing has in many ways followed the same path as the development of grammars. In small scale systems, researchers were contented simply to choose a core list of words they would like to be able to deal with, and hand-craft the lexicon entries with phonological forms, syntactic categories and semantic fields and representations, etc. The lexicon would perhaps be adequate for the few sentences the researcher was interested in, but useless for anyone concerned with unrestricted English. In the present project, we would ideally like to be able to provide, for any wordform in the language,

appropriate grammatical tags from the terminal categories in the POW corpus, and preferably a probability measure of the occurrence of the wordform with each particular tag. For instance, given the wordform *bricks*, the lexical lookup process which initialises the parser might return

[[73 BRICKS H][1 BRICKS M]]

which would provide the parser with the information that *bricks* can be a noun (H) or main verb (M) with frequencies of 73 and 1 respectively. Such frequencies could be turned into probabilities by dividing the frequency of occurrence with a particular tag by the total frequency of occurrence with any tag (in this case 74). In a speech recognition application, this probability should then be multiplied by the probability of the wordform occurring in the language, which can be estimated from a raw corpus. However, in the current application, I use only tag probabilities, since the nature of the input is not in question.

There are at least three approaches one might adopt to the provision of a large-scale lexicon for robust parsing. Firstly, we could attempt to extract a list of wordform/wordtag correspondences with their frequencies from the chosen corpus itself, or use a corpus-trained probabilistic tagger akin to the constituent likelihood automatic word-tagging system (CLAWS) (Leech et al 1983, Atwell et al 1984). Alternatively, we could use a traditional dictionary-style morpheme list and a morphological analyser to strip off affixes before looking up a word. As a third option we could list all the morphological variants in the lexicon itself without taking advantage of the regular rules of English morphology, and thereby make the lexicon much larger. In practice, each of these options has its advantages and disadvantages. One problem common to them all is how to handle wordforms not covered by the lexicon[5], such as proper nouns, neologisms, compounds and idioms, akin to the problem of grammatical undergeneration.

2.3.1 Corpus-Based Tag Assignment.

Currently, wordlists with disambiguated frequency information can only be obtained from grammatically tagged or fully annotated corpora. These lexicons tend to be larger than could

[5] The CLAWS approach has a set of affix rules it uses to help tag assignment, rather than lexical look up. However, a lexicon is used for the cases when the affix rules fail, and for idioms.

easily be produced by hand (see section 3.2 for an example from the POW corpus), but still not really adequate for a project aiming to handle unrestricted English. They have the advantage that they do provide disambiguated frequencies for wordforms in the corpus, and consequently can be used as lexicons for prototype probabilistic parsers which do not have pretensions of unrestricted lexical coverage. The POW corpus, which contains 65,000 words, yields a lexicon of 4,618 unique words with syntactic categories and their disambiguated frequencies. To obtain a larger wordlist would require an extremely large SFG tagged corpus (which sadly doesn't exist).

An alternative to straightforward lexical look-up from a corpus-derived lexicon is a tagging program based on the same SF syntax model. Two approaches to building such a tagger exist, based on either a probabilistic model, or on co-occurrence rules. A probabilistic method has been used to produce the 1 million tagged LOB corpus (Johansson et al 1986), which resulted from a semi-automatic approach to word tagging, called constituent likelihood automatic word-tagging system (CLAWS). A portion of the Brown corpus which had first been grammatically tagged by a rule-based program, and then corrected by hand was used to extract a probabilistic model of the relations between word tags in context. The program was then able to indicate unambiguously what the grammatical tag should be for some new word in the corpus, achieving 95-96% accuracy. Remaining errors were corrected by a manual post-editing phase to create a completely tagged corpus. The 1 million word tokens are examples of around 50,000 word types, which is a sizeable lexicon, although this does include many proper nouns and other items such as punctuation marks which would not normally be contained in a traditional dictionary.

The main aim of the CLAWS project was, however, not to produce a lexical look up procedure for a parser, but to create a tagging program which could also be re-used on other corpora (Garside 1987). To be used as a first step in SFG parsing, the tagger would need to be retrained to deal with the SFG grammar, by extracting tag co-occurrence frequencies from the POW corpus. Although an early version of CLAWS is now publicly available, it has not to my knowledge been designed to automatically accommodate other tagging schemes. Alternatively, the output of CLAWS as it stands might be mapped onto SFG categories. These options would probably reduce its accuracy, depending on how bound CLAWS is to the type of published, written language the LOB corpus contains.

A further probabilistic (Markov model) tagger has been developed by Church (1988), trained on the tagged Brown corpus, and employing corpus-based bigrams and trigrams, as well as lexical probabilities. The PARTS tagger has reported accuracy rates of between 95-99%, depending on text type and evaluation measure.

Recently, two different teams have developed tagging programs able to surpass (the lower end of) these success rates, using context rules, with limited use of probabilities. In Helsinki, a formalism to describe English grammar has been built called ENGCG (English Constraint Grammar, see for example Karlsson et al 1995). This model is essentially a hand-crafted approach using linguistic knowledge, in the form of a large lexicon and of morphological rules. As well as assigning parts of speech to each word, a skeletal parse outlining noun phrase structure is performed, and some functional elements such as a clause's subject can be recognised. The same formalism is being used for English, Finnish, Swedish, German and Basque. It is not clear how one could make use of this tagger without also subscribing to the tagging scheme.

Another recent alternative to CLAWS, which **can** be trained on the POW corpus data, is the Brill tagger (Brill 1992, 93, 94). Instead of using a purely stochastic approach of modelling co-occurrence of parts-of-speech, Brill's model instead learns a small set of context rules from a tagged corpus. He uses a technique called transformation-based error-driven learning to acquire these rules, which are like a hybrid between a rule-based and a pure stochastic model. The method consists of

(i) extracting a lexicon from part of a manually tagged training corpus which he refers to as the truth;

(ii) stripping off the tags from the truth, to make a raw corpus;

(iii) making a first guess at the best tag by choosing the most frequent one from the lexicon;

(iv) comparing the guessed tags to the truth, to calculate an error rate.

(v) generating all possible transformation rules (from a set of about six templates). An example template is 'change tag *a* to tag *b* when the preceding word is tagged *y* and the following word is tagged *z*'. An example transformation rule resulting from this might be 'change the tag from noun to verb if the previous tag is modal'.

(vi) apply each of the many rules in turn, and evaluate by comparing the updated tags to the truth.

(vii) select the rule that reduces the most errors, go back to (iii) and repeat until no new rule reduces the error rate, or a preset threshold is reached.
Brill originally trained his model to tag the Penn Treebank, with success rates of 96-7% for single tag assignment (just in excess of CLAWS), and up to 99% success when the evaluation criteria are relaxed to allow an average of 1.5 tags per word. The original tagger described here now also contains facilities to permit tagging of words not found in the training corpus (although with a lower success rate of 85%), and to permit re-training on other tagged corpora. John Hughes has trained the Brill tagger trained on the POW corpus, among others (Hughes and Atwell, forthcoming). Examples of context rules and the lexical tagging rules produced by this training process are found in Appendices 10 and 11. The POW corpus is relatively small, compared to other tagged corpora used by Brill (of up to 1 million words), so we may expect a reduction in tagging success rate. In particular, its lexical coverage is very small, so the rules for tagging unknown words will be based on a limited training corpus.

The Brill tagger assigns just one tag per lexical item, so if it fails, it will have a significant impact on parsing. Equally, it has no mechanism for dealing with multi-word lexical items, unless these are tokenised together by means of hyphens. Consequently, we may wish to explore alternative lexical look-up techniques in parallel, such as using dictionary material.

2.3.2 Dictionaries and Morphological Analysers.

Machine readable dictionaries (MRDs) such as the Longman Dictionary of Contemporary English (LDOCE: Procter 1978) and the Oxford Advanced Learners Dictionary (OALD: Hornby 1974) are lexical resources which offer some hope for computational linguists interested in robust parsing, but until very recently have developed using the lexicographer's competence, rather than a large scale observational survey of the language. One dictionary which has resulted from the analysis of a large corpus (of 20 million words) is COBUILD (Sinclair 1987), but unfortunately the machine readable version is not, to my knowledge, freely available for academic research.

Substantial reformatting and re-organisation of a MRD is often necessary before it becomes a lexicon tractable for NLP work (Atwell 1987, Boguraev and Briscoe 1987, 1989, Wilks et al

1988), but this work is undoubtedly time saving compared to compiling your own lexicon of the same size. An example of the reformatted bracketed LDOCE lexicon is shown in Figure 7.

Figure 7. Section of Lispified LDOCE.

```
((abate)
        (1 A0001600 !< a *80 bate)
        (3 E!"beIt)
        (5 v !<)
        (7 100 !< I *DE !< ---- !< ----T)
        (8 (of winds !, storms !, disease !, pain !, etc. !.) to become less strong !; decrease : *46 The ship waited till
                the storm abated before sailing out to sea)
        (7 200 !< T1 *46 often pass !. !< ---- !< ----T----T)
        (8 *46 lit *44 to make less : *46 His pride was not abated by his many mistakes)
        (7 300 !< T1 !< LW-- !< ----H----T)
        (8 *46 law *44 to bring to an end(esp !. in the phr !. *45 abate a nuisance *44))
        (10 1 !< !  ment !< !< n !< U !<))
((abattoir)
        (1 A0001800 !< ab *80 at *80 toir)
        (3 !" *67 bEtwA : R)
        (5 n !<)
        (6 !< BrE)
        (7 0 !< !< AHBZ !< B---NF--X)
        (8 *CA slaughterhouse))
((abbess)
        (1 A0001900 !< ab *80 bess)
        (3 !" *67 b9s !, !" *67 bes)
        (5 n !<)
        (6 C !; N !<)
        (7 0 !< !< RL-- !< ----F---Y)
        (8 a woman who is the head of a religious establishment (*CA CONVENT *CB) !, formerly called
                an *CA ABBEY *CB !, for women *63 compare *CA ABBOT))
```

A reduced version of such a computational lexicon was derived from LDOCE for use in the Alvey Natural Language Toolkit's morphological analyser (Carroll and Grover 1989). The analyser was reviewed by Souter and Atwell (1988b), who drew the following conclusions.

It should be possible to expand the ANLT lexicon to cope with the full capacity of LDOCE, but this would exacerbate a problem with the morphological analyser: its slow speed in producing solutions. This would be ameliorated by improving the system hardware, but counter-balanced by the extra load of the parsing program itself.

The morphological analyser does not assign weights to its analyses, a problem common to all MRDs. Frequency information in the entries would be useful for probabilistic NLP.

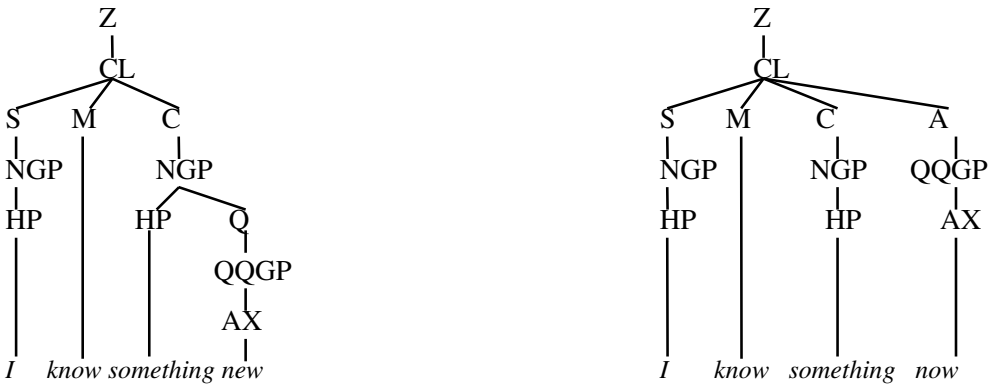
A further problem with the analyser and with MRDs in general is that the grammar used in the syntax entries will certainly not be amenable to use in NLP (Akkerman et al 1985, 1988). It may categorize idiosyncratic features which were of interest to the dictionary editor, but not to the computational linguist, making it difficult or impossible to achieve an automatic mapping between the dictionary syntax and that chosen for the parsing scheme. However, a preliminary manual mapping between LDOCE syntactic categories and those in SFG was produced for the COMMUNAL project (Souter and Atwell 1988b; 73-76), and is reproduced in Appendix 4.

2.3.3 Lexical Databases.

Although MRDs such as LDOCE contain a wide variety of information which could potentially be used in NLP systems, much of this information is not needed for parsing. MRDs also carry the overhead of a morphological analyser, which it may be possible to avoid. Some MRDs have been processed beyond simply a computationally tractable list format (as in Figure 7) into lexical databases. Such databases may include all variant wordforms of a stem, rather than just the stem itself, which saves the expense of using a morphological analyser. Furthermore, the lexical information derived from MRDs can be supplemented easily with new database fields, for example for lexical probabilities. One such lexical database is the CELEX database (Burnage 1990) of Dutch, English and German, compiled in Nijmegen, Holland. The English component contains over 80,000 wordforms (rather than stems) obtained from the intersection of the headwords of LDOCE and OALD, and expanded into all their morphological variants. The syntactic categories in the database are derived from LDOCE. Each wordform is given a frequency from the Birmingham (COBUILD) Corpus, but unfortunately the frequencies are not disambiguated; the entry for *books* as a third person verb and the entry as a plural noun have the same frequency (163 occurrences per million words). CELEX has nevertheless been used as a source lexicon for the aforementioned ENGCG tagger at Helsinki.

CELEX by no means represents the final word on lexical database resources. A team directed by Yorick Wilks at New Mexico State University, Las Cruces, has also converted LDOCE into database form (Wilks et al 1989) as has a team from Cambridge University (Alshawi et al 1989). The European Community has funded a collaborative project ACQUILEX which is developing a multilingual lexical database primarily for use in semantic taxonomies, semantic equivalence and feature-based syntactic work (see eg. Meijs 1993a, 1993b).

The need remains when using a lexical database for a mapping from its grammatical description to that in the parsed corpus, but if the database's grammar has been compiled from more than one dictionary, there is a better chance of achieving such a mapping automatically or semi-automatically. In practice it is highly likely that manual intervention will be needed to map to a fine-grained corpus grammar model. In the case of systemic functional grammar trees, the labels directly dominating the words are elements of structure depicting the word's function, rather than (an absent level of labelling) describing the form of the word. For instance, were the apex of a quantity-quality group to be further distinguished as either an adjective or and adverb, then in lexical look-up and parsing we would be able to quickly separate the analyses for *new* as a qualifier (*Q*) and *now* as an adjunct (*A*) in the following:



As it stands, the final word of both of these examples will be tagged *AX*, which licenses both the qualifier and adjunct reading for either example.

2.3.4 Choice of Lexical Resource.

It is necessary to select at least one approach to provide a lexical resource compatible with the chosen grammar, which is sufficiently computationally tractable to be used together with the

grammar in a wide-coverage parsing program. Initial attempts to map between the syntactic category information of the CELEX and the POW corpus SF syntax were promising, and as this resource also contained wordform frequencies, a lexicon drawn from the CELEX English database has been adopted as one solution to the tagging problem. CELEX was selected in preference to the other databases because it included frequency information, was freely and readily available, and was not still under development[6]. The modification of the CELEX lexicon and look up performance tests on the POW corpus are presented in section 3.2. In addition to CELEX, the POW-trained version of the Brill tagger was used for a separate parsing test, with results for both described in chapter 5.

2.4 Parsing Techniques.

Parsers can be classified in different ways, according to the purposes they are built for and the techniques they employ (which in turn is related to the syntactic formalisms they adopt). Weerasinghe (1994; 7-10) separates parsers on the basis of the disciplines they emerge from - psychology, logic, mathematics, statistics and AI. A more coarse-grained classification would still have to allow for parsers belonging to the mathematical/theoretical tradition (exemplified by Tomita 1991), the computing tradition of parsers as compilers of programming languages, or the linguistic tradition of parsers for assigning syntactic structure and resolving ambiguity.

The purposes of parsing have already been discussed (see section 1.3), and vary according to whether the syntactic analysis is the final goal of the exercise, or whether the output of the parser is intended to be the input to some higher level semantic interpretation. In the latter case, if our parser is intended to be unrestricted, then an important issue is the development of the related semantic model and successfully relating it to the syntactic structures. Much of the work in parsing and interpretation using systemic functional grammar has focused on the difficult problem of reversing the process of system network traversal and realisation rule application, as discussed in section 2.2.1. It would be preferable for the association between syntactic structure and semantic selection expression to be able to be derived automatically in some way, perhaps using the generator output (which can be adjusted to include the semantic features chosen in

[6] CELEX has since been further extended, but for our purposes it was a complete and deliverable system at the time we needed it.

generation). This kind of approach will only yield semantic analyses for sentences an SFG NL generator can generate, however. In other grammatical implementations which include a semantic model, such as the Alvey NL Toolkit (described in section 2.4.1), or the Core Language Engine (Alshawi 1992), the semantic associations with the syntax have to be hand-crafted. At present, the only possibility for automated semantic analysis appears to be the harnessing of resources from dictionaries and encyclopaedias to semi-automatically develop semantic taxonomies and networks of semantic relations (such as those produced by the ACQUILEX (Meijs 1993a, 1993b) and Wordnet (Princeton University) projects). In principle, such knowledge could automatically be acquired from suitably semantically and syntactically annotated corpora, but these have yet to be created[7]. The lack of agreement as to what constitutes a correct parse is replicated in the world of semantics, where possible semantic models include Montague grammar, type theory and lambda calculus, models of temporal and spatial relations, SFG system networks, preferences and selection restrictions, or even SQL expressions. Hence we restrict ourselves in this thesis to the already difficult problem of syntactic analysis of unrestricted natural language, without concerning ourselves with the issue of formalising and learning or hand-crafting an equally large semantic model.

When we consider the different techniques used in syntactic parsing, one key distinction can be drawn, which is to separate techniques which rely on deterministic search (which have been used for finite search spaces of up to around a thousand grammar rules), from techniques for probabilistic search (which presume an almost infinite search space).

The former could be called deterministic or rule-based approaches, although strictly speaking they do not all employ the formalism of a set of rules, and have been used traditionally with competence grammars. The term determinism will not be used because of its ambiguity in artificial intelligence circles between an approach which, at every point of choice in the parsing process, successfully determines the right direction to take without recourse to back-tracking (Marcus 1980), and a more liberal interpretation in which a system is guaranteed to find at least one solution (where one or more exist in the grammar) without using probabilities.

---

[7] One exception is Bod (1995), who has attempted to create a semantic performance (Montague) grammar.

Probabilistic techniques have been adopted in response to the presumed open-endedness of grammars used for the analysis of corpus texts. Such techniques usually aim to find a best-fit solution, even for semi-grammatical sentences. Probabilistic parsers may use rules augmented with probabilities, but in this thesis I reserve the term rule-based parsing for rules devoid of probabilities or weights.

Karlsson (1995; 9) expresses a similar distinction in his excellent review of six possible approaches with respect to grammar-based versus probabilistic models, recognising that researchers do not always pick one or the other, but often use a hybrid approach:

1. grammar-based rules only, e.g. Alvey GPSG, Fidditch, TOSCA, The Core Language;
2. probabilistic modules only (PARTS part of speech tagger, CLAWS1 part of speech tagging + UCREL syntax);
3. grammar-based rules strictly followed by probabilistic modules;
4. probabilistic modules strictly followed by grammar-based rules (the combination of PARTS and Fidditch: de Marcken's (1990) model for part of speech disambiguation and syntactic analysis trained on the LOB corpus);
5. grammar-based rules interleaved with probabilistic modules; grammar rules interleaved with heuristic metrics for solving ambiguities, followed by a fitting procedure for handling parsing failure: EPISTLE (Heidorn 1982; Jensen and Heidorn 1983; also cf. the papers in Jensen, Heidorn and Richardson 1993); McCord's (1990) slot grammar; skeleton parsing (Black, Garside and Leech 1993); realistic annealing parsing (Souter and O'Donoghue 1991); unification-based grammar rules supplanted with statistical information drawn from pre-tagged corpora (Briscoe and Carroll 1991); optimal linguistic constraints eventually followed by more heuristic constraints if the optimal constraints fail, then reapplication of the optimal constraints: Constraint Grammar;
6. probabilistic modules interleaved with grammar-based rules

It would be impractical to review here in any great detail all the different approaches which have been proposed for parsing natural language, so instead I will summarise rule-based parsing and probabilistic alternatives, including some hybrid approaches, commenting on the pros and cons of each.

2.4.1 Rule-Based Parsing.

By rule-based parsing, I mean parsing which proceeds by searching through the set of rules in a grammar (and lexicon) to determine which rules may jointly be applied to produce a well formed

syntactic structure for a sentence of the language described in the grammar. If no analysis for a sentence can be found using the rules in the grammar, then the sentence is ungrammatical. If more than one analysis is found, then the sentence is syntactically ambiguous.

Finite state or context-free grammars are simple formalisms adequate for many of the structures of natural language . Such grammars are usually implemented as recursive transition networks (RTNs), or sets of phrase-structure rules. Parsing then involves traversing the network, recording the structures built as each arc is crossed, or for phrase-structure rules, combining them recursively to build a nested list or tree structure.

However rule-based grammars are limited in their ability to adequately capture unbounded dependencies in English such as topicalisation and wh-question formation, unless the grammars have been supplemented with features, feature constraints, and metarules. Associated parsers must then have recourse to unification of these features.

Two commonly used rule-based parsing algorithms for context-free grammars are shift-reduce parsing and chart parsing. These differ only in the efficiency with which analyses are found, and not in the analyses that are found.

2.4.1.1 Shift-Reduce Parsing.

Shift-reduce parsing is a relatively simple parsing algorithm for context-free phrase-structure grammars, and is widely described in the literature; for an introduction, see (Winograd 1983: 87-111). Two operations are applied successively until the input sentence is exhausted: The leftmost word in the sentence is *shifted* onto a stack, and then a rule in the grammar is chosen whose right hand side matches the contents of the stack, in order to *reduce* the stack to the category label on the rule's left hand side. The POPLOG programming environment contains a built-in PROLOG parser of this kind. Atwell and Souter (1988) and Atwell et al (1988) describe experiments with this parser using grammars extracted from the Lancaster/Leeds Treebank and POW corpora. In both cases, grammars of several thousand rules are unable to be fully loaded because of system limitations, restricting grammar size to around a thousand rules. An alternative rule-based technique which is more efficient, but slightly more complex is chart parsing.

2.4.1.2 Chart Parsing.

One of the problems with shift-reduce parsing algorithms is the choice of when to shift and when to reduce. Much inefficiency results from repeated backtracking to find successful combinations of the two operations and the right rules, and the matter is made worse by the ambiguity common to all natural languages. This means that the same structures are constantly being deleted and then rebuilt in shift-reduce parsing, and the inefficiency is scaled up considerably with very large grammars.

One way of overcoming this inefficiency is to keep a record of all legal sub-structures built, so they never need to be reconstructed. This can be done in a well-formed substring table or *chart*, hence the name chart parser. In a chart parser (Winograd 1983: 116-127, Gazdar and Mellish 1989; 181-213), words in a sentence are combined as *edges*, with a record kept of i) the syntactic category of the edge, ii) its start and end point, iii) its contents, and iii) any other edges needed to make a full constituent. The syntactic category of the edge is the mother of a phrase-structure rule, and the contents together with the edges needed are the daughters in a phrase-structure rule. The daughters may be separated by a full stop to indicate which have been found and which have yet to be found, in which case the edge represents a *dotted rule*. An edge which has found all the daughters it needs to be a full constituent is called *inactive*, whereas an edge which is still looking for further daughters is called *active*. Edges are recursively combined until all the words have been included and no further edge combinations are possible, using a procedure called the *fundamental rule of chart parsing*. Informally, this means combining an inactive edge with an active one which finishes at the inactive edge's starting point, and which is looking for a daughter category which matches the category of the inactive edge. This combination rule is described formally by Gazdar and Mellish (1989; 197):

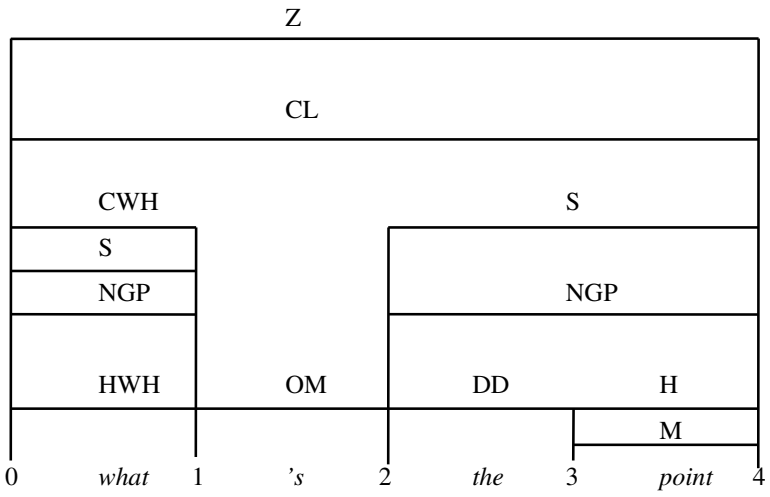
> If the chart contains edges *[i,j,A->W1.B W2]* and *[j,k,B->W3],* where *A* and *B* are categories and *W1, W2* and *W3* are (possibly empty) sequences of categories or words, then add edge *[i,k,A->W1 B.W2]* to the chart.

In practice, such combined edges are not added directly to the chart, but put on an agenda of edges to be added to the chart. When all edges have been taken off the agenda, and no more combinations are possible, then parsing is finished. The set (possibly empty) of edges labelled with the root or sentence category label which contain all the words in the sentence is then

considered to be the list of grammatical analyses. The order in which edges are combined (and hence solutions produced) can be manipulated to perform depth or breadth-first search, by the ordering the agenda, but the same list of solutions will always be found, for a static lexicon and syntactic model. Since modified chart parsing will be the approach employed in the present parser, I will illustrate in Figure 8 the way a chart parser constructs an analysis for the sentence *what's the point*, using a toy lexicon and SF syntax.

Figure 8. Building a Chart.

Mini-lexicon		**Mini-grammar**	
what	HWH	Z ⌖ CL	
's	OM	CL ⌖ S OM C	CL ⌖ CWH OM S
the	DD	S ⌖ NGP	CWH ⌖ NGP
point	H, M	NGP ⌖ HWH	NGP ⌖ DD H



In Figure 8, only some of the inactive edges are shown, and all the active edges are omitted, for ease of reading. The example is somewhat contrived, since some rules have also been omitted for the sake of clarity. In a bottom-up left-to-right chart parser, the chart in Figure 8 would be constructed as follows: The first word *what* would be looked up in the lexicon, and found to have a possible label *HWH*. An edge would be created in the chart spanning just that word (from position 0 to 1), with that label. In a breadth first approach, the second word would then be looked up. In a depth first approach, we would next look in the grammar to see if any rules exist with *HWH* as their first daughter, and find the rule *NGP* ⌖ *HWH*. This would result in the addition of an active edge labelled *NGP* seeking an inactive edge labelled *HWH* being added to

the chart. The fundamental rule could then apply, combining the active and inactive edge to produce a new inactive edge labelled *NGP*. The grammar would then be consulted again, looking for rules with NGP as the first daughter, and two would be found, *S* ⌖ *NGP* and *CWH* ⌖ *NGP*. Two new active edges would be created in the chart, which would then each combine with the inactive *NGP* edge using the fundamental rule. Next the grammar would be consulted again, looking for rules with either *S* or *CWH* as first daughter, and the two clause rules *CL* ⌖ *S OM C* and *CL* ⌖ *CWH OM* S would be found and added to the chart as active edges. Each will combine with the *S* and *CWH* edge respectively. At this point, no new rules can be applied to inactive edges for the first word, so the second word is looked-up in the lexicon. and the edge building process continues, finding an inactive *OM* edge for the enclitic verb *'s*. The *OM* edge can combine with both of clause rules, leaving them both active, seeking one more daughter. The third word *the* is then found to be labelled as a deictic determiner, *DD*, and a new active *NGP* edge is created which combines with the *DD* edge. Finally, the last word, *point*, is looked up and found to have two possible tags, *H* and *M*, which results in two lexical edges being created. Only the first of these can combine with the neighbouring active *NGP* edge seeking an *H*, to produce an inactive *NGP* edge. Again the grammar is consulted, and two new active edges labelled *S* and *CWH* are created, which then combine with the inactive *NGP* edge to span the last two words of the sentence. Of the two resulting inactive edges, only one can combine with an active clause rule, which creates a spanning inactive edge labelled *CL*. The grammar is then consulted one more time, to find rules with *CL* as first daughter, and the *Z* ⌖ *CL* rule matches, resulting in the creation of an inactive edge labelled *Z* which spans the sentence. These conditions are those required for a solution parse tree (the *Z* label being that of the root or target), and the structure contained inside the *Z* edge is then recursively built and added to the list of solutions. The parser continues looking for further edges until no more combinations can be made using the lexicon and grammar, at which point the list of solutions is returned.

Variations on this algorithm have been used by many designers of parsers for small and large grammars alike, with enhancements to improve performance on large-scale lexicons and grammars. One of the best known of these is the Alvey Natural Language Toolkit, containing a chart parser written in Common LISP (Phillips 1986, Phillips and Thompson 1987) which was intended to serve as a general-purpose tool for the natural language community. As explained in section 2.1.1, the grammar used roughly follows the Generalised Phrase Structure Grammar formalism, including categories as sets of features and metarules (Gazdar et al. 1985). The

expanded object grammar contains well over 1,000 rules, which is very large, for a hand-built competence grammar. Experiments have shown the accuracy of the parser to be very good (Atwell et al 1988, chapter 4), but the process of loading and running to be very slow. The Toolkit's authors recommend that a dedicated machine of at least 16 Mb. working memory be used. Were the grammar used to be one of several thousand rules extracted from a parsed corpus, the running performance would be expected to deteriorate still further, (assuming, of course, that it is possible to integrate other grammar descriptions and formalisms such as SFG into the framework of the Toolkit). One poor aspect of rule-based parsing which is difficult to overcome is its total failure in cases where the structure of the sentence cannot be described using the existing set of rules. Most rule-based parsers (including the chart parser of the Alvey NL Toolkit) simply give a null response when their grammar (or lexicon) is inadequate, not even offering a partial, best-fit solution. This problem of undergeneration can be minimised by using a much larger grammar (such as one derived from a parsed corpus), but even then it is theoretically possible that the parser will fail to produce a solution. In this case the contents of the chart could be use to offer some kind of ad hoc solution, or parsing could restart with the grammar relaxed in some way (Kwasny and Sondheimer 1981).

However, with very large grammars contained in rule-based parsers, a further problem tends to occur (see, for example, Oostdijk 1991, Keenan 1993); tens or even hundreds of legitimate analyses are produced by the parser. Briscoe (1994; 99) gives an extreme example where the definition of *youth hostel* is parsed using the ANLT to produce a *parse forest* of over 2500 different analyses. Clearly it would be desirable if these were to be ordered according to some measure of likelihood, or even by having a semantic component interact with the parser. The latter suggestion is preferred by some theoretical linguists including the developers of the ANLT, but has proved difficult to implement, not least because of the lack of consensus as to how to formalise an adequate semantic model for natural language. But some researchers have introduced corpus-based probabilities into the chart in order to influence the parser's search strategy, thereby classifying them as hybrid rather than purely rule-based techniques (Magerman and Marcus 1991, Briscoe and Waegner 1992, Pocock and Atwell 1993).

2.4.1.3 Probabilistic Chart Parsing.

Although approaches to probabilistic parsing are discussed in section 2.4.2, probabilistic adaptations to chart parsers will be dealt with here, since they represent a modification to the search strategy of the standard chart parsing algorithm. The lexicon and grammar used by a probabilistic chart parser are augmented with probabilities extracted from a parsed corpus. Samples of the probabilistic lexicons and context-free grammars which can be extracted from the POW corpus are given in Figures 9 and 10. The leftmost column contains the observed corpus frequency of the word-wordtag pair (Figure 9) or rule (Figure 10).

Figure 9. The 20 Most Frequent Word-Wordtag Pairs in the POW Corpus.

2679	I	HP	691	GOT	M
2250	THE	DD	610	THEY	HP
1901	A	DQ	585	NO	F
1550	AND	&	554	IN	P
1525	IT	HP	523	TO	I
1298	YOU	HP	482	PUT	M
1173	'S	OM	417	HE	HP
1117	WE	HP	411	DON'T	ON
1020	THAT	DD	401	ONE	HP
897	YEAH	F	400	OF	VO

Figure 10. The 20 Most Frequent Context-Free Rules in the POW Corpus.

8882	S --> NGP	1738	NGP --> DQ H
8792	NGP --> HP	1526	NGP --> H
8251	Z --> CL	1496	C --> PGP
6698	C --> NGP	1272	C --> QQGP
4443	QQGP --> AX	1234	CM --> QQGP
2491	PGP --> P CV	1221	NGP --> DD
2487	CV --> NGP	1215	NGP --> HN
2283	NGP --> DD H	1182	C --> CL
2272	CL --> F	1011	MO --> QQGP
1910	Z --> CL CL	1004	CL --> C

As each edge for a new word or rule is added to the chart, the probability of the word or rule is also added to the edge. When edges are combined, their probabilities are combined, usually being multiplied together, since the over-arching edge represents the occurrence of the active edge and the inactive edge. The recursive combination of edges results in a global probability for each tree. The parser's search strategy is “most-likely first”, since as the combined edges are added to the agenda, the agenda is reordered using the probability of the edges. When the next edge is

added to the chart, it will have the highest probability of those remaining on the agenda. When all solutions have been found, they can be ordered according to their probabilities, or alternatively, they can be produced on the fly while the parser is still running.

Pocock and Atwell took Gazdar and Mellish's (1989; 429-431) POP11 implementation of a chart parser and made the changes described above so that the search strategy would be probabilistic. They enhanced the program in a number of other ways to improve the efficiency of the parser, since they intended to use the parser with a grammar extracted from the Spoken English corpus (SEC: Knowles and Lawrence 1987). The lexicon and grammar were separated, and stored as property lists, rather than flat lists. Probabilities were logarithmised and normalised to integer form to allow for fast integer arithmetic. The chart was stored as an array of lists, so only a subpart need be consulted, and a number of methods were employed to reduce garbage collection when parsing. The SEC grammar consisted of 15 non-terminal and 187 terminal categories, and several thousand context-free phrase-structure rules. Their findings with the complete grammar were not promising; performance was very slow, so to achieve adequate performance for their application (disambiguating speech recognition word lattices) they had to select a subset of the grammar. With a large grammar, much of the chart parser's time is spent searching through lists (the agenda and the chart), which are particularly slow to process in POP11.

Magerman and Marcus achieved more satisfactory results with their *Perl* parser by making the probability combination function conditional and adding context-sensitive constraints (having recourse to the parent of the mother category of a rule). They achieved a success rate of 35 out of 40 successful parses (87.5%) on test sentences from the *Voyager* domain. However, the corpus material used to derive the grammar was artificially produced by a natural language generator, and then this grammar was manually augmented with syntactic constraints. While these results are extremely good, it is desirable instead to be able to use a grammar extracted totally automatically from a non-artificial parsed corpus.

Further experiments with probabilistic chart parsers have been conducted with context-free grammars (both with and without features and unification) taken from the ANLT by Briscoe (1994). His work attempts to solve the problem of undergeneration of competence grammars by re-estimating the grammar probabilities using the Baum-Welch algorithm (Baum 1972). Briscoe uses this technique to optimise the probabilities on the existing rules (which he calls explicit),

and to select good ones from a set of implicit potential rules. When applied using a chart parser on sentences taken from the SEC and Associated Press corpora, he improves the success rate of the untrained grammar (50%) to between 63-66% with the re-estimated grammar (depending on whether unification is used). The number of sentences parsed (even if the correct parse was not the most likely) rose from 53% to 96% using re-estimation of a grammar containing explicit and implicit rules. This method of expanding an incomplete competence grammar looks to be very promising, and could also be used to optimise and expand a corpus-based probabilistic context-free grammar.

2.4.2 Probabilistic Parsing.

Instead of trying to improvise with a number of ad hoc 'failure' rules, or to improve the search strategy of a rule-based approach, a number of researchers have developed novel probabilistic parsing techniques, which use a stochastic model based on corpus frequencies, and abandon the phrase-structure rule-based formalism altogether.

Much of the thinking behind probabilistic parsing originates from the observation of corpora which have been hand parsed. These display the sort of grammatical open-endedness discussed in section 2.2.3 (Sampson 1987b, Garside et al 1987). In many probabilistic approaches, the strict grammatical/ungrammatical distinction is foregone, and replaced with a scale of observed frequency. Frequently occurring structures are hopefully those which are acceptable to most speakers of the language, while less frequent structures may be uncontroversial and genuinely rare, or may be semi-grammatical and acceptable only to some speakers. Structures which are never attested in a large, representative corpus (and representative here is quite difficult to specify) are considered to be "ungrammatical" or not to be part of the language. Alternatively, these very rare structures may be given a very low default probability, such that they are recognised by the parser, but only after the structures found in the corpus.

The parsing process then becomes a matter of stochastic optimisation; a search for the most frequently observed structure, given some probabilistic model of the language concerned. Consequently probabilistic parsing may be divided into the choice of a probabilistic language model, and a suitable optimal search technique.

2.4.2.1 Probabilistic Language Models.

The simplest models are for bigrams or trigrams; describing the co-occurrence of two or three-word strings, and avoiding any explicit representation of higher level syntactic structure. The advantage of such models is that only a raw, unanalysed corpus is needed, so the data from very large corpora can be easily assimilated. Such co-occurrences are used by Sharman (1990) to train a Hidden Markov Model (HMM) based word tagger for English, the accuracy of which increases with the length of the polygrams. However, the complexity of the process of training and running the tagger also increases with the length of the word strings used. Using the same approach in parsing might seem desirable, but it is unclear how constituents would be formed, and even less clear whether they would match linguists intuitions of how constituents should be labelled. For promising experiments in this direction though, see (Hughes 1994, Jost and Atwell 1994, Jost 1994).

Other models incorporate some type of grammatical description, whether only terminal categories (word tags) for the words in the sentence (as used in the CLAWS project described in section 2.3.1), or full syntactic trees. The raw material for such models is either a tagged, or a fully analysed corpus. APRIL (Sampson et al 1989) and COMMUNAL (Phase 1) (Atwell et al 1988) were successive projects at Leeds University which used first-order Markov models represented as recursive transition networks (RTNs) extracted from fully hand-analysed English corpora (the LOB Treebank and POW Corpus, respectively). An example of the probabilistic RTN formalism extracted from the edited POW corpus is shown in Figure 11.

Figure 11. A Fragment of Probabilistic RTN from EPOW.

A	#	CL	250
A	#	NGP	156
A	#	PGP	970
A	#	QQGP	869
A	#	TEXT	1
A	CL	$	250
A	CL	CL	6
A	NGP	$	156
A	PGP	$	970
A	PGP	PGP	2
A	QQGP	$	869
A	QQGP	QQGP	4
A	TEXT	$	1

This RTN fragment contains 4 columns. The first is the mother in the tree, in this example *A* (= Adjunct). The second and third are possible ordered daughters of the mother (including the start symbol (#) and the end symbol ($)). The fourth column contains the frequency of the combination of the pair of daughters for a particular mother.

The RTN shown above can be used straightforwardly in hybrid parsing by attempting to pass through the network for a given set of words (and tags). Where a choice of possible arc traversals presents itself, the probabilities can be used to bias the choice (rather like the re-ordering of the agenda in chart parsing). However, in the APRIL and COMMUNAL projects the networks were instead used purely to evaluate the likelihood of subtrees found by another parsing technique (simulated annealing). The crucial issue here is extracting the grammar from the parsed corpus in a form which captures the full complexity of natural language, but which is simple enough to be integrated with the chosen search technique. Experiments with an implementation of a simulated annealing parser (see section 2.4.2.2, below) suggest that a higher order Markov model will be needed to handle some long-distance dependencies which occur in wh-questions and topicalisation, for example.

2.4.2.2 Search Techniques.

One of the advantages of probabilistic parsing is its robustness; a solution is always produced, even for input which might be considered syntactically deviant in some way. However, the price for such robustness is paid in the search time, which is enormous for all the permutations of grammatical labels and potential tree structures. Fortunately, there are established techniques for searching large solution spaces relatively efficiently, such as simulated annealing (van Laarhoven and Aarts 1987, Kirkpatrick et al 1983), or biologically inspired algorithms (Kempen and Vosse 1988).

Simulated annealing works by starting with any possible solution parse tree (eg the flat tree consisting only of the root label and the word tags) and proposing random changes in tree structure and labelling. At each change, the new tree is evaluated against a probabilistic language model, and as a general trend, if the change is an improvement, it is accepted. Initially in the annealing run, worsening changes are also accepted so that the global, rather than just the local, optimal solution can be found. As annealing progresses, fewer and fewer worsening changes are

accepted, until only moves which improve the value of the tree are allowed. Both the APRIL and COMMUNAL projects employed simulated annealing, with potential solution trees being evaluated against corpus-based probabilistic RTNs (Sampson et al 1989, Atwell et al 1988). The Realistic Annealing Parser (RAP), which was developed by Tim O'Donoghue under the auspices of the COMMUNAL project, attempted to improve on the total randomness used in APRIL by the introduction of left-to-right parsing, and careful use of probability density functions to influence the place and choice of changes to the tree structure (Souter and O'Donoghue 1991). It finds just one (hopefully optimal) solution for each sentence, although there is no guarantee that simulated annealing will produce a global optimum. Equally unusual compared to rule-based parsing is the fact that it can find different solutions on different attempts to parse the same sentence. One would hope that this would only happen if the values of the two parses are extremely close, representing more or less equally weighted ambiguous readings. An example of this produced by the RAP when parsing the sentence *fight on to save art* is given in Figure 12. The overall likelihood of both analyses is, in this case, very similar: -18.1732 and -18.6086 are the logarithms of the total probability of each tree, calculated by multiplying the individual probabilities for each branch.

Figure 12. Ambiguous Analyses of the Sentence *Fight on to save art* .

```
<push Z> 1.0
    <push CL> 0.972759
      <cat M> 0.079384 "FIGHT"
      <push C> 0.73187
        <push PGP> 0.13343
          <cat P> 0.866249 "ON"
        <pop PGP> 0.014601
      <pop C> 0.992598
    <pop CL> 0.688811
    <push CL> 0.227848
      <cat I> 0.025948 "TO"
      <cat M> 0.962891 "SAVE"
      <push C> 0.73187
        <push NGP> 0.620802
          <cat H> 0.07593 "ART"
        <pop NGP> 0.936869
      <pop C> 0.957554
    <pop CL> 0.688811
  <pop Z> 0.758537
Total: -18.1732
```

```
<push Z> 1.0
  <push CL> 0.972759
    <push S> 0.315439
      <push NGP> 0.98825
        <cat H> 0.07593 "FIGHT"
        <push Q> 0.058878
          <push QQGP> 0.083591
            <cat AX> 0.827548 "ON"
          <pop QQGP> 0.965217
        <pop Q> 1.0
      <pop NGP> 0.978261
    <pop S> 0.996161
    <cat I> 0.002828 "TO"
    <cat M> 0.962891 "SAVE"
    <push C> 0.73187
      <push NGP> 0.620802
        <cat H> 0.07593 "ART"
      <pop NGP> 0.936869
    <pop C> 0.957554
  <pop CL> 0.688811
<pop Z> 0.758537
Total: -18.6086
```

Extensive testing on unseen corpora was never carried out on the RAP, due to its lexical limitations, and the fact that optimal settings for the array of parameters involved in annealing

were never arrived at. (At the time it was suggested a second annealing process might be necessary just to discover the optimal parameter settings for the first!). O'Donoghue abandoned the simulated annealing approach because of the speed and unreliability of its performance, and its inability to capture unbounded dependencies, in favour of his vertical strip parser (described in section 2.2.2). Nevertheless it represents a good example of the extent to which a probabilistic RTN can be stretched to handle even semi-grammatical input.

Two more promising examples of probabilistic techniques are those of Magerman (1994) and Bod (1993, 95). Instead of using limited Markov models for evaluating potential parse trees, they use the frequencies of sub-trees found in a fully analysed training corpus. Magerman assumes first that any tree matching the sentence length is a potential solution, so he generates all possible (right-branching) trees for the sentence. All these potential analyses are then evaluated using corpus-based probabilities, pruning out the less likely ones, until one or more optimal solutions are reached. Bod's approach is similar, utilising a corpus of tree structures, together with a set of operations that combine corpus subtrees into new trees (not restricted to be in Chomsky Normal Form), and an associated evaluation function. The input to his parser is a sequence of part of speech labels, which automatically reduces the parsing/ambiguity problem. Nevertheless, he reports accuracy rates of between 87 and 96% on tests from the ATIS corpus. He even extends his approach to propose a method for semantic analysis using a performance model (Bod 1995; 99-115). Both Magerman and Bod use the Penn Treebank as their corpus source, which is about the coarsest of current syntactic annotation schemes, and includes a limited set of formal labels, with no functional labels.

2.4.3 Selection of a Parsing Technique.

The vast array of attempts to find a successful technique for parsing natural language (let alone unrestricted language) make it somewhat difficult to select one to pursue with a view to providing a parser for systemic functional syntax found in the POW corpus. However it does testify to the fact that the problem is not yet solved. One of the stumbling blocks for NLP is the lack of any standard metric for judging the success of a parser, although two methods are described in (Sampson et al 1989) and (Black et al 1991).

The present project will experiment with two lexical tagging resources, and two variants on a chart parsing algorithm. One will employ a probabilistic context-free phrase-structure formalism for SFG extracted automatically from the EPOW corpus, and integrate this with the CELEX-SFG lexicon into a probabilistic chart parser. The grammar will be modified by collapsing simple phrase-structure rules into a smaller set of rules which allow optional categories. At the same time, the grammar will be implicitly expanded by providing a mechanism for co-ordination beyond that which is observed in the corpus. It is hoped that by so reducing the grammar, and modifying the chart parsing algorithm to allow for optionality and coordination, the slow performance of the parser described in (Pocock and Atwell 1993) can be improved.

In a second experiment, the lexical look-up facility will be provided by the POW-trained Brill tagger, and the syntactic model will be context-sensitive, consisting of a combined context-free rule and vertical trigram model. The second experiment will also limit vertical depth of the edges proposed in the chart, to constrain the number of hypotheses the syntactic model licenses.

This method will allow the parser to be adopted for other parsed corpora and their descriptions, since most of these readily permit the extraction of a probabilistic context-free grammar and vertical trigram model (should it always be needed). The main effort in changing grammar descriptions would then be in adapting the CELEX lexicon tags to that description, or retraining the Brill tagger on the new tagged corpus.

The process of extracting and transforming the syntax model from the corpus, and developing the lexical resources will be described in chapter three. Chapter four will explain the parsing technique in detail.

Chapter 3. Developing the Resources for Parsing.

This chapter will be divided into three sections: The first section will describe the extraction and transformation of the syntactic model from the Polytechnic of Wales (POW), and the edited POW (or EPOW) corpus. The second will explain how the CELEX English database was used to provide a probabilistic lexicon whose grammatical tags were transformed to match those the POW corpus systemic functional syntax. The third briefly describes how the Brill tagger was trained on the POW corpus.

3.1 Developing a Probabilistic Systemic Functional Syntax.

Atwell (1988), Atwell and Souter (1988a), Souter (1990) and Souter and Atwell (1992) all describe methods for extracting simple context-free rules (among other formalisms) from parsed corpora, such as the Lancaster/Leeds treebank and the POW corpus. In the earliest of these experiments, rules were extracted in a PROLOG definite clause grammar (DCG) form without probabilities, and attempts were made (unsuccessfully) to load these rules into PROLOG and run them as a parser using a parser-generator facility. System restrictions prevented the entire grammar from being loaded (POPLOG PROLOG's memory was exhausted).

In further experiments, the rules were extracted from the POW corpus in a simple context-free formalism (see Figure 10) and lexical ‘rewrite’ rules between words and tags were extracted separately as a lexicon (see Figure 9).

For the POW corpus rules to be extracted, it was first necessary to write a procedure which could handle the POW corpus numerical tree format, effectively treating them as bracketed lists. The extraction algorithm itself is then relatively straightforward; taking the corpus one tree at a time, rewriting all the mother-daughter relationships as phrase structure rules, and deleting all the duplicates after the whole corpus has been processed. A count is kept on how many times each rule occurred. The method is described more fully in (Atwell and Souter 1988a). The lexical items are extracted separately into a wordlist with the elements of structure they expound, and again a frequency count is kept (see section 3.2). The most recent code for the extraction process

was written in POP11 by Tim O'Donoghue, for use with the EPOW corpus. As well as a stochastic context-free grammar, O'Donoghue extracted a stochastic finite state automaton, and his vertical strip grammar. I have since supplemented these with a further POP11 program for extracting vertical trigrams from the EPOW corpus. I first explain some important policy decisions and experiments which Eric Atwell and I conducted before O'Donoghue's work. Nevertheless, I will later adopt (in section 3.1.4) O'Donoghue's context-free syntax as the starting point for the modifications which I refer to as grammar expansion and collapsing below.

Initially Atwell and I experimented with two alternative methods for rule extraction which differ regarding their treatment of *filling*. Filling is the relationship between the elements of structure and units (functional and formal categories) in the grammar. Normally in hand-drawn parse trees and those generated by the GENESYS NL generator, functional and formal categories are conflated together *within one node* of a tree, such as *S_ngp* (S = Subject, ngp = nominal group). A separate relationship exists between the formal categories in the grammar (units) and their daughters (elements of structure), called *componence*. Componence is the same relationship as that in standard rewrite rules such as

ngp -> dd h

(*dd* = deictic determiner, *h* = head) where different nodes on the same level are horizontally linked together (Fawcett 1981; 6-8). Our first extraction method was to maintain Fawcett's distinction between filling and componence relationships and resulted in just over 8,500 distinct rules, a sample of which (in PROLOG DCG form) is included in Appendix 5. The second treats filling as componence (S -> ngp etc.), which still yields over 4,500 distinct rules, a sample of which is given in Appendix 6. The second strategy is both more economical and intuitively more sensible, since the internal structure of a nominal group would appear to be the same whether it fills a subject or a complement, and filling must allow for branching when, for example, nominal groups are co-ordinated under one subject anyway. However, this simplification will fail to capture the difference between main clauses and relative clauses. Both will be simply represented by the symbol *cl*, with no reference to the fact that for the former the mother will be the top symbol of the grammar, *Z* (= sentence), and for the latter the mother will be the qualifier in a nominal group. Failing to capture such information will lead to overgeneration in the parser, since it will permit a relative clause structure to be found in main clause position, for example, and vice-versa. However, this loss of information was considered preferable to effectively having to store the same componence rules many times over in the context-free grammar, once without

any conflated element of structure, and then again for each different element of structure the mother of the componence rule can fill.

3.1.1 Inconsistencies in the Corpus.

Inevitably, Atwell and I identified some errors in the processing of the POW trees to bracketed form. Empty brackets occur when the sole content of a tree is a 'non-verbal' string enclosed in square brackets. Empty brackets are also found deep in a tree if there is some typing error in the hand parsing which results in a subtree which does not conform to the general pattern of alternation between functional and formal labels (elements of structure and units). Such erroneous trees are filtered out using a bracket-checking program. This practice has enabled the processing of the POW corpus without having to do any tedious editing of the original text. On a subsection of the corpus, it was calculated that about one in every thirty three trees was ignored in this way. Some minor adjustments were required to the extraction program to allow 'words' of extraordinary length, which exceeded the programmer's anticipation of 30 characters as a maximum:

THE-WILLIE-WONKA-AND-THE-CHOCOLATE-FACTORY
[VROOM-BROOM-VROOM-BROOM-VROOM-BROOM-VROOM-BROOM...]

Without manual intervention it is impossible to filter out all the errant data from the POW corpus, so some typographical errors remain in the wordlist, and in some of the rules, where the conventions for drawing the numerical trees were not strictly adhered to. Obviously, extracting trees from the output of a NL generator such as GENESYS in the form of an Ark corpus (Souter 1990, O'Donoghue 1990) presents far less of a problem from the point of view of such 'noise'.

3.1.2 Distribution and Frequency of the Rules.

Of the total of 8,522 unique rules extracted from the POW corpus using the method which treats formal and functional categories as part of the same node, (*S_ngp, Z_cl* etc.) some occur very frequently and some only once. Inevitably, some rules have been extracted from malformed or inconsistent input, but these will be relatively rare. Other rules will be problematic because they contain portmanteau categories. For instance, when the hand-parsing was being undertaken, if the

annotator could not decide between two labels, they would both be included (eg: *S/C?*) or even a single label could be uncertain (eg: *NGP?*). Some *ad hoc* procedures can be introduced into the grammar extraction to remove such problems. Simply ignoring that tree would lose valuable information contained in the well-formed remainder of the tree. The alternative which was finally adopted was to remove all such question marks and in the case of portmanteau-style tags, use the first of two (or more) options.

Another alternative which at first appeared attractive was to threshold out the rare rules, in the hope that this would remove all such unusual category labels, and allow a core grammar to be defined using only the fairly frequently occurring rules. Such a frequency threshold by which a rule is accepted or not would have to be carefully established, but initial examination of the rare rules was not encouraging. More than 6,000 rules of the 8,522 total occur only once. Two-thirds of the 4,647 unique rules yielded by the second method of extraction (S -> ngp etc.) were also singletons. Many of these singletons are not produced by errant input, but are genuine examples of rare structures in (spoken) English, (Actually some would be considered to be not so rare, if we trusted our native-speaker intuition!). Other rare rules result from the elaborate category labelling for a particular construct, or the flat shape of the tree immediately below the clause level. Consider the following four examples, in which the frequency count (1) is on the far left of each rule, followed by a reference to the sentence in the original corpus, and then the actual rewrite rule. Below the rule in each example appears the section of the corpus from which the rule was extracted, and below that, the sentence without its analysis.

Figure 13. Rules which occur only once (6047 out of 8522).

a) An unexpectedly rare tag question structure: *isn't there*

1 /*10DGPSSM55*/ ATG_CL --> OXN STH

55 Z 1 CL 2 S NGP HP I 2 M KNOW 1 CL 3 STH THERE? 3 OM 'S? 3 C NGP 4 H? ? 4 Q CL 5 M BUILDING? 5 C NGP 6 DD THE? 6 H HOUSE [UN:?] 3 ATG CL 7 OXN ISN'T 7 STH THERE [NV:EM]

(I know, there's ? building? the house, isn't there)

b) A genuinely rare structure; interrupted speech: *There's a....*

1 /*10DGPSSS27*/ Z_CL --> STH OM C_NGPUN V_NGP

27 Z CL 1 STH THERE 1 OM 'S 1 C NGPUN DQ A [UN:THERE-LIKE?] 1 V NGP HN DAWN

(There's a ... Dawn!)

c) Rare structure resulting from the fine distinctions made in adverbials, here, an affective adverbial (AA): *preferably*

1 /*12BBPSMB318*/ AA_QQGP --> AX

316 Z CL CWH NGP 1 DT NGP 2 DDWH WHAT 2 H SORT 1 VO OF 1 MOTH NGP H THREE 1 H BRICK 318 Z CL 1 C? NGP 2 MO QQGP AX WHITE 2 (H) 1 AA QQGP AX PREFERABLY

(What sort of three brick? White, preferably.)

d) Rare because of non-standard hand-parsing.

The following is extracted as a terminal category in the grammar, i.e. a word class instead of a word itself.

1 /*10BBPSMJ299*/ ABOUT

299 [FS:A...] Z CL ? PGPUN P ABOUT

(About...)

I plotted the distribution of frequent versus rare rules on a scatter diagram, with frequency of a rule on one axis, and frequency of that frequency on the other axis. It is interesting to note that the distribution for rules is very similar to that for word frequency. The word and rule graphs have been plotted on logarithmic scales for comparison in Appendix 8. The characteristic features of both plots are that only a few words/rules occur very frequently, and very many occur only once or twice, a distribution known as Zipf's law (Zipf 1936).

3.1.3 Coverage of the Rules.

As a bench mark against which to evaluate these rulesets, I hand-wrote a formal competence grammar based on the structures given in (Fawcett 1981) using context-free rules which allowed repeated and optional daughters (see Appendix 9). When this grammar was expanded into simple context-free rules, (with a limit of three put on possible co-ordinations, which is by no means

generous with respect to the co-ordination of clauses, for example), over 18,000 distinct rules were produced. Assuming the competence grammar is not wildly overgenerative, this confirmed our suspicions that there were actually gaps in the rules extracted from even the POW corpus. The observation that a comprehensive context-free grammar for English, like its vocabulary, has a distribution adhering to Zipf's law serves to justify the use of probabilities with such a formalism. The parser using such a large grammar is likely to find a large parse forest of solutions, which we would like to order by their frequency. Observations of the extracted context-free syntax lead us to conflicting conclusions. The grammar's coverage appears to fall short of that which we know should be covered, so we would like to expand the set of rules. At the same time, the grammar is so large, that, if possible, we would like to reduce its size without losing any empirical information. Partial solutions to this quandary are presented below.

3.1.4 Editing the Corpus.

One way the grammar can be reduced is to edit out the inconsistent tree structures and category labels, and I am grateful to Tim O'Donoghue for having performed this task, resulting in the Edited POW corpus (O'Donoghue 1991b, 1991c). O'Donoghue also adopted the method of rule extraction which treats filling and componence as essentially the same relationship. Having edited the POW corpus, he applied his own rule and lexicon extraction programs to EPOW. The set of rules extracted (treating filling as componence) numbers only 2820. In my remaining work on the POW SFG, I will actually use O'Donoghue's more restricted (but cleaner) set of probabilistic context-free rules.

Part of O'Donoghue's extraction process also removed some category labels which might preferably have been retained[1], from a linguistic perspective. For instance, the unfinished unit labels *CLUN, NGPUN, PGPUN* etc. were removed, as were bracketed labels signalling ellipsis; *(S), (M)* and <*S*>, <*M*> etc. This will mean that unfinished sentences, and those with missing elements will be treated in the same way as complete sentences. We would still like our parser to find analyses for such sentences, but to distinguish them from complete sentences.

---

[1] Note that these categories were only removed during the rule extraction process. The edited version of the corpus retains them.

Further ways of reducing the physical size of the syntactic model (to make parsing more efficient) are to take account of optionality, repeated daughters and mutually exclusive daughters. Weerasinghe (1990; 48) provides a brief account of how he incorporates optionality and mutual exclusivity into a chart parser for a limited SFG model (limited with respect to the range of syntax found in the POW corpus, at least). Here we consider just optionality and repetition (in co-ordination, for example).

3.1.5 Collapsing a Syntax Model Using Optionality.

It is possible to further reduce the size of the syntax model with no loss of linguistic information by introducing a notation for optionality of daughters in a rule. For instance the rules in Figure 14 could be collapsed into just one rule. (There is an assumed rewrite arrow between the second and third elements of these rules).

Figure 14. Syntax Rule Reduction Using Optionality.

[-0.21922 QQGP AX]
[-2.73498 QQGP T AX]
[-6.82655 QQGP AX FI]
[-5.72793 QQGP T AX FI]

can be reduced to

[-0.137776 QQGP [T] AX [FI]]

I wrote a POP11 program to recursively collapse all the componence rules in the grammar (those with the mothers *CL, NGP, QQGP, PGP, GC* or *TEXT*), after preprocessing them using a suite of AWK programs into a POP11 list format, with frequencies transformed into logs of their probabilities. The transformation of frequency into probability uses the following equation. Given a rule R with a mother category M:

Prob(R/M) = Freq(R)/Freq(M)

For example, if the following two rules (with raw frequencies) were the only ways the mother unit genitive cluster could be expanded

[1 GC G]

[3 GC G OWN]

then their probabilistic versions would be

[0.25 GC G]

[0.75 GC G OWN].

When collapsing the two rules into one permitting optionality, their probabilities would be added, since the collapsed rule represents the disjunction of the two source rules:

[1 GC G [OWN]].

Unfortunately, in combining these rules we lose the information that the second rule is three times as likely as the first, i.e.. that the optional element is much preferred. This information could be preserved by annotating the optional brackets with probabilities:

[1 GC G [0.75 OWN]].

The solution is much more complicated when more than one element is optional. For *n* optional daughters, we would need a matrix of 2^n probability entries. It is anticipated that *n* could be at least four. The matrix size and complexity would add to that of the grammar itself, so initially I have experimented with the simpler format which involves some loss of frequency information.

Assuming a basic procedure for combining rules akin to that exemplified above, but which also allows rules containing optional daughters to combine with each other, various algorithms might be used to achieve the reduction of the rules using optionality, three of which are discussed here:

1. Order the rules for any particular mother by the number of daughters, greatest first. Attempt to combine successive rules which have only one different daughter. After each combination, reorder the rules, and repeat until no further combinations are possible.
2. Using linguistic knowledge of the intended head daughter in a rule, start with a rule containing just the head, and recursively build on optional sisters to the head as they are found. When no further sisters can be added which are compatible with the current optional rule, restart creating a new optional rule combining the head with a different sister. Repeat until no further combinations are possible.
3. Order the rules by frequency, most likely first. Successively attempt to combine the top rule with those that are less likely. Each time a combination is made, put the combined rule on top of the list and recurse. When no more combinations are possible with the top rule, recurse with the second rule. Repeat until no further combinations are possible.

I tried each of these approaches manually on a subset of the grammar (the rules with *QQGP* as mother) to assess which would achieve the greatest overall reduction in rules. The second algorithm would be difficult to implement as it stands, since in many case in the corpus, the linguistic head of a constituent is missing[2]. The third algorithm was seen as the most productive, since it achieved the greatest reduction in the number of rules, and the optional rules created tended to automatically take account of the linguistic notion of a head, since the most frequent rules included one. The first algorithm failed to deliver this feature.

I implemented the third algorithm in POP11, and tested the program against the manually reduced sample, before applying it to the whole of the componence model (2655 rules), and reducing it to only 1787 rules. Figures 15 and 16 show the effect of the reduction on the quantity-quality group (*QQGP*) syntax; 31 original rules in Figure 15 are collapsed to 16 rules in Figure 16.

Figure 15. A Probabilistic QQGP Syntax.

[-0.21922	QQGP AX]	[-5.15257	QQGP DD AX]
[-2.73498	QQGP T AX]	[-5.25101	QQGP DD AXT]
[-3.10892	QQGP AXWH]	[-5.48281	QQGP T AXT]
[-3.81428	QQGP AX SC]	[-5.62257	QQGP AXT FI]
[-3.87337	QQGP AXT]	[-5.72793	QQGP T AX FI]
[-5.15257	QQGP & AX]	[-5.72793	QQGP T]

[2] It is considered an advantage of the corpus-based approach being adopted here that the grammar includes such rules. A competence grammar such as that included in Appendix 9 or in GENESYS will normally have the restriction that each rule should contain a head, whereas in unrestricted English corpora this is not the case. A further approach has been proposed by Eric Atwell: assume the head is required, collapse all the rules which include the head, then create a set of fall-back rules without the head. The parser applies the headed rules first, and only adds fall-back rules when the parser fails.

[-5.78509	QQGP T AX SC]	[-8.6183	QQGP AX T FI]
[-6.1334	QQGP AX T]	[-8.6183	QQGP AXWH SC]
[-6.53886	QQGP TWH AX]	[-8.6183	QQGP DD AXT FI]
[-6.82655	QQGP AX FI]	[-8.6183	QQGP DD T AX]
[-6.82655	QQGP AXT SC]	[-8.6183	QQGP DQ AXT]
[-6.82655	QQGP DQ AX]	[-8.6183	QQGP DQ T FI]
[-7.23201	QQGP T AXT FI]	[-8.6183	QQGP DQ]
[-7.51969	QQGP T FI]	[-8.6183	QQGP SC]
[-7.92516	QQGP INF AX]	[-8.6183	QQGP T AX SC FI]
[-8.6183	QQGP & T AX]		

Figure 16. A Reduced Probabilistic QQGP Syntax.

[-0.137776	QQGP [T] AX [FI]]	[-5.62257	QQGP AXT FI]
[-2.40972	QQGP [AXWH] [SC]]	[-6.53886	QQGP TWH AX]
[-3.64157	QQGP [DD] AXT [FI]]	[-6.6724	QQGP DQ [AX]]
[-3.68383	QQGP [T] AX SC]	[-6.82655	QQGP AXT SC]
[-4.81164	QQGP T [AXT] [FI]]	[-7.23201	QQGP [DQ] T FI]
[-5.1218	QQGP & [T] AX]	[-7.92516	QQGP INF AX]
[-5.1218	QQGP DD [T] AX]	[-8.6183	QQGP DQ AXT]
[-5.15257	QQGP AX [T] [FI]]	[-8.6183	QQGP T AX SC FI]

This simplistic way of combining probabilities in merged rules can produce high probabilities for rules which include rare options. Extreme examples in Figure 16 are

i) the most likely collapsed rule

[-0.137776 QQGP [T] AX [FI]]

includes the relatively rare rule

[-6.82655 QQGP AX FI];

ii) the second most likely collapsed rule

[-2.40972 QQGP [AXWH] [SC]]

generates the uniquely occurring rule

[-8.6183 QQGP SC].

Although it may appear that this second collapsed rule also generates the rule

[-2.40972 QQGP []]

in fact it is not possible for the empty daughter case to occur in a chart parser. The rule will only ever be added to the chart if a potentially combining inactive edge labelled *AXWH* or *SC* has already been found.

3.1.6 Expanding a Syntax Model using Co-ordination.

It is possible to reduce the set of syntax rules slightly further by collapsing the 165 filling rules, which allow for co-ordination and subordination of units in SFG. For example, the rules which permit a sentence to contain up to ten co-ordinated clauses in the corpus

[-0.26847 Z CL]
[-1.73170 Z CL CL]
[-3.15969 Z CL CL CL]
[-4.50744 Z CL CL CL CL]
[-5.88536 Z CL CL CL CL CL]
[-6.72161 Z CL CL CL CL CL CL]
[-7.90027 Z CL CL CL CL CL CL CL]
[-7.90027 Z CL CL CL CL CL CL CL CL]
[-8.59341 Z CL CL CL CL CL CL CL CL CL]
[-9.28656 Z CL CL CL CL CL CL CL CL CL CL]

could all be reduced to one rule

[[-0.26847 -1.73170 -3.15969 -4.50744 -5.88536 -6.72161 -7.90027 -7.90027 -8.59341 9.28656] Z CL].

This method has been implemented using an AWK[3] program, and reduces the filling grammar to 111 rules (Appendix 12). Their probabilities are stored in an ordered list as the first element of the rule (rather than as an atomic element), and we can exploit this difference in data structure from a canonical probabilistic rule in order to make the parser handle the rule differently, allowing optionally repeated daughters. This technique actually offers a way of expanding the grammar's coverage at the same time, since we could allow the parser to analyse potentially

[3] For an introduction to this string processing language see (Aho et al 1988).

infinitely co-ordinated structures. The probability of the rule decreases with increased co-ordination, and a function can be estimated to predict the probability of unobserved co-ordination, using the evidence from the corpus. Figure 17 shows the average probability of co-ordination for all units filling all elements of structure in the EPOW corpus. Such a function is also useful in influencing co-ordination in an SFG NL generator, such as GENESYS.

Figure 17. Co-ordination Likelihood in the EPOW Corpus.

Daughters	Probability (%age)
1	47.58515
2	0.51551
3	0.10314
4	0.04766
5	0.02206
6	0.02409
7	0.01235
8	0.03706
9	0.01853
10	0.00927

Note that the percentage probabilities do not add up to 100. They represent the likelihood of any particular mother containing a particular daughter (not just any daughter) and the likelihood of co-ordination of that daughter. For example, consider the uncollapsed rules with the mother adjunct (*A*):

10.9083	A CL
0.178094	A CL CL
0.0445236	A CL CL CL
6.94568	A NGP
43.0988	A PGP
0.0890472	A PGP PGP
38.5129	A QQGP
0.178094	A QQGP QQGP
0.0445236	A TEXT

The percentage probabilities in the left column here do add up to 100. But when collapsing rules, we do so for each unique daughter separately. Hence:

```
[[10.9083       0.178094       0.0445236]     A CL]
[[6.94568]                                    A NGP]
[[43.0988       0.0890472]                    A PGP]
[[38.5129       0.178094]                     A QQGP]
[[0.0445236]                                  A TEXT]
```

The probabilities in each column are then averaged to obtain those in Figure 17. The probabilities do not decrease smoothly, since there is only sparse data for co-ordination of more than five daughters (the co-ordination of clauses in sentences). The estimated probability degradation function will therefore be smoothed to allow for lack of data. The function currently in use is given in section 4.2.2.

These methods will go some way to reduce the problem of undergeneration in the grammar, but do not address the gaps in the componence grammar, which can only be rectified by the parsing of a larger corpus, or the re-estimation of the grammar with added implicit rules. Alternatively we could relax the grammar formalism: It would be possible to use the context-free grammar to create a wider-coverage probabilistic immediate dominance grammar (in which no ordering of the daughters is assumed), and re-parse using this relaxed format in the case where a chart parser fails. As yet, none of these solutions have been tried, so the grammar will still 'leak' occasionally.

At the same time as leaking occasionally, a simple context-free syntax model of SFG will tend to over-generate, without some form of constraint on the vertical relationships in the tree. The full range of matrix clause structures is not appropriate in a relative clause, for example, yet a simple context-free model is unable to capture this fact. Earlier in section 3.1, we eschewed the idea of a extracting a full context-sensitive model from the corpus (one in which elements of structure and units are conflated on one node) for the sake of efficiency. One way of recouping the lost empirical information is to extract a separate model of the vertical relationships in the corpus, as indeed O'Donoghue has done in his vertical strip grammar. His vertical strips are somewhat limited by their being of restricted depth, and not very general, since they span from leaf to root in one go. Instead, I have extracted a probabilistic vertical trigram model, using a recursive POP11 program to search every tree in the corpus for possible grandmother, mother, daughter

triples. These are of the form [*probability daughter mother grandmother*]. For example, the trigrams

[38 & CL AL]
[7 & CL A]
[3 & CL CV]
[26 & CL C]

show that a linker *(&)* has a *CL* mother and *AL* (logical adjunct) grandmother 38 times in the corpus, whereas the same daughter-mother combination under a simple adjunct (*A*) occurs only seven times. A larger fragment of the model is to be found in Appendix 7, and the total model contains 968 trigrams.

Using these methods I have reduced the overall componence and filling grammar to about two-thirds of its original size (from 2820 to 1898 rules), while increasing its coverage potentially infinitely. Possible over-generation which would have been the outcome of a pure context-free model has been constrained by the provision of a vertical trigram model. The changes made to the chart parsing algorithm to accommodate the new rule format are described in chapter 4. We will now consider the development of accompanying lexical resources.

3.2 Developing a Probabilistic Lexicon for Systemic Functional Grammar.

One characteristic of many existing NLP systems is the lack of attention paid to provision of an adequate lexicon. Much of the corpus-based computational linguistic work at Leeds is no exception, since we have focused our efforts on parsing techniques (see for example Atwell et al 1988, O'Donoghue 1993, Pocock and Atwell 1993). The research presented here, however, attempts to reverse the trend. Prior to the present work, the largest lexicons we tended to use have been word lists extracted from the corpora themselves, along with the possible parts of speech (word-tags), and frequencies of each word-form word-tag pair. This sort of list differs from that typically contained in a machine-readable dictionary (MRD) in that morphological variants of a word-stem are separate entries in the corpus word list, and frequencies are included for each reading of a syntactically ambiguous word. The entries corresponding to the stem *brick*

are shown in Figure 18, with word-tags *H* for head (of a nominal group), and *M* for main verb. (Out of a total number of tokens of 65,000 in the corpus, the frequencies may appear unduly high. However, the children were recorded in interviews, and at play during a *Lego* building task!).

Figure 18. Variants of the Word-Stem *brick* in the POW Lexicon.

16	BRICK	H
1	BRICK	M
1	BRICKED	M
73	BRICKS	H

The modest lexicon extracted from the POW corpus contains 4984 unique lexical items, and serves as an adequate facility for prototype parser development. Its advantages are that the grammatical word-tags correspond exactly to those used in the large systemic functional grammar we have extracted from the corpus, and that it contains 'realistic' disambiguated frequencies for each word-form[4]. However, in the context of a parser for unrestricted English, the inadequacies of such a lexicon far outweigh these advantages: the parser will fail at the lexical look up point too frequently to be of any practical use. This section will describe these inadequacies, and discuss the method used to select and convert the CELEX English database into a suitable form for probabilistic parsing with the corpus-based systemic grammar. This work has been reported in (Souter 1990b, 1993a).

3.2.1 Inadequacies of the POW Corpus Word List.

The most important problem with the POW word-list is its lack of coverage. With fewer than 5,000 words in the lexicon, many attempts at parsing a new sentence will fail immediately at the lexical look-up stage, unless such a sentence has been carefully chosen with the subject matter of the corpus in mind. It is difficult to put a figure on the size of lexicon required, but, ignoring proper nouns, it is likely to be between 50-100,000 wordforms, if not stems. The frequency distribution of words in corpora is well known to follow Zipf's law, so even such a large lexicon

[4] Note that even this lexicon is over five times as large as the largest lexicon used by either O'Donoghue (1993) or Weerasinghe (1994).

will fail in coverage occasionally, and require *ad hoc* solutions for assigning categories (and frequencies) to a word.

A second problem is the number of inaccuracies in the format of the hand parsed corpus, which result in a few false 'words' being included in the lexicon (and some false rules in the extracted grammar). These consist of typographical errors, misspellings, word-tags being treated as words, occasions where the transcriber was uncertain what had been uttered (and inserted question marks) and spaces being omitted between tags and words causing them to be treated as one item. It may be surmised that these occurrences will be rare, compared to the genuine lexical items, and that removing all items below a threshold frequency of, say, one, or even five, would suffice to leave only the desired data. As was the case with rule extraction, this would actually be a negative step, since thresholding out the noisy data will also remove some genuinely infrequent words, as can be seen from the list of singleton items in Figure 19.

Figure 19. Singleton Word/Word-tag Co-occurrences in the POW Corpus Lexicon.

1	ABANDONED	M	1	ADD?	M
1	ABLE	CP	1	ADDED	M
1	ABLE?	AX	1	ADDING	M
1	ABOUT		1	ADJUST	M
1	ABOUT?	T	1	ADN-THEN	&
1	ABROAD	AX	1	ADRIAN	HN
1	ABROARD	CM	1	ADVENTRUE	H
1	ACCOUNTANT	H	1	ADVERT	H
1	ACHING	M	1	ADVERTS	H
1	ACROSS	CM	1	AERIAL?	H
1	ACTUALLY	AL	1	AEROPLACE?	H

One way round the problem of noise in the corpus is to employ checking programs which post-edit the structure of each tree. O'Donoghue's edited version of the POW corpus has been spelling checked to try to minimise the occurrence of noisy lexical data. The wordlist extracted from the EPOW corpus is reduced to 4618 unique lexical items, which emphasizes its poor lexical coverage, but at least is now a 'cleaner' lexical resource. A sample of the most frequent words in the EPOW wordlist is included as Appendix 13. The overall problem of lack of coverage remains, though, and requires a more radical solution. Before we consider the first of two proposed solutions, our general aims for the lexicon will be presented.

3.2.2 Lexical Aims and Policy Decisions.

Our aims in providing an improved lexical facility for our parsing programs are as follows: The lexicon should be large-scale, that is, contain several tens of thousands of words, and these should be supplemented with corpus-based frequency counts. The lexicon should employ systemic functional grammar for its syntax entries, to conform with the grammar used in parsing. Frequencies should ideally be available for the different readings of a syntactically ambiguous wordform. The lexicon format should be versatile, so it can be utilised by other projects.

Having established these aims, we took several decisions as to how best to achieve them. While it may not be psycholinguistically desirable, it was decided to adopt a wordform list, rather than a list of morphemes. This avoids the need for a time consuming morphological analyser, but will substantially increase the space required to store the lexicon. Secondly, we naturally would prefer to make use of an existing lexical resource, rather than building our own. The definition of systemic functional syntax will be taken to be that used in the POW corpus, which is stable, relatively fine grained and provides many examples of its use for analysing real English sentences. Finally, we will not initially attempt to include any semantic components, or word definitions, as we are presently focusing on syntactic parsing.

3.2.3 Selection of Lexical Resources.

Ideally, we would have an extremely large corpus of several million words from various text types which have been perfectly hand parsed (or even just tagged with terminal categories) according to the same systemic functional grammar as the POW corpus. From such a corpus, a much larger lexicon could be extracted, including more reliable frequencies than those provided by POW. This resource is unlikely to become a reality, so it is necessary instead to look at what is currently available. The alternatives were discussed in section 2.3, and the solution proposed here is to adapt an existing lexical database, and to train a tagging program on the EPOW corpus (see section 3.3). The former involves manipulating the selected entries into a machine tractable form for rapid look up, (we would not wish to load the whole database and associated user interface software), and transforming the grammatical entries to SF syntax form. The lexicon will also have to be augmented with a mechanism for dealing with proper nouns, and any word formats not handled by the database, such as hyphenated and abbreviated words. The most

suitable lexical database option, in combined terms of availability, cost, coverage and tractability was the CELEX database.

3.2.4 The CELEX English Database.

At the Centre for Lexical Information (CELEX), University of Nijmegen, the Netherlands, a team led by Hans Kerkman has built a large lexical database for Dutch, German and English (Burnage 1990, Piepenbrock 1993). The English section of the CELEX database comprises the intersection of the word stems in LDOCE (Procter 1978, ASCOT version, see Akkerman et al 1985, 1988) and OALD (Hornby and Cowie 1974), expanded to include all the morphological variants; a total of 80,429 wordforms. Moreover, the wordform entries include frequency counts from the COBUILD (Birmingham) 18 million word corpus of British English (Sinclair 1987), normalised to frequencies per million words. However, these are not disambiguated for wordforms with more than one syntactic reading. CELEX offers a fairly traditional set of syntax categories, augmented with some secondary stem and morphological information derived mainly from the LDOCE source dictionary. At the time we acquired the data from Nijmegen, it was free to academic researchers, but subsequently a charge has been imposed to new academic and commercial licensees. A menu-driven user interface to the database, called FLEX, allows relative novices to inspect the lexicons available, and extract the data they require to be imported into their own research environment. On a visit to Nijmegen in 1989, we exported from the database a lexicon containing wordform, stem, category, frequency, and morphology information.

A small sample showing the format of our selected CELEX lexicon is given in Figure 20.

Figure 20. A Fragment of our CELEX English Lexicon.

wordform	*stem*	*category*	*stem info*	*freq*	*ambiguity*	*morphol. info*
abaci	abacus	N	Y N N N N N N N N Y	0	N	irr
aback	aback	ADV	Y N N N N N	3	N	
abacus	abacus	N	Y N N N N N N N N N	0	N	
abacuses	abacus	N	Y N N N N N N N N Y	0	N	+es
abandon	abandon	N	N Y N N N N N N N N	16	Y	

abandon	abandon	V	Y N N N N N N N N N N	16	Y	
abandoned	abandon	V	Y N N N N N Y N N Y N N	36	Y	+ed
abandoned	abandoned	A	Y N N N N	36	Y	N N

Columns four and eight indicate subcategorisation and morphological information about each entry, with a binary (Y/N) value. Other columns show the frequency of the word-form, and morphological conversion from the stem to the word-form. The position of the frequency column differs according to the primary syntactic category of the entry, which is one of the set {N, V, A, ADV, PREP, C, I, PRON, EXP}. The position of the subcategorisation and morphology indicators also varies according to the grammatical category of the entry. A key to their meaning is given in Figure 21.

It is inevitable that some manipulation of any imported lexicon will be necessary. In our case, this task involved the removal of unwanted duplicate entries, the creation of a mapping between the different syntax descriptions, the reformatting of the lexicon structure, and the addition of some new entries and frequencies.

Figure 21. Structure of our CELEX Lexicon according to Grammatical Category.

Col	*Nouns*	*Verbs*	*Adjectives*	*Adverbs*	*Others*
1	Wordform	Wordform	Wordform	Wordform	Wordform
2	Stem	Stem	Stem	Stem	Stem
3	Class(N)	Class(V)	Class(A)	Class(ADV)	Class(C,EXP,&,I,PREP,PRON)
4	Count	Transitive	Ordinary	Ordinary	Personal
5	Uncount	Trans+Compl	Attributive	Predicative	Demonstrative
6	Sing. use	Intransitive	Predicative	Prepositive	Possessive
7	Plural use	Ditransitive	Postpositive	Combinatory	Reflexive
8	Group count	Linking	Group uncount	Comparative	Wh-pronoun
9	Group uncount	Plural morph	Frequency	Superlative	Frequency
10	Attributive	Past tense	Ambiguous	Frequency	Ambiguous
11	Postpositive	1st person	Comparative	Ambiguous	Co-ordinating
12	Vocative	2nd person	Superlative		Subordinating
13	Plural morph	3rd person			
14	Frequency	Participle			
15	Ambiguous	Rare verb form			

16	Inflec-affix	Inflec-affix
17		Frequency
18		Ambiguous

3.2.5 Converting the CELEX Lexicon.

Various transformations were performed on the lexicon to make it more suitable for use in parsing (using systemic functional syntax). By means of the UNIX utility *uniq*, a substantial number of the verb entries were removed, which would be duplicates for our purposes, reducing the lexicon to only 59,322 wordforms. This was necessary because the CELEX lexicon includes distinctions between, for example, 1st and 2nd person forms of the verb. These are rightly recorded as separate word-form entries in the database for German and Dutch, but such a distinction is unnecessary for English regular verbs. The full paradigm of irregular verbs such as *be* and *have* was, of course, retained.

Using the AWK program and some UNIX tools, the lexicon was reformatted along the lines of the bracketed, often called 'lispified', LDOCE dictionary, which is perhaps the nearest to an established lexicon format norm. Columns of the lexicon were reordered to bring the frequency information into the same column for all entries, irrespective of syntactic category.

The most difficult change to perform was the transforming of the CELEX syntactic labels into SF syntax. Some mappings were achieved automatically because they were one to one, eg:

prepositions:	*PREP -> P*
main verbs:	*V -> M*
head nouns:	N -> H

Other mappings required use of the stem information in the CELEX lexicon eg:

subordinating and co-ordinating conjunctions:	
	C -> & (= linker, for co-ordinating conjunctions, with Y in column 11), or
	C -> B (= binder, for subordinating conjunctions, with Y in column 12)
pronouns:	PRON -> HP (the general case), or
	PRON -> HWH (= wh-pronoun, with Y in column 8), or
	PRON -> HPN (= negative pronoun, for no-one, nobody, nothing, noone or none)

comparative and superlative adjectives:

A -> AX (the general case), or

A -> AXT (= comparative or superlative apex, with Y in column 8 or 9)

Elsewhere, however, the two grammatical descriptions diverge so substantially (or CELEX did not contain the relevant information) that manual intervention was required to create a file of some 300 word-forms for which SFG is more finely grained than CELEX, or which CELEX had omitted. These can be divided into the following categories:

auxiliaries and modals (O, OM, OMN, ON, OX, OXN, X, XM),
determiners (DD, DDWH, DQ, DQN, DQWH, DO),
interjections (F, EX),
temperers (see below).

The systemic functional grammar category temperer, *T* , is particularly problematic, and even a good definition of the term is difficult to give (see Fawcett 1981 section 6). Temperers are 'adverbs' of degree or scope which may modify other adverbs, prepositions or adjectives, but not verbs. Examples of degree temperers are given in (1).

(1) *about, all that, amazingly, brand, dead, fairly, hardly, how, nearly, particularly, pretty, rather, raving, really, so, terribly, very*.

Scope temperers modify or specify the field of an adjective, such as

(2) *economically* viable, *ergonomically* sound, *mathematically* brilliant.

The production of scope temperers from adverbs seems possible 'on the fly', and would require a large proportion of the adverb entries in the lexicon to be duplicated with syntactic category *T* in the 3rd column, as well the usual *AX* (= apex) of the quantity-quality group. Consequently, only the more common degree temperers have been duplicated in the lexicon. This has a parallel in the case of verb participles being used as adjectives. The CELEX lexicon chooses to include these only in their verbal form. A procedure permitting the relabelling of an apex as a temperer and a verb participle as an adjective could instead be included at the parsing stage. While this may appear to be an *ad hoc* solution, it can be viewed more positively as a mechanism for handling a specific feature of SFG: The problem with such adverbs arises because in SFG parse trees, the

terminal category which is expounded by a lexical item is in fact a functional label (eg. head, temperer) rather than a formal label (noun, adverb). In most cases, the function is realised by only one formal category (eg. the head of a nominal group is a noun) and this formal category only fills one function (a noun is used as the head of a nominal group), so the formal category is omitted from the grammar. Occasionally however, as in the case of adverbs produced from an adjective + *ly* , a category can fill more than one function. To prevent wholesale duplication in the lexicon, this feature could be dealt with more economically by the parser. This mechanism has not yet been implemented, which will result in overgeneration in the analyses that are found for such structures.

The extra entries which were added manually to the lexicon also required some frequency information. I added frequencies from the EPOW corpus normalised to a frequency per million words, although the ideal solution would have been to send the list of extra entries to the COBUILD team and request frequency counts from the same material as had been included in CELEX. I do not expect the variation between corpus genres to seriously affect the probabilistic parsing technique. A fragment of the transformed lexicon is shown in Figure 22.

Figure 22. A Fragment of the Reformatted CELEX SFG Lexicon.

((abaci)(abacus)(H)(0)(N)(Y)(N)(N)(N)(N)(N)(N)(N)(Y)(irr))

((aback)(aback)(AX)(3)(N)(Y)(N)(N)(N)(N)(N))

((abacus)(abacus)(H)(0)(N)(Y)(N)(N)(N)(N)(N)(N)(N)(N)())

((abacuses)(abacus)(H)(0)(N)(Y)(N)(N)(N)(N)(N)(N)(N)(N)(Y)(+es))

((abandon)(abandon)(H)(16)(Y)(N)(Y)(N)(N)(N)(N)(N)(N)(N)(N)())
((abandon)(abandon)(M)(16)(Y)(Y)(N)(N)(N)(N)(N)(N)(N)(N)(N)(N)())

((abandoned)(abandon)(M)(36)(Y)(Y)(N)(N)(N)(N)(N)(Y)(N)(N)(Y)(N)(N)(+ed))
((abandoned)(abandoned)(AX)(36)(Y)(Y)(N)(N)(N)(N)(N)(N))

KEY: ((wordform)(stem)(category)(frequency)(ambiguity)(...stem info...)(...morphol. info...))

3.2.6 Testing the Lexicon.

To assess the adequacy of the CELEX lexicon, a program was written (in AWK) to look up each of the 4618 unique words of the Edited Polytechnic of Wales Corpus word list in the CELEX

lexicon. This has the advantage that both lexical coverage and the syntax entries can be evaluated. Using another larger corpus word list (say from the LOB corpus) would provide a more rigorous test of lexical coverage, but disallow testing of the syntactic classes, or tags. The testing program assigned words from the EPOW list to three sub-lists; words not in the CELEX lexicon, words in CELEX, and words in CELEX for which CELEX also contains the same syntactic category. Results are shown in Figure 23, both for word types (4,618), and, more realistically, word tokens (60,784).

Figure 23. Testing the CELEX SFG Lexicon with the EPOW Word List.

	type/token total		not in lexicon		in lexicon		in lexicon with same tag	
types:	4618	100%	1301	28.2%	3317	71.8%	2613	56.6%
tokens:	60784	100%	4232	6.96%	56552	93.04%	49605	81.61%

These initial results suggest that lexical coverage is quite good, with only a little under 7% of word tokens not recognised. Inspection of the list of words which were not found in the CELEX lexicon shows that these belong almost entirely to three groups: hyphenated words (compounds)[5], proper nouns (tagged *HN*) and words containing apostrophes (mainly abbreviations). Apart from these three groups, the residue consists of only 764 tokens (1.2%), and includes words which would be classed as ungrammatical in standard adult English (*bringed, builded, chimley, comed, digged, drawed, drownded, goodest*) and words which are typical of spoken language (*cor, flippin, hoick, na, oops, tata, wonkety*) or of Welsh English (*ay, bach, boyo, yere*).

Areas of weakness in the grammatical mapping can be identified by considering the words which were found to be in the CELEX lexicon, but for which the transformed CELEX-SFG lexicon did not contain the same word tag. These consist primarily of a group of prepositions also labelled in POW as apexes (AX), (*on, up, out, in, down*), adjuncts (A, AI, AF, AL ...) *such as just, sometimes, only, and formulae (F) yeah*, no, yes, what, pardon, alright, aye, right. These are typical of areas where systemic functional grammar makes fine-grained distinctions which are

[5] Note that the POW corpus contained an unusually high proportion of hyphenated words, due at least in part to the SFG requirement for a single-word head of a nominal group. This led to compound noun sequences being transcribed as single hyphenated wordforms. Eg. *adventure-books air-hostess ball-bearings*.

not found in the CELEX source lexicon. Furthermore, from time to time in the POW corpus, a label normally used on a non-terminal node (for a quantity-quality group functioning as an adjunct) may be used as a terminal to label a single word if the group is non-productively expounded by only one word. This is a short cut in the labelling of the parsed corpus which it would be unrealistic to expect to be included in a non-SFG lexicon, such as CELEX.

Some of these observations have led to the syntax mapping being amended, such as the inclusion of the label *AXWH* for *wh-* apexes (*where, when* etc.). Further amendments and supplements will continue to be made to the lexicon, which is an ongoing task for its developer. Recognising the fallibility of the lexicon, within the parser itself two default mechanisms have been introduced to deal with the occasions when the lexical look up process fails. Firstly, if a word begins with a capital letter (and the word is not *I*), then it is assigned the label *HN* for namelike-head (proper noun). Secondly, if after consulting CELEX and checking for proper nouns, there has still not been a tag assigned, three default labels are attached to a word: *H, M* and *AX*. This is analogous to the CLAWS default tagging of *NN VB JJ*, see (Leech et al 1983, Johansson et al 1986). Hence uncapitalised hyphenated words will fall into this catch-all category. Input to the parser is expected to have been preprocessed with respect to the occurrence of apostrophes (see section 1.6.1). As the lexicon is refined, the default mechanisms will be brought into action less and less.

3.2.7 Rapid Lexical Lookup.

Storing a lexicon containing 60,000 entries within working memory whilst running the parser will have an adverse affect on parser speed, since it will limit the memory available for the parsing process itself. It will also restrict the choice of machines the parser will run on, if the size of the process exceeds the memory of the machine, and swap space has to be used.

As a consequence, it was decided to keep the lexicon in secondary memory, and use an indexing program to look up words rapidly. Such a program was supplied to CCALAS with the LDOCE MRD, which created an indexed file for all the entries in the dictionary. This C program was modified to work with the CELEX SF lexicon, which had been formatted to match LDOCE's structure. I am grateful to George Demetriou of Leeds University for identifying and solving an inconsistency between the UNIX alphabetical *sort* command, and the alphabetical ordering needed for the C look up program, which had caused a small number of entries not to be found,

despite their presence in the lexicon. Demetriou also amended the C code to allow entries to be accessed by a UNIX command. For example the command:

% celex5 bottle

returns the entries:

((bottle)(bottle)(H)(83)(Y)(Y)(N)(Y)(N)(N)(N)(N)(N)(N)(N)())

((bottle)(bottle)(M)(83)(Y)(Y)(N)(N)(N)(N)(N)(N)(N)(N)(N)(N)(N)())

This type of command could then be executed from within the parsing program, and the UNIX output stream diverted back into the parser. In this way rapid lexical look up was achieved without the cost of search through a lengthy lexicon list structure, and the damaging effect on working memory. The implementation of the look up process is further described in chapter 4.

3.2.8 Lexical Probabilities.

In Pocock and Atwell's parser, the probability of a word with a particular tag was calculated by multiplying the lexical probability of the word by the probability of the word being labelled with a specific tag. This makes sense in their application (speech recognition) in which the likelihood of a word should play its part as well as its possible tags. In CELEX, the tag likelihood information is not known, since the CELEX entries are not syntactically disambiguated. As a result, tag probabilities have simply been calculated by dividing 1 by the number of tags for a given word. This will clearly fail to bias the parser in the way we would prefer, to take account of a specific tag being the most likely. For instance the word *yes* has a lexical probability of 740/1000000 = 0.00074 (frequencies in the CELEX lexicon are per million words). It is assigned two tags, *F* and *H*, so the tag probability is 0.5. The combined lexical and tag probability of *yes* as a head noun would therefore be calculated as 0.00037. When the lexical edges for the word *yes* are added to the agenda (along with edges for other words in the sentence), they would be likely to be removed quite quickly from the agenda and added to the chart, (remember the agenda is ordered by probability), since *yes* is quite a common word. Once added to the chart, its edges would combine with others derived from grammar rules to build a subtree. However, lexical edges for rare words, and those that are highly ambiguous, may remain on the agenda for some time. It would be preferable to influence the search strategy to include the other lexical edges at an earlier stage, to produce more of a breadth first method. To achieve this, the lexical probabilities were removed from the function for assigning probabilities to tagged words, and the tag probability used on its own. A lexical edge for an unambiguous word would therefore have a probability of 1. This change served to bring lexical edges into the chart more quickly, and is

justifiable because the input itself is not in question (i.e.. the parser is not being used to disambiguate speech or handwriting recognition lattices).

3.3 Training the Brill Tagger.

Hughes (Hughes and Atwell, forthcoming) has copied the publicly available source code for the Brill tagger (Brill 1992, 93, 94) to Leeds, for use in the AMALGAM project (Atwell et al 1994). In this project, we are attempting to produce a multi-tagged and multi-parsed corpus of English, using the Spoken English Corpus as a benchmark corpus for future tagging and parsing programs. In order to obtain the SEC tagged according to several syntactic annotation schemes, we can re-train the Brill tagger for each scheme on a suitable tagged corpus, and then use the tagger to tag the SEC. One of our chosen schemes was that used in the EPOW corpus. Hughes has modified the original fully parsed corpus by removing all the non-terminal annotation, leaving a tagged version of POW. It was also necessary to 'tokenise' the corpus into the input form expected by the tagger's training module. This consisted of adding some punctuation, to act as sentence separators, making all characters lower case (except for those labelled as proper nouns) and other minor amendments.

Most tagged corpora are significantly larger than POW's 65,000 words[6], so all of the corpus was used in training to maximise the performance of the resulting lexicon, context rules and bigram model, which constitute the main components of the tagger. (The working of the tagger was described in section 2.3.1, and example fragments of these components are to be found in Appendices 10 and 11). The tagger was then tested on some of the POW data on which it had been trained, with a success rate of around 95%. On untrained material, we would therefore expect this rate to be somewhat lower. The Brill tagger also learns general morphological and context rules in order to be able to guess tags for words on which it has never been trained. In particular, it is trained to assign the label *HN* to any word beginning with a capital letter (other than *I*). In the absence of a lexical entry or any of the learned bigrams and context rules being applicable, the default tag *H* is assigned. Again, the POW corpus will be a relatively small source of data for these general rules.

---

[6] Brill 1994 refers to training data of up to 1 million words, for maximum success rates.

The parsing code for the chart parser has been adjusted to handle either the CELEX lexicon look-up, or to call the Brill tagger from within POP11. In the first case, one or more tags are assigned to each word of input, along with the probabilities described in section 3.2.8. For the Brill tagger, however, only one tag is added per word, so the tag probability is always 1.

3.4 Conclusions.

This chapter has shown how a wide-coverage probabilistic context-free grammar can be derived from the EPOW corpus, and then reduced in size but increased in coverage by the introduction of rules containing optional and co-ordinated daughters. A vertical trigram model has also been extracted from the corpus to capture the vertical dependencies of SF syntax. A large lexical database was accessed to provide a probabilistic lexicon which permitted rapid look up, and was modified to match the corpus-based syntactic annotation. As well as producing an SF syntax lexicon specifically for an SF parser, we have also established a methodology for conversion between lexico-grammatical labelling schemes, which can be applied in translating CELEX to other corpus-based grammatical descriptions, eg. for the SEC and LOB corpora. As a second lexical resource, the Brill tagger has been trained on the POW corpus. The corpus-based systemic functional lexical and syntax models form the primary resources for the parsing process itself, which will be described in chapter four.

Chapter 4. A Probabilistic Chart Parser for Systemic Functional Syntax.

In section 2.4 I described various approaches to the problem of parsing unrestricted English, and proposed an architecture with three separate components: a probabilistic lexical resource (achieved either by lexical look-up or by a tagging program), a probabilistic corpus-based syntax model and the probabilistic chart parsing algorithm. The parsing component can be developed in a relatively modular fashion, allowing for it to be applied to different grammar descriptions, providing they each adhere to the same formalism. In chapter three I showed how large lexical and syntactic resources could be harmonised within a systemic functional grammar (SFG) description, such as that found in the POW corpus.

Instead of designing a parser from scratch, I have decided to adapt the chart parsing algorithm of Pocock and Atwell (1993) which was itself derived from Gazdar and Mellish (1989). Pocock and Atwell experimented with chart parsing using a grammar extracted from the Spoken English Corpus, but later resorted to a faster Markov model approach. Some changes are necessary to the parsing program itself because of the amendments to the syntactic formalism (to handle optionality, co-ordination and vertical trigrams) I used to reduce the grammar size and prevent overgeneration, and these are described in this chapter. The results of testing the parser are given in chapter 5, but I include some discussion of informal trials with trivial examples here for illustration purposes.

The terminology and standard procedure for chart parsing was presented in section 2.4.1.2. Pocock and Atwell's amendments to the basic chart parsing algorithm are given in Appendix 14. Their chart parser was written in POP11, and designed to allow word lattices produced by a speech recogniser to be disambiguated, and employed a large grammar and modest lexicon extracted from the IBM/Lancaster Spoken English Corpus (Knowles and Taylor 1987). They used rule probabilities taken from the corpus, combined with lattice and lexical probabilities for the words, to force the chart parsing search strategy to produce the most likely tree first. Although Pocock and Atwell finally abandoned the chart parser owing to speed and system limitations, it is being adapted here to work with a smaller grammar (in terms of number of

rules), and an external lexicon. The use of POP11 as a programming language in this context is well justified. It offers typical AI features such as lists, properties (fast look-up hash tables) and pattern matching along with facilities one would associate with a procedural language, such as array handling and fast arithmetical processing. External function calls are possible to a number of languages within the POPLOG environment (PROLOG, LISP, ML), and to C and the UNIX operating system. The COMMUNAL project is using POPLOG PROLOG as its programming language, which would mean that the two systems could be interleaved if desired. O'Donoghue's semantic interpreter (section 2.2.1) was also written in POP11. POPLOG does however have some drawbacks which will have an influence on the parser's performance.

Pocock classified his final code as being version 5 of his probabilistic chart parser. Two developments to this parser will now be described:

Version 6:

Lexical look-up:	the CELEX SF lexicon.
Syntax formalism:	Probabilistic context-free rules, with modification only to permit optionality and repeated daughters.
Parsing algorithm:	Chart parser, ordering agenda according to likelihood.
Probabilistic function:	Simple multiplication of combined edges, with a probability degradation function applied to co-ordinated daughters.
Edge format	[weight start finish label found tofind]

Version 7:

Lexical look-up:	The Brill tagger.
Syntax formalism:	Probabilistic context-free rules with modifications to permit optionality and repeated daughters supplemented with probabilistic vertical trigrams.
Parsing algorithm:	Chart parser, ordering agenda according to likelihood, with constraints on subtree depth and overall tree depth. New edges only proposed if well-formed with respect to the vertical trigram model. Vertical recursion severely restricted. Extra parsing termination conditions incorporated.

Probabilistic function:	Multiplication of combined edges, but with an extra weighting factor contributed by the product of the component vertical trigrams (for all but leaf nodes).
Edge format	[weight start finish label found tofind depth]

Test results for each of these versions are to be found in chapter 5, where they are referred to as tests 1 and 2.

The loading process for versions 5-7 takes only a few seconds. From within POP11 the file *parseV5/6/7.p* is loaded, which itself consults all the parsing code and datafiles containing the syntax and lexicon. A prompt is given for three types of parsing: parsing word sequences, tag sequences, or lattices of words. The first of these was used exclusively in the tests that follow. The diagnostic messages output during loading are shown in Appendix 15. The parser can be run in two modes, with or without a trace facility. A trace facility is used if the user wishes to study the parser whilst it is working, by seeing the (inactive) edges it is adding to the chart. This is achieved by the parser outputting such edges to a file. Using the trace facility slows down the parser, though, so for batch mode it is omitted. However, the trace facility has been used for all of tests described here.

4.1 Preliminary tests using Parser Version 5.

Minor amendments were made to the input format of version 5, to permit input of upper and lower case words, and deal with variations between the SEC and POW with respect to punctuation. Since the SEC includes punctuation, its minimum sentence length was hard-wired to be two items, and this was adjusted to permit single-word utterances. A small part of the version 5 parsing code was specifically written with the SEC corpus grammar in mind, and had to be swapped for its equivalent within SFG. For instance, when initialising the agenda, the default grammatical tags assigned to words not found in the lexicon are the three for noun, verb and adjective in the SEC. The lexical probabilities are combined with *lattice probabilities* in Pocock's implementation. These are provided by a speech recognition device as a guide to how 'confident' the recogniser is of the word(s) it has produced. Since this is not the primary application for our SF parser, the lattice probabilities were excluded. Pocock's parser also waited until parsing had finished entirely before producing any output. I amended the code to output

successful parses to the screen as soon as they were found, since using a very large grammar can mean that the parser will run for some considerable time.

The most significant change was to incorporate the CELEX SFG lexicon. In Pocock's implementation, the lexicon is stored as a property list within POP11. For reasons put forward in section 3.2, I stored the lexicon externally, and accessed it via a UNIX call to a C look up program. Once an entry has been retrieved, the word, probability, SF syntax tag, and its start and end point within the sentence are then combined in an edge, stored as a list structure. The edge is used to initialise the chart parser's agenda. If more than one category can be assigned to the word (i.e. it is syntactically ambiguous) then further edges are added to the agenda. In version 6, the agenda is so initialised for each word in the sentence before parsing begins. For example, if the sentence being parsed had the first word *why*, the following edges produced by CELEX look-up would be added to the agenda:

```
[-6931 0 1 AX [why] []]
[-6931 0 1 EX [why] []]
```

The edges containing the word *why* are labelled apex (*AX*) or exclamation (*EX*), have a logarithmic lexical probability of -6931 (log(0.5) since there are two tags)[1], start at position 0 (the start of the sentence) and finish at position 1. Their contents are the word itself. The empty brackets at the end of each list represent the fact that no further information is needed to complete the lexical edge; it is inactive.

Before embarking on any formal trials, I experimented with some 'trivial' input. The output produced for the single word utterance "yes" is included in Figure 24.

Figure 24. A Trivial Test of Parser Version 5.

```
[-29742 0 1 Z [[CL [F yes]]] []]
```

[1] For further discussion of lexical/tag probabilities see section 3.2.8.

[-63486 0 1 Z [[CL [F [QQGP [AX [NGP [H yes]]]]]]] []]
[-69092 0 1 Z [[CL [C [NGP [H yes]]]]] []]
[-76664 0 1 Z [[CL [F [QQGP [AX [QQGP [AX [NGP [H yes]]]]]]]] []]
[-80403 0 1 Z [[CL [C [CL [F yes]]]]] []]
[-81034 0 1 Z [[CL [AL [CL [F yes]]]]] []]
[-83607 0 1 Z [[CL [S [NGP [H yes]]]]] []]
[-89842 0 1 Z [[CL [F [QQGP [AX [QQGP [AX [QQGP [AX [NGP [H yes]]]]]]]]]] []]
[-91782 0 1 Z [[CL [V [NGP [H yes]]]]] []]
[-93287 0 1 Z [[CL [C [QQGP [AX [NGP [H yes]]]]]] []]
[-100481 0 1 Z [[CL [A [CL [F yes]]]]] []]
[-101484 0 1 Z [[CL [A [QQGP [AX [NGP [H yes]]]]]] []]
[-102837 0 1 Z [[CL [AF [CL [F yes]]]]] []]
[-103020 0 1 Z [[CL [F [QQGP [AX [QQGP [AX [QQGP [AX [QQGP [AX [NGP [H yes]]]]]]]]]]] []]
[-105239 0 1 Z [[CL [C [TEXT [Z [CL [F yes]]]]]] []]
[-106465 0 1 Z [[CL [C [QQGP [AX [QQGP [AX [NGP [H yes]]]]]]]] []]
[-106522 0 1 Z [[CL [CWH [NGP [H yes]]]]] []]
[-110115 0 1 Z [[CL [AWH [QQGP [AX [NGP [H yes]]]]]] []]
[-111031 0 1 Z [[CL [A [NGP [H yes]]]]] []]
[-111215 0 1 Z [[CL [SWH [NGP [H yes]]]]] []]
[-114147 0 1 Z [[CL [C [CL [F [QQGP [AX [NGP [H yes]]]]]]]] []]
[-114662 0 1 Z [[CL [A [QQGP [AX [QQGP [AX [NGP [H yes]]]]]]]] []]
[-114778 0 1 Z [[CL [AL [CL [F [QQGP [AX [NGP [H yes]]]]]]]] []]
[-115403 0 1 Z [[CL [F [QQGP [AX [NGP [DQ [NGP [H yes]]]]]]]] []]
[-116166 0 1 Z [[CL [ATG [CL [F yes]]]]] []]
[-116198 0 1 Z [[CL [F [QQGP [AX [QQGP [AX [QQGP [AX [QQGP [AX [QQGP [AX [NGP [H yes]]]]]]]]]]]]]]]]
[]]
[-116997 0 1 Z [[CL [F [QQGP [T [NGP [H yes]]]]]] []]
[-117625 0 1 Z [[CL [F [QQGP [AX [PGP [CV [NGP [H yes]]]]]]]] []]
[-117685 0 1 Z [[CL [F [QQGP [AX [NGP [DP [NGP [H yes]]]]]]]] []]
[-118763 0 1 Z [[CL [C [PGP [CV [NGP [H yes]]]]]] []]
[-118786 0 1 Z [[CL [F [QQGP [AX [NGP [DQ [QQGP [AX [NGP [H yes]]]]]]]]]] []]
[-119643 0 1 Z [[CL [C [QQGP [AX [QQGP [AX [QQGP [AX [NGP [H yes]]]]]]]]]] []]
[-119753 0 1 Z [[CL [C [CL [C [NGP [H yes]]]]]] []]
[-120220 0 1 Z [[CL [AP [CL [F yes]]]]] []]
[-120384 0 1 Z [[CL [AL [CL [C [NGP [H yes]]]]]] []]
[-121009 0 1 Z [[CL [C [NGP [DQ [NGP [H yes]]]]]] []]
[-123291 0 1 Z [[CL [C [NGP [DP [NGP [H yes]]]]]] []]
[-123293 0 1 Z [[CL [AWH [QQGP [AX [QQGP [AX [NGP [H yes]]]]]]]] []]
[-124375 0 1 Z [[CL [AM [CL [F yes]]]]] []]
[-124392 0 1 Z [[CL [C [NGP [DQ [QQGP [AX [NGP [H yes]]]]]]]] []]
[-124393 0 1 Z [[CL [CWH [QQGP [AX [NGP [H yes]]]]]] []]

It was still necessary to interrupt the parsing process after over an hour since it was still active, but inspection of the output made me realise that recursion in the grammar would be likely to cause it to continue indefinitely. Solutions are output in descending logarithmic probability order (most likely first). The label of each successful edge is *Z*, it must start at 0 and finish at 1 (for a single word utterance), and not have anything left in the final set of brackets (the list of daughters needing to be found). The first solution is the desired one, and should certainly be found, since it is about the most frequent utterance (and analysis) in the corpus! However, the multiplicity of other solutions is some cause for concern. Some are legitimate SF syntactic structures. For instance, the edges

[-69092 0 1 Z [[CL [C [NGP [H yes]]]]] []]
[-91782 0 1 Z [[CL [V [NGP [H yes]]]]] []]

would be legitimate analyses in answer to the questions '*Was that a yes or a no?*' and '*Did you say yes or no?*'.

However, the solutions of the form

[-76664 0 1 Z [[CL [F [QQGP [AX [QQGP [AX [NGP [H yes]]]]]]]] []]

will give rise to infinite mutual recursion. The quantity-quality group can be rewritten as a single apex, and the apex can itself be filled by a quantity-quality group. This sort of recursion is normally prevented in a chart parser by checking to see if the same rule has been applied previously at the same point in the chart, before adding a new edge. In this case that procedure obviously wasn't succeeding, even though such a procedure had been included in the program. The reason for its failure was that the actual test performed was to see if a newly proposed edge was already a member of the agenda or the chart. The POP11 *member* function would fail in the cases of recursion shown above because at each test the newly proposed edge will have a different probability and different contents, as a result of combining with other edges. A less stringent test than *member* was required, which allowed for variation in the 'probability' and 'found' elements of the edge. The test was re-implemented using the POP11 pattern matcher, which allows the specification of a partial pattern within a list. While this test succeeds, it does degrade the efficiency of the program. The member test checks for a specific pattern, whereas the pattern matcher requires several permutations to be checked before it can either fail or succeed. With long lists of edges on the agenda and chart, this is a necessary but significant drawback.

The solutions produced with the amended parser still display the grammar's ability to combine its many terminal and non-terminal categories in an amazing variety of ways. It appears that virtually every unit can fill every element of structure, creating a massive permutation in parse trees, as Figure 24 illustrates. This is exacerbated when parsing longer sentences of three or four words. These do not constitute lengthy sentences in a written corpus, (for example, in the LOB corpus the average sentence length is around 25 words), but are more normal in conversational dialogue (in POW the average sentence length is only around 6 words).

On inspecting the chart after parsing had finished, it also became clear that many inappropriate edges were being added for such a short sentence. Edges (rules) which require more than one daughter could never be successfully completed in the chart, since it is a feature of SFG that each

terminal category must be expounded by a word[2]. Unlike GPSG, there are no empty categories marking the place where a 'moved' constituent would normally be placed. As a consequence, it is pointless adding an edge which has more required daughters than there are remaining in the sentence. This change was made to the parser, and increases the speed quite markedly for very short sentences of fewer than four words, but will have a less marked effect on longer sentences.

4.2 An Improved Parsing Algorithm: Version 6.

Having ensured that the prototype probabilistic chart parser worked, albeit slowly, and having ironed out a few difficulties I was then in a position to make the more important changes which would permit the introduction of the reduced grammar. These changes can be divided into two categories: those for rules containing optional daughters, and those for rules with potentially co-ordinated daughters.

4.2.1 Incorporating Rules with Optional Daughters.

The first change to the program which was necessary to permit the inclusion of rules with optional daughters was to the way the grammar property table was created. Pocock had written a program *property.p* which took a probabilistic context-free grammar as input, and produced a list structure suitable for loading into POP11 as a property. He used properties as an efficiency measure, since they enable sub-parts of the grammar (those with a particular category label on the first daughter) to be accessed without needing to search through all the other rules. To enable a property list to work correctly for the reduced grammar, we had to account for the fact that the first daughter might be optional. Consider the rule

[-1378 QQGP [T] AX [FI]].

This needs to be added not only to the list of rules with *T* as their first daughter in the form

[-1378 QQGP T AX [FI]]

but also to the list for *AX*, in the form

[-1378 QQGP AX [FI]]

---

[2] More precisely, by a morpheme. The COMMUNAL generator produces separate morphemes, which are later combined into wordforms. In the POW corpus, each leaf node must consist of a word or an ellipted item. But in the syntax model extracted from EPOW, ellipted items are ignored.

since the temperer is optional. It might appear that this mechanism will undo all the good work we have done in reducing the grammar in the first place, and to a certain extent it will (when a rule has an optional first daughter). But the options will still remain in the rest of the daughters. In practice owing to the technique adopted for reducing the rules, relatively few of them begin with an optional daughter.

This mechanism has been implemented and the program *property.p* revised. When parsing, edges can now be created from the grammar rules with optional daughters in the list of those to be found. This has two knock-on effects with respect to the fundamental rule of parsing:

1. An active edge with a first optional daughter in its 'needed' list is really looking for two kinds of inactive edges; those labelled with the optional daughter as their category, and those labelled with the optional daughter's right sister. In the case of repeated optional daughters, even more inactive edges need to be considered, up to and including the first obligatory daughter if there is one.
2. An edge which contains only optional daughters in its 'needed' list must be considered as potentially inactive. It should be added to the chart with its optional daughters intact, but if combined with an active edge, these will be ignored.

These two amendments result in the blurring of the distinction between active and inactive edges when they contain only optional daughters and care must be taken not to add them to the chart twice, once as an inactive and once as an active edge. The chart parser algorithm was modified to take account of optional daughters in the fashion just described.

4.2.2 Incorporating Rules with Co-ordinating Daughters.

The fundamental rule of chart parsing also needed to be amended to handle the structure of rules for co-ordinated daughters. In section 3.1, I distinguished filling rules (which allow co-ordination and subordination) from componence rules (which do not). Filling rules are identified by a different data structure in the reduced grammar; their probability element is contained in a list rather than being atomic. A conditional check on the form of the probability element in an edge leads to the rules being treated differently. Componence rules are dealt with as in the prototype parser. Filling rules will have the structure

[[-22157 -63306 -77169] A CL],

for example. This rule will be accessed normally from the grammar. Once it becomes an edge, it can combine with a clause (CL) to form an inactive adjunct (A) edge. Whenever it combines in this fashion, *two* new edges are proposed. The first is a normal inactive edge with an atomic probability element, but the second is an active edge which has a probability list, has found one clause, but still needs to find another, i.e.. a new daughter is added to the 'needed' list. This process can go on indefinitely when combining with an inactive clause edge until the end of the sentence is reached, when the lack of words left in the sentence will prevent the new active edge from being created. As the number of co-ordinated daughters increases, their probability is degraded by the following function:

$$P(\text{mother with } n+1 \text{ daughters}) = (P(\text{mother with } n \text{ daughters}) - 120000/2^{n}) + P(\text{daughter } n)$$

which is an estimated (smoothed) function approximating to the observed probability degradation (presented in section 3.1.6). The function is initialised with the observed probability of only one daughter. It can be adjusted to increase or decrease the likelihood of co-ordination. A minor adjustment was also necessary in the ordering of the agenda to check if an edge represented a filling or a componence rule before accessing its probability.

With the two changes in place, the new version of the weight-driven chart parser (version 6) was ready to test. A few trial runs again on short sentences were performed as a comparison with the prototype. The results for the parsing of the utterance "yes" are again presented in Figure 25. This time parsing concluded, and a full set of solutions can be shown.

Figure 25. Trivial Output from a Weight-Driven Chart Parser (version 6).

```
Type words to parse in lower case except for "I" and proper nouns :
-> : yes
Starting parsing process - Version 6
Lexical lookup results:
yes
[[-6931 F] [-6931 H]]

[-22496 0 1 Z [[CL [F yes]]] []]
[-36080 0 1 Z [[CL [AL [CL [F yes]]]]] []]
[-49164 0 1 Z [[CL [C [NGP [H yes]]]]] []]
[-56287 0 1 Z [[CL [S [NGP [H yes]]]]] []]
[-62713 0 1 Z [[CL [C [CL [F yes]]]]] []]
[-66611 0 1 Z [[CL [C [PGP [CV [NGP [H yes]]]]]]] []]
[-73512 0 1 Z [[CL [S [NGP [H yes]]]]] []]
[-78122 0 1 Z [[CL [F yes]]] []]
```

[-78855 0 1 Z [[CL [A [PGP [CV [NGP [H yes]]]]]]] []]
[-80163 0 1 Z [[CL [V [NGP [H yes]]]]] []]
[-84201 0 1 Z [[CL [A [PGP [P [QQGP [AX [NGP [H yes]]]]]]]] []]
[-84291 0 1 Z [[CL [A [QQGP [AX [NGP [H yes]]]]]]] []]
[-86341 0 1 Z [[CL [A [CL [F yes]]]]] []]
[-89886 0 1 Z [[CL [AF [CL [F yes]]]]] []]
[-91541 0 1 Z [[CL [V [NGP [H yes]]]]] []]
[-91809 0 1 Z [[CL [AWH [QQGP [AX [NGP [H yes]]]]]]] []]
[-94036 0 1 Z [[CL [CWH [NGP [H yes]]]]] []]
[-94652 0 1 Z [[CL [A [NGP [H yes]]]]] []]
[-99724 0 1 Z [[CL [SWH [NGP [H yes]]]]] []]
[-99925 0 1 Z [[CL [A [CL [AL [CL [F yes]]]]]]] []]
[-101567 0 1 Z [[CL [F [QQGP [AX [NGP [H yes]]]]]]] []]
[-103002 0 1 Z [[CL [A [NGP [DD [GC [PS [NGP [H yes]]]]]]]] []]
[-103222 0 1 Z [[CL [A [PGP [CV [NGP [H yes]]]]]]] []]
[-103811 0 1 Z [[CL [ATG [CL [F yes]]]]] []]
[-107966 0 1 Z [[CL [AM [CL [F yes]]]]] []]
[-108568 0 1 Z [[CL [A [PGP [P [QQGP [AX [NGP [H yes]]]]]]]] []]
[-108658 0 1 Z [[CL [A [QQGP [AX [NGP [H yes]]]]]]] []]
[-108817 0 1 Z [[CL [A [NGP [MO [QQGP [AX [NGP [H yes]]]]]]]] []]
[-108920 0 1 Z [[CL [AP [CL [F yes]]]]] []]
[-110708 0 1 Z [[CL [A [CL [F yes]]]]] []]
[-111012 0 1 Z [[CL [AI [QQGP [AX [NGP [H yes]]]]]]] []]
[-111093 0 1 Z [[CL [CWH [QQGP [AX [NGP [H yes]]]]]]] []]
[-119019 0 1 Z [[CL [A [NGP [H yes]]]]] []]
[-119910 0 1 Z [[CL [AF [QQGP [AX [NGP [H yes]]]]]]] []]
[-124292 0 1 Z [[CL [A [CL [AL [CL [F yes]]]]]]] []]
[-124348 0 1 Z [[CL [CANTIC [NGP [H yes]]]]] []]
[-125164 0 1 Z [[CL [AM [QQGP [AX [NGP [H yes]]]]]]] []]
[-125569 0 1 Z [[CL [SANTIC [NGP [H yes]]]]] []]
[-127369 0 1 Z [[CL [A [NGP [DD [GC [PS [NGP [H yes]]]]]]]] []]
[-128480 0 1 Z [[CL [AWH [NGP [H yes]]]]] []]
[-128562 0 1 Z [[CL [CWH [CL [F yes]]]]] []]
[-130471 0 1 Z [[CL [ALWH [QQGP [AX [NGP [H yes]]]]]]] []]
[-130471 0 1 Z [[CL [AI [QQGP [AX [NGP [H yes]]]]]]] []]
[-130937 0 1 Z [[CL [AWH [PGP [CV [NGP [H yes]]]]]]] []]
[-133184 0 1 Z [[CL [A [NGP [MO [QQGP [AX [NGP [H yes]]]]]]]] []]
[-136283 0 1 Z [[CL [AWH [PGP [P [QQGP [AX [NGP [H yes]]]]]]]] []]
[-136830 0 1 Z [[CL [AWH [NGP [DD [GC [PS [NGP [H yes]]]]]]]] []]
[-141747 0 1 Z [[CL [CWH [PGP [CV [NGP [H yes]]]]]]] []]
[-142146 0 1 Z [[CL [CWH [CL [AL [CL [F yes]]]]]]] []]
[-142645 0 1 Z [[CL [AWH [NGP [MO [QQGP [AX [NGP [H yes]]]]]]]] []]
[-147093 0 1 Z [[CL [CWH [PGP [P [QQGP [AX [NGP [H yes]]]]]]]] []]
[-147421 0 1 Z [[CL [SANTIC [CL [F yes]]]]] []]
[-155230 0 1 Z [[CL [CWH [CL [C [NGP [H yes]]]]]]] []]
[-161005 0 1 Z [[CL [SANTIC [CL [AL [CL [F yes]]]]]]] []]
[-168405 0 1 Z [[CL [A [TEXT [Z [CL [F yes]]]]]]] []]
[-174089 0 1 Z [[CL [SANTIC [CL [C [NGP [H yes]]]]]]] []]

Number of edges in chart = 572
Number of inactive edges = 409
Number of almost inactive edges = 0
Number of active edges = 236
Number of edges in output file = 572
Number of solution parse trees : 44
E = edge number and W = weight
Time to execute (variable TIME) = 244.05
Number chart edges (variable chart_edges) = 572

The grammar licenses 44 parses out of 572 edges in the chart. Note that the number of inactive edges plus the number of active edges (645) is greater than the total number of edges, which illustrates the extent to which optional rules have been treated as both active and inactive. In a small number of cases (four times here), the parser produces duplicate parses with different probabilities. This results from the reduced grammar containing two rules for a mother rewriting as the same daughter, surrounded by different optional sisters. Ideally this should be rectified by amending the componence rule reduction program, and reducing the grammar again. Since this program is highly recursive and takes several days to run on the clause grammar, this has not yet been done. A temporary *ad hoc* solution is simply to delete the second version of the parse from the final solutions list.

The solutions are in a slightly different order than the prototype parser, which is a result of the combination of probabilities in rules with optional edges. The prototype solutions which contained recursion have not been produced, but the number of parses is still high. 17 solutions are found (including two duplicates) for the formula label, and 27 for the head noun label (including two duplicates). Some of these analyses are not acceptable in SFG terms, but are licensed by the (over-generative) context-free version of the grammar. For example, the parse

[-94036 0 1 Z [[CL [CWH [NGP [H yes]]]]] []]

should not be accepted since the element of structure *wh*-complement (CWH) should contain a *wh*-head (HWH), such as *what*. The dependency between the two will be captured by version 7 of the parser, through the use of the vertical trigram model. This feature alone accounts for 15 unwanted parses. A similar lack of dependency exists between adjunct labels (of which there are many; *A, AF, AL, AWH, ATG, AM, AP, AI, ALWH*) and the words they are eventually expounded by in these solutions, but this is not captured by the vertical trigram model. The lexical look-up facility would need to recognise the likelihood of a particular lexical item having a particular tag (and great-grandmother in these cases) to adequately handle the relationship between these adjuncts and their eventual exponent lexical items. The combination of the vertical trigram model and the Brill tagger achieves this, in the case of *wh*-variants of heads and apexes, for example.

None of the solutions produced can be strictly described as parsing errors, since they are all legitimate combinations of lexical items and context-free rules. The ordering is important

therefore in selecting one or more parses as 'desired' solutions with respect to the overall SFG description. Using linguistic intuition and knowledge of the POW corpus grammar, I would expect to find at least the following parses, given the lexical tags assigned to the word *yes*:

[-22496 0 1 Z [[CL [F yes]]] []]
[-49164 0 1 Z [[CL [C [NGP [H yes]]]]] []]
[-56287 0 1 Z [[CL [S [NGP [H yes]]]]] []]
[-80163 0 1 Z [[CL [V [NGP [H yes]]]]] []]

These are at least all found within the first ten parses, (eight if you exclude the duplicate parses), which provides some evidence that the probabilities are playing the selective role we would hope.

The next version incorporates the improvements we have just discussed (and some others).

4.3 Version 7: Combining Context-Free and Vertical-Trigram Rules.

Version 7 of the parser uses the Brill-tagger trained on the POW corpus, which serves to limit some of the ambiguity found in version 6 by only adding one tag per word to the chart. As we shall see in chapter 5, it also slightly increases the lexical tagging success rate, but still results in several tagging errors.

4.3.1 The Vertical Trigram Model.

The biggest advance in terms of parsing accuracy is gained by combining vertical dependency information in the corpus with the context-free rules. This has been experimented with by Magerman and Marcus (1991) in their Pearl chart parser. They incorporated daughter-grandmother relationships into the syntax model, giving the formalism more context-sensitive power, thereby improving the accuracy of their parser. The success comes at a price of increased size of syntactic model, as discussed in section 3.1. The vertical trigram model has been incorporated into the working memory of the parser, (as increases in the hardware have allowed),

using a further property list for efficient access. Resultant parsing process sizes are larger than for version 6.

What we are trying to capture in combining a probabilistic context-free grammar with a probabilistic vertical dependency grammar is the most likely subtree associated with a particular lexical item or grammatical category. Our implementation of chart parsing then begins to approach the subtree combination method of Bod (1995), combining the most likely subtrees for each word/wordtag pair, by identifying overlapping structures. Unlike O'Donoghue's work, which was restricted in depth to that observed in his artificial corpus of generator output, if we store a grammar of vertical dependencies smaller than the distance from word tag to root category, but greater than between just two nodes (mother and daughter), we will be able to permit a constrained level of vertical recursion. Such constraints limit the mutual recursion of adjuncts, for example, while still allowing for deeper trees than those observed in the corpus.

The vertical trigram model is incorporated into the chart parsing algorithm as follows. As a new edge is proposed, an additional check is performed before it is added to the agenda. This involves looking up all the relationships between the label of the active edge (the grandmother), the label of the inactive edge (the mother) and each of the possible daughters in the *found* element of the inactive edge. If any of the proposed vertical trigrams are absent from the corpus-trained model, the proposed edge is rejected. Otherwise, the product of their combined probabilities is multiplied by the weights of the component active and inactive edges, to form the weight of the new edge[3].

4.3.2 Limiting Tree Depth.

In order to further constrain the over-generation shown in version 6 of the parser, I have also introduced a factor limiting the depth of solution trees to that observed in the corpus for trees of a comparable breadth (counted in words). This involves recording the depth of each edge in the chart (a seventh element in the data structure for the edge). When new edges are proposed, if the combined depth of the over-arching edge exceeds that observed for a similar breadth in the

[3] In the present implementation of version 7, the vertical weight is moderated by dividing it by 1.1, which has been found by experiment to achieve an appropriate balance (for SF syntax) between breadth first and depth first search, still within an overall most-likely first strategy.

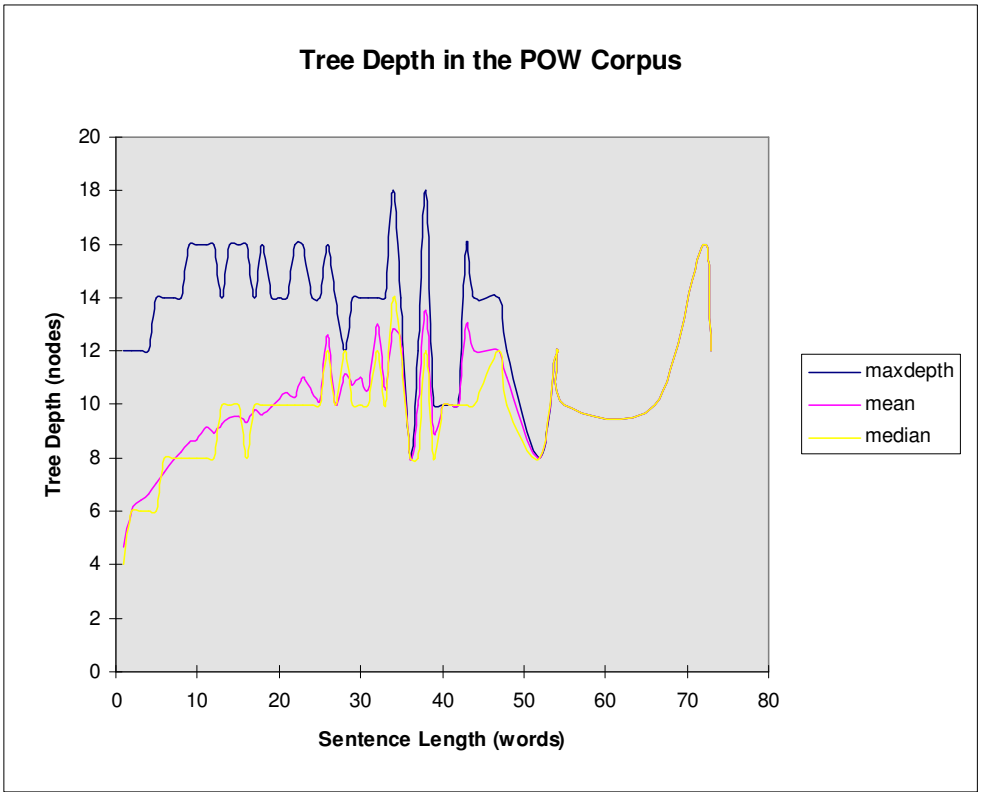
corpus, the edge is prohibited. This affects both sub-trees and overall trees. The function defining the depth limit is derived from the corpus. Figure 26 shows the empirical evidence for (sub)tree in the POW corpus.

The depth limit has been set to the maximum observed in the corpus, but could be more or less. If we wish to improve parsing speed, we can eliminate very deep trees, which are quite rare, by setting the function to be nearer the median rather than the maximum observed depth. Note that tree depth in POW tends to jump in multiples of two nodes, in line with the alternation between labels elements of structure and units. The combination of the depth constraint and the vertical trigram model is still more general than O'Donoghue's vertical strip grammar, but is similar in that it is tied to the depth observed in the corpus (in the current implementation).

4.3.3 Stopping the Parser.

Version 6 of the parser illustrated that, when chart parsing with a very large syntax model, the parser can sometimes run and run. Three (variable) measures have been implemented to stop the parser after (what I consider) a reasonable time. Firstly, there is an arbitrary limit (currently of six) set on the number of parses to be produced. This will illustrate the extent to which the probabilistic method results in legitimate parses being found first. Secondly, the weight of the first solution

Figure 26.



produced is recorded, and no new edges are generated with a weight currently set to be 1.5 times that of the first solution's weight. Finally, in the absence of any solutions being found, parsing stops when the agenda reaches a certain size, currently set to be (50,000 x sentence length) edges.

The results of formal testing of version 6 and 7 of the probabilistic chart parser are now presented in chapter 5.

Chapter 5. Parser Testing and Evaluation.

This chapter will report test results for two parsing methods: Firstly, a prototype parser using CELEX lexical look-up, and a corpus-based context-free syntax. Secondly, a more successful parser, using the Brill tagger for lexical annotation, and an amended chart parsing algorithm which uses a context-sensitive syntax model incorporating vertical trigrams. The results will be evaluated and compared. First of all, we will consider the selection of test data.

5.1 The Test Data.

Preliminary testing material for a corpus-based parser should in the first instance come from the corpus itself. One might assume that a parser should at least handle the structures in its training material. In the case of the CELEX-SFG lexicon though, the parser is using a lexicon not derived from the corpus, so lexical tagging will not necessarily be 100% accurate. In the second case, the Brill tagger, which *was* trained on the EPOW corpus, achieves an accuracy rate of between 90-95% of word tokens correctly tagged. Furthermore, it may be that the manual annotation chosen for a sentence by the corpus annotators is not the only legitimate SF syntax analysis, and perhaps not the most likely (based purely on syntactic criteria, rather than using other semantic and contextual cues found in the corpus), so we may reasonably expect some different solutions to those found in the corpus to be produced by a purely syntactic parser. Parsers employing very large grammars often find tens, or even hundreds of possible solutions, and it may take several days to allow the parser to run and run until all solutions are produced . In the first test described in section 5.2, the parser was interrupted manually after the first few solutions has been found (or my patience ran out). In the second test (section 5.3), for efficiency reasons, an arbitrary limit has been set on the maximum number of analyses (currently six) the parser may produce. It will be an interesting test of the value of probabilistic parsing to discover how often the POW analysis is found in the first few parses. Therefore parsing a sample of the POW corpus itself is not a purely trivial task, because of tagging errors, and because of time and computing resource limits.

Other corpora of a different genre to that which provided the training material should ideally also be used as secondary testing material, to give an objective assessment of the parser's success. A half-way house strategy adopted by some (eg. Sharman 1990, Pocock and Atwell 1993) is to deliberately set aside a fraction of the corpus from training, and use this in testing.

This is more difficult than testing on the training material, but not as stringent as parsing new genres.

In this experiment, I have used the whole of the EPOW corpus from which to extract the syntactic model, in order to maximise the subset of English it covers and to enhance the reliability of the probabilistic part of the model. Therefore, provided the lexical tagging works correctly, the syntactic model is guaranteed to contain the required rules to produce solutions for test sentences from the POW corpus, but not for those outside the corpus. It is useful to use corpus sentences as test data, since we have a hand-annotation to act as the target solution the parser should produce, (among others, if the utterance is syntactically ambiguous). For non-corpus-based test sentences, we have no such target, and must instead rely on 'expert' knowledge of the syntactic description used in the corpus. In the second parsing test, in which the corpus-based test material has been supplemented with some from outside the corpus, I have used my own judgment to gauge the correctness of the solutions.

The sample of the POW corpus given in Figure 1 (chapter 1) was used as the first test material for the parser. The sample consisted of 17 utterances[1], shown in Figure 27a. This sample was chosen arbitrarily from the material provided by twelve year olds, and consists of part of one of the Lego play sessions (corpus file '12abpspg'). It may be argued that such material is less likely to contain grammatical errors (than that of the six-year olds, for example), but equally, it may contain greater syntactic variety.

A further small set of six sentences taken from those used by the Limerick International Workshop on Industrial Parsing of Software Manuals (Koch and Sutcliffe 1995) were used as examples of data on which the parser had not been trained (Figure 27b). Although at least sixty sentences were used as test data at the workshop, a core of just six were chosen for a detailed comparative study of the results, and it is these which have been used here. The genre is entirely different from the POW corpus, being written formal material taken from software manuals. Of particular note is the difference in utterance length, which, as we shall see, seriously affects parser performance. These 'untrained' sentences were modified slightly to remove their sentence-level punctuation (commas, full-stops, question marks etc.), since, although the Brill tagger can handle these, there are no relevant rules in the syntax model for the parser to use to incorporate them into the tree structure. (Being a spoken corpus, the transcription contains no such punctuation - it was not deemed useful to insert some

[1] 19 utterances if we include no's. 11, 11a and 11b as separate parsing tests.

artificially. In written corpora containing such punctuation marks, parsing can actually be ameliorated, since they often serve to delimit phrases.) Only the trained test sample was used for testing the context-free prototype parser, whereas both were used for the context-sensitive parser.

Figure 27a. The Trained Test Sample.

1. why
2. what ‘s the point
3. you put these on for windows
4. you don’t have to
5. won’t be long
6. it ‘s easiest mind
7. I know something easy
8. build a garage
9. fantastic
10. or something like a skyscraper
11. this worked out it won’t fit

11a. this worked out

11b. it won’t fit

12. go on
13. we can always move it along can’t we
14. will that one fit in by there
15. come on
16. let’s get going
17. I can’t even

Figure 27b. The Untrained Test Sample.

18. press SHIFT+INS or CTRL+V
19. what do we mean by this
20. select the text you want to protect
21. that is these words make the source sentence longer or shorter than the TM sentence
22. displays the records that have a specific word or words in the TITLE CONTENTS SUBJECT or SERIES fields of the BIB record depending on which fields have been included in each index
23. enter the line number of the alphabetical title search option

5.2 Test 1: Prototype context-free parser with CELEX look-up.

The first trial was conducted with a version of the probabilistic chart parser which I have called version 6. (The code which I adopted from Pocock and Atwell being version 5). The components of this parser are:

(1). The CELEX English lexicon, modified to match the systemic functional syntax labels in the POW corpus, with a rapid look-up program supplying multiple parts-of-speech in the case of lexically ambiguous items.

(2). A probabilistic context-free systemic-functional syntax model extracted from the whole of the POW corpus, in which the simple context-free rules have been modified (as described in section 3.1) to account for optionality, co-ordination and subordination.
(3). A probabilistic chart parser whose search strategy is 'most likely first', and which has been adapted to elegantly handle the modified context-free syntax in (2).

The first test was conducted on a Sun SPARCstation 2[2] running UNIX and POPLOG POP11. In virtually every case, the parser had to be terminated prematurely, and in some cases before any solutions had been found. Performance on sentences of five words and over was so slow that no solutions were produced within three days. Consequently, after trying to parse several such sentences, and having to prematurely interrupt the process each time, I divided the two-clause utterances in 7/8, 13/14 and 16/17 into separate trials for the parser. (Some of the utterances in the corpus consist of two or more 'co-ordinated' clauses, even in the absence of a co-ordinating lexical item, but have not been segmented separately in the transcribed version, perhaps owing to intonational cues). Even when the parser produces solutions for the shorter single-clause input, it continues to run for several hours, so I interrupted such processes after the first few solutions had been output. As a consequence, it is difficult to classify the output in traditional terms, such as number of solutions produced, since many more could have been produced given a faster machine and plenty of time. However, we can compare the first few parses with those in the corpus, and see if they are the same, or even remotely similar. The results are summarised in Figure 28.

These results are very poor. Only two out of 17 utterances were correctly parsed, and in only one of these cases was the correct parse the first solution to be found. (If the EPOW analysis differs from the one found by the parser only in its inclusion of ellipted elements, then such parses were deemed to be correct). Five of the longer utterances failed to be assigned a single parse before they were interrupted. Of the remainder, many parses were found, and it was usually the first solution which was the nearest to the desired solution (shown in brackets in column five). Nearness can be indicated by the overlap in tree structure and category labels. Usually, these near misses were the result of lexical look up problems, where the desired tag was never found in the CELEX lexicon. The failure of this parser is now analysed in more detail.

---

[2] The first test was performed on the School of Computer Studies machine csparc16, a 40 MHz SPARCstation 2, with 32 Mb. of main memory, executing 28.5 MIPS.

Figure 28. Test 1: Results.

Utterance number	*Parsing completed*	*Parses found*	*POW analysis found*	*Solution no*
1	Y	55	Y	12
2	N	5	N	(2)
3	N	0	N	-
4	N	2	N	(1)
5	N	6	N	(1/5)
6	N	7	N	(4)
7	N	2	N	(1)
8	N	3	Y	1
9	Y	44	N	(1)
10	N	0	N	-
11	N	0	N	-
12	Y	6	N	(1)
13	N	0	N	-
14	N	0	N	-
15	N	15	N	(1)
16	N	2	N	(1)
17	N	7	N	(1)

5.2.1 Lexical Look-Up.

Lexical failure was responsible for several parsing failures (9 of the 17 utterances contained a single word which was not assigned the correct tag, and in one utterance two words failed to be assigned the correct tag, although one of these was because the word *one* had been incorrectly tagged in EPOW). A total of 66 words were contained in the test utterances, of which 55 were correctly tagged (83.3%). The words which the lexicon failed to tag correctly along with the desired tag were:

Lexical Item	*Tag(s) found in CELEX*	*POW tag*
in	P, PM	AX
on	P, PM	AX (3 times)
out	PM, M	AX
why	AX, EX	AXWH
always	AX	AI
mind	H, M	AF
fantastic	AX	EX
one	H, M, AX	HP
even	T, AX, H	AI

have-to	OX, M, X / AX, EX, PM, P	XM
by-there	PM, P / AX, F, STH	AX

In the case of the last two items in this list. it would have been unreasonable to expect the CELEX lexicon to reproduce POW's hyphenated format, so the words were input separately. The failures can be classified as follows:

The prepositions *on, in* and *out* are labelled as apexes in POW when part of phrasal verbs:

[CL [M COME] [CM [QQGP [AX ON]]]].

This feature has yet to be incorporated into the lexical mapping from CELEX to SFG. It would effectively mean duplicating all prepositional entries with the new *AX* tag, since it is difficult to specify those which can occur in phrasal verbs. A more reasonable solution would be to incorporate a lexicon of phrasal verbs (eg. Cowie et al 1975), and allow multi-word idiom tags and lexical edges. See section 6.1.3 for further explanation.

Where a fine-grained grammatical distinction has yet to be incorporated into the lexicon, such as for the various SFG adjuncts, or for *wh* versions of an element of structure, then a solution which contained the less delicate description was accepted as correct. For instance *AX* is considered a match for *AXWH*, since the mapped CELEX lexicon does not yet provide the *AXWH* label.

The category label *EX* is assigned to the word *fantastic* in the corpus. It is difficult to specify an exhaustive list of the lexical items that can be used in exclamations, so only the most frequent have been included (by mapping the CELEX label *I*: interjection). The *AX* label is the correct label within the POW corpus for an adjective, but here the word *fantastic* is functioning as an exclamation.

The problem of the multiplicity of (normally non-terminal) adjunct labels (such as *AI* and *AF*) being used directly as terminal categories has already been identified (see section 3.2.6), and two of the 'errors' can be attributed to my solution; labelling these as apexes of quantity-quality groups which can fill a non-terminal adjunct, i.e. treating them as productive constituents.

The first parsing test, therefore, shows that there are still some improvements to be made to the lexical mapping from CELEX to SFG syntactic categories. The difficult problems associated with the lexical labelling of adjuncts, exclamations, phrasal verb particles, and temperers (among others) mean that completely watertight lexical look-up has not been achieved. It is precisely by running such parsing tests that lexical deficiencies in the hand-crafted mapping algorithm from CELEX labels to the POW corpus SFG can be discovered and rectified. Some deficiencies could be resolved by further lexical entries being added to CELEX (eg. for phrasal verb particles and terminal adjunct categories), others would involve fine-tuning the mapping (eg. wh-apexes) and others would require the parsing program to be adjusted to change lexical labelling 'on the fly' (temperers, adjectival verb participles). What is surprising about these results is not the fact that some errors have occurred, but rather that there were relatively few of them. In many cases, CELEX look-up introduced more than one tag into the chart for each lexical item, which will of course have made the parsing problem more perplex. However, the second parsing test will show that using the Brill tagger for lexical 'look-up' increases the success rate from 83% to over 90% of word tokens in the test data, and has the advantage of selecting only one tag per lexical item.

5.2.2 Limitations of the syntax formalism.

Since in all but two cases the parser had to be terminated prematurely due to the size of the search space, it is difficult to classify the performance of the syntactic model. The context-free formalism is responsible for the overgeneration described in section 4.3.2, which is displayed again in many of the test solutions. Context-sensitive constraints need to be added to the context-free rules to capture knowledge of likely grandmothers, as well as mothers of a daughter (and in some cases even more distant relationships between nodes), for instance in the possible subconstituents of the various adjunct labels. Currently, typical parses for adjuncts are duplicated for each particular adjunct label (*A, AA, AD, AF, AI AL, ALWH, AM, AML, AN, AP, ATG, AWH*). The overgeneration is exacerbated by the possibility of an adjunct being filled by a clause, which itself contains just an adjunct. The vertical recursion of the same adjunct is prevented by the parser, but the successive alternation of each adjunct label sandwiching a clause label is not. Hence the current syntax licenses the following structure:

...[CL [A [CL [AL [CL [...

Given there are thirteen such adjunct labels in the syntactic description used in POW, this could potentially result in 8192 different adjunct/clause structure combinations being

proposed each time a clause label is added to the chart. In the corpus however, only seven of the adjuncts occur being filled with a clause, so the combinatorial explosion is limited to a mere 128 possibilities. But this feature of the syntactic formalism means that every time a clause edge is generated, the explosion takes place. Similar combinations recur for other elements of structure and units, which helps to explain the non-termination of parsing within a reasonable period (the writer's patience limit).

To illustrate the size of the explosion problem, in Figure 29 I present some data taken from the chart and agenda when a six-word utterance was parsed, and the process halted after four days. The utterance, *this worked out it won't fit*, should have contained two clauses (one might also argue that this could really have been divided into two separate three-word utterances by the corpus analysers).

Figure 29. Data from a Six-Word Parse.

```
listlength(agenda)==>
** 157188
agenda(1)==>
** [[-134326] 0 2 MOTH [[NGP [MO [QQGP [AX [NGP [DD this]]] [FI [CL [M worked]]]]]]] [NGP]]
agenda(157188)==>
** [[-298506]1 4 AM [[CL [C [CL [M worked]]]] [CL [M out] [A [CL [AL [CL [F [QQGP [AX [NGP
[HP it]]]]]]]]]] [CL]]
listlength(chart(0,1))=>
** 3651
listlength(chart(0,2))=>
** 826
listlength(chart(0,3))=>
** 213
listlength(chart(0,4))=>
** 110
listlength(chart(0,5))=>
** 0
listlength(chart(0,6))=>
** 0
listlength(chart(1,6))=>
** 0
listlength(chart(2,6))=>
** 13
listlength(chart(1,5))=>
** 0
chart(0,3)==>
...[-91517 0 3 Z [[CL [S [NGP [DD this]]] [M worked] [C [PGP [PM out]]]]] []]...
chart(3,6)==>
...[-81469 3 6 Z [[CL [S [NGP [HP it]]] [OMN won t] [M fit]]] []]...
vars prob;
chart(0,3) matches [== [?prob 0 3 Z [[CL ==] ==] [CL]] ==]=>
** <false>
agenda matches [== [?prob 0 3 Z [[CL ==] ==] [CL]] ==]=>
** <true>
prob=>
** [-151517]
```

We see from Figure 29 that the agenda has reached an enormous size (157,188 edges) with logarithmic probabilities ranging from -134326 for a two-word edge at the start of the

agenda, to -298506 for a two-word edge at the end of the agenda. No spanning edges have been found, and the longest edges cover only four words. However, among the three-word spanning edges are those which correctly parse the two clauses. In fact, these clause edges were the first to be found for the span from position 0 to 3 and from 3 to 6. I checked to see if the active clause co-ordinating rule had been activated, and it had, and had already combined with the first clause (*this worked out*), with a logarithmic probability of -151517. This active edge still looking for an inactive matching clause is placed well below the top of the agenda (the top edge has a logarithmic probability -134326), so will take some time to reach. Once it has found the inactive second clause, they will combine to form a new spanning edge with a probability of -78784 + -151517 = -230301, which will be added to the agenda. Only when this spanning edge has reached the top of the agenda, will the solution be produced. It seems likely, given the speed with which the two clauses were found, that this will also be the first solution that would be produced. The prognosis for this unconstrained chart parser is therefore not good. Longer single clause utterances would take much longer to parse, and only by dividing multi-clause utterances into separate parses could we hope to produce a solution. The syntactic formalism is working as it should, but the opportunity for combinatorial explosion because of mutually recursive rules is far too great.

5.2.3 Near Misses.

For the shorter sentences which were not correctly parsed, I made a judgment as to which of the solutions produced was the best-fit to the desired parse in the corpus. For example Figure 30 shows the desired and best-found solutions for three of the near misses. The parser solutions are annotated with the ranking amongst other solutions.

Figure 30. Near Misses.

i) ***what 's the point***

POW solution:

[Z [CL [CWH [NGP [HWH WHAT]]] [OM 'S] [S [NGP [DD THE] [H POINT]]]]]

Parser solution (2):

[-107832 0 4 Z [[CL [S [NGP [HWH what]]] [OM 's] [C [NGP [DD the] [H point]]]]]

ii) ***you don't have to***

POW solution:

[Z [CL [S [NGP [HP YOU]]] [ON DON'] [XM HAVE-TO] [(M)] [(C)] [(CM)]]]

Parser solution (1):

[-92454 0 4 Z [[CL [S [NGP [HP you]]] [ON don t] [M have] [C [NGP [MO [QQGP [AX to]]]]]]]

iii) ***won't be long***

POW solution:

[Z [CL [(S)] [OMN WON'T] [M BE] [C [QQGP [AX LONG]]]]]

Parser solution (1):

[-94797 0 3 Z [[CL [OMN won t] [M be] [C [NGP [MO [QQGP [AX long]]]]]]]

Example i) shows how lack of dependency in a context-free syntax between daughters and their grandmothers causes overgeneration. The *HWH* tag should be dominated by a *wh*-grandmother, such as *SWH* or *CWH*. These will eventually be found by the parser, but they are less likely than their non- *wh* counterparts. Until the question structure information is found at the clause level, the clause rule reversing the *wh*- complement and subject will fail to be applied.

Example ii) is a near miss because of the hyphenation of the modal verb *have-to* in the POW SFG. In my CELEX lexicon, the modal is listed as two wordforms, which are looked up separately. The verb tag found is for a main verb, rather than a modal, and the infinitival *to* fails to be labelled as such. The resulting parse treats *have* as a main verb and *to* as a complement. This provides further justification for the need to incorporate collocations and idioms in the lexicon (see section 6.1.3).

Example iii) illustrates an occasion on which the combination of probabilities itself appears to select a more complex structure in favour of a simple one. The complement in a clause is more likely to be filled by a nominal group than by a quantity-quality group, so the single apex *long*
is classified as an adjectival modifier in a nominal group rather than as a predicative adjective.

Although I have been rather conservative in classifying such examples as failures, several are legitimate systemic-functional syntactic structures, so cannot be regarded as incorrect. The nearest solution to the desired one (in terms of matching categories and structure) was frequently the first one produced, as Figure 28 column 5 shows, so the probabilistic method should not immediately be rejected. The real message from such examples is that there is a need to compensate edges which contain words so that they can be activated equally or in preference to 'wordless' edges created from the grammar.

In the probabilistic model used in test one, the weight of a tree is calculated as the product of the weights of each of its constituent edges. A spanning edge containing, say, six words would have a combined probability derived by multiplying together probabilities of all its many constituent edges. Such a potential edge would be competing in the agenda with edges spanning only one or two words, which would be prioritised until they were extremely deep. It would be desirable to normalise the edges that span several words to allow them the chance to act ahead of the implausible very deep structures being built for single words. Indeed, some very deep trees and subtrees created in the chart (caused for example by mutual vertical recursion of adjuncts) are not well-formed according to the SF description used to annotate the corpus, but are well-formed with respect to an unmodified context-free formalism. A mechanism is needed to constrain vertical growth of trees to the limits observed in the language as it is performed, and to require that the vertical relationships between nodes are linguistically acceptable. A context-free formalism essentially controls horizontal ordering (linear precedence) relationships between potential sister nodes in a tree, as well as the immediate vertical dominance between a mother and its possible daughter constituents. However, the alternative vertical layering in SFG syntax trees between elements of structure and units would suggest a model which can capture relationships at least between grandmother, mother and daughter nodes. Intuitively, one can envisage this being needed to model the difference between clause structure for a matrix clause and a relative clause, for example.

O'Donoghue (1993) used vertical strips extracted from leaf node to root in his vertical strip parser, which limited parse trees to the depth found in his artificial corpus of NL generator output. Weerasinghe (1994; 86) uses a combined probabilistic model derived from the POW corpus and then mapped onto SF labels used in the GENESYS NL generator. His model includes lexical tag (exponence) probabilities, filling probabilities (a vertical bigram model) and element transition probabilities (akin to probabilistic linear precedence rules).

In the second parsing test, I will combine the context-free model (used in test one) with a vertical trigram model extracted from the EPOW corpus, which together act in a context-sensitive fashion to control both horizontal and vertical relationships in the solutions the parser finds. The use of additional vertical weights, on top of the probabilities of the context-free rules, tends to mitigate against deep thin trees. Whilst the search strategy is still most-likely first, a more breadth first pattern is encouraged, by penalising vertical growth without corresponding horizontal growth in the tree. This appears to be a feasible alternative to other ways of compensating wider spanning edges, such as taking the probability of a tree as a

geometric mean of its constituent node probabilities (Sampson et al 1989). There is as yet no generally accepted solution to this problem.

5.3 Test 2: Brill tagging and a context-sensitive chart parser.

The second trial was conducted with a version of the probabilistic chart parser which I have called version 7. The components of this parser are:

(1). Lexical look-up provided by the Brill tagger, trained on the EPOW corpus. This ensures that at most one tag is assigned to each lexical item. Since the tagger is trained on the corpus' lexical data, it can handle the hyphenated multi-word combinations POW contains. The training process also includes the learning of general morphological rules for English words, which allow prediction of word category for words not found in the corpus. (However, as we shall see, this is process does not yet yield perfect results.)

(2). A probabilistic context-free systemic-functional syntax model extracted from the whole of the EPOW corpus, in which the simple context-free rules have been modified (as described in section 3.1) to account for optionality, co-ordination and subordination.

(3). A probabilistic vertical trigram model, also extracted from the EPOW corpus.

(4). A probabilistic chart parser whose search strategy is 'most likely first', and which has been adapted to elegantly handle the modified context-free syntax in (2) and (3). In particular, whenever a non-terminal edge is being proposed (by activation of the fundamental rule of chart parsing), each of the vertical trigrams from the new spanning edge's mother label down to its constituent granddaughters is checked against the vertical trigram model. If any are not found in the corpus-extracted model, the proposed edge is rejected. If all the vertical trigrams are found to be valid, then the logarithms of their respective probabilities are added, and combined with the weights of the active and inactive edges in making the new spanning edge.

(5). A limiting factor place on the production of all edges in the chart, which prevents them growing deeper than the observed depth of (sub)tree found in the EPOW corpus. The factor varies according to the length of the (sub)sentence, i.e. according to the breadth of the tree, measured in leaf nodes. Once the first solution has been found by the parser, the maximum depth of any further solutions is capped to 4 nodes greater than the first solution (provided this is still less than the original maximum depth limit).

(6). A modified termination condition on parsing. The standard chart parsing algorithm terminates when the agenda is empty, and no more new edges can be proposed for the particular lexicon and syntax model. The current version 7 of the parser instead terminates when

a) six solutions have been produced, or
b) the weight of an edge at the top of the agenda is 1.5 times the weight of the first solution found, or
c) the number of edges on the agenda has grown to 50,000 times the length of the input in words.
Each of these terminating conditions has been set arbitrarily, after observation of the parser performance without them, and also with them set at different values.

The addition of the vertical trigram model (3) to the context-free syntax rules (2) effectively transforms the complexity of the model to be context-sensitive (a production is only activated when the mother is itself the daughter of a particular grandmother). It serves to limit (but not entirely prevent) the vertical recursion of, for example, adjuncts sandwiched by clauses, which were partially responsible for the very slow parsing times displayed in test one. It is more flexible than O'Donoghue's vertical strips from leaf to root, since a vertical trigram model theoretically allows deeper trees than those observed in the corpus, but these will be penalised by having very low probabilities. However, in the current implementation, a further vertical limiting factor (5) has been introduced, to prevent the growth of subtrees of greater depth than those found in the corpus. The vertical trigrams are comparable to (but more restrictive than) the vertical bigram model of filling used by Weerasinghe (1994; 87).

The second test was performed on the Sun general purpose server of the School of Computer Studies, running UNIX and POPLOG POP11[3]. The results are summarised in Figure 31, and Appendix 15 contains a detailed log of the parser output.

Figure 31. Test 2: Results.

Utterance	Words	Tags correct	Parsing completed	Parses found	POW analysis found	Good SF analysis found	Sol. no.	CPU time	Chart edges
1	1	1/1	Y	6	Y	Y	2	3.47	133
2	4	4/4	Y	6	Y	Y	3	3340.83	3637
3	6	6/6	N	5	Y	Y	5	7 days	19498

[3] A SPARCserver-1000E (csgps1), with main memory of 192 Mbytes, 2 x 60 MHz processors, running Unix Solaris 2.3, POPLOG version 14.5.

4	3	3/3	Y	1	Y	Y	1	107.32	846
5	3	3/3	Y	4	Y	Y	1	148.01	1026
6	4	3/4*	Y	5	N	Y	(5)	-	11766
7	4	4/4	Y	6	N	Y	(4)	6564.48	4361
8	3	3/3	Y	1	Y	Y	1	21.9	618
9	1	0/1	Y	4	N	Y	(1)	9.06	218
10	5	5/5	Y	6	N	Y	(1)	59180.1	10548
11	6	6/6	Y	6	N	Y	(3)	106244	17647
11a	3	3/3	Y	4	Y	Y	3	201.38	1322
11b	3	3/3	Y	1	Y	Y	1	26.72	616
12	2	1/2	Y	2	N	Y	(1)	2.74	161
13	8	7/8	N	0	N	N	-	3 days	19518
14	6	6/6	Y	6	Y	Y	4	7715.26	8074
15	2	1/2	Y	2	N	Y	(1)	2.5	161
16	3	2/3**	N	0	N	N	-	2 hrs	6577
17	3	3/3	Y	1	Y	Y	1	119.6	873
18	4	4/4	Y	6	-	Y	(2)	1986.51	3640
19	6	6/6	N	3	-	Y	(1)	16 hrs	12367
20	7	6/7	N	0	-	N	-	-	-
21	15	14/15	N	0	-	N	-	-	-
22	32	31/32	N	0	-	N	-	-	-
23	10	9/10	N	0	-	N	-	-	-

* The word *easiest* is incorrectly labelled *AX* in EPOW, so the tag assigned (*AXT*) was actually marked as correct.

** Once *get* has been incorrectly labelled *X*, it is impossible to find any solution given the corpus-based grammar, so parsing was manually terminated early.

These results are a lot more successful than those for the first test. If we first consider the 19 'trained' test sentences taken from the POW corpus, (treating the two-clause utterance in 11 as three separate tests as shown), on ten occasions the POW analysis is among the first six found, and in five of these cases the POW solution is the first one to be produced. Of the further nine cases where the exact POW solution was not found, seven produced syntactically acceptable trees, and three of these seven produced the best solution first. In only two of the nineteen cases, no solution was found before parsing was terminated, once (no. 16) because an incorrect tag made it impossible for a solution to be found using the current syntax model, and once (no. 13) because I manually terminated parsing of an eight word utterance after three and a half days before the termination conditions had been reached.

In the smaller test on the six sentences of 'untrained' material, the results are less positive. Parsing was completed in only two of the six cases, but both produced syntactically acceptable analyses. The failures are partly due to lexical tagging error, and partly to the length of the input. Each of the unparsed tests contained a tagging error which would have made it impossible to find the 'correct' analysis.

To illustrate the improvement with respect to version 6 of the parser in test one, the two-clause utterance discussed as a problem in section 5.2.2 (Figure 29), *this worked out it won't fit* (utterance 11), has now been successfully parsed, both as two separate utterances, and as one co-ordinated clause utterance, as found in the corpus. Although the exact POW analysis is not found in the first six solutions, the third solution to be found (see Appendix 15) differs from it only in that the word *out* is dominated (indirectly) by a simple complement label (*C*), rather than the finer-grained main-verb-completing complement (*CM*) which identifies the occurrence of a phrasal verb. (The finer-grained label is less frequent than the general complement label. Note that in the separate test parsing just the first clause, the parser finds both solutions.) We will now consider the issues of parser efficiency and lexical tagging in more detail.

5.3.1 Test 2: Parser Efficiency.

Version 7 of the parser has improved efficiency because only one lexical tag is added to the chart for each word. Relatively deep edges are prohibited by a depth limit across all edges in the chart, and some potential edges are never added to the chart by virtue of being vertically ill-formed. Although version 7 of the parser has constrained edge generation markedly compared to version 6, it has not eliminated the combinatorial explosion licensed by the syntactic formalism.

Parse times range from 2-20 CPU seconds for very short utterances to 7715.26 CPU seconds for a six-word utterance. However, parse time is not purely related to sentence length, but also to structural complexity. Another six-word utterance (no. 3) was manually terminated after 7 days having produced 5 solutions, with the agenda having reached around half a million edges.
The size of the parser process reaches up to 85 Mb in such cases. Once the agenda has reached such a size, much of the time is spent on simple sequential list search. The real time devoted by the CPU to such a process obviously varies depending on the number of other

jobs/users the operating system must attend to. The parser is not unique in suffering such memory and speed problems when dealing with a large, perplex grammar: Sekine and Grishman (1995; 221) report limiting active nodes and inactive nodes to 3 million and 10 thousand respectively, and use a small back-up grammar when their chart-parser is 'unable to produce a parse due to memory limitation'. Osborne (1995; 5) reports the need for similar constraints and a 'packing' mechanism when using a modified version of the Alvey NL Tools grammar and parser. Keenan (1993) actually abandons the use of the ANLT, considering it too unwieldy to be applied to the problem of parsing handwriting recognition lattices. He instead resorts to a less powerful Markov-model-based formalisms to find a practical solution.

The success of the test on the POW corpus utterances can be ascribed to their relatively short length, for which the size of the chart is less than around 1,000 edges. With longer sentences, the chart reaches a size of several thousand edges and the agenda is usually an order of magnitude larger, so test material from written corpora would naturally make the efficiency results considerably worse. To a certain extent, this problem is constantly being reduced in importance, as computer processing speed and working memory increase. Nevertheless, it does currently limit the practical use of the parser for unrestricted English.

5.3 2 Test 2: Lexical Tagging Results.

The EPOW-trained Brill tagger assigns one tag for each word of input, and is reported by Hughes and Atwell (forthcoming) to achieve around 95% accuracy. On both the untrained and trained sentences in test two, the accuracy is somewhat lower, at just over 92% (11 errors in 138 words). The errors are summarised below:

Lexical item	Tag assigned	EPOW tag
's	OX	OM
mind	M	AF
fantastic	AX	EX
on (x2)	P	AX
always	AX	AI
get	X	M

select	P	M
shorter	H	AXT
displays	P	M
enter	P	M

Interestingly, the error rate is actually worse on the POW corpus material (7/74) than on the untrained tests (4/64). However, of the seven errors found in the POW tests, only one proves really fatal: the labelling of *get* as an auxiliary *X* prevents any analysis at all being found for utterance no. 16. (All of the other errors for the POW tests still permit an analysis to be produced, even if it doesn't exactly match that in the corpus). In contrast, all four of the untrained test errors are serious, and three fall into the same category; labelling a sentence-initial main verb (*M*) incorrectly as a preposition (*P*). This occurs because none of these words is found in the POW corpus, so the tagger resorts to its morphological and bigram/context rules to try to determine the word-tag. In each of the three cases, the main verb is followed (not unusually) by the sequence determiner (*DD*) + head (*H*), which happens to match the structure for the complement of a preposition as well as that of a main verb. The fourth serious error is the labelling of *shorter* as a head (*H*) rather than comparative apex (*AXT*) in the co-ordinating expression *longer or shorter*. It is likely that this would have produced an analysis of co-ordinated *NGP*s rather than *QQGP*s.

The capitalised items in the untrained tests are assigned the label *HN* (namelike head) which I deem to be acceptable, although in the corpus the alternative analysis of a simple head (*H*) dominated by *NGP* filling a thing-modifier (*MOTH*) is often found labelling compound nouns and related collocations.

The nature of the errors occurring with either CELEX look-up or Brill tagging is strikingly similar. The areas for potential improvement are phrasal verb particles, adjuncts acting as terminal labels, open-ended categories such as exclamations, and SFG's fine-grained description for auxiliaries. The Brill tagger is more successful, and has the added advantage that it uses the context of each item in the sentence to determine just one tag per item.

5.4 Formal Evaluation.

The formal evaluation of parsing programs is still fraught with problems, as (Koch and Sutcliffe 1995) illustrates. Researchers have different views as to what constitutes a correct parse. The broadest difference corresponds to the various grammatical descriptions:

dependency grammar, phrase-structure grammar and functional grammar. The aim of this project has fallen within the bounds of the syntactic element of systemic functional grammar, which involves building a tree structure akin to that in phrase-structure grammar, but also labelling alternate layers of nodes with functional labels, called elements of structure. The trees produced by my model are a notational variant, in which elements of structure and units appear on separate nodes. Our task has been to try to reproduce the particular systemic-functional syntax used to annotate the POW corpus, which omits the functional role labels found in the more recent development of SFG for the COMMUNAL NL generator, GENESYS. Nevertheless, it still constitutes a very rich syntactic description. The richness of a syntactic description is a further area in which researchers differ in what they view as a successful parse.

On one end of the spectrum, there are very coarse-grained descriptions, with only a handful of terminals and non-terminals, often referred to as skeletal parsing, epitomised by that found in the UPenn treebank (Marcus et al 1993), and in the parser of Sekine and Grishman (1995), which contains only two non-terminals! At the other end, there are extremely detailed descriptions, devised to illustrate fine-grained linguistic distinctions, such as those found in POW, LOB, ICE and the Susanne Corpus, for example.

Parsing tests may also include or exclude the problem of lexical look-up, by guaranteeing correct lexical tagging for the test sentences, or not. Different teams also vary in judging which solution to include as a parse to be entered into the evaluation process; the first only, one of the first few, or any of the parses produced. Even when we have a supposed target parse to aim for, we cannot be entirely confident in judging correctness, since it is possible with corpus data (as shown in utterance no. 6) for the target to contain an error, (albeit minor in this case). Perhaps the most obvious ways of comparing solutions found against a target solution is to match node labellings and tree structure overlap. It requires linguistic knowledge of the syntactic category labels used in a grammatical description to make judgments of correctness of node labelling, and to classify labelling errors as minor or major, as I have done here. Such judgment is of course prone to occasional error, just as the original manual transcribers and annotators were. Similarly, judgments can be made as to the distance from the target tree structure by positing the number of structural changes required to transform a solution parse tree into the target tree. Such changes would have to be defined and accepted as a standard, (four possibilities would be those used in the simulated annealing parser projects at Leeds: relabel, merge, hive and reattach (Sampson et al 1989, Souter and O'Donoghue 1991), but who is to decide whether each change should be equally weighted?

A final problem, moreover, is that although we know that a robust parser should eventually be able to produce a corpus-based target sentence if it was trained on the parsed corpus (and has completely accurate tagging), there is no guarantee that the target will be (or even should be) the first solution to be produced (or even one of the first six), since other legitimate syntactic analyses might be produced first, which the annotators ruled out using intonation cues, semantic and pragmatic knowledge.

These comments explain why any one measure of the success of a parser has yet to be generally accepted. Sampson et al (1989) proposed an evaluation metric which measures similarity of node labels and overall structure by assessing the overlap in all the vertical strips from leaf to root node, and report values of 75% for the APRIL parser tested on 50 sentences from the parsed LOB corpus. To my knowledge, the metric has not been re-used elsewhere, although O'Donoghue (1994; 112) uses a variant on this theme.

Black et al (1991) propose an evaluation function 'based only on the constituent boundaries ... (and not on the names it assigns to these constituents)', which first eliminates a list of known discrepancies between different grammatical descriptions, and then compares crossing parentheses and recall with a similarly reduced parse taken from the Penn treebank. This metric has been criticised by some researchers as too coarse and not linguistically motivated (Magerman 1994; 91, Weerasinghe 1994; 140). Two good reasons for not considering such *crossings* to be an acceptable measure are that when dealing with accuracy within just one parsing scheme, the labels on the nodes are clearly important in judging how correct a parse is. Secondly, the measure assumes only one solution parse is produced, whereas a parser may reasonably produce more than one (acceptable) parse. Weerasinghe concludes that an exact match can be the only fully acceptable measure, but, as we have just discussed, this has its own shortfalls, since the target may not be the only legitimate syntactic parse, or may itself be flawed.

Nevertheless, for the sake of those who wish to try to quantify results, the performance of the parser (in Test 2) can be recorded as shown in Figure 32 (and readers wishing to quote these figures are duty bound to quote the preceding four paragraphs as well).

An exact match success rate of 26% is found for the trained test (there is no such concept for this unseen test). If we broaden our window to the first six parses found, the exact match success rate is 53% (a+b). If we broaden our success criteria still further, to include any legitimate SF syntax tree in the first six parses, the success rate rises to 89% (a+b+c+d) on

the POW corpus tests. Since the number of tests for the unseen material is quite low, we will not evaluate them separately.

For all 25 sentences, the most conservative criterion we can set is that a legitimate solution must be produced first (a+c), which has a success rate of 40%. If we broaden this again to say a legitimate (POW SF syntax) parse may be produced within the first six solutions, the overall success rate (a+b+c+d) is 76%.

Figure 32: Evaluation of test results on 19 'seen' and 6 unseen test utterances.

Type of Measure	Number of 'Seen' Utterances	%age	Number of Unseen Utterances	%age	Overall %age
(a) Exact match, first parse	5	26.3	n/a	n/a	
(b) Exact match, within top 6	5	26.3	n/a	n/a	
(c) Good SF syntax tree produced first	4	21.1	1	16.6	40
(d) Good SF syntax tree produced in top 6	3	26.3	1	16.6	36
(e) Only bad parses produced	0	0.0	0	0	0
(f) No parses produced	2	10.5	4	66.6	24
Total sentences	19	100	6	100	100

To a certain extent, the intended use of a parser should also play a part in evaluating its success rate. Parsers can be used in a variety of applications, including
1. tasks which generally pass the parsing results onto further levels of processing (semantic, pragmatic, machine translation, speech synthesis),
2. evaluating best paths through the lattices in recognition tasks (speech and handwriting recognition),
3. evaluating acceptability of grammar (grammar and style checkers in word processing tools), 4. semi-automatically annotating raw corpora, in order to expand the corpus data set from which further broad-coverage probabilistic and syntactic models can be obtained for a natural language, or to obtain parses from several different grammatical descriptions for the same raw language data.

The eventual application for this parser has deliberately been only loosely specified as belonging to either the tasks in 1. or 4. Most of the work conducted on parsing the syntax of SFG has been with a view to the higher goal of semantic interpretation (falling squarely under task 1), in order to permit NL analysis and generation within the same grammatical

description (if not the same formalism). See for example the contributions by Fawcett (1994) and O'Donoghue (1994) in Strzalkowski (1994), and the work of Kasper (1988), O'Donnell (1993) and Weerasinghe (1994). The results presented here do not compare at all favourably with those of Weerasinghe's POP and O'Donoghue's Vertical Strip Parser, which achieve exact match success rates of 85% and 81% respectively. However, both of their projects were based on much less complex versions of SF syntax with narrower coverage of English than that found in the POW corpus: those extracted from the COMMUNAL NL generator GENESYS 'midi' and PG 1.5 versions respectively. Equally, their syntactic models did not yet include notations for handling ellipted, replacement and unfinished elements within a constituent. One would also assume that all of their training material consisted of 'headed' constituents, which cannot be said for the POW corpus. It may also be argued that the small quantity of unseen testing material used here presents a greater challenge (in terms of sentence length and genre) than the POW and GENESYS Ark sentences used by Weerasinghe and O'Donoghue, although they have used a greater number of test sentences.

Even though the POW SFG notation is different from that used in the COMMUNAL NL generator, it would theoretically be possible for version 7 of the present parser to be used as a precursor to semantic interpretation, since Weerasinghe (1994; 176) has devised a partial mapping between the POW and COMMUNAL SF syntactic descriptions[4]. However, with the present algorithm and computing resources, it would not be a feasible option for this or any other real-time application, because of the slow performance of the parser.

Instead, the present parser offers greater potential as a tool for aiding corpus linguists in annotating new material according to the POW scheme. It is a time-consuming job for any linguist to learn the syntactic annotation scheme for any parsed corpus, even with some kind of annotation training manual (which doesn't always exist). The parser could be used to suggest possible analyses which the corpus annotator could choose from or modify in a manual post-editing process. This is the kind of tool the AMALGAM team at Leeds has already used to annotate the Spoken English Corpus with the ICE syntactic description, consisting of a tagger and parser provided by the TOSCA group from Nijmegen University, Holland (Willis 1996). In this kind of interactive mode, it is possible to first check that the tagging has been accurate, before continuing with parsing, which would certainly benefit our

[4] It is a partial mapping because Weerasinghe's list describes only 42 of the 98 node labels used POW, and listed in Appendix 3.

own probabilistic chart parser. Even to do the same for the POW SF syntax scheme, however, the present parser would need to be improved in terms of speed and sentence length.

5.5 Conclusions.

The results presented here represent a qualified success. We have first seen that when using a corpus-trained context-free syntax model, the probabilistic chart parser is effectively unable to assign structures quickly to sentences of five words or more, due to combinatorial explosion in the grammar. Inadequacies in the CELEX lexical look-up also had a small part to play, although modifications to the lexical grammar mapping would minimise this problem. With the exception of lexical failure, the context-free chart parser *would* be able to produce correct analyses for longer sentences given sufficient time and adequate hardware resources, but these analyses would be part of an enormous list of possible solutions.

By modifying the chart-parsing algorithm to employ both probabilistic context-free rules and a probabilistic vertical trigram model (which together form a context-sensitive model), the success of the parser is significantly improved, being able to find at least one legitimate SF syntax parse in its first six solutions for 76% of the tests performed. The combined model serves to improve the validity of the solutions produced, and prune the parser's search space. However, until the processing speed of computing hardware is greatly increased, such results can still only be produced after an impractically long delay for any real-time applications, particularly when the test sentences contain more than 6-7 words. In order to minimise this delay, a set of termination conditions for parsing have been introduced, and the depth of solution trees has been capped to that observed in the POW corpus.

The use of probabilistic search does tend to cause the 'desired' solution to be among the first few produced (given the evidence of the small test sample included here), but is not always the first solution. Using syntactic probability alone cannot be responsible for selecting a correct parse from the forest of trees produced, and my results confirm the need for semantic and pragmatic knowledge to be used as well in order to select the target solution found in the corpus. I would however raise the question of the validity of tests seeking to find only an exact match with a corpus-based target for a purely syntactic parser.

As far as lexical look-up and tagging are concerned, both the CELEX lexicon (modified to include SF syntactic categories) and the Brill tagger trained on the POW corpus have proved quite successful, with accuracy levels of 83% and 92% respectively for the word tokens in

the test data. However, both tools show some room for improvement. The Brill tagger outperforms the CELEX lexicon in my tests, but would perhaps be less successful were a greater number of unseen tests to be included. If we consider our aim to be that of finding a corpus target solution, then the Brill tagger has the advantage of only introducing one tag per word into the chart, which improves parser performance. But if we have a wider aim of producing all syntactically legitimate parses, then the selection of only one tag per word will compromise parsing success, and a large-scale lexicon such as CELEX may offer a better solution.

In the next chapter, I will assess how the components of the parser could be further improved, to go a step further toward practical parsing using a wide-coverage systemic functional syntax model of English, such as that found in the POW corpus.

Chapter 6. Improvements to the Parser.

At the end of chapter five I drew the conclusion that the components used in version 7 of the parser are a qualified success, but still leave room for improvement. A feasible method has been proposed and implemented for creating a wide-coverage lexicon using the CELEX lexical database, and an alternative lexical tool, the POW-trained Brill tagger, is even more successful. The syntactic model extracted from the POW corpus consists of both probabilistic context-free rules and probabilistic vertical trigrams, and serves to limit the over-generation found in a purely context-free model. The probabilistic chart parsing algorithm I have adopted and modified is able to use these resources with some success, but not yet with sufficient speed to be employed in a real-time application. This chapter considers how the parsing algorithm and its lexical and syntactic resources might be further improved and controlled. I will first consider the possibilities for improving the lexical resources.

6.1 Improving the Lexical Resources.

6.1.1 Disambiguated Tag Probabilities.

One of the most obvious improvements to the POW syntax modified CELEX lexicon would be to include the frequencies of each ambiguous tag for a word in the parsing process. This would ‘enhance’ the most likely lexical edges, making it more likely that the desired parse is found first. Having disambiguated tag frequencies would not however solve the problem of ambiguity itself. In the absence of a several million word SFG tagged corpus, the frequencies could be obtained from a large tagged corpus such as the British National Corpus or the Bank of English, but in each of these cases a mapping of grammatical categories to the POW SF syntax would be involved. In areas where such a tagged corpus used a less delicate grammatical description than the POW SF syntax, tag frequency assignment would be complicated. But for basic category distinctions (such as noun/verb/adjective) broad preferences for one particular tag could be derived.

6.1.2 Refining the CELEX to POW Syntax Mapping.

Further amendments to the mapping between the original CELEX grammar categories and those of the POW corpus have been identified during the course of this study. Many of these will have to be in the form of hand-crafted entries, or exception lists to the general mapping, because they result from areas where SFG is particularly delicate. The principal areas of deficiency are described in sections 3.2.6 and 5.2.

A mechanism has yet to be devised for permitting and controlling the production of temperers and adjectival verb participles within the parser itself. Implementation within the parser is proposed because there appears to be no real restriction in English on the spontaneous production of degree and scope temperers from adverbs and adjectives, for example (see section 3.2.5 for exemplification). A first attempt solution would be for the parser to identify cases of adjacent adverbs, or adverbs occurring to the left of prepositions and adjectives, and check if there is an active edge at that point in the chart looking for a temperer.

6.1.3 Handling Recurrent Word Combinations.

A more radical modification to the lexicon needs to be made to handle the linguistic phenomena of idiomatisation (*he kicked the bucket*) and compounding (*piston engine*), neither of which is supported by CELEX. In both of these cases, several wordforms need to be classed together as one lexical item. In the context of a chart parser, we can imagine this being implemented (when the chart is initialised) as an edge spanning several words, but with only one tag. Idioms and compounds represent one extreme of a cline in language between totally free combination of wordforms and increasingly restricted co-occurrence of words. Restricted word co-occurrences (variously called *collocations, phrases* or *phrasemes*), have not been accounted for in any serious way by computational linguists writing parsers. Altenberg (1994) claims that around 70% of the words in the London-Lund Corpus are part of recurrent word combinations, and these can vary from two to around forty words in length. If such a large number of words in the language are in some way bound lexically, syntactically and semantically to each other, this should surely be captured in the lexicon, and this information used to prime the parser. However, until the lexicon is revamped to become more of a phrase-bank, there is little hope for parsers to take advantage of collocation. Omitting such information condemns the parser to producing all the ambiguities the

grammar and lexical tags will permit, without finding the idiomatic or collocational structure. The POW SF annotation makes a small attempt at capturing such word combinations through its use of the labels *CM* and *PM* (Main verb completing complement and main verb preposition respectively), but a simple context-free grammar formalism fails to capture these horizontal dependencies with the main verb. In other cases, the annotators of the corpus hyphenate together multi-word lexical items. Other than requiring these to be hyphenated in the input to the parser (in which case they would be treated as one item and labelled with default tags), the lexicon can be the only source of collocation, compound and idiom information. A comprehensive collocational/idiom lexicon should also handle phrasal verbs and at least the most common temperer combinations.

6.1.4 Improving the Brill tagger.

The main deficiency in the Brill tagger is in the size of its lexicon, which is obtained directly from the POW corpus. Any words not found in the corpus are tagged using a small set of default rules, which consider the morphology of the unknown word, and the surrounding words and their tags. These rules have been learnt from the morphology of only the words in the corpus, so would clearly benefit from being re-trained on a larger POW-SF annotated corpus. The Brill tagger's lexicon could be supplemented using the wordforms and tags in the CELEX lexicon, which would expand its size from around 4,500 words to at least 60,000 words. Unfortunately, this is not a trivial exercise, since for ambiguous words, the Brill tagger's lexicon requires the alternative tags to be ordered by likelihood, which is effectively the same problem as discussed in section 6.1.1.

One other way the error rate of the Brill tagger could be reduced, is (somewhat bizarrely) to attempt to independently tag the test sentence with the Brill tagger trained on other (perhaps larger) corpora. The tags so obtained from other schemes could be broadly compared with the POW tag, and if the POW tag was found in some way to be the odd one out, it could be mapped to a better tag in the POW scheme by use of a tag interlingua. Such an interlingua is currently under development by John Hughes in the AMALGAM project (Hughes and Atwell, forthcoming).

6.2 Improving the Syntactic Formalism.

6.2.1 A Probability Matrix for Optional Rules.

One improvement to the context-free syntax formalism as it stands would be to retain all individual rule probabilities in the collapsed rules, as discussed in section 4.3.1. A possible way of retaining the original probabilities would be to store them in a matrix as the initial probability element of the collapsed rule. A collapsed rule containing *n* optional daughters would need 2^n places in the probability matrix. Observing the collapsed context-free rules derived from the corpus, *n* would be at most four, so a 16 place ordered matrix would be needed. However, the simplest way to implement such a structure when collapsing the rules would be to apply it to all rules, regardless of their number of optional daughters. This would result in a vast number of zero probabilities being needlessly stored. Since the syntactic model is currently loaded as part of the POP11 process, this would use up valuable memory. A method which only creates the matrix as it is required would be preferable. During parsing, when an optional edge is instantiated, the matrix probability would then be accessed to derive the combined edge probability. A further difficulty with this solution however is that in a corpus the size of EPOW, few rules are frequent enough for us to predict probability with great confidence. Collapsed rules at least include combined probabilities we can be surer of.

6.2.2 Grammatical Coverage.

The modification to all the filling rules extracted from the corpus currently permits a theoretically unlimited number of subordinated or co-ordinated units to fill an element of structure. This propensity is restricted by a probability degradation function (estimated from the evidence in the corpus), which progressively penalises edges representing an element of structure containing more and more units of the same type. The degradation function can, of course, be modified to produce solutions displaying more or less co-ordination.

The vertical trigram model theoretically allows the parser to permit vertical relationships greater than three nodes deep which are not observed in the corpus, but tends to constrain them to be very near to the corpus patterns. A vertical sequence of three nodes or less is however required to have been observed in the corpus.

No such flexibility has yet been introduced in the handling of the componence relationship (that which expands a unit as a sequence of elements of structure), which forms the bulk of the combined syntactic model, particularly with respect to the expansion of the clause. This is modelled by a probabilistic context-free formalism, with a modification allowing optional daughters. To permit componence relationships not observed in the corpus, we would need to relax the context-free model to some kind of regular grammar, such as linear precedence rules, or a bigram or trigram model (Weerasinghe (1994) uses a bigram model of transition probabilities between elements of structure). The componence grammar is already enormous, given the inclusion of rules for replacement, ellipted and unfinished constituents, and relaxing the formalism here would further increase the permutations to be added to the chart, so this step has not been taken.

6.3 Improving the Probabilistic Chart Parser.

6.3.1 Efficiency Checks.

The first step one might take to improve the performance of the probabilistic chart parser as it stands is to do some kind of efficiency audit. This would involve monitoring and minimalising the time spent on garbage collection, as well as assessing each procedure in turn to check it has been written in an efficient way. Whilst I have attempted to bear efficiency in mind in the modifications I have made to Pocock and Atwell's weight-driven chart parser[1], some of my changes will most probably have compromised the efficiency of their implementation. Two such cases have already been identified:

Firstly, a small number of duplicate edges are produced as a result of applying different optional rules which instantiate to the same rule in parsing.

Secondly, to prevent mutual vertical recursion of the same pair of categories (discussed in section 5.2.2) I introduced a less stringent check on the contents of the chart and the agenda

[1] Indeed, that was the purpose of collapsing the grammar in the first place.

when proposing a new edge. (The less stringent check allowed for variation in the probability and the lower levels of the 'found' elements of an edge, whereas Pocock's implementation required an exact match, using the *member* function). My revision improved accuracy of the parser at the expense of efficiency (the POP11 *matches* facility was used instead of *member*). Pocock's code was carefully checked to see if efficient alternatives to many POP11 functions could be used, whereas I have not yet done the same for my own supplementary code.

A further efficiency gain may be possible through external storage of the syntactic model. However, unlike the lexical facility, which needs to be accessed only once per word at the start of the parsing run, the syntactic model must be consulted throughout parsing. Potential efficiency gains could only be measured by implementing an external grammar access function, and testing its performance. Note however that any such improvements are only likely to be minor, since Pocock and Atwell's improvement of Gazdar and Mellish's original algorithm was incrementally developed taking efficiency into account at each stage (see Pocock and Atwell 1993, Atwell 1994, for details).

6.3.2 Applying Multi-Word Edges in a Chart.

One advantage of the chart parsing algorithm is that only minor modifications would be necessary to accommodate multi-word lexical edges (for collocations, idioms etc. see section 6.1.3) being added to the chart initialisation procedure. Instead of simply taking each word in turn and looking it up in the lexicon, all permutations of neighbouring word combinations would also be looked up in the lexicon (supposing a good collocational lexicon with SF syntax labels were available). A further amendment might be to look up non-adjacent combinations of words, since some collocations can be discontinuous: eg. *run up* in *run a big bill up*. Multi-word edges would tend to be activated ahead of productive syntactic combinations, since the former would not involve probability combination. Productive combinations still need to be generated by the parser for cases of non-idiomatic use of the same structure. For example, in (1) there is a genuine ambiguity between the idiomatic and syntactic readings, whereas in (2) and (3) syntactic and semantic clues in the adverbial phrase (time versus place adjuncts) suggest one reading as more likely.

(1) He kicked the bucket.

(2) He kicked the bucket last week.
(3) He kicked the bucket down the corridor.

Although the idea of parsing idioms and collocations is attractive, it would actually serve to increase the perplexity of the syntax model. Whereas some lexicons of idioms and phrasal verbs do exist, like the CELEX lexicon, they would need to be mapped to the POW SF syntax labels. Alternatively, the Brill tagger could be re-trained on collocation-annotated corpus material.

6.3.3 Restricting Unnecessary Rule Application.

I have already implemented a limited look-ahead function within the chart parser to restrict rules with more daughters than there are words remaining from being added to the chart (or agenda). A more general look ahead facility to prevent rule application at the start and middle of a sentence is needed. As the algorithm stands, the only check that is performed (apart from the sentence length check just mentioned) when the grammar is consulted, is to ensure that the grammar rule contains a first daughter matching the label of an inactive edge in the chart. If the grammar rule contains more than one daughter, additional checks should be performed on each daughter to ensure that the word tags on the right-adjacent words could either be such daughters, or be constituent parts of the daughters. Implementing this in SF syntax would probably require a record being kept of a node's vertical position in the tree, i.e. its position in the rankscale (Fawcett 1981: 54). Were a look-ahead function able to be applied in this way, it would severely restrict the combinatorial explosion produced by a simple context-free grammar and unmodified chart parsing algorithm. On the other hand, such a look-ahead check every time a potential rule is considered would add other processing overheads, since it would effectively involve checking for the existence of a vertical strip, in (O'Donoghue 1994)'s terms, between a right-adjacent word and the next daughter category.

6.3.4 Controlling the agenda.

Perhaps the single most important improvement to the algorithm of version 7 of the parser would be in controlling the agenda to accelerate the production of spanning edges. The agenda is currently implemented as a POP11 list, ordered simply by the weight of each edge, no matter how many words the edge spans. The size of the agenda produced for even a six-word sentence

can reach half a million edges after a week of parsing, so managing it efficiently is crucial to efficient parsing overall.

In a standard chart parser, the agenda is treated as either a stack or a queue, to achieve depth or breadth first search. Ordering the agenda by probability is different to both of these. We can produce a search path approaching either depth or breadth first (whilst still retained a most-likely first strategy) by altering the probabilistic function determining the weight of a subtree. If we want to achieve breadth first style search, we can penalise new edges being proposed which don't increase their combined word span, (i.e. those with one edge which is entirely active, and has found no daughters yet), by adding to their weight. We can encourage the early production of deeper narrower trees by lessening the same weight. In the current parser, version 7, more of a breadth first effect is produced by modifying the combined weight derived from the vertical trigram model for any new edge. Since POW SFG trees are relatively broad and flat (compared to say a tree in Chomsky Normal Form), the vertical weight parameter has been set to reproduce fairly flat trees (although this can be altered by adjusting a single parameter value).

Very deep trees are prohibited from ever being added to the agenda by a further function limiting edge depth. This function depends on the length of the edge (in words), and the limit can be set to the maximum depth observed in the corpus for a particular word span, or somewhere between the maximum and the median, if we wish to further prune potential edges from entering the agenda (at risk of not finding some rarer, deep solution trees). The function must rise in steps of two nodes, (since I treat the filling relationship as covering two nodes, rather than one), to account for the alternation between units and elements of structure. It is currently set to be the maximum observed depth in the corpus for the range of sentence lengths (from one word to 73 words, the maximum depth ranges from 12 nodes to 18 nodes). It would be worth experimenting with limits nearer to the median, rather than the maximum depth to see how much the agenda is reduced, and thereby parsing speed improved.

However, probably the most effective modification we could make to the agenda would be to prioritise multi-word edges over those spanning fewer words. The agenda could be ordered first by edge breadth into several sub-agendas, and then by weight within each sub-agenda, as Weerasinghe (1994; 117) has done with apparent success. Agenda search could also be speeded

up by physically storing each sub-agenda as a separate element of an array, as we have in the current parser for the chart itself.

6.3.5 Feature-Based Parsing.

It is tempting to suggest that feature-based categories should replace the simple atomic labels used in SFG syntax, and combined using unification during parsing. This change has been one of the hallmarks of several recent grammatical theories (GPSG, HPSG, LFG, Categorial Grammar), but such formalisms have rarely been used for corpus annotation (in which case it has been achieved by automatic parsing using a feature-based parser, such as the ANLT parser, which relies on the headedness of constituents). Such a proposal would serve to reduce the grammar size, and capture some of the long-distance dependencies (vertical and horizontal) which are found in the POW corpus trees. For example, the relationship between a main verb and a main verb completing complement or preposition could be enforced using feature propagation principles (such as GPSG's Control Agreement Principle, see Gazdar et al 1985; 83). Likewise the dependency between a *wh-* word, wordtag and its *wh-* grandmother could be captured by the GPSG Head Feature Convention (Gazdar et al 1985; 50), instead of using the current vertical trigram model. In the POW corpus grammar, though, we have observed many cases where the head is absent, which would prevent such conventions/principles from being successfully applied.

However while using feature-based grammars and unification obviously reduces parser overgeneration, it does not solve the problem of massive ambiguity that all wide-coverage grammars experience (Oostdijk 1991, Keenan 1993). A further drawback with feature-based grammars is that they are hand-crafted competence grammars, and not directly extractable from corpora, along with their rule probabilities. So producing a feature-based formalism on the scale of lexicon and grammar we have used here would be an enormous manual task, and be contrary to the general ethos of this work, which is to minimise the manual intervention in the creation of the syntactic model, by extracting it from suitable annotated corpus material. The only feasible way a large feature-based SFG could be automatically produced would be to use a NL generator such as GENESYS to produce syntax trees annotated with the relevant features taken from the system network, should the network contain them. Even if this were possible, this grammar

would not have as wide a coverage as the POW corpus SFG, nor would it have realistic frequencies to guide parsing.

6.3.6 Semantic Solution Pruning.

When faced with the problem of massive ambiguity, most developers of wide-coverage grammars assume some kind of compositional semantic model will work alongside the syntax to select the correct analyses. A Montague-style compositional approach is used to attempt to build semantic representations for each syntactic analysis, and if a representation cannot be built, then the parse is rejected. Problems with this approach are that again each semantic rule has to be hand crafted rather than automatically learned, and it assumes that the semantic grammar is complete. Sentences which are not assigned semantic representations may well be legitimate, but just not covered adequately by semantic rules yet. A mechanism for assigning likelihoods to semantic representations is not normally incorporated, and so the buck is passed on to a (usually even less well defined) pragmatic model for further disambiguation.

A more promising semantic constraint model for disambiguating parse forests would be one which could be automatically learned or extracted, would allow a weight to be assigned to a sentence, and would not rely directly on the precise syntactic structure of the sentence for its success. Demetriou and Atwell (1994) present a combination of such semantic constraints which offer a more practical approach to semantic disambiguation. Among others, they include semantic domain codes and selectional restrictions/preferences extracted from machine-readable dictionaries and lexical databases, and measures of mutual semantic information and distance between words, which can be extracted from corpora. Some of these measures rely first on the production of a syntactic structure, to provide the semantic head of each constituent, for example, while others can be applied independently of the syntax tree. The former are useful in disambiguating syntax trees, while the former and the latter can be applied in speech and handwriting recognition tasks where the recogniser produces a lattice of words as output. Although an optimal function for combining each of these constraints has yet to be established, they offer a more practical (learnable) way of assessing the semantic integrity of a sentence and its parse trees.

6.4 Conclusion.

In this chapter I have presented some potential improvements to the probabilistic chart parser algorithm, and to the lexical, corpus and syntactic resources it employs. Each of these are non-trivial amendments, so have not yet been implemented. In some cases they will not be implemented in the foreseeable future owing to the absence of relevant lexical or corpus resources. These improvements address the issues of poor speed, over and undergeneration, and disambiguation of multiple solutions which were evident in the parsing test results detailed in chapter 5. It is difficult to assess the gains that can be made in each case, but adjusting the ordering of the agenda to prioritise broader edges over narrower edges would appear to offer the greatest potential for improving parser performance to a more adequate speed for real-time applications. At the same time, the current implementation would certainly benefit from faster hardware which is currently or soon will be available.

Chapter 7. Conclusions.

The original aims of this thesis were twofold: to propose a re-usable method for deriving a probabilistic parser for relatively unrestricted spoken English, and to use this method to implement such a parser for the systemic functional syntax used to annotate the POW corpus. Both of these aims have been achieved, but not without some qualifications on their success. The main shortfall has been in the implementation of a parsing algorithm which can rapidly traverse the enormous search space presented by the large lexical and syntactic resources needed for unrestricted parsing. The parser achieves a reasonably creditable success rate of 76%, if the criteria for success are liberally set at at least one legitimate SF syntax tree in the first six produced for the given test data. But it doesn't do this quickly enough for real-time applications. I will now discuss these two aims in turn, beginning with the different elements of the parser implementation for the corpus-trained systemic functional syntax.

7.1 Lexical Resources.

Two lexical resources have been developed to support the parser; a large-scale lexicon (from CELEX), and a corpus-trained tagging program (due to Brill).

The first consists of a 60,000 wordform lexicon derived from the CELEX database, in which the original 'theory-neutral' grammatical tags have been semi-automatically transformed to those of the POW SF syntax. The mapping has been tested by looking up all the unique word-wordtag pairs from the Edited POW corpus in the CELEX lexicon, and checking if the POW-assigned tag was present. The success rate for word tokens with the correct tag was around 82%, which means the two default mechanisms described below will come into play for approximately one in every five words. On the full parsing tests described in section 5.2, look-up success rate was 83% of the words and tags found in the POW corpus.

The lexical probabilities contained in the CELEX database are extracted from the Birmingham COBUILD corpus. In the present parser, these probabilities are not used, since they are not disambiguated for each possible tag. However, were the parser being used for a speech or handwriting recognition application, the likelihood of one wordform over another would be valuable information. The lexicon is supported by two default mechanisms within the parser. The first assigns the label for proper nouns to any words other than the first

person singular pronoun (*I*) which begin with a capital letter. This default assumes some form of lexical preprocessing has been performed on the input, either spoken or written. The second default assigns the three most common tags (noun, verb, adjective) to any words which have failed to have been assigned a tag by lexical look up or the first default. Rapid lexical look up is achieved as an external process to the parser using an indexed and subindexed file structure.

The second lexical resource is the Brill tagger (Brill 1992, 93, 94), trained on the POW corpus by John Hughes (Hughes and Atwell, forthcoming). This incorporates a small lexicon based on the words in the corpus, and a set of context an bigram-like rules learned from the corpus, as well as some rules for assigning tags to unknown words. The tagger assigns a single tag to each word of input, and does so with 92% success rate on my test data (see section 5.3). Apart from an improved success rate, the Brill tagger is preferred because it makes parsing more efficient, by only introducing one edge per word when initialising the chart. It is also re-usable, since it can be trained on other corpora, as Hughes and Atwell (forthcoming) demonstrate for the LOB, London-Lund, Brown, SEC, Penn and ICE corpora. The tagging success rate could be improved by supplementing the tagger's small lexicon with the aforementioned CELEX lexicon, by checking possible tags against those assigned by the Brill tagger trained on other schemes, and identifying discrepancies, or by using the tagger to tag new material, manually post-editing it, and retraining on the enlarged tagged corpus.

7.2 Grammatical Resources.

The grammatical resources consist firstly of a probabilistic context-free grammar extracted automatically from the Edited Polytechnic of Wales corpus, of just over 2,800 rules. This size of grammar is achieved by treating the SFG relationships of componence and filling as the same phenomenon. This means formal and functional grammar labels are not treated as though they were placed on the same node in a tree, but handled as if they were on separate nodes. Were the two relationships distinguished when extracting rules, the grammar size would grow to at least twice this size. The benefit gained through the smaller set of rules is offset by the loss of context sensitive grammatical information. In order to compensate for this loss, and capture the vertical relationships between units and elements of structure (and vice-versa), a probabilistic vertical trigram model is also extracted from the corpus.

To reduce the set of context-free rules, and thereby improve parsing efficiency, an algorithm for collapsing similar rules into new rules containing optional daughters is devised and

applied to the componence rules. Similarly, the filling rules which handle co-ordination (and sub-ordination) in the grammar are collapsed into a smaller set of rules with only one daughter, while maintaining the probability of the repeated daughters. A general degradation function for the increasing occurrence of co-ordination is extracted from this data, which will then permit the parser to recognise co-ordinated structures not realised in the corpus. No default mechanism has been implemented to allow for gaps in the componence grammar which would cause the parser to fail. These reduction techniques led to the final syntactic model containing just under 1900 probabilistic context-free rules, in addition to 968 vertical trigrams. The data structures were in the form of a property table, indexed by the (first) daughter of each rule/trigram, in order to afford more rapid access to specific subparts of the syntax.

The combined context-sensitive syntactic model, when tested with the parser, was found to overgenerate marginally with respect to vertical SFG relationships between words, elements of structure and units.

7.3 Parser Implementation and Testing.

An existing weight-driven chart parsing algorithm was amended to handle rules containing optional daughters and permit co-ordination beyond that exhibited in the corpus. Modifications were also made to allow the lexicon and tagger to be accessed externally to the main parsing process. Two SF syntax specific features were incorporated into the parsing algorithm: a restriction is placed on the fundamental rule of chart parsing such that only edges that are well-formed with respect to the vertical trigram model may be combined. Secondly, the notation for filling rules (which permits repeated co-ordinated and subordinated daughters) was accommodated to distinguish them from componence rules.

Although a working parser for the POW corpus syntax has been produced, it is certainly not an unmitigated success, but neither is it a complete disaster. Two versions of the parser were tested on 25 test sentences taken partly from the POW corpus, and partly from unseen material of a different genre. Of the two parsing algorithms which are tested in chapter 5, it is fair to say that the simple context-free model leaves a great deal to be desired, since it fails to find a target solution in all but two cases. The context-sensitive parser created by the combination of a corpus-trained context-free syntax with a vertical trigram model performs with some success (ranging between 26-89%, depending on evaluation criteria), but still fails to find a solution within the timescale required by any real-time application for parsing all

but very short sentences. For sentences of over seven words the parser can run for several days without producing a solution. In its current form, it would only be truly useful in helping a non-systemicist begin to analyse a raw corpus using the POW corpus annotation scheme. The parser has essentially three component parts, a lexicon, a syntactic model, and an algorithm for combining these into grammatical analyses for sentences, I have provided large-scale lexical and syntactic SFG resources for parsing a substantial subset of English grammar, but a partially inadequate algorithm with which to harness their size and complexity in a real-time application.

Having assessed how well I met this first aim, I will now turn my attention to the second aim, that of demonstrating a reusable method for parsing resource development.

7.4 The Research Method.

The general research principle in this project has been to build computationally tractable linguistic resources which account for characteristics of the language as it is performed, rather than model native speakers' intuitions of how the language should theoretically be used. This principle exposes the computational linguist to many of the messy features of real language use, such as repeated words, interrupted sentences, missing or incorrect words, and constituents without syntactic or semantic heads. Such exposure is exactly what the machine will get in human-computer interaction via written or spoken language, however, so an adequate natural language processing system for unrestricted English needs to be able to account for language as it is performed. Whilst it is tempting to suggest that users of future language technology be constrained in their freedom to use the full range of their natural language, the work presented here instead explores the possibility of parsing relatively unrestricted language, since there are no guarantees that the methodologies used for constrained-NL systems can be scaled up easily to unrestricted language.

I have also tended to address linguistic issues ahead of computational resources, preferring ideally to preserve linguistic information where possible. However, there have been some compromises for the sake of providing a working implementation of the parser. I have furthermore adopted reusable, transferable methods ahead of those devised specifically for one grammatical theory, where practicable.

With these principles in mind, the primary resource for the computational linguist developing a parser is the parsed corpus, which provides a rich set of examples of lexical items and

grammatical structures, and offers the potential of extracting a probabilistic syntactic description in a number of formalisms suitable for parsing. Parsing programs can then be devised for each such formalism, irrespective of the grammatical description it houses. Using the corpus as a primary resource also has the advantage of offering probabilistic information, which is crucial in unrestricted NLP work, as a practical means of ordering potentially massive search spaces and selecting the most likely solutions. The only restriction within this framework is that a parsed corpus is available for a chosen grammatical description, and that it is in a format which permits automatic extraction. In practice this last constraint means writing a bespoke program for translating the corpus into some kind of standard form, such as bracketed trees. Existing parsed corpora have been stored in a variety of formats (Souter 1993), but there are some grounds for hoping that formats for parsed corpora will converge on a standard soon, as the recommendations of the ACL/ALLC sponsored text encoding initiative and the EAGLES language engineering standards project begin to be adopted, and parsed corpus browsing tools become available.

Although parsed corpora yield both grammars and lexicons, these can be developed as separate resources in a standard formalism, and this set formalism adopted as a standard for the parsing program. It becomes sensible to consider separate grammar and lexical development when we make the following observation: While grammars derived from such corpora are relatively large, compared to their hand-written rivals, the same cannot be said for probabilistic lexicons extracted from parsed corpora. Larger lexical resources need to be provided, and apart from tagged corpora, the only other reusable resource available is a lexical database or machine-readable dictionary. The 1 million word tagged LOB corpus furnishes us with a list of around 50,000 unique words and tags, but some lexical databases are even larger. This situation has recently been greatly improved, with the release of the 100 million word tagged British National Corpus as a public resource. Researchers are just beginning to explore this enormous bank of the English language, but are yet to exploit its potential for language engineering tools. Even so, many of the words in a corpus are singleton occurrences of ‘noise’ such as typographical errors, foreign words and proper nouns, which we may not wish to store in the lexicon. No matter how large, corpora tend to display gaps in the typical morphological paradigm for some words, so in the end, a combination of corpus and lexical database may be preferable.

If the lexicon is developed independently, the grammatical description it contains must be harmonised with that in the corpus-based grammar for use in parsing. The relative success of the parser will depend on this harmonisation, the chosen syntactic formalism, and the parsing

algorithm itself. Using either lexical databases or tagged corpora leaves us with the problem of syntactic tag mapping to our target grammar (unless we are fortunate enough to have a very large tagged corpus which contains our target grammatical description). This 'mapping' can be achieved semi-automatically, as I have shown for the CELEX lexicon, or quasi-automatically, as has been done with the Brill tagger. These procedures should be able to be repeated quite easily for other grammatical descriptions, since the POW SF syntax represents one of the richer syntactic annotation schemes, including both formal and functional levels.

For the systemic-functional syntax found in the POW corpus, a combined context-sensitive formalism has had to be adopted to accommodate this dual tree labelling in the parsing program. This step may not be necessary for other corpus-trained syntactic models, for which a context-free formalism may suffice.

The only area in which the research method still leaves real room for improvement is in the parsing algorithm itself. The modified probabilistic chart parser is not yet able to efficiently handle the enormous number of hypotheses licensed by the lexical and syntactic resources in a way that produces solutions in real time. The probabilistic model does tend to generate legitimate solutions in the first few (if not always the first) to be produced, but there is evidence that additional semantic and contextual knowledge are required to select the target solutions found in the corpus. It is anticipated that the greatest further gains in efficiency will come from better handling of the agenda, to prioritise wider-spanning edges.

7.5 The Last Word.

In section 1.3, I asked several questions which arise when developing a parser for unrestricted English. I will conclude by presenting answers to these questions, as discovered during this project. We have seen that the formalism required to capture the complex syntactic relations in a systemic-functional description of unrestricted natural language must be at least as powerful as a context-sensitive grammar. Traditional parsing techniques such as chart parsing therefore have to be modified substantially to incorporate probabilistic (non-immediate) dominance rules with context-free rules, especially where the grammatical description includes formal and functional annotation. The phenomena of unrestricted spoken natural language, such as ellipsis, unfinished sentences, repeated and replacement elements, as displayed in the. systemic functional annotation of the POW corpus, make a parsing algorithm based on headed constituents impracticable. Capturing these syntactic phenomena even in the kind of context-sensitive systemic-functional syntax model described here results

in widespread ambiguity and combinatorial explosion within a chart parsing framework. Control of the agenda using probability, with restrictions on the generation of implausible subtrees, and encouraging a search strategy in keeping with the general shape of the corpus trees only partially addresses the problem, but does result in a promising, if slow, parser. It is likely that only careful prioritisation of wider-spanning edges and general advances in computer hardware speed and memory size will transform the current implementation into one suitable for real-time syntactic parsing of unrestricted English.

References.

Aho, Alfred, Brian Kernighan and Peter Weinberger. 1988. The AWK Programming Language. Addison-Wesley.

Akkerman, Eric, Pieter Masereeuw and Willem Meijs. 1985. Designing a Computerized Lexicon for Linguistic Purposes. ASCOT Report No 1. Amsterdam: Rodopi.

Akkerman, Eric, Henry Voogt-van Zutphen and Willem Meijs. 1988. A Computerized Lexicon for Word-Level Tagging. ASCOT Report No 2. Amsterdam: Rodopi.

Akmajian, A., and F. Heny. 1975. Introduction to the principles of transformational syntax. Cambridge, Mass.: MIT Press.

Allwood, Jens, Lars-Gunnar Andersson and Osten Dahl. 1977. Logic in Linguistics. Cambridge: C.U.P.

Alshawi, H. 1992. The Core Language Engine. Cambridge Mass.: MIT Press.

Altenberg, Bengt. 1994. On the phraseology of spoken English: the evidence of recurrent word combinations. In Proceedings of the Leeds International Symposium on Phraseology. 18-20 April 1994. Leeds University.

Atwell, Eric Steven, Geoffrey Leech and Roger Garside. 1984. Analysis of the LOB Corpus: progress and prospects. In Jan Aarts and Willem Meijs eds. Corpus Linguistics. Amsterdam: Rodopi.

Atwell, Eric Steven. 1987. Converting the Oxford Advanced Learner's Dictionary into a structured database. In Robert Oakman and Barbara Pantonial eds. ICCH87: Abstracts for the Eighth International Conference on Computers and the Humanities. Columbia (South Carolina): Association for Computers and the Humanities.

Atwell, Eric Steven. 1988. Transforming a Parsed Corpus into a Corpus Parser. In Merja Kytö, Ossi Ihalainen and Matti Rissanen (eds.). Corpus Linguistics, hard and soft. 61-70. Amsterdam: Rodopi.

Atwell, Eric and Clive Souter. 1988a. Experiments with a very large corpus-based grammar. In Proceedings of the 15th ALLC conference, June 5-13. Jerusalem.

Atwell, Eric and Clive Souter. 1988b. Probabilistic Criteria in Prototype Parser 1. COMMUNAL Research Report No. 15. School of Computer Studies, The University of Leeds.

Atwell, Eric, Clive Souter and Tim O'Donoghue. 1988. Prototype Parser 1. COMMUNAL Research Report No. 17. School of Computer Studies, The University of Leeds.

Atwell, Eric. 1994. The Speech Oriented Probabilistic Parsing Project: Final Report. School of Computer Studies, University of Leeds.

Atwell, Eric, John Hughes and Clive Souter. 1994. AMALGAM: Automatic Mapping Among Lexico-Grammatical Annotation Models. In Judith Klavans and Philip Resnik eds. Proceedings of The Balancing Act - Combining Symbolic and Statistical Approaches to Language, Workshop in conjunction with the 32nd Annual Meeting of the Association for Computational Linguistics. New Mexico State University, Las Cruces, New Mexico, USA, 27th-30th June 1994.

Barrett, R., A. Ramsay and A. Sloman. 1985. POP-11: A practical language for artificial intelligence. Chichester: Ellis Horwood.

Baum, L. 1972. An inequality and associated maximisation technique in statistical estimation for probabilistic functions of Markov processes. Inequalities (3): 1-8.

Black, Ezra, S. Abney et al. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In Proceedings of the DARPA Workshop on Speech and Natural Language. (Feb 1991). 306-311. San Mateo, Ca.: Morgan Kaufmann.

Black, E., R. Garside and G. Leech (eds.). 1993. Statistically-driven Computer Grammars of English: The IBM Lancaster Approach. Amsterdam: Rodopi.

Bod, Rens. 1993. Monte Carlo Parsing. Proceedings of the 3rd International Workshop on Parsing Technologies (IWPT '93) Tilburg.

Bod, Rens. 1995. Enriching Linguistics with Statistics. Ph.D. Thesis. Department of Computational Linguistics. University of Amsterdam.

Boguraev, Bran, and Ted Briscoe. 1987. Large Lexicons for Natural Language Processing: Exploring the Grammar Coding System of LDOCE. Computational Linguistics 13 (3-4): 203-218.

Boguraev, Bran, and Ted Briscoe eds. 1989. Computational Lexicography for Natural Language Processing. London: Longman.

Brill, Eric. 1992. A simple rule-based part of speech tagger. Proceedings of the Third Conference on Applied Natural Language Processing, ACL, Trento, Italy, 1992.

Brill, Eric. 1993. A Corpus-Based Approach to Language Learning. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania, 1993.

Brill, Eric. 1994. Some advances in rule-based part of speech tagging. Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, Wa., 1994.

Briscoe, Ted, and John Carroll. 1991. Generalised Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars. Cambridge: University of Cambridge Computer Laboratory, Technical Report No. 224.

Briscoe, Ted and Nick Waegner. 1992. Robust stochastic parsing using the inside-outside algorithm. In Proceedings of the AAAI workshop on Statistically-based NLP techniques. July 12-16. San Jose. 39-53.

Briscoe, Ted. 1994. Robust statistical parsing techniques. In Corpus-based research into language. N. Oostdijk and P. de Haan eds. 97-119. Amsterdam: Rodopi.

Burnage, Gavin. 1990. CELEX - A Guide for Users. Nijmegen: Centre for Lexical Information (CELEX).

Butler, C.S. 1985. Systemic Linguistics: Theory and Applications. London: Batsford.

Carroll, John and Claire Grover. 1989. The derivation of a large computational lexicon for English from LDOCE. In Computational Lexicography for Natural Language Processing. Bran Boguraev and Ted Briscoe eds. 117-134. London: Longman.

Chomsky, Noam. 1957. Syntactic Structures. The Hague: Mouton.

Church, K. 1988. A stochastic parts program and noun phrase parser for unrestricted text. Proceedings of the 2nd Conference on Applied Natural Language Processing. ACL. pp. 136-143.

Cowie, A. and R. Mackin. 1975. Oxford Dictionary of Current Idiomatic English. Vols. 1 & 2. Oxford: OUP.

Davey, A. 1974. Discourse Production. Ph.D. thesis, University of Edinburgh. Published by Edinburgh University Press in 1978.

Davey, A. 1978. Discourse Production: A computer model of some aspects of a speaker. Edinburgh: Edinburgh University Press.

Demetriou, George C. and Eric S. Atwell. Machine-Learnable, Non-Compositional Semantics for Domain Independent Speech or Text Recognition. In Proceedings of 2nd Hellenic-European Conference on Mathematics and Informatics (HERMIS), Athens University of Economics and Business. 1994.

Dowty, David R., Robert Wall and Stanley Peters. 1981. Introduction to Montague Semantics. Dordrecht: Reidel.

Ellegård, A. 1978. The Syntactic Structure of English Texts: A computer-based study of four kinds of text in the Brown University Corpus. Gothenburg Studies in English. 43. Gothenburg.

Fawcett, Robin P., and Michael Perkins. 1980. Child Language Transcripts 6-12. (With a preface, in 4 volumes). Department of Behavioural and Communication Studies, Polytechnic of Wales.

Fawcett, Robin P. 1980. Language Development in Children 6-12: Interim Report. Linguistics 18 pp. 953-958.

Fawcett, Robin P. 1980. Cognitive Linguistics and Social Interaction. Julius Groos Verlag.

Fawcett, Robin P. 1981. Some Proposals for Systemic Syntax. Journal of the Midlands Association for Linguistic Studies (MALS). 1.2, 2.1, 2.2 (1974-76). Re-issued with light amendments, 1981, Department of Behavioural and Communication Studies, Polytechnic of Wales.

Fawcett, Robin P. 1984. System networks, codes and knowledge of the universe. In Fawcett, R., Halliday, M.A.K., Lamb, S.M. and Makkai, A. (eds.) The Semiotics of Culture and Language. London, Pinter. pp. 135-179.

Fawcett, Robin P. and Gordon Tucker. 1987. What a parser needs to know about twenty main verb forms. Mimeo. SESCP, UWCC.

Fawcett, Robin P. 1988a. A note on the relationship between the syntactic categories used in (1) the analysis of the Polytechnic of Wales Corpus and (2) generation and analysis in the COMMUNAL project. (personal communication)

Fawcett, Robin P. 1988b. The English Personal Pronouns: An Exercise in Linguistic Theory. In Linguistics in a systemic perspective. J. Benson, M. Cummings and W. Greaves eds. 185-220. Amsterdam/Philadelphia: John Benjamin.

Fawcett, Robin P. and Gordon H. Tucker. 1989. Prototype Generators 1 and 2. COMMUNAL Report No. 10, Computational Linguistics Unit, University of Wales College of Cardiff.

Fawcett, Robin P. and Gordon H. Tucker 1990. Demonstration of GENESYS: a very large semantically based systemic functional grammar. In Proceedings of COLING 90 Vol. 1. Helsinki. pp. 47-9.

Fawcett, Robin P. 1990. The computer generation of speech with semantically and discoursally motivated intonation. In Proceedings of 5th International Workshop on Natural Language Generation. Pittsburgh. pp. 164-73a.

Fawcett, Robin P. 1992. The state of the craft in computational linguistics: a generationist's viewpoint. COMMUNAL Working Papers No. 2. Computational Linguistics Unit, University of Wales College of Cardiff.

Fawcett, Robin P., Gordon Tucker and Yuen Q. Lin. 1993. How a systemic functional grammar works: the role of realisation in realisation. In Horacek, H and M. Zock (eds.) New Concepts in NL Generation: Planning, Realisation and Systems. London, Pinter. 114-186.

Fawcett, Robin P. 1994. A generationist approach to grammar reversibility in NLP. In Tomek Strzalkowski (ed.). Reversible Grammar in NLP. 1994. Kluwer. 365-413

Frege, Gottlob. 1952. Ueber Sinn und Bedeutung. In P. Geach and M. Black eds. Translations from the Philosophical Writings of Gottlob Frege. 56-78. Oxford: Blackwell.

Garside, Roger, Geoffrey Leech and Geoffrey Sampson eds. 1987. The Computational Analysis of English. London and New York: Longman.

Garside, Roger. 1987. The CLAWS word-tagging system. In The Computational Analysis of English. Roger Garside, Geoffrey Leech and Geoffrey Sampson eds. 30-41. London and New York: Longman.

Gazdar, Gerald, Ewan Klein, Geoff Pullum and Ivan Sag. 1985. Generalised Phrase Structure Grammar. Oxford: Blackwell.

Grover, Claire, Ted Briscoe, John Carroll and Bran Boguraev. 1989. The ALVEY natural language tools grammar (second release). Technical Report 162. Computer Laboratory, University of Cambridge.

Halliday, Michael. 1961. Categories of the theory of grammar. Word (17) 241-92.

Halliday, Michael. 1985. An Introduction to Functional Grammar. London: Edward Arnold.

van Halteren, Hans, and Theo van den Heuvel. 1990. Linguistic Exploitation of Syntactic Databases: the use of the Nijmegen Linguistic Database program. Amsterdam: Rodopi.

Heidorn, George. 1982. Experience with an easily computed metric for ranking alternative parses. Proceedings of the 20th Annual Meeting of the ACL. 82-84.

Hindle, Donald. 1983. User Manual for Fidditch. Technical memorandum 7590-142, US Naval Research Laboratory.

Hornby, A, and Tony Cowie eds. 1974. Oxford Advanced Learners' Dictionary of Current English. Oxford: Oxford University Press.

Houghton, G. and S. Isard. 1988. Why to speak, what to say and how to say it: modelling language production in discourse. In P. Morris (ed.) Modelling Cognition. Chichester: Wiley. pp. 112-30.

Hughes, John. 1994. Automatically Acquiring a Classification of Words. Ph.D. Thesis. School of Computer Studies, The University of Leeds.

Hughes, John and Eric S. Atwell. (forthcoming). The AMALGAM Project: Annotating the Spoken English Corpus with several annotation schemes. To appear in Magnus Ljung (ed.) Proceedings of the 17th ICAME Conference, Stockholm, May 15-19th, 1996. Rodopi Press.

Jensen, Karen, and George Heidorn. 1983. The Fitted Parse: 100% parsing capability in a syntactic grammar of English. Proceedings of the Conference on Applied NLP, ACL. 93-98.

Jensen, K., G. Heidorn and S.D Richardson (eds.). 1993. NLP: the PLNLP Approach. Boston: Kluwer.

Johansson, Stig, Eric Atwell, Roger Garside, and Geoffrey Leech. 1986. The Tagged LOB Corpus. University of Bergen, Norway: Norwegian Computing Centre for the Humanities.

Jost, Uwe and Eric Atwell. 1994. A hierarchical mutual-information based probabilistic language model. In L. Evett and T. Rose eds. Computational Linguistics for Speech and Handwriting Recognition. AISB Workshop Series 1994, University of Leeds.

Jost, Uwe. 1994. Probabilistic Language Modelling for Speech Recognition. M.Sc. dissertation. School of Computer Studies, University of Leeds.

Karlsson, Fred. 1994. Robust parsing of unconstrained text. In Corpus-based research into language. N. Oostdijk and P. de Haan eds. 121-142. Amsterdam: Rodopi.

Karlsson, F. 1995. Designing a parser for unrestricted text. In Karlsson, F., A Voutilainen, J. Heikkila and A. Anttila. (eds.) 1995. Constraint Grammar. Berlin/New York: Mouton De Gruyter. pp. 1-40.

Karlsson, F., A Voutilainen, J. Heikkila and A. Anttila. (eds.) 1995. Constraint Grammar. Berlin/New York: Mouton De Gruyter.

Kasper, Robert T. 1988. An experimental parser for systemic grammars. ISI Reprint Series 88-212, University of Southern California.

Kasper, Robert T. 1989. Unification and classification: an experiment in information-based parsing. In Proceedings of International Workshop on Parsing Technologies. Pittsburgh, PA.

Kay, Martin. 1985. Parsing in Functional Unification Grammar. In D. Dowty, L. Karttunen and A. Zwicky eds. Natural Language Parsing. Cambridge: CUP.

Keenan, Frank. 1993. Large-vocabulary syntactic analysis for text recognition. Ph.D. Thesis. Department of Computing. Nottingham Trent University.

Kempen, Gerard, and Theo Vossen. 1988. Incremental syntactic tree formation in human sentence processing. NICI, University of Nijmegen. Appeared in Cahiers de la Fondation Archives Jean Piaget, 1989. Geneva.

Keulen, F. 1986. The Dutch Computer Corpus Pilot Project. In Corpus Linguistics II. J. Aarts and W. Meijs eds. 127-161. Amsterdam: Rodopi.

Kirkpatrick, S., C.D. Gelatt and M.P. Vecchi. 1983. Optimization by simulated annealing. Science 220: 671-80.

Knowles, Gerry, and Lita Lawrence. 1987. Automatic Intonation Assignment. In The Computational Analysis of English. Roger Garside, Geoffrey Leech and Geoffrey Sampson eds. 139-148. London and New York: Longman.

Koch, Heinz-Detlev and Richard F. E. Sutcliffe. eds. 1995. Proceedings of the International Workshop on Industrial Parsing of Software Manuals. Limerick, May 4-5 1995.

Kwasny, S. and Norman Sondheimer. 1981. "Relaxation techniques for parsing grammatically ill-formed input in natural language understanding systems" in American Journal of Computational Linguistics 7(2): 99-108.

van Laarhoven, Peter J. M. and Emile H. L. Aarts. 1987. Simulated Annealing: theory and applications. Dordrecht: Reidel.

Lari, K. and S. Young. 1990. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. Computer Speech and Language 4: 35-56.

Leech, Geoffrey, Roger Garside, and Eric Steven Atwell. 1983. The Automatic Grammatical Tagging of the LOB Corpus. In Newsletter of the International Computer Archive of Modern English (ICAME NEWS) 7: 13-33, Norwegian Computing Centre for the Humanities, Bergen University.

Leech, Geoffrey, and Roger Garside. 1991. Running a grammar factory: the production of syntactically analysed corpora or `treebanks'. In English Computer Corpora: Selected Papers and Research Guide. S. Johansson and A.-B. Stenström eds. 15-32. Berlin: Mouton de Gruyter. Lyons, John. 1970. Chomsky. London: Fontana-Collins.

Magerman, D. 1994. Natural Language Parsing as Statistical Pattern Recognition. Ph.D. Thesis. Stanford University, USA.

Magerman, D. and M. Marcus. 1991. Pearl: a probabilistic chart parser. In 2nd International Workshop on Parsing Technologies. Cancun, Mexicon. 193-199.

Mann, William C. 1983. A linguistic overview of the Nigel Text Generation Grammar. ISI Reprint Series 83-9, University of Southern California. de Marcken, C.G. 1990. Parsing the LOB corpus. In Proceedings of the ACL Meeting, Pittsburgh PA. 243-251.

de Marcken, C. 1990. Parsing the LOB corpus. Proceedings of the 28th Annual Meeting of the ACL. 243-251.

Marcus, M.P. and B. Santorini. 1991. Building very large natural language corpora: the Penn Treebank. CIS report. University of Pennsylvania.

Marcus, M.P., B. Santorini and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. Computational Linguistics 19 (2). 313-330.

Matthiessen, C. M. I. M, and J. A. Bateman. 1991. Text Generation and Systemic-Functional Linguistics. London: Pinter.

McCord, Michael. 1990. Slot Grammar: A system for simpler construction of practical NL grammars. In R. Studer (ed.). Natural Language and Logic (Lecture Notes in AI: 459). Berlin: Springer-Verlag. pp. 118-145.

Meijs, Willem. 1993a. Exploring Lexical Knowledge. In Corpus-based Computational Linguistics. 249-260. C. Souter and E. Atwell eds. Amsterdam: Rodopi Press.

Meijs, Willem. 1993b. Analyzing nominal compounds with the help of a computerized lexical knowledge system. In English Language Corpora: Design Analysis and Exploitation. 299-312. J. Aarts, P. de Haan and N. Oostdijk eds. Amsterdam: Rodopi Press.

O'Donnell, Michael. 1993. Reducing Complexity in a Systemic Parser. Proceedings of the 3rd ACL/SIGPARSE International Workshop on Parsing Technologies (IWPT3),Tilburg and Durbuy, August 10-13th 1993. 203-218.

O'Donnell, Michael. 1994. Sentence Analysis and Generation: A Systemic Perspective. Ph.D. Thesis, Department of Linguistics, University of Sydney, Australia.

O'Donoghue, Tim F. 1990. The theory behind REVELATION1: a semantic interpreter for systemic grammars. Research Report 90.28. School of Computer Studies, University of Leeds.

O'Donoghue, Tim F. 1991a. An Expert System for semantic interpretation in Systemic grammar. Research Report 91.7. School of Computer Studies, University of Leeds.

O'Donoghue, Tim F. 1991b. EPOW: The Edited Polytechnic of Wales Corpus. Research Report 91.11. School of Computer Studies, University of Leeds. Also appeared in Proceedings of the 5th International Conference on Symbolic and Logical Computing. Dakota State University, USA, April 1991.

O'Donoghue, Tim F. 1991c. Taking a parsed corpus to the cleaners: the EPOW corpus. ICAME Journal 15: 55-62.

O'Donoghue, Tim F. 1991d. The Vertical Strip Parser: A lazy approach to parsing. Research Report 91.15, School of Computer Studies, University of Leeds.

O'Donoghue, Tim F. 1991e. A semantic interpreter for systemic grammars. In Tomek Strzalkowski (ed.). Reversible Grammar in Natural Language Processing: Proceedings of a workshop sponsored by the Special Interest Groups on Generation and Parsing of the Association for Computational Linguistics. 129-138. Morristown, New Jersey: ACL.

O'Donoghue, Tim F. 1993. Reversing the process of generation in Systemic Grammar. Ph.D. thesis. School of Computer Studies, Leeds University.

O'Donoghue, Tim F. 1994. Semantic Interpretation in a Systemic Functional Grammar. In Tomek Strzalkowski (ed.) 1994. Kluwer.

Oostdijk, Nelleke. 1991. Corpus Linguistics and the Automatic Analysis of English. Amsterdam: Rodopi Press.

Osborne, Miles. 1995. Parsing Computer Manuals using a Robust Alvey NL Toolkit. In Koch and Sutcliffe (eds.) International Workshop on Industrial Parsing of Software Manuals, Limerick, 4-5th May 1995.

Patten, T. 1988. Systemic Text Generation as Problem Solving. Cambridge: CUP.

Pereira F. and Y. Schabes. 1992. Inside-Outside re-estimation for partially bracketed corpora. In 30th Annual Meeting of the ACL. Newark, Delaware. 128-135.

Phillips, J. D., and Henry S. Thompson. 1987. A parsing tool for the Natural Language Theme: version 13. Software paper no. 5. Department of Artificial Intelligence, Edinburgh University.

Piepenbrock, Richard. 1993. A Longer Term View on the Interaction between Lexicons and Text Corpora in Language Investigation. In Corpus-based Computational Linguistics. 59-70. C. Souter and E. Atwell eds. Amsterdam: Rodopi Press.

Pocock, Rob and Eric Atwell. 1993. Probabilistic Grammatical Models For Treebank-Trained Lattice Disambiguation. Research Report 93.30. School of Computer Studies, Leeds University. Also available in Eric Atwell. 1994. The Speech Oriented Probabilistic Parsing Project: Final Report. School of Computer Studies, University of Leeds.

Pulman, S.G., G.J. Russell, G.D. Ritchie and A.W. Black. 1989. Computational morphology of English. Technical Report 155. Computer Laboratory, University of Cambridge.

Procter, Paul. 1978. Longman Dictionary of Contemporary English. London: Longman.

Quirk, R., S. Greenbaum, G. Leech and J. Svartvik. 1972. A Comprehensive Grammar of the English Language. London and New York: Longman.

Radford, Andrew. 1981. Transformation syntax. Cambridge: CUP.

Sampson, Geoffrey. 1987a. The Grammatical Database and Parsing Scheme. In The Computational Analysis of English. Roger Garside, Geoffrey Leech and Geoffrey Sampson (eds.). 82-96. London and New York: Longman.

Sampson, Geoffrey. 1987b. Evidence against the "grammatical"/"ungrammatical" distinction. In Corpus Linguistics and Beyond. Willem Meijs ed. 219-226. Amsterdam: Rodopi.

Sampson, Geoffrey, Robin Haigh and Eric S. Atwell. 1989. Natural language analysis by stochastic optimization: a progress report on Project APRIL. Journal of Experimental and Theoretical Artificial Intelligence 1: 271-287.

Sampson, Geoffrey. 1990. Notes for Corpus Meeting, Wadham College, Oxford, January 1990. In G. Leech ed. Proceedings of a workshop on Corpus Resources. 13-15. DTI/SERC Speech and Language Technology Club.

Sampson, Geoffrey. 1992. Analysed Corpora of English: a consumer guide. In Computers in Applied Linguistics. M.C. Pennington and V. Stevens eds. 181-200. Multilingual Matters.

Sampson, Geoffrey. 1994. SUSANNE: A Domesday Book of English Grammar. In Corpus-based Research into language. N. Oostdijk and P. de Haan eds. 169-188. Amsterdam: Rodopi.

Sekine, S. and R. Grishman. 1995. A corpus-based probabilistic grammar with only two non-terminals. Proceedings of the ACL SIGPARSE Fourth International Workshop on Parsing Technologies (IWPT '95), Prague and Karlovy Vary. Sept 1995. 216-223.

Sharman, Richard. 1990. Hidden Markov models for word tagging. Technical Report 214. Winchester: IBM UK Scientific Centre.

Sinclair, John McH. ed. 1987. Looking Up: An account of the COBUILD project in lexical computing. London: Collins COBUILD.

Souter, Clive and Eric Atwell. 1988a. Constraints on Legal Syntactic Configurations. COMMUNAL Research Report No. 14. School of Computer Studies, The University of Leeds.

Souter, Clive and Eric Atwell. 1988b. Morphological Analysis. COMMUNAL Research Report No. 16. School of Computer Studies, The University of Leeds.

Souter, Clive. 1989a. The COMMUNAL Project: Extracting a grammar from the Polytechnic of Wales corpus. ICAME Journal 13: 20-27.

Souter, Clive. 1989b. A Short Handbook to the Polytechnic of Wales Corpus. ICAME, Norwegian Computing Centre for the Humanities, P.O. Box 53, Bergen University, N-5027 Bergen, Norway.

Souter, Clive. 1990a. Systemic Functional Grammars and Corpora. In J. Aarts and W. Meijs eds. Theory and Practice in Corpus Linguistics. 179-211, Rodopi Press, Amsterdam.

Souter, Clive. 1990b. Probabilistic Parsing and the CELEX lexicon. CELEX Newsletter 5: 9-11, Centre for Lexical Information, University of Nijmegen, The Netherlands.

Souter, Clive. 1990c. Corpus Linguistics: the State of the Science. In Gerhard Leitner ed. Computer Corpora des Englischen. (CCE) Newsletter 4: 1-15, Institut fur Englische Philologie, Freie Universitat Berlin, Germany.

Souter, Clive and Tim O'Donoghue. 1991. Probabilistic Parsing in the COMMUNAL Project. In Stig Johansson and Anna-Brita Stenström eds. English Computer Corpora, Selected Papers and Research Guide. 33-48, Berlin: Mouton de Gruyter.

Souter, Clive. 1992. The Nijmegen Linguistic Database program. ICAME Journal 16: 70-79.

Souter, Clive and Eric Atwell. 1992. A Richly Annotated Corpus for Probabilistic Parsing. Research Report 92.13, School of Computer Studies, University of Leeds. Also appeared in Proceedings of AAAI workshop on Statistically-Based NLP Techniques, San Jose, California, July 12-17, 1992.

Souter, Clive. 1993a. Harmonising a lexical database with a corpus-based grammar. In C. Souter and E. Atwell eds. Corpus-based Computational Linguistics. 181-193, Amsterdam: Rodopi Press.

Souter, Clive. 1993b. Towards a Standard Format for Parsed Corpora. In Jan Aarts, Pieter de Haan and Nelleke Oostdijk eds. English Language Corpora: Design, Analysis and Exploitation. 197-214, Amsterdam: Rodopi Press.

Souter, Clive and Eric Atwell eds. 1993. Corpus-Based Computational Linguistics. Amsterdam: Rodopi Press.

Souter, Clive. 1994. Using parsed corpora: a review of current practice. In Corpus-based Research into language. N. Oostdijk and P. de Haan eds. 143-158. Amsterdam: Rodopi.

Strzalkowski, Tomek. (ed.) 1994. Reversible Grammar in NLP. Kluwer.

Tomita, M. 1991. Generalised LR Parsing. Kluwer Academic Publishers.

Weerasinghe, A. Ruvan. 1990. A chart parsing approach to building a systemic parser for a non-trivial subset of English. M.Sc. Dissertation. Computational Linguistics Unit, University of Wales College of Cardiff.

Weerasinghe, A Ruvan and Robin P. Fawcett. 1993. Probabilistic Incremental Parsing in Systemic Functional Grammar. Proceedings of the 3rd ACL/SIGPARSE International Workshop on Parsing Technologies (IWPT3),Tilburg and Durbuy, August 10-13th 1993. Appendix 1.

Weerasinghe, A. Ruvan. 1994. Probabilistic Parsing in SFG. Ph.D. thesis. School of Computing Mathematics, University of Wales College of Cardiff.

Wilks, Yorick., Dan Fass, Cheng-Ming Guo, James E. McDonald, Tony Plate and Brian M. Slator. 1988. Machine Tractable Dictionaries as Tools and Resources for Natural Language Processing. In Proceedings of the 12th International Conference on Computational Linguistics (COLING '88), Budapest, 750-755.

Willis, Tim. (1996) Annotating the Spoken English Corpus with the Nijmegen ICE parser. In Proceedings of the 16th ICAME Conference, Toronto, May 1995.

Winograd, Terry. 1972. Understanding Natural Language. San Diego: Academic Press.

Winograd, Terry. 1983. Language as a Cognitive Process. Reading, Mass. Addison-Wesley.

Zipf, George K. 1936. The psycho-biology of language: an introduction to dynamic philology. George Routledge.

Appendices.

Appendix 1. Sample Fragments of Parsed Corpora. 166
Appendix 1.1 Lancaster/Leeds Treebank. 166
Appendix 1.2 Nijmegen Corpus (CCPP). 167
Appendix 1.3 Polytechnic of Wales Corpus. 168
Appendix 1.4 Susanne Corpus. 169
Appendix 1.5 IBM/Lancaster Spoken English Corpus. 170
Appendix 2. A Brief Description of the POW Corpus. 171
Appendix 3. Systemic-Functional Syntax Categories in the POW Corpus. 172
Appendix 4. A Mapping from LDOCE to POW SF Syntax Tags. 175
Appendix 5. A Fragment of a Context-Free SF Syntax Maintaining the Distinction between Filling and Componence. 179
Appendix 6. A Fragment of a Context-Free SF Syntax Ignoring the Distinction between Filling and Componence. 180
Appendix 7. A Fragment of a Vertical Trigram Model from the POW Corpus. 181
Appendix 8. Rule and Word-Wordtag Frequency Distribution in the POW Corpus. 182
Appendix 9. A Prototype Competence Systemic Functional Syntax. 183
Appendix 10. Brill Tagger Context Rules Learned from POW 186
Appendix 11. General lexical tagging rules used by the Brill tagger for untrained words. 187
Appendix 12. The Reduced EPOW Filling Grammar. 188
Appendix 13. The 100 Most Frequent Word-Wordtag Pairs in the EPOW Lexicon. 189
Appendix 14. Pocock and Atwell's Weight-Driven Chart Parser. 190
Appendix 15. Parser Version 7: Test Results. 191

Appendix 1. Sample Fragments of Parsed Corpora.

Appendix 1.1 Lancaster/Leeds Treebank.

R01 69 001

[S[Np[NP[Jones]NP][NNS[ideas]NNS]Np][Vb[BER[are]BER]Vb][Nw[RN[now]RN]Nw][J[QL[so]QL][RB[firmly]RB][JJ[established]JJ][RB[abroad]RB][Fc[CS[that]CS][Fa[CSA[as]CSA][Np[JJ[primitive]JJ][NNS[states]NNS]Np][V[VB[develop]VB]V]Fa][,[,],][Ni[PP3[it]PP3]Ni][Vzeb[BEZ[is]BEZ][XNOT[not]XNOT]Vzeb][Ns[ATI[the]ATI][JJ[old]JJ][NN[country]NN]Ns][Fr[Pq[IN[on]IN][WDT[which]WDT]Pq][Nap[PP3AS[they]PP3AS]Nap][V[VB[model]VB]V][Nop[PPLS[themselves]PPLS]Nop]Fr][,[,],][Nns+[CC[but]CC][ATI[the]ATI][JJ[new]JJ][NP[Jones]NP]Nns+]Fc]J][.[.].]S]

R01 72 001

[S[P[IN[in]IN][NP[Africa]NP]P][,[,],][Np[NP[Jones]NP][NNS[hotels]NNS]Np][V[VB[spring]VB]V][R[RP[up]RP]R][R[RB[even]RB]R][Fa[CSA[as]CSA][Ns[ATI[the]ATI][NPT[Prime]NPT][NPT[Minister]NPT][JJ[elect]JJ]Ns][Vzp[BEZ[is]BEZ][BEG[being]BEG][VBN[let]VBN]Vzp][P[IN=[IN21[out]IN21][IN22[of]IN22]IN=][NN[prison]NN]P]Fa][.[.].]S]

R01 73 072

[S[P[IN[in]IN][Ns[ATI[the]ATI][JJ[middle]JJ][NR[east]NR]Ns]P][,[,],][Np[NN[oil]NN][NNS[royalties]NNS]Np][Vp[BER[are]BER][VBN[turned]VBN]Vp][P[IN[into]IN][Np[NP[Jones]NP][NNS[amenities]NNS][,[,],][P[IN=[IN21[such]IN21][IN22[as]IN22]IN=][N&[NN[ice]NN][,[,],][N-[JJ[big]JJ][NNS[cars]NNS]N-][,[,],][N+[CC[and]CC][NNS[night-clubs]NNS][Fr[Nq[WP[that]WP]Nq][Veb[MD[would]MD][XNOT[not]XNOT][BE[be]BE]Veb][P[IN=[IN21[out]IN21][IN22[of]IN22]IN=][NN[place]NN]P][P[IN[on]IN][Nns[NP[Miami]NP][NPL[Beach]NPL]Nns]P]Fr]N+]N&]P]Np]P][.[.].]S]

R01 75 112

[S[P[IN[in]IN][NP[Brazil]NP]P][,[,],][Ncs[AT[an]AT][J[RB[entirely]RB][JJ[new]JJ]J][NN[capital]NN]Ncs][Vzp[HVZ[has]HVZ][BEN[been]BEN][VBN[hacked]VBN]Vzp][P[IN=[IN21[out]IN21][IN22[of]IN22]IN=][Ns[ATI[the]ATI][NN[jungle]NN]Ns]P][P[IN[as]IN][Ncs[AT[a]AT][JJ[living]JJ][NN[monument]NN][P[IN[to]IN][N&[NP[Jones]NP][N+[CC[and]CC][ABN[all]ABN][Fr[Nas[PP3A[he]PP3A]Nas][Vz[VBZ[stands]VBZ]Vz][Ps[INF[for]INF]Ps]Fr]N+]N&]P]Ncs]P][.[.].]S]

R01 78 001

[S[Np[JJ[foreign]JJ][NNS[visitors]NNS]Np]S]

R02 79 001

[S&[Nas[PP3A[he]PP3A]Nas][V[VBD[felt]VBD]V][P[IN[in]IN][Ns[PP$[his]PP$][NN[jacket]NN][NN[pocket]NN]Ns]P][S+[CC[and]CC][V[VBD[pulled]VBD]V][R[RP[out]RP]R][Ncs[AT[a]AT][NN[key]NN][NN[ring]NN]Ncs]S+][.[.].]S&]

Appendix 1.2 Nijmegen Corpus (CCPP).

0800104 C.[9500 IN 9102 THE 2103 DARK 4103 DAYS 3203 OF 9104 WINTER, 3101
0800104 THIS 5101 IS E201 THE 2103 KIND 3103 OF 9104 WIMBLEDON 9905 DAY 3102
0800104 ONE 7703
0800105 DREAMS A203 ABOUT. 8800 IT 0201 'S F201 ARRIVED- A801 THE 2102
0800105 PERFECT 4102 SETTING 3102 FOR 9103 FINALS 9904 DAY. 3100 A 2502
0800106 SUN 3102 VERY 8104 STRONG 4104 INDEED, 8103 BUT 6103 NOT 8504 TOO 8104
0800106 HOT 4101 , 6501 VERY 8103 LITTLE 2802 WIND 3101 , 6501
0800106 SHIRT-SLEEVED 4202
0800107 CROWD 3102 SETTLING A903 DOWN 8803 TO 9204 ENJOY B704 THIS 2105
0800107 FIRST-EVER 7605 OPEN 9906 CHAMPIONSHIP 9906 FINAL 3105
0800108 AT 9106 WIMBLEDON. 3500 FITTING, 4101 OF 9902 COURSE, 8101 THAT 6302
0800108 ROD 9905 LAVER, 3504 THE 2106 WORLD'S 3306 NUMBER 9907
0800109 ONE 9907 PLAYER, 3105 THE 2107 FAVOURITE 3106 AND 6106 THE 2107
0800109 NUMBER 9908 ONE 9908 SEED 3103 HAS F203 WON B803 HIS 2004 WAY 3104
0800110 THROUGH 8303 TO 9104 THIS 2105 FINAL. 3100 TWENTY-NINE 7602 YEARS 3202
0800110 OF 9103 AGE 3101 , 6501
0800111 COMES A202 FROM 9103 QUEENSLAND... 3500
0800112 THE 2102 WORLD'S 3302 LEADING 4102 PLAYER, 3101 REALLY, 8101 SINCE 9102
0800112 THE 2103 END 3103 OF 9104 1965 7203 WHEN 8904 HE 5104 TOOK B304
0800113 OVER 8804 THE 2105 TOP-BILLING 3104 FROM 9105 KEN 9906 ROSEWALL. 3500
0800114 LAVER 3501 PLAYING A901 THIS 2103 AFTERNOON 3101 IN 9102 HIS 2003
0800114 SIXTH 7603 FINAL 3103 AT 9104 WIMBLEDON. 3500 AS 9302 AN 2503
0800115 AMATEUR, 3101 HE 5101 REACHED B301 THE 2102 FINAL... 3101 FOR 9102
0800115 FOUR 7603 YEARS 3203 IN 9104 SUCCESSION; 3100 THAT 5102
0800116 WAS E302 1959 7251 , 6501 1960 7252 HE 5102 WAS E302 RUNNER-UP 3101
0800116 , 6501 IN 9103 61 7202 HE 5102 WON B302 THE 2103 TITLE 3102 WHEN 6303
0800116 HE 5104 BEAT B304
0800117 ER 1104 MACKINLEY 3504 IN 9105 THE 2106 FINAL 3101 , 6501 IN 9103
0800117 62 7202 HE 5102 RETAINED B302 THE 2103 TITLE, 3102 BEATING B903
0800117 MARTIN 9904
0800118 MULLIGAN. 3500 OF 9902 COURSE 8101 ER 1101 HIS 2002 FIFTH 7602
0800118 APPEARANCE 3102
0800119 IN 9103 A 2504 WIMBLEDON 9905 FINAL 3101 WAS E301 IN 9102
0800120 THE 2103 FIRST-EVER 7603 PROFESSIONAL 4103 TOURNAMENT 3103 AT 9104
0800120 WIMBLEDON 3503 SPONSORED A804 BY 9105 THE 2106
0800121 BBC, 3101 LAST 4103 SUMMER, 3101 WHEN 6302 LAVER 3503 WON B303 THE 2104
0800121 PROFESSIONAL 4104 CHAMPIONSHIP. 3100 HE 5101 'S E201
0800122 HERE 8101 TODAY 8101 IN 9102 HIS 2003 SIXTH 7603 FINAL. 3100 THE 2103
0800122 OPPOSITE 4103 SIDE 3103 OF 9104 THE 2105 NET 3101 ANOTHER 4102
0800123 PROFESSIONAL, 3101 OF 9902 COURSE, 8101 TONY 9903 ROCHE- 3502 BORN A803
0800123 IN 9104 TARCUTTA, 3505 WHICH 5406 IS E206 A 2507
0800124 NEW 9908 SOUTH 9908 WALES 9908 SHEEPTOWN. 3100 TWENTY-THREE 7603
0800124 YEARS 3203 OF 9104 AGE 3102 , 6502 SEEDED E803 FIFTEEN, 7701
0800125 HE 5101 REALLY 8101 CLEARED B301 UP 8801 THE 2102 WAY 3102 TO 9103
0800125 THE 2104 FINAL 3101 FOR 9102 HIMSELF 5001 WHEN 6302 HE 5103 KNOCKED B303
0800126 KEN 9905 ROSEWALL, 3504 THE 2105 NUMBER 9906 TWO 9906 SEED, 3103
0800127 OUT 9905 OF 9104 THE 2105 CHAMPIONSHIP 3103 ON 9104 THE 2105 NUMBER 9906
0800128 ONE 9906 COURT 3103 THE 2105 OTHER 4105 DAY. 3100 THE 2102 OTHER 4102
0800128 SEEDED 4202 PLAYER 3102 HE 5103 BEAT A303 ON 9104 THE 2105 WAY 3101
0800129 WAS E301 BUTCH 9903 BUCHHOLZ, 3502 THE 2103 NUMBER 9904 TEN 9904
0800129 SEED 3103 FROM 9104 THE 2105 UNITED 9906 STATES; 3600 AND, 8201
0800130 OF 9902 COURSE, 8101 IN 9102 THE 2103 SEMI-FINAL 3101 HE 5101 BEAT B301
0800130 TOM 9902 GRAEBNER. 3500 [[9400

Appendix 1.3 Polytechnic of Wales Corpus.

**** 58 1 1 1 0 59

6abicj

1 [FS:Y...] Z 1 CL F YEAH 1 CL 2 S NGP 3 DD THAT 3 HP ONE 2 OM 'S 2 C NGP 4 DQ A 4 H RACING-CAR
2 Z CL 1 S NGP 2 DD THAT 2 HP ONE 1 OM 'S 1 C NGP 3 DQ A 3 MO QQGP AX LITTLE 3 H TRUCK
3 [HZ:WELL] Z 1 CL 2 S NGP HP I [RP:I] 2 AI JUST 2 M HAD 2 C NGP 3 DQ A 3 MO QQGP AX LITTLE 3 H THINK 1 CL 4 & THEN 4 S NGP HP I 4 M THOUGHT 4 C CL 5 BM OF 5 M MAKING 5 C NGP 6 DD THIS 6 HP ONE
4 Z 1 CL 2 S NGP HP I 2 AI JUST 2 M FINISHED 2 C NGP 3 DD THAT 3 HP ONE 1 CL 4 & AND 4 S NGP HN FRANCIS 4 M HAD 4 C NGP 5 DD THE 5 H IDEA 5 Q CL 6 BM OF 6 M MAKING 6 C NGP 7 DQ A 7 H RACING-CAR
5 [FS:THEN-I] Z CL 1 & THO 1 S NGP HP I 1 M MADE 1 C NGP DD THIS
6 Z CL 1 & THEN 1 S NGP HP FRANCIS 1 OX WAS 1 AI JUST 1 X GOING-TO 1 M MAKE 1 C NGP HP ONE 1 A CL 2 B WHEN 2 S NGP H YOU 2 M CAME 2 CM QQGP AX BACK 2 CM QQGP AX IN
7 [NV:MM] Z 1 CL F NO [FS:FRAN...] 1 CL 2 S NGP HP WE 2 M HAD 2 C NGP 3 DQ AN 3 H IDEA 3 Q CL 4 BM OF 4 M MAKING 4 C NGP 5 DQ FOUR 5 H THINGS
8 Z 1 CL F YEAH 1 CL 2 S NGP HP I 2 M PLAYED 2 C PGP 3 P WITH 3 CV NGP HP IT 2 A PGP 4 P AT 4 CV NGP H HOME
9 Z CL F YEAH
10 [FS:I] [FS:I] Z 1 CL F NO 1 CL 2 S NGP HP I 2 OX 'VE 2 AI JUST 2 M GOT 2 C NGP 3 DQ ONE 3 MO QQGP AX BIG 3 H TIN [FS:OF?] 3 Q QQGP 4 AX FULL 4 SC PGP 5 P OF 5 CV NGP HP IT
11 [NV:ER] Z CL 1 (S) 1 (M) 1 C NGP 2 DQ NGP 3 DQ ALL 3 H SORTS 2 VO OF 2 H THINGS
12 Z 1 CL 2 S NGP HP I 2 M MAKE 2 C NGP H CARS 2 A QQGP AX ALWAYS 1 CL 3 & AND 3 A SOMETIMES 3 S NGP HP I 3 M MAKE 3 C NGP H HOUSES 1 CLUN & AND
13 Z 1 CL F YEAH 1 CL 2 S NGP HP I 2 M GOT 2 C NGP HN KERPLUNK
14 [FS:IT] [FS:IT] [NV:UM] Z 1 CL 2 S NGP HP YOU 2 M PUT 2 C NGP H STRAWS 2 C PGP 3 PM INTO 3 CV NGP 4 DQ A [RP:A] [RP:A] 4 MOTH NGP H GLASS 4 H TUB 4 Q PGP 5 P WITH 5 CV NGP 6 H HOLES 6 Q PGP 7 P IN 7 (CV) 1 CL 8 & THEN 8 S NGP HP YOU 8 M PUT 8 C NGP 9 DD THE 9 H STRAWS 8 C PGP 10 PM IN 10 CV NGP 11 DD THE 11 H HOLES 1 CL 12 & THEN 12 S NGP HP YOU 12 M PUT 12 C NGP 13 DD THE 13 H MARBLES 12 CM QQGP AX DOWN 1 CL 14 & AND 14 (S) 14 M PULL 14 C NGP 15 DQ A 15 H STRAW 14 CM QQGP AX OUT 14 A CL 16 I TO 16 M SEE 16 C CL 17 B IF 17 S NGP 18 DQ A 18 H MARBLE 17 M GOES 17 C PGP 19 P INTO 19 CV NGP 20 DQ A 20 H POINT
15 Z CL 1 S NGP HP I 1 ON DUN 1 M NO 1 (C)
16 [NV:ER] [NV:ER] Z CL 1 S NGP HP I 1 M PLAY 1 C PGP 2 P WITH 2 CV NGP 3 DD MY 3 H BIKE
17 Z 1 CL 2 S NGP HP I 2 M PLAY 2 C PGP 3 P WITH [FS:MY-CHIP] 3 CV NGP 4 DD MY [RP:MY] 4 MO QQGP AX BIG 4 H TIPPER-LORRY 1 CL 5 & AND 5 S NGP HP I [RP:I] 5 M CALL 5 C PGP 6 PM FOR 6 CV NGP HN DAVID
18 Z 1 CL F YEAH [FS:HE'S-ONE-MY] [FS:HE'S-ROUND] 1 CL 2 S NGP HP HE 2 OM 'S 2 C PGP 3 P IN 3 CV NGP 4 DD MY 4 H CLASS
19 [NV:OH] [FS:WE-JUST] Z CL 1 S NGP HP WE 1 M PLAY 1 C PGP 2 P AT 2 CV 3 NGP H FOOTBALL [HZ:AND-STUFF] 3 NGP 4 & AND 4 H CRICKET
20 [NV:ER] [FS:WE-PLAY-S...] Z CL 1 S NGP HP WE 1 M PLAY 1 C 2 NGP H FIREMEN 2 NGP 3 & AND 3 H POLICE
21 [NV:ER] Z CL 1 AI JUST 1 M READ 1 C NGP H COMICS

Appendix 1.4 Susanne Corpus.

A01:0030f#-#YB#<minbrk>#-#[Oh.Oh]
A01:0030g#-#AT#The#the#[O[S[Ns:s.
A01:0030h#-#NN1c#jury#jury#.Ns:s]
A01:0030i#-#RRR#further#far#[R:c.R:c]
A01:0030j#-#VVDv#said#say#[Vd.Vd]
A01:0030k#-#II#in#in#[P:p.
A01:0030m#-#NNT1c#term#term#[Np[Ns.
A01:0030n#-#YH#+<hyphen>#-#.
A01:0030p#-#NN1c#+end#end#.Ns]
A01:0040a#-#NN2#presentments#presentment#.Np]P:p]
A01:0040b#-#CST#that#that#[Fn:o.
A01:0040c#-#AT#the#the#[Nns:s101.
A01:0040d#-#NNL1c#City#city#.
A01:0040e#-#JB#Executive#executive#.
A01:0040f#-#NNJ1c#Committee#committee#.
A01:0040g#-#YC#+,#-#.
A01:0040h#-#DDQr#which#which#[Fr[Dq:s101.Dq:s101]
A01:0040i#-#VHD#had#have#[Vd.Vd]
A01:0040j#-#JB#over<hyphen>all#overall#[Ns:o.
A01:0050a#-#NN1n#charge#charge#.
A01:0050b#-#IO#of#of#[Po.
A01:0050c#-#AT#the#the#[Ns.
A01:0050d#-#NN1n#election#election#.Ns]Po]Ns:o]
A01:0050e#-#YC#+,#-#.Fr]Nns:s101]
A01:0050f#-#YIL#<ldquo>#-#.
A01:0050g#-#VVZv#+deserves#deserve#[Vz.Vz]
A01:0050h#-#AT#the#the#[N:o.
A01:0050i#-#NN1n#praise#praise#[NN1n&.
A01:0050j#-#CC#and#and#[NN2+.
A01:0050k#-#NN2#thanks#thank#.NN2+]NN1n&]
A01:0050m#-#IO#of#of#[Po.
A01:0050n#-#AT#the#the#[Nns.
A01:0060a#-#NNL1c#City#city#.
A01:0060b#-#IO#of#of#[Po.
A01:0060c#-#NP1t#Atlanta#Atlanta#[Nns.Nns]Po]Nns]Po]N:o]
A01:0060d#-#YIR#+<rdquo>#-#.
A01:0060e#-#IF#for#for#[P:r.
A01:0060f#-#AT#the#the#[Ns:103.
A01:0060g#-#NN1c#manner#manner#.
A01:0060h#-#II#in#in#[Fr[Pq:h.
A01:0060i#-#DDQr#which#which#[Dq:103.Dq:103]Pq:h]
A01:0060j#-#AT#the#the#[Ns:S.
A01:0060k#-#NN1n#election#election#.Ns:S]
A01:0060m#-#VBDZ#was#be#[Vsp.
A01:0060n#-#VVNv#conducted#conduct#.Vsp]Fr]Ns:103]P:r]Fn:o]S]
A01:0060p#-#YF#+.#-#.O]

Appendix 1.5 IBM/Lancaster Spoken English Corpus.

SK01 1 v
SK01 2 v
[Good_JJ morning_NNT1] !_!
SK01 3 v
[Nr Every_AT1 three_MC months_NNT2 Nr] ,_, [here_RL [P on_II [N Radio_NN1 4_MC N]P]] ,_, [N I_PPIS1 N][V present_VV0 [N a_AT1 programme_NN1 [Fn called_VVN [N Workforce_NP1 N]Fn]N]V] ._.
SK01 4 v
[N It_PPH1 N][V 's_VBZ [N a_AT1 quarterly_JJ bulletin_NN1 [P on_II [N employment_NN1 -_- not_XX unemployment_NN1 -_- employment_NN1 N]P]N] V] !_!
SK01 6 v
[Nr Last_MD year_NNT1 Nr] ,_, [N the_AT workforce_NN1 N][V grew_VVD V] .
SK01 9 v
[N Stephen_NP1 N][V left_VVD [N school_NN1 N][Nr last_MD year_NNT1 Nr] V] ._.
SK01 10 v
Now_RT ,_, [N unemployment_NN1 [P among_II [N school_NN1 leavers_NN2 N]P]N] ,_, [Fa as_CSA [N I_PPIS1 N][V 'm_VBM [J sure_JJ [Fn[N you_PPY N][V know_VV0 V]Fn]J]V]Fa] ,_, [V is_VBZ [J very_RG high_JJ J]V] ._.
SK01 11 v
[N One_PN1 [P in_II [N four_MC teenagers_NN2 N]P]N] ,_, [P over_II [N the_AT country_NN1 [P as_II [N a_AT1 whole_NN1 N]P]N]P] ,_, [V is_VBZ [P out_II21 of_II22 [N work_NN1 N]P]V] ._.
SK01 13 v
[N It_PPH1 N][V 's_VBZ [J even_RR worse_JJR [Fa if_CS [N you_PPY N][V e_VBR black_JJ J]V]Fa]J]V]
SK01 14 v
[N It_PPH1 N][V 's_VBZ almost_RR [N one_MC1 [P in_II [N two_MC N]P]N] V] ._.
SK01 15 v
[N Every_AT1 other_JB black_JJ teenager_NN1 [P in_II [N Britain_NP1 N] P]N][V is_VBZ [P out_II21 of_II22 [N work_NN1 N]P]V] ._.
SK01 16 v
And_CC ,_, [Fa although_CS [N unemployment_NN1 [P at_II [N any_DD age_NN1 N]P]N][V is_VBZ [N a_AT1 tribulation_NN1 N]V]Fa] ,_, [N it_PPH1 N][V does_VDZ seem_VV0 [J especially_RR unfair_JJ [P to_II [N the_AT young_JJ N]P]J]V] ._.
SK01 24 v
But_CCB [N the_AT DHSS_NNJ N][V was_VBDZ [J wrong_JJ J]V] !_!
SK01 25 v
[N Stephen_NP1 N][V was_VBDZ entitled_VVN [P to_II [N 18.20_NNU [a_AT1 week_NNT1]N]P][Ti to_TO help_VV0 [N him_PPHO1 N] look_VV0 [P for_IF [N work_NN1 N]P]Ti]V] ,_, [Fa as_CSA [N he_PPHS1 N][V discovered_VVD [Fa when_CS [N he_PPHS1 N][V sought_VVD [N the_AT advice_NN1 [P of_IO [N Youthaid_NP1 (_([Fr[N which_DDQ N][V is_VBZ [N an_AT1 organisation_NNJ [Fr[N which_DDQ N][V helps_VVZ [N young_JJ unemployed_JJ people_NN N]V]Fr]N]V]Fr])_) N]P]N]V]Fa]V]Fa] ._.
SK01 27 v
Oh_UH ,_, and_CC [N he_PPHS1 N][V did_VDD pass_VV0 [N his_APP$ exams_NN2 N]V] ._.
SK01 28 v
[N Youthaid_NP1 N][V is_VBZ [N a_AT1 small_JJ [national_JJ charity_NN1] ,_, now_RT [[ten_MC years_NNT2] old_JJ] ,_, [Tn dedicated_VVN ,_, [Fa as_CSA [N I_PPIS1 N][V said_VVD V]Fa] ,_, [P to_II [Tg helping_VVG [N unemployed_JJ young_JJ people_NN N]Tg]P]Tn] N]V] ._.

Appendix 2. A Brief Description of the POW Corpus.

Date of Compilation: 1978-84

Location: Polytechnic of Wales, Pontypridd, S. Wales.

Compiled by: Dr. Robin P. Fawcett and Dr. Michael R. Perkins

Type of Data:

Spoken corpus, recordings transcribed using conventions from the Survey of Modern English Usage at University College London, and those of a similar project at Bristol, with pitch movements marked by trained phonetician.

Fully hand parsed, using a Systemic Functional Grammar developed by Fawcett, with rich syntactico-semantic categories, capable of handling raising, dummy subject clauses, ellipsis, replacement strings. Parse trees stored in a numerical format (not standard bracketed) to capture discontinuities in syntactic structures. Children's English from Pontypridd, S. Wales. Informal register. The subjects were screened to exclude those with strong second language influence (Welsh or otherwise). 120 children aged between 6-12, (all within 3 months either side of their 6th, 8th, 10th or 12th birthday) divided equally according to sex, age, and socio-economic class established by profession and highest educational level of parents. Small cells of 3 children were recorded at play with Lego bricks, and each child also interviewed by the same `friendly' adult on his/her favourite games and TV programmes.

Size:

65,000 words approximately, in 11,396 lines. 1 parsed sentence per line, hence some very long lines. (also available in 80 chars wrap round format) 1.1 Mb. storage.

Format:

194 ASCII text files, each with a reference to age, social class, sex, play session or interview, and child's initials. (each file is a sample of a single child's speech in a play session or an interview).

Availability:

Only the parsed corpus is available in machine readable form; the recorded tapes and 4-volume transcripts with intonation contours are available in hard copy from the British Library Inter-Library Loans System. Original recordings are available from: Dr. Robin Fawcett, Computational Linguistics Unit, University of Wales College of Cardiff, Cardiff.

Original reason for collection:

Psycholinguistic research into development of children's' English between ages of 6 and 12, investigating the growing use of a variety of syntactico-semantic structures.

Current research (1987-9):

COMMUNAL project; Natural Language Processing at UWCC and Leeds University Extracting machine-readable systemic functional grammars and lexicons for use in parsing. Suites of programs developed to achieve this, including converting the corpus into bracketed form. The grammar used for the hand parsing in the corpus was not formalised in terms of phrase-structure rules, or RTNs, but in system networks of semantic/functional features and their realisation rules more suitable for NL generation than parsing.

Appendix 3. Systemic-Functional Syntax Categories in the POW Corpus.

Name of Category	Symbol in POW	NT/T	Examples (for Terminals)
TEXT AND SENTENCE			
Text	text	NT	-
Unfinished Text	textun	NT	-
Sentence	Z (for sigma)	NT	-
CLAUSE			
Clause	Cl	NT	-
Unfinished Clause	Clun	NT	-
Adjunct (= Experiential Adjunct)	A	NT/T	really, mostly
Affective Adjunct	Aa	NT	-
Discourse organizational Adjunct	Ad	NT/T	first-of-all, anyway
Replacement Adjunct	Arepl	NT	-
Feedback-seeking Adjunct	Af	NT/T	look, right, you know
Inferential Adjunct	Ai	NT/T	just, only
Logical Adjunct	Al	NT/T	really, though, as well
Replacement Logical Adjunct	Alrepl	NT	-
Wh-logical Adjunct	Alwh	NT	-
Modal Adjunct	Am	NT/T	maybe, probably
Metalingual Adjunct	Aml	NT/T	say, I mean
Negative Adjunct	An	NT/T	never, neither
Politeness Adjunct	Ap	NT/T	there, please
Tag Adjunct	Atg	NT/T	is it, isn't it
Wh-Adjunct	Awh	NT/T	how, when, where, why
Binder	B	NT/T	because, cos, if, so, when
Main-verb-completing Binder	Bm	T	of
Negative Binder	Bn	T	-
Complement	C	NT	-
Anticipatory Complement	Cantic	NT	-
Replacement Complement	Crepl	NT	-
Main-verb-completing Complement	Cm	NT/T	across, in, on, up
Predicative Complement	Cp	NT/T	able
Formula	F	NT/T	alright, yes, no, pardon, what
Frame	Fr	NT/T	right, now
Infinitive element	I	T	to
Main verb	M	T	builds, kicked, went
Negator	N	T	not, no

Operator	O	T	did, does, do, let's
Modal Operator	Om	T	'll, 'd, 'm, are, can, could, is
Negative Modal Operator	Omn	T	can't, couldn't, isn't, won't
Negative Operator	On	T	didn't, doesn't, don't
Auxiliary Operator	OX	T	'm, 're, 've, have, was
Negative Auxiliary Operator	OXn	T	haven't, wasn't
Subject	S	NT	-
Anticipatory Subject	Santic	NT	-
Replacement Subject	Srepl	NT	-
Dummy it Subject	Sit	T	it
Dummy there Subject	Sth	T	there
Wh-Subject	Swh	NT	-
Vocative	V	NT	-
Auxiliary	X	T	be, going to, have, used
Modal/Necessity Auxiliary	Xm	T	better, got to, have to
Negative Modal Auxiliary	Xmn	T	mustn't
Negative Auxiliary	Xn	T	don't, hadn't, haven't
NOMINAL GROUP			
nominal group	ngp	NT	-
unfinished nominal group	ngpun	NT	-
deictic determiner (also in qqgp)	dd	NT/T	the, this, that, her, my
wh-deictic determiner	ddwh	T	what, which
quantifying determiner (also in qqgp)	dq	NT/T	a, an, one, four, any, all
negative quantifying determiner	dqn	NT/T	no, none
wh-quantifying determiner	dqwh	T	how many, how much
ordinative determiner	do	NT/T	first, sixth, last
partitive determiner	dp	NT/T	part
superlative determiner	ds	NT	-
typic determiner	dt	NT	-
selector (of)	vo	T	of
modifier (= experiential modifier)	mo	NT	-
affective modifier	moa	NT/T	flipping
comparison modifier	moc	NT/T	other, else, same, different
quantifying modifier	moq	NT/T	five, only, ten
situation modifier	mosit	NT/T	opening
thing modifier	moth	NT/T	plastic, square, table
head (i.e. 'common noun')	h	T	brick, books, men
('proper') name head	hn	T	America, Alf, Barry-Island, Batman
pronoun head	hp	T	anything, he, her, him, I, it
negative pronoun head	hpn	T	no-one, nobody, nothing
situation head	hsit	NT/T	painting, reading
wh-pronoun head	hwh	T	what, which, who

qualifier	q	NT/T	ago, left
replacement qualifier	qrepl	NT	-
PREPOSITIONAL GROUP			
prepositional group	pgp	NT	-
unfinished prepositional group	pgpun	NT	-
preposition	p	NT/T	on, in, up, under
Main-verb-completing preposition	pm	T	about, after, at, for, into
completive	cv	NT	-
replacement completive	cvrepl	NT	-
wh-completive	cvwh	NT	-
QUANTITY-QUALITY GROUP			
quantity-quality group	qqgp	NT	-
unfinished quantity-quality group	qqgpun	NT	-
temperer (also in pgp)	t	NT/T	a bit, about, all, over, very
wh-temperer	twh	T	how
apex	ax	NT/T	always, away, back, big, black
tempering apex	axt	T	biggest, better, higher, smaller
wh-apex	axwh	NT/T	how, where, why, when
scope	sc	NT/T	more
finisher	fi	NT/T	of all, together
GENITIVE CLUSTER			
genitive cluster	gc	NT	-
genitive element	g	T	's
possessor	ps	NT	-
owner	own	T	own
ELEMENTS OCCURRING IN MORE THAN ONE UNITS NOT SPECIFIED ABOVE			
inferrer	inf	T	just, only
Linker	&	T	and, and then, but, or, so, then
Negative linker	&N	T	nor

Appendix 4. A Mapping from LDOCE to POW SF Syntax Tags.

(First version 23-5-88)

LDOCE POW

1) Adjectives. adj [A,B,E,F,GU,P,Wa] [3,5,6,9] Default= [B]

LDOCE		**POW**		
Description of item	*LDOCE tag/code*	*POW tag*	*POW unit*	*Comments*
attributive(prenom)	[A]	AX	(MO_QQGP)	eg overall
both at+pr (default)	[B]	AX	(MO/C_QQGP)	eg nice, fast
attributive(postnom)	[E]	AX	(MO_QQGP)	eg elect
predicative	[F]	AX	(C_QQGP)	eg asleep
group uncountable	the+[GU]	AX	(NGP MO_QQGP)	eg the accused
plural	the+[P]	AX	(NGP MO_QQGP)	eg the dead
comp=er, superl=est	Wa1[]	AX	(MO/C_QQGP)	eg nice, fast
comp=more/er,superl=most/est	[Wa2]	AX	"	eg secure
comp/superl,schwa-losing	[Wa3]	AX	"	eg simple
no-comp/superl	[Wa5]	AX	"	eg main, atomic
comparative/superlative (formsare not in LDOCE)	-	AXT	"	eg larger, largest

2) Nouns. n [A,C,E,GC,GU,N,P,R,S,U,Wn] [3,5,6,9] Default= [C]

LDOCE		**POW**		
Description of item	*LDOCE tag/code*	*POW tag*	*POW unit*	*Comments*
attributive(prenominal)	[A]	H	(MOTH_NGP)	eg 'football' in 'football kit' (freq one word hyphenated in POW)
countable (default)	[C]	H	(NGP)	
attributive(postnominal)	[E]	?	"	v. rare
group(countable)	[GC]	H	"	eg committee
group(uncountable)	[GU]	H	"	eg the Admiralty
vocative	[N]	H/HN	(V_NGP)	eg doctor!
plural	[P]	H	(NGP)	eg the police
namelike	[R]	HN	"	eg God,the Earth
singular(uncountable)	[S]	H	"	eg think,babble
uncountable	[U]	H	"	eg sugar,love
odd-plural:				
usu. change in pl.	[Wn1]	H	"	eg lion, pheasant
usu. don't change in pl.	[Wn2]	H	"	eg quail, salmon
never change in pl.	[Wn3]	H	"	eg sheep, grouse

3) Pronouns. pron [Wp] default= []

LDOCE	**POW**

Description of item	*LDOCE tag/code*	*POW tag*	*POW unit*	*Comments*
personal: subject, object, possessive, reflexive.	[Wp1]	HP	(NGP)	eg I, me, mine, myself.
relative: restric & non-restrictive	[Wp2]	HWH	"	eg who(m), whose, which, what.
restrictive only				that, { }
negative	-	HPN	"	eg no-one, none, nothing, nobody.
other	-	HP	"	eg any, everything, anything, something, another, others, each, both.

4) Determiners. determiner [Wp] default= []

	LDOCE		**POW**	
Description of item	*LDOCE tag/code*	*POW tag*	*POW unit*	*Comments*
possessive determiner	[Wp1]	DD	(NGP)	eg my, our, his.
deictic	-	DD	(NGP)	eg the, that, this those, these.
wh-deictic	[Wp2]	DDWH	(NGP)	eg which, what, whose.
quantifying non-ordinal (includes predeterminers)	-	DQ	(NGP)	eg a, any, a-bit, some, all, each, every, both.
quantifying ordinal (MOQ when preceded by another determiner)	-	DQ	(NGP)	eg one, two, three..
	-	DO	(NGP)	eg first, second..
negative	-	DQN	(NGP)	eg no.
wh-	-	DQWH	(NGP)	eg how-many/much
partitive/representative (quite rare, could be given DQ)	-	DP	(NGP)	eg part of, one of, much of, a bit of
genitive element	-	G	(NGP_PS)	eg 's.

5a) Verbs. v [D,I,L,T,V,Wv,X] [0-9] always have a number.
(includes modals, auxiliaries and linking verbs)

	LDOCE		**POW**	
Description of item	*LDOCE tag/code*	*POW tag*	*POW unit*	*Comments*
ditransitive	[D]	M	(all CL)	eg give, tell
intransitive	[I0]	M		eg sleep, pause.
	[I2]	OM(N)/ XM(N)		modal auxiliaries
	[I8]	OX(N)/ X(N)		auxiliaries
				(see below [Wv2])
	[I5/6]	M		eg seems, appears (it + verb)

linking/copula (subj refers to complement)	[L]	M	eg be, become, act, start last, cost, end up, go.
transitive	[T]	M	eg kick, say, help.
non-finite-clause complement	[V]	M (+ BM)	eg help(to do sth) see, ask.
transitive+(sth. else)	[X]	M	eg consider, put, regard(as), make.
to be	[Wv1]	M	
operator/auxiliary	[Wv2]	O/X	eg do.
		OM/XM	eg can, might, 'd, 'll, will, ought..
		OX/X	eg have, be.

(Operator is used in POW for the first auxiliary, which agrees, is finite, tensed, and moves to clause-initial position in Y/N questions. Hence most auxiliaries are given an operator and an auxiliary tag in POW. All such POW tags may be followed by N (eg OXN) for the negated, often shortened form of the verb; *hasn't, won't, couldn't* etc.).

schwa-losing	[Wv3]	M	eg couple
-ing as adjective	[Wv4]	M	eg fly
-ed as adjective	[Wv5]	M	eg plait
no *-ing* form (present tense)	[Wv6]	M	eg know, see
infinitival element	[-3]	I	eg to

5b) Phrasal Verbs. v adv, v adv prep, v prep, v adv;prep. [I,T]

LDOCE		**POW**		
Description of item	*LDOCE tag/code*	*POW tag*	*POW unit*	*Comments*
adverb particle	v adv	M AX	(CM_QQGP)	eg pull out, put down
preposition particle	v prep	M PM	(C_PGP)	eg get into, go around.
adv/prep particle	v adv;prep	M AX	(CM_QQGP)	eg lay off
		M PM	(C_PGP)	eg lay off
adv+prep particles	v adv prep	M AX	(CM_QQGP)	
		P	(A_PGP)	eg pop up through

(some phrasal verbs with 2-particles have the particles hyphenated in pow, eg *see out-of, make out-of*, in which case they are treated as single preposition particles; v prep).

6) Adverbs adv [E,F,H,Wa] (mostly no upper-case letters)
(never followed by a number, except [Wa1] etc.)

LDOCE		**POW**		
Description of item	*LDOCE tag/code*	*POW tag*	*POW unit*	*Comments*
default (no code)	-	AX/T/F/ FR		most adverbs are both [F] and [H]
postnominal	[E]	Q	(QQGP)	eg ago, left, previously.
complement	[F]	AX	(A_QQGP)	eg abroad, quickly, easily.
combining with adj,prep,adv	[H]	T	(MO_QQGP)	eg very, so, well, right.
schwa-losing	[Wa3]	AX/T		eg simply

schwa-reducing	[Wa4]	AX/T	eg poetically

(pow tag F (formula) is used for LDOCE adverbs which are usually single-word adverbials, such as *yes, no, OK, right*. pow tag FR (frame) is used similarly, for time adverbials; *now, right-now*).

7) Prepositions. (LDOCE) prep = P (pow)

8) Conjunctions.

pow distinguishes two types, binder(B) and linker(&) for LDOCE conj, which could perhaps be conflated.

LDOCE		**POW**		
Description of item	*LDOCE tag/code*	*POW tag*	*POW unit*	*Comments*
linker	conj	&	(CL)	eg and, but, then, so, or.
binder	conj	B	(CL)	eg cos, as, after, before, if, for, when, since, till

9) Interjections.

Again pow distinguishes two types, exclamation(EX) and formula(F) for LDOCE interj, which could be conflated.

LDOCE		**POW**		
Description of item	*LDOCE tag/code*	*POW tag*	*POW unit*	*Comments*
exclamation	interj	EX	(CL)	eg hey,dear,oh
formula	interj	F	(CL)	eg yes,no,OK,right

Appendix 5. A Fragment of a Context-Free SF Syntax Maintaining the Distinction between Filling and Componence.

6949 /*10ABIHS1*/ 'S_NGP' (['S_NGP' , X1]) --> 'HP' (X1).
1366 /*10ABIHS10*/ 'Z_CL' (['Z_CL' , X1]) --> 'F' (X1).
1242 /*10ABIHS2*/ 'C_NGP' (['C_NGP' , X1]) --> 'HP' (X1).
1198 /*10ABIHS19*/ 'CM_QQGP' (['CM_QQGP' , X1]) --> 'AX' (X1).
1138 /*10ABIHS1*/ 'C_NGP' (['C_NGP' , X1, X2]) --> 'DQ' (X1), 'H' (X2).
905 /*10ABIHS19*/ 'C_PGP' (['C_PGP' , X1, X2]) --> 'P' (X1), 'CV_NGP' (X2).
895 /*10ABIHS1*/ 'Z' (['Z' , X1, X2]) --> 'CL' (X1), 'CL' (X2).
853 /*10ABIHS25*/ 'C_QQGP' (['C_QQGP' , X1]) --> 'AX' (X1).
845 /*10ABIHS2*/ 'MO_QQGP' (['MO_QQGP' , X1]) --> 'AX' (X1).
798 /*10ABIHS2*/ 'C_NGP' (['C_NGP' , X1, X2]) --> 'DD' (X1), 'H' (X2).
733 /*10ABIHS14*/ 'A_QQGP' (['A_QQGP' , X1]) --> 'AX' (X1).
733 /*10ABIHS12*/ 'CV_NGP' (['CV_NGP' , X1, X2]) --> 'DD' (X1), 'H' (X2).
732 /*10ABIHS12*/ 'A_PGP' (['A_PGP' , X1, X2]) --> 'P' (X1), 'CV_NGP' (X2).
582 /*10ABIHS5*/ 'S_NGP' (['S_NGP' , X1]) --> 'DD' (X1).
547 /*10ABIHS12*/ 'C_NGP' (['C_NGP' , X1]) --> 'H' (X1).
486 /*10ABIHS55*/ 'Z' (['Z' , X1, X2]) --> 'CL_F' (X1), 'CL' (X2).
352 /*10ABIHS16*/ 'S_NGP' (['S_NGP' , X1, X2]) --> 'DD' (X1), 'H' (X2).
336 /*10ABIHS14*/ 'C_NGP' (['C_NGP' , X1]) --> 'HN' (X1).
316 /*10ABIHS16*/ 'CV_NGP' (['CV_NGP' , X1]) --> 'HP' (X1).
288 /*10ABIHS14*/ 'C_NGP' (['C_NGP' , X1]) --> 'DD' (X1).
281 /*10ABIHS42*/ 'Z_CL' (['Z_CL' , X1, X2, X3]) --> 'S_NGP' (X1), 'M' (X2), 'C_NGP' (X3).
260 /*10ABIHS31*/ 'MOTH_NGP' (['MOTH_NGP' , X1]) --> 'H' (X1).
254 /*10ABPSHS222*/ 'V_NGP' (['V_NGP' , X1]) --> 'HN' (X1).
245 /*10ABIHS11*/ 'CV_NGP' (['CV_NGP' , X1]) --> 'DD' (X1).
231 /*10ABIHS1*/ 'CWH_NGP' (['CWH_NGP' , X1]) --> 'HWH' (X1).
226 /*10ABIHS11*/ 'Q_PGP' (['Q_PGP' , X1, X2]) --> 'P' (X1), 'CV_NGP' (X2).
222 /*10ABIHS71*/ 'CV_QQGP' (['CV_QQGP' , X1]) --> 'AX' (X1).
216 /*10ABIRL51*/ 'CV_NGP' (['CV_NGP' , X1]) --> 'H' (X1).
208 /*10ABIHS36*/ 'CL' (['CL' , X1, X2, X3]) --> 'S_NGP' (X1), 'M' (X2), 'C_NGP' (X3).
192 /*10ABIHS16*/ 'Z' (['Z' , X1, X2, X3]) --> 'CL' (X1), 'CL' (X2), 'CL' (X3).
181 /*10ABIRL3*/ 'C_PGP' (['C_PGP' , X1, X2]) --> 'PM' (X1), 'CV_NGP' (X2).
169 /*10ABPSHS273*/ 'C_NGP' (['C_NGP' , X1, X2, X3]) --> 'DQ' (X1), 'MO_QQGP' (X2), 'H' (X3).
169 /*10ABIHS61*/ 'CV_NGP' (['CV_NGP' , X1]) --> 'HN' (X1).
166 /*10ABIHS5*/ 'Z_CL' (['Z_CL' , X1, X2, X3]) --> 'S_NGP' (X1), 'OM' (X2), 'C_NGP' (X3).
163 /*10ABIHS2*/ 'S_NGP' (['S_NGP' , X1]) --> 'HN' (X1).
157 /*10ABIRL3*/ 'CV_NGP' (['CV_NGP' , X1, X2]) --> 'DQ' (X1), 'H' (X2).
156 /*10ABIHS24*/ 'C_QQGP' (['C_QQGP' , X1, X2]) --> 'T' (X1), 'AX' (X2).
143 /*10ABPSHS38*/ 'S_NGP' (['S_NGP' , X1]) --> 'H' (X1).
138 /*10ABPSHS235*/ 'Z_CL' (['Z_CL' , X1, X2, X3]) --> 'S_NGP' (X1), 'M' (X2), 'C_CL' (X3).
135 /*10ABIHS18*/ 'C_CL' (['C_CL' , X1, X2, X3]) --> 'I' (X1), 'M' (X2), 'C_NGP' (X3).
133 /*10ABIHS71*/ 'C_PGP' (['C_PGP' , X1, X2]) --> 'P' (X1), 'CV_QQGP' (X2).
128 /*10ABIHS41*/ 'Z_CL' (['Z_CL' , X1, X2, X3]) --> '(S)' (X1), '(M)' (X2), 'C_NGP' (X3).
120 /*10ABIRL55*/ 'AWH_QQGP' (['AWH_QQGP' , X1]) --> 'AXWH' (X1).
116 /*10ABIRL69*/ 'Z_CL' (['Z_CL' , X1, X2, X3, X4]) --> 'S_NGP' (X1), 'OX' (X2), 'M' (X3), 'C_NGP' (X4).
116 /*10ABIHS2*/ 'DP_NGP' (['DP_NGP' , X1, X2]) --> 'DD' (X1), 'H' (X2).
114 /*10ABITG15*/ 'CL' (['CL' , X1, X2, X3]) --> 'S_NGP' (X1), 'OM' (X2), 'C_NGP' (X3).
112 /*10ABIHS2*/ 'CL' (['CL' , X1, X2, X3, X4]) --> '&' (X1), 'S_NGP' (X2), 'M' (X3), 'C_NGP' (X4).
102 /*10ABPSRL117*/ 'ATG_CL' (['ATG_CL' , X1, X2]) --> 'OMN' (X1), 'S_NGP' (X2).
101 /*10ABPSHS215*/ 'C_NGP' (['C_NGP' , X1]) --> 'DQ' (X1).
100 /*10ABIHS32*/ 'Z_CL' (['Z_CL' , X1, X2, X3]) --> 'S_NGP' (X1), 'OM' (X2), 'C_QQGP' (X3).
92 /*10ABIRL69*/ 'C_TEXT' (['C_TEXT' , X1]) --> 'Z_CL' (X1).

Appendix 6. A Fragment of a Context-Free SF Syntax Ignoring the Distinction between Filling and Componence.

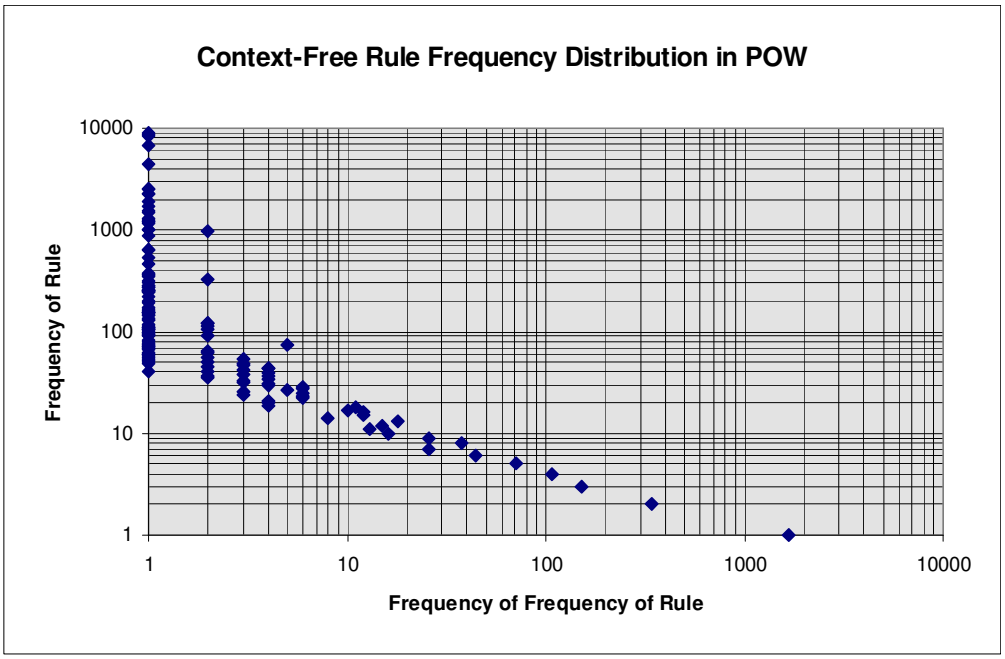
8765	10ABIHS000001	S --> NGP
8598	10ABIHS000001	NGP --> HP
8013	10ABIHS000004	Z --> CL
6520	10ABIHS000001	C --> NGP
4344	10ABIHS000002	QQGP --> AX
2365	10ABIHS000002	CV --> NGP
2361	10ABIHS000001	PGP --> P CV
2225	10ABIHS000004	CL --> F
2195	10ABIHS000002	NGP --> DD H
1705	10ABIHS000001	NGP --> DQ H
1702	10ABIHS000001	Z --> CL CL
1473	10ABIHS000012	NGP --> H
1460	10ABIHS000001	C --> PGP
1238	10ABIHS000024	C --> QQGP
1223	10ABIHS000002	CM --> QQGP
1178	10ABIHS000002	NGP --> HN
1169	10ABIHS000005	NGP --> DD
1149	10ABIHS000002	C --> CL
951	10ABIHS000002	MO --> QQGP
933	10ABIHS000002	A --> PGP
931	10ABIHS000007	CL --> S M C
846	10ABIHS000014	A --> QQGP
618	10ABIHS000005	CL --> S OM C
461	10ABIHS000022	AL --> CL
432	10ABIHS000001	CL --> M C
394	10ABIHS000016	Z --> CL CL CL
357	10ABPSHS00079	V --> NGP
346	10ABIHS000024	QQGP --> T AX
336	10ABIHS000001	NGP --> HWH
334	10ABPSHS00273	ATG --> CL
326	10ABIHS000031	MOTH --> NGP
310	10ABIHS000001	CL --> & S M C
278	10ABIHS000011	Q --> PGP
273	10ABIHS000041	CL --> (S) (M) C
271	10ABIRL000047	CL --> M
266	10ABIHS000001	CWH --> NGP
261	10ABIHS000022	Q --> CL
248	10ABIRL000015	CL --> S OM M C
243	10ABPSHS00273	NGP --> DQ MO H
241	10ABIHS000018	CL --> I M C
240	10ABIHS000071	CV --> QQGP
239	10ABIHS000001	CL --> S OX M C
238	10ABIHS000026	Z --> CLUN
230	10ABIHS000002	A --> CL
226	10ABIHS000055	QQGP --> AXWH
207	10ABITG000037	CL --> S M
203	10ABIRL000003	PGP --> PM CV
199	10ABIRL000003	NGP --> DQ
189	10ABIHS000028	CL --> B S M C
176	10ABIHS000051	DQ --> QQGP
171	10ABIHS000013	C --> NGP NGP
164	10ABIHS000055	CL --> ???
160	10ABIHS000002	DP --> NGP
158	10ABIHS000018	CL --> (S) (OM) C
153	10ABIHS000009	A --> NGP
151	10ABIHS000011	NGP --> DQ H Q
150	10ABPSRL00071	CL --> EX
148	10ABIRL000055	AWH --> QQGP
146	10ABIHS000071	CL --> M CM
127	10ABIRL000017	NGP --> MO H
126	10ABIHS000061	CL --> STH OM C
124	10ABIRL000049	TEXT --> Z
121	10ABPSHS00047	??? --> NGP
121	10ABIHS000012	CL --> S M C A
119	10ABITG000023	CL --> M C C
118	10ABIRL000049	C --> TEXT
117	10ABIHS000024	Z --> CL CLUN
116	10ABIHS000059	QQGP --> AX SC
116	10ABIHS000025	DQ --> NGP
115	10BBPSHW00065	C? --> NGP
113	10ABIHS000002	NGP --> DD MO H
112	10ABPSRL00158	CL --> C?
111	10ABIHS000053	QQGP --> AXT
103	10ABPSRL00027	CL --> OMN S
102	10ABIHS000016	NGP --> DD H Q
101	10ABIRL000034	SWH --> NGP
101	10ABIRL000004	NGP --> DQ DD H
99	10ABIRL000022	NGP --> DQ MOTH H
99	10ABIHS000013	CL -->(S) (OX) (M) C
97	10ABIRL000047	Z --> CL CL CL CL
96	10ABIHS000070	CWH --> QQGP
95	10ABPSRL00039	CL --> S ON M C
95	10ABIHS000027	PGP --> (P) CV
93	10ABPSRL00194	CL --> FR
91	10ABPSHS00047	NGP --> DD HP
90	10ABPSHS00281	NGP --> DQ MO HP
88	10ABIHS000025	CL --> & S OM C
87	10ABIHS000002	NGP --> DP
77	10ABITG000043	CREPL --> NGP
76	10ABIHS000025	PGP --> P C
74	10ABIHS000025	C --> NGPUN
74	10ABIHS000009	NGP --> DQ VO H
71	10ABPSHS00301	CL --> CWH OM S
71	10ABPSHS00038	CL --> B S OM C
71	10ABIRL000021	CL --> (I) (M) C
70	10ABIRL000002	CLUN --> &
70	10ABIRL000001	MOA --> QQGP
69	10ABITG000051	DD --> GC
69	10ABIRL000061	NGP --> DQ VO HP
69	10ABIRL000051	NGP --> H Q

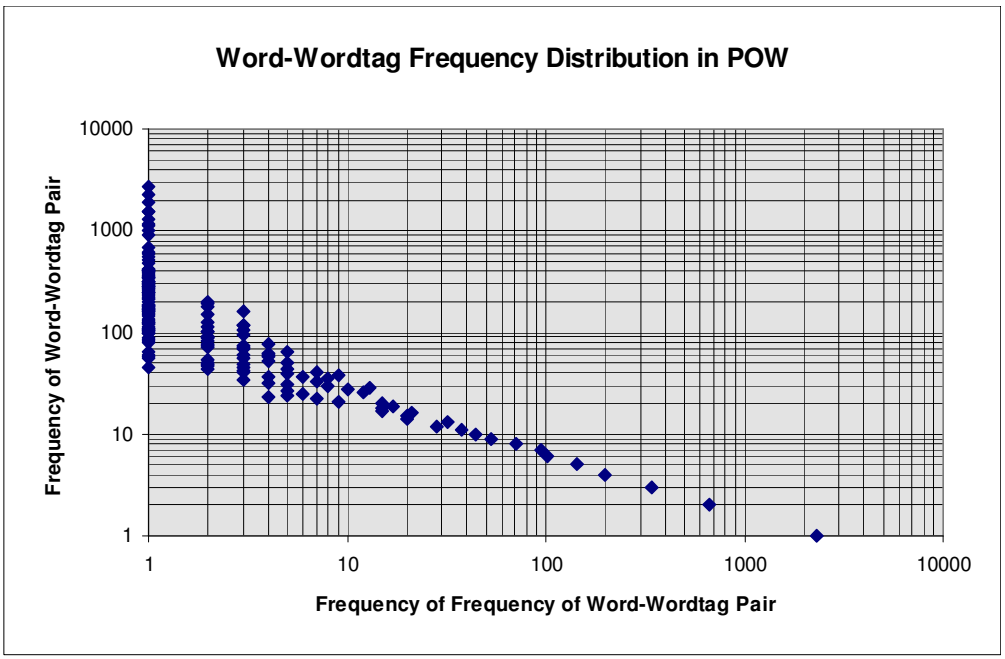
Appendix 7. A Fragment of a Vertical Trigram Model from the POW corpus

This sample illustrates all the possible vertical trigrams dominating the linker (&) label, with their frequencies from the POW corpus. Note that the linker is one of the few elements of structure which may occur in more than one unit (CL, CLUN, NGP, NGPUN, PGP, QQGP). There are 968 unique vertical trigrams in all.

[38 & CL AL]
[7 & CL A]
[3 & CL CV]
[26 & CL C]
[2 & CL FI]
[8 & CL Q]
[3 & CL SANTIC]
[1 & CL SC]
[1745 & CL Z]
[3 & CLUN AL]
[1 & CLUN A]
[2 & CLUN C]
[151 & CLUN Z]
[1 & NGP CANTIC]
[4 & NGP CREPL]
[1 & NGP CVREPL]
[44 & NGP CV]
[211 & NGP C]
[3 & NGP DQ]
[6 & NGP SANTIC]
[26 & NGP S]
[1 & NGPUN CREPL]
[1 & NGPUN CV]
[18 & NGPUN C]
[2 & NGPUN S]
[2 & PGP A]
[7 & PGP C]
[3 & QQGP A]
[4 & QQGP CM]
[2 & QQGP CV]
[9 & QQGP C]
[3 & QQGP DQ]
[12 & QQGP MO]

Appendix 8. Rule and Word-Wordtag Frequency Distribution in the POW Corpus.





Appendix 9. A Prototype Competence Systemic Functional Syntax.

A Formal Systemic-Functional Grammar
Clive Souter (June 1988)

(This grammar is taken from Fawcett's "Some Proposals for Systemic Syntax", using his example trees and descriptions of structure for all levels of the sentence. It will be expanded to a 'full' set of simple PS rules to be compared with those extracted from the pow corpus. Two kinds of rewrite arrow are used here:

=> for filling of an element of structure by a unit, such as S => NGP
-> for componence, (traditional rewrite in PS grammars), eg NGP -> H

Rules may include daughters followed by '+' which allows for repeated co-ordination of that category, and will be specified to a certain maximum value for the sake of generation of a set of simple PS rules later on. Where a daughter is optional, it is placed in round brackets (DD). If a daughter may optionally be repeated up to a limited no of times, this figure is included in braces {} immediately following the daughter category, eg DD{3}. It so happens in the current version of the grammar that no daughters are both obligatory and allow repetition, i.e. all daughters which may be repeated up to a limited number of times are optional i.e. (MO{2}). The notation could therefore be collapsed to just use the {} facility, with (MO) = MO{1}, but it remains to be seen whether this will always be the case.

Filling Relationships:

A => CL
A => NGP
A => PGP+
A => QQGP+
C => CL+
C => NGP+
C => PGP+
C => QQGP+
C => TEXT
CV => CL
CV => NGP+
CV => PGP
CV => QQGP
DD => GC
DP => NGP
DQ => NGP
DQ => QQGP
FI => CL+
FI => PGP
H => CL
MO => QQGP+
MOTH => NGP ;; perhaps not all the nominal group structure is needed here
PS => NGP+
Q => CL+
Q => NGP
Q => PGP+
Q => QQGP
S => CL+

S => NGP+
S => PGP
S => QQGP+
SC => CL
SC => PGP+
T => NGP
T => QQGP
TEXT => Z
Z => CL+

Componence Relationships:

Adverbial/Adjectival Group = Quantity/Quality Group.

(should this distinction be reintroduced if we no longer have the difference between MO_QQGP and A_QQGP? Clearly while the structure of the two is the same, the items which may expound the various daughters of QQGP ARE different according to whether they are part of a MO or A. NB: still causes probs for dict look-up)

QQGP -> (&) (T) AX (SC) (FI)
QQGP -> (&) AX (T) (SC) (FI)

i.e. T is mutually exclusive. AX is necessary.

Nominal Group (including genitive cluster)

A full range of determiner structures has been attempted, with the exception that DO(ordinative) and DS(superlative) have been treated as one category, DS. Partitive(DP) and Quantifying(DQ) determiners have been distinguished. Modifiers are of three types, MOQ(quantifying), MO('normal'), and MOTH('thing'). MOTH replaces the original nominator, and is used for the first element in strings like "paper bag" and "football match". There is clearly some overlap with the idea of compound nouns, whose characteristics are difficult to specify, as not only phonological (stress) criteria are needed. Compounds are normally hyphenated in pow, but not always in everyday usage. The head is nearly always present, except for cases like "this", which is traditionally treated as a pronoun as well as a determiner, but here only as a determiner.

The deictic determiner (DD) may be filled by a genitive cluster, or expounded directly by a word.

NGP -> (&) DP VO DQ VO DS VO DD (MOQ) (MO{3}) (MOTH{3}) H (Q{5})
NGP -> (&) DP VO DS VO DD (MOQ) (MO{3}) (MOTH{3}) H (Q{5})
NGP -> (&) DP VO DS VO (MOQ) (MO{3}) (MOTH{3}) H (Q{5})
NGP -> (&) DP VO DD (MOQ) (MO{3}) (MOTH{3}) H (Q{5})
NGP -> (&) DQ VO DS VO DD (MOQ) (MO{3}) (MOTH{3}) H (Q{5})
NGP -> (&) DQ VO DS (MOQ) (MO{3}) (MOTH{3}) H (Q{5})
NGP -> (&) DQ VO DD (MOQ) (MO{3}) (MOTH{3}) H (Q{5})
NGP -> (&) DQ DD (MOQ) (MO{3}) (MOTH{3}) H (Q{5})
NGP -> (&) DQ (MOQ) (MO{3}) (MOTH{3}) H (Q{5})
NGP -> (&) DS VO DD (MOQ) (MO{3}) (MOTH{3}) H (Q{5})
NGP -> (&) DS (VO) (MOQ) (MO{3}) (MOTH{3}) H (Q{5})
NGP -> (&) DD (MOQ) (MO{3}) (MOTH{3}) H (Q{5})
NGP -> (&) DD (MOQ)
NGP -> (&) (MOQ) (MO{3}) (MOTH{3}) H (Q{5})
NGP -> (&) HP
NGP -> (&) HN
GC -> (PS) G (OW) ;; NB no co-ordination

Prepositional Groups

Main verb completing prepositions (PM) (i.e. in phrasal verbs) not included. What should their structure be? (temperers disallowed, no co-ordination)

PGP -> (&) (T) P CV ;; should P be able to be filled by a QQGP?
PGP -> PM CV

Clauses

Does not yet account for all medial adjuncts; particularly between S, O and Xs.

declarative:

CL -> F
CL -> (FR) (&) (B) (A+) S (O) (A) (X{5}) M (CP) (C{2}) (A+)
CL -> (FR) (&) (A+) C S (O) (X{5}) M (C) (A+) ;; topicalised complement
                                                ;; or rel. clause.
CL -> (FR) (&) (A+) (C) M S (A+) ;; here is.. into it ran the mouse.

imperative:

CL -> (&) (B) (A+) M (C{2}) (A+)

interrogative: (NB wh-subjects treated as in non-interrogative position)

CL -> (FR) (&) (A+) O S (A) (X{5}) (M) (C{2}) (A+)
CL -> (FR) (&) (A+) C O S M (C) (A+)

Appendix 10. Brill Tagger Context Rules Learned from POW

csgps1:POW>more CONTEXT-RULEFILE

OM OX NEXT1OR2TAG M
STH AX NEXT1OR2TAG .
P AX NEXTTAG .
P M PREV1OR2WD I
F M PREVTAG ON
I P NEXTTAG DD
M O RBIGRAM do you
M X NEXTTAG M
STH AX NEXT1OR2TAG HP
M H PREVWD a
OX OM CURWD 'd
DQ AX NEXTTAG SC
P AX NEXTTAG P
P PM PREVWD look
HWH F SURROUNDTAG STAART .
AF FR PREV1OR2OR3TAG STAART
P AX NEXT1OR2TAG M
HP DQ NEXT1OR2TAG VO
AX & WDNEXTTAG then HP
OM OX NEXT2TAG X
AX T NEXTTAG AX
I P NEXTTAG HN
AX P NEXTWD the
P AX NEXTTAG &
M OX RBIGRAM have you
HP DQ NEXTTAG H
M O NEXTTAG M
STH AX NEXT1OR2TAG &
OM G SURROUNDTAG H H
OX OM NEXTTAG DQ
I P NEXTTAG H
AX STH NEXTTAG OM
P AX NEXT1OR2WD we
P PM PREVWD looking
I P NEXTTAG DQ
OM G PREVTAG HN
AX F WDAND2BFR STAART alright
H M PREVBIGRAM & HP
OX OM NEXTTAG AX
DQ AX PREVTAG T
STH AX NEXTTAG P
HP DD WDNEXTTAG her H
HWH DDWH NEXT1OR2TAG H

Appendix 11. General lexical tagging rules used by the Brill tagger for untrained words.

The first of these are default labels for nouns and proper nouns. Others represent (fairly esoteric) rules for helping predict tags with respect to a neighbouring tag or word, or by analysing affixes. For example

M my fgoodright H 5.66868686868687

suggests a good tag to follow a main verb followed by the word my is a head noun.
(The rules are presented in 3 columns to save space).

NN a fchar H 0
NN b fchar H 0
NN c fchar H 0
NN d fchar H 0
NN e fchar H 0
NN f fchar H 0
NN g fchar H 0
NN h fchar H 0
NN i fchar H 0
NN j fchar H 0
NN k fchar H 0
NN l fchar H 0
NN m fchar H 0
NN n fchar H 0
NN o fchar H 0
NN p fchar H 0
NN q fchar H 0
NN r fchar H 0
NN s fchar H 0
NN t fchar H 0
NN u fchar H 0
NN v fchar H 0
NN w fchar H 0
NN x fchar H 0
NN y fchar H 0
NN z fchar H 0
NNP A fchar HN 0
NNP B fchar HN 0
NNP C fchar HN 0
NNP D fchar HN 0
NNP E fchar HN 0
NNP F fchar HN 0
NNP G fchar HN 0
NNP H fchar HN 0
NNP I fchar HN 0
NNP J fchar HN 0
NNP K fchar HN 0
NNP L fchar HN 0
NNP M fchar HN 0
NNP N fchar HN 0
NNP O fchar HN 0
NNP P fchar HN 0
NNP Q fchar HN 0
NNP R fchar HN 0
NNP S fchar HN 0
NNP T fchar HN 0
NNP U fchar HN 0
NNP V fchar HN 0
NNP W fchar HN 0
NNP X fchar HN 0
NNP Y fchar HN 0
NNP Z fchar HN 0
the goodleft M 103.741851499632
ng hassuf 2 M 63.3301587301587
the goodright H 54.2394733115649
I goodright M 45.756450713517
very goodright AX 33.0682856984206
he goodright M 29.2066457866229
H ed fhassuf 2 M 21.3333333333333
H it fgoodright AX 19.1383883614615
AX s fdeletesuf 1 M 13.3333333333333
can goodright M 12.7017211703959
M with fgoodright HP 10.5027834347864
't hassuf 2 OMN 9.6165286494636
M out fgoodright P 9.14219217900017
H ly faddsuf 2 AX 8.9
H was fgoodright AX 8.62216837090389
M never fgoodleft OM 8.53591985129854
ly deletesuf 2 AX 8.35227272727273
M - fchar H 6.98913043478261
-to deletesuf 3 XM 6.94565217391304
a deletepref 1 AX 6.12766116941529
ere hassuf 3 AX 6.01641414141414
H are fhassuf 3 F 6
ing deletesuf 3 M 5.95238095238095
H don't fgoodright M 5.83333333333333
H years fgoodleft DQ 5.39849012775842
M er fhassuf 2 H 5.2
M my fgoodright H 5.66868686868687
H we fgoodright M 4.8
H be fgoodright AX 5.28767957810511
AX s faddsuf 1 H 5.18349068349068
H ones fgoodleft AX 5.05492610837439
mother goodleft DD 4.3646096289248
AX es fhassuf 2 H 4
H u fhassuf 1 F 4
then hassuf 4 & 4
H has fgoodright HP 3.95789473684211
H too fgoodright AX 3.95402298850575
of hassuf 2 P 3.92857142857143
M isn't fgoodleft AX 3.88399570354458
elf hassuf 3 HP 3.66666666666667
H you fgoodright M 3.5995670995671
M some fgoodright H 4.60498687664042
H could fgoodright M 3.72834645669291
OMN got fgoodleft OXN 3.4010989010989
H came fgoodright AX 3.20498866213152
HP big fgoodleft DD 3.10564226578085
M dle fhassuf 3 H 3
M ty fhassuf 2 AX 3
ous hassuf 3 AX 3
H years fgoodright AX 3
ah hassuf 2 F 3
H no fhassuf 2 F 3
H ' fhassuf 1 P 3
H oh fhaspref 2 EX 3
est hassuf 3 AXT 3
-it deletesuf 3 ATG 3
P she fgoodleft B 2.96361471861472

Appendix 12. The Reduced EPOW Filling Grammar.

[[-2.21565 -6.33061 -7.71691] A CL]
[[-2.66705] A NGP]
[[-0.84167 -7.02376] A PGP]
[[-0.95418 -6.33061] A QQGP]
[[-7.71691] A TEXT]
[[0.00000] AA QQGP]
[[-0.69315] AD PGP]
[[-0.69315] AD QQGP]
[[-0.13353] AF CL]
[[-2.07944] AF QQGP]
[[0.00000] AI QQGP]
[[-0.07036 -3.51942 -4.96634] AL CL]
[[-5.65948] AL PGP]
[[-3.58004] AL QQGP]
[[0.00000] ALREPL CL]
[[0.00000] ALWH QQGP]
[[-0.41552] AM CL]
[[-1.07881] AM QQGP]
[[0.00000] AML CL]
[[0.00000] AN QQGP]
[[0.00000] AP CL]
[[-2.30259] AREPL CL]
[[-2.30259] AREPL NGP]
[[-0.91629] AREPL PGP]
[[-0.91629] AREPL QQGP]
[[0.00000] ATG CL]
[[-4.36945] AWH NGP]
[[-4.36945] AWH PGP]
[[-0.02564] AWH QQGP]
[[-0.53900] AX NGP]
[[-2.48491] AX PGP]
[[-1.09861] AX QQGP]
[[-2.23685 -5.97961 -8.21320] C CL]
[[-0.50225 -4.07007 -5.57414 -7.11459 -8.61867] C NGP]
[[-2.00126 -7.36590 -8.61867] C PGP]
[[-2.16347 -7.52005 -8.61867 -9.31181] C QQGP]
[[-4.45200] C TEXT]
[[-0.06454 -2.77259] CANTIC NGP]
[[-7.13409] CM CL]
[[-5.52466] CM NGP]
[[-4.83151] CM PGP]
[[-0.01608 -5.74780] CM QQGP]
[[0.00000] CP QQGP]
[[-3.16969] CREPL CL]
[[-0.26826 -3.39283 -4.77912] CREPL NGP]
[[-2.29422] CREPL PGP]
[[-2.98736] CREPL QQGP]
[[-3.91797 -7.26787 -7.96102] CV CL]
[[-0.14219 -4.22335 -6.86241 -7.96102 -7.96102] CV NGP]
[[-4.96529] CV PGP]
[[-2.43956 -7.96102 -7.96102] CV QQGP]
[[-7.96102] CV TEXT]
[[-0.08004 -2.56495] CVREPL NGP]
[[-0.69315] CVWH NGP]
[[-0.69315] CVWH QQGP]
[[-4.16148] CWH CL]
[[-0.32923] CWH NGP]
[[-4.85463] CWH PGP]
[[-1.35812] CWH QQGP]
[[-0.05481] DD GC]
[[-3.62434] DD NGP]
[[-3.62434] DD QQGP]
[[0.00000] DO QQGP]
[[-0.00593] DP NGP]
[[-5.12990] DP QQGP]
[[-5.69373] DQ GC]
[[-0.94880 -4.30744] DQ NGP]
[[-0.52895 -5.00058] DQ QQGP]
[[0.00000] DQN QQGP]
[[0.00000] DR NGP]
[[0.00000] DS QQGP]
[[0.00000] DT NGP]
[[0.00000] F QQGP]
[[-1.17865 -3.25810] FI CL]
[[-0.42488] FI PGP]
[[0.00000] HSIT CL]
[[-4.85593] MO NGP]
[[-0.01668 -5.32593 -5.54908] MO QQGP]
[[0.00000] MOA QQGP]
[[-3.41773] MOC NGP]
[[-0.03334] MOC QQGP]
[[0.00000] MOQ QQGP]
[[0.00000] MOSIT CL]
[[-0.02118] MOTH NGP]
[[-3.86523] MOTH QQGP]
[[0.00000] P QQGP]
[[0.00000] PS NGP]
[[-0.88099 -5.38450 -5.78996] Q CL]
[[-2.87219] Q NGP]
[[-0.83413 -5.78996] Q PGP]
[[-2.47577] Q QQGP]
[[-0.69315] QREPL CL]
[[-0.69315] QREPL PGP]
[[-6.69872] S CL]
[[-0.00483 -6.00557 -7.48717 -9.09661] S NGP]
[[-8.40346] S PGP]
[[-8.40346] S QQGP]
[[-2.75154 -3.85015] SANTIC CL]
[[-0.18659 -3.15700 -3.15700] SANTIC NGP]
[[-2.11296 -4.51086] SC CL]
[[-2.90142] SC NGP]
[[-0.23419] SC PGP]
[[-3.81771] SC QQGP]
[[-3.25810] SREPL CL]
[[-0.03922] SREPL NGP]
[[0.00000] SWH NGP]
[[-0.38137] T NGP]
[[-1.14862] T QQGP]
[[-0.00265] V NGP]
[[-5.93489] V QQGP]
[[-0.26847 -1.73170 -3.15969 -4.50744 -5.88536 -6.72161 -7.90027 -7.90027 -8.59341 -9.28656] Z CL]

Appendix 13. The 100 Most Frequent Word-Wordtag Pairs in the EPOW Lexicon.

2679	I	HP
2250	THE	DD
1901	A	DQ
1550	AND	&
1525	IT	HP
1298	YOU	HP
1173	'S	OM
1117	WE	HP
1020	THAT	DD
897	YEAH	F
691	GOT	M
610	THEY	HP
585	NO	F
554	IN	P
523	TO	I
482	PUT	M
417	HE	HP
411	DON'T	ON
401	ONE	HP
400	OF	VO
398	THERE	AX
397	THIS	DD
383	THERE	STH
373	CAN	OM
366	ON	P
353	MY	DD
350	YES	F
349	LOOK	M
341	HAVE	M
320	KNOW	M
318	BE	M
316	ON	AX
314	'LL	OM
305	DO	M
304	NOT	N
295	THEM	HP
294	GO	M
287	ALL	DQ
281	HOUSE	H
279	MAKE	M
278	'VE	OX
269	IF	B
261	WHAT	HWH
253	'S	OX
247	WITH	P
245	GET	M
243	IS	OM
238	NOW	AX
227	WAS	OM
224	UP	AX
211	HERE	AX
210	LIKE	P
198	LITTLE	AX
198	BUT	&
196	'M	OX
192	LIKE	M
192	FOR	P
187	SOME	DQ
186	TO	P
185	TWO	DQ
182	JUST	AI
182	DOOR	H
179	SHE	HP
177	AND-THEN	&
172	THESE	DD
170	BY-THERE	AX
169	OUT	AX
168	GOOD	AX
164	ONE	DQ
162	DO	O
161	PLAY	M
161	GOING-TO	X
161	CAN'T	OMN
160	NEED	M
154	HAVEN'T	OXN
152	ROOF	H
152	HAVE-TO	XM
148	ME	HP
144	IN	AX
137	WHEN	B
131	WANT	M
128	THINK	M
125	WENT	M
125	COME	M
124	WHERE	AXWH
123	MAN	H
120	THINGS	H
118	WINDOW	H
118	GOT-TO	XM
118	COULD	OM
117	WINDOWS	H
117	BUILD	M
117	BIG	AX
114	OFF	AX
112	ONES	HP
112	'RE	OM
109	THING	H
108	CAR	H
107	AN'	&
106	AT	P

Appendix 14. Pocock and Atwell's Weight-Driven Chart Parser.

Major amendments made to Gazdar and Mellish's POP11 chart parser:

```
;;; file ~rob/Chart/V5/chartV5.p

WEIGHT - DRIVEN CHART PARSER: Version 5
Rob Pocock 6/1/93

See '~rob/Chart/0.READ_ME' for information on how to run the chart parser
and an explanation of code.

Changes Made To Basic Algorithm in [Gazdar & Mellish 89] :
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
(1) Separate grammar and lexicon. Lexicon is only accessed when initialising
    agenda.
(2) Grammar and lexicon are stored as property lists (indexed according to
    first rhs constituent), allowing direct access to grammar rules invoked
    in function 'inactive_edge_procedure'.
(3) Edges augmented with a weight field :
        [ ?weight ?start ?finish ?label ?found ?tofind ]
(4) Weights may be either (logarithmised) probabilities or frequencies
    depending on the grammar & lexicon loaded.
(5) 'initialise_agenda' assigns three default wordtags (weighted) to words
    which aren't in the lexicon  -->  NN1, VV0, JJ
(6) Chart is stored as an array of many lists rather than one very long list.
    Allows more precise access - much faster.
(7) Edges in the agenda are ordered according to highest weight first - main
    influence on parser search strategy.
(8) Use a list 'update_list' to store intermediate edges (also ordered). This
    list is then combined with the agenda to create a new updated agenda.
(9) Only check whether new edges to be added to the agenda already exist ( in
    update_list, agenda or chart), in function 'inactive_edge_procedure'.
    These are completely active edges created from invoked grammar rules. All
    other new edges created in 'add_edge' derive from these 'invoked' edges,
    so there is no need to check for them.
(10) Spanning edges (complete parse trees) are augmented with their edge
     number, allowing the 'stage' at which parse trees are found to be
     studied.
(11) Three types of input :
     - sentence of words
     - sequence of grammar wordtags
     - lattice of recognition-candidates
(12) With lattice input, data is read from a textfile whose filename is
     contained in global variable 'lattice_filename'. For an example of
     required format, examine ~rob/Chart/Data/STEPHEN.l
(13) Pop11 efficiency details :
     - attempted to avoid creating new lists when updating old lists, so as to
       save on memory and garbage collection. eg. NCREV, NC_<>
     - fast integer functions eg. fi_+, fi_>
     - compile time macros #_< and >_#
     - compile procedures as being constant
       eg. compile_mode:pop11 +defcon   (in ~rob/Chart/V5/initV5.p)
     - use '==' wherever possible
     - use 'until list == [] do' ... 'enduntil' for accessing list contents
     - use 'syslockheap' and 'sysgarbage'
```

Appendix 15. Parser Version 7: Test Results.

Diagnostics displayed when version 7 of the parser is loaded:

```
:load parseV7.p
;;; LOADING parseV7.p

Using EPOW trained BRILL tagger:

;;; LOADING /home/csuna_cl/staff/cs/pop/parser/lookup.p

Loading parsing code :

;;; LOADING /home/csuna_cl/staff/cs/pop/parser/initV5.p
;;; LOADING /home/csuna_cl/staff/cs/pop/parser/outputV7.p
;;; LOADING /home/csuna_cl/staff/cs/pop/parser/chartV7.p
;;; LOADING /home/csuna_cl/staff/cs/pop/parser/inputV5.p
;;; LOADING /home/csuna_cl/staff/cs/pop/parser/problem.p
;;; LOADING /home/csuna_cl/staff/cs/pop/parser/out.p
;;; LOADING /home/csuna_cl/staff/cs/pop/parser/property.p
;;; LOADING /home/csuna_cl/staff/cs/pop/parser/TRACE_chartV7.p

Loaded parsing code - Version 7

Look at textfile ~nlp/SOPP/Chart/0.READ_ME for more information.

Popmemlim = 20000000

Loading collapsed POW systemic functional grammar (1898 rules) :

Creating grammar property list (gpl) :

Creating vertical triples property list (vpl) :

Grammar loaded, ready to parse
Use commands      - parse("W",[]) - for sentence of words
                  - parse("G",[]) - for sequence of wordtags
                  - parse("L",[]) - for lattice of words

;;; Parameters set to the following three way stopping mechanism for parser:
;;; maxparses = 6,
;;; finalweight = 1.5 * firstweight
;;; max agenda length = 50000 * strlen
```

Utterance 1

```
Type words to parse in lower case except for "I" and proper nouns :
 -> : why

POW solution:
[Z [CL [AWH [QQGP [AXWH WHY]]]]]

 Starting TRACE parsing process - Version 7

[why AXWH]

 Lexical lookup results for sentence:
why
AXWH


lattice_filename = LATTICE
chart_filename =  EDGES
chart_to_file = <false>
only_inactive = <true>
almost_inactive = <false>

Result of parse :
[[[W -128437 Z [Z [CL [A [QQGP [AXWH why]]]]] 6]]
   [[W -130227 Z [Z [CL [AWH [QQGP [AXWH why]]]]] 6]]
   [[W -131189 Z [Z [CL [C [QQGP [AXWH why]]]]] 6]]
   [[W -155665 Z [Z [CL [CWH [QQGP [AXWH why]]]]] 6]]
   [[W -173309 Z [Z [CL [AL [QQGP [AXWH why]]]]] 6]]
   [[W -175388 Z [Z [CL [C [PGP [CV [QQGP [AXWH why]]]]]] 8]]]

Number of edges in chart = 133
Number of inactive edges = 40
Number of almost inactive edges = 0
Number of active edges = 97

Number of solution parse trees : 6
E = edge number   and   W = weight

Time to execute (variable TIME) = 3.47
Number chart edges (variable chart_edges) = 133
```

Utterance 2

```
Type words to parse in lower case except for "I" and proper nouns :
 -> :what 's the point

POW solution:
[Z [CL [CWH [NGP [HWH WHAT]]]
       [OM 'S]
       [S [NGP [DD THE] [H POINT]]]]]

 Starting TRACE parsing process - Version 7


 Lexical lookup results for sentence:
what
```

HWH

's
OM

the
DD

point
H

lattice_filename = LATTICE
chart_filename = EDGES
chart_to_file = <true>
only_inactive = <true>
almost_inactive = <false>

Result of parse :
```
[[[W -149624 Z
      [Z [CL [S [NGP [HWH what]]]
            [OM 's]
            [C [NGP [DD the] [H point]]]]]
      6]]
   [[W -185372 Z
      [Z [CL [S [NGP [HWH what]]]
            [OM 's]
            [C [NGP [DD the]]]
            [C [NGP [H point]]]]]
      6]]
   [[W -196983 Z
      [Z [CL [CWH [NGP [HWH what]]]
            [OM 's]
            [S [NGP [DD the] [H point]]]]]
      6]]
   [[W -199622 Z
      [Z [CL [S [NGP [HWH what]]]
            [OM 's]
            [C [PGP [CV [NGP [DD the] [H point]]]]]]]
      8]]
   [[W -207834 Z
      [Z [CL [S [NGP [HWH what]]]
            [OM 's]
            [A [PGP [CV [NGP [DD the] [H point]]]]]]]
      8]]
   [[W -210985 Z
      [Z [CL [C [NGP [HWH what]]]
            [OM 's]
            [S [NGP [DD the] [H point]]]]]
      6]]]
```

Number of edges in chart = 3637
Number of inactive edges = 1772
Number of almost inactive edges = 769
Number of active edges = 2033
Number of edges in output file = 1604

Number of solution parse trees : 6
E = edge number and W = weight

Time to execute (variable TIME) = 3340.83
Number chart edges (variable chart_edges) = 3637

listlength(agenda)==> 18112

Utterance 3

Type words to parse in lower case except for "I" and proper nouns :
-> : you put these on for windows

POW solution:
```
[Z [CL [S [NGP [HP YOU]]]
          [M PUT]
          [C [NGP [DD THESE]]]
          [CM [QQGP [AX ON]]]
          [A [PGP [P FOR] [CV [NGP [H WINDOWS]]]]]]]
```

Starting TRACE parsing process - Version 7

Lexical lookup results for sentence:
```
you
HP

put
M

these
DD

on
AX

for
P

windows
H
```

```
lattice_filename = LATTICE
chart_filename =  EDGES
chart_to_file = <true>
only_inactive = <true>
almost_inactive = <false>

[-230118 0 6 Z
        [[CL [S [NGP [HP you]]]
            [M put]
            [C [NGP [DD these] [MO [QQGP [AX on]]]]]
            [A [PGP [P for] [CV [NGP [H windows]]]]]]
        [] 8]
[-239344 0 6 Z
        [[CL [S [NGP [HP you]]]
            [M put]
            [C [NGP [DD these]
                  [MO [QQGP [AX on] [FI [PGP [P for]]]]]
                  [H windows]]]]]
        [] 10]
[-240631 0 6 Z
        [[CL [S [NGP [HP you]]]
            [M put]
```

```
            [C [NGP [DD these]]]
            [A [QQGP [AX on]]]
            [A [PGP [P for] [CV [NGP [H windows]]]]]]]
        [] 8]
[-240813 0 6 Z
        [[CL [S [NGP [HP you]]]
            [M put]
            [C [NGP [DD these] [MO [QQGP [AX on]]]]]
            [C [PGP [P for] [CV [NGP [H windows]]]]]]]
        [] 8]
[-241124 0 6 Z
        [[CL [S [NGP [HP you]]]
            [M put]
            [C [NGP [DD these]]]
            [CM [QQGP [AX on]]]
            [A [PGP [P for] [CV [NGP [H windows]]]]]]]
        [] 8]
```

suspended after one week running at nice value 5. - agenda reached:

WEIGHT = -242092 ; Chart edge number 19498
listlength(agenda)=> 419619

Utterance 4

Type words to parse in lower case except for "I" and proper nouns :
-> : you don’t have-to

POW solution:

```
[Z [CL [S [NGP [HP YOU]]]
    [ON DON']
    [XM HAVE-TO]
    [(M)]
    [(C)]
    [(CM)]]]
```

Starting TRACE parsing process - Version 7

Lexical lookup results for sentence:
you
HP

don't
ON

have-to
XM

Result of parse :

```
[[[W -115180 Z
    [Z [CL [S [NGP [HP you]]] [ON don't] [XM have-to]]]
    6]]]
```

Number of edges in chart = 846
Number of inactive edges = 203
Number of almost inactive edges = 209
Number of active edges = 658
Number of edges in output file = 188

```
Number of solution parse trees : 1
E = edge number   and   W = weight

Time to execute (variable TIME) = 107.32
Number chart edges (variable chart_edges) = 846
listlength(agenda)=> 2170
```

Utterance 5

```
Type words to parse in lower case except for "I" and proper nouns :
 -> : won't be long

POW solution:
[Z [CL [(S)]
      [OMN WON'T]
      [M BE]
      [C [QQGP [AX LONG]]]]]

 Starting TRACE parsing process - Version 7


 Lexical lookup results for sentence:
won't
OMN

be
M

long
AX

Result of parse :
[[[W -117536 Z
      [Z [CL [OMN won't] [M be] [C [QQGP [AX long]]]]]
      6]]
   [[W -124709 Z
      [Z [CL [OMN won't] [M be] [C [NGP [MO [QQGP [AX long]]]]]]]
      8]]
   [[W -136655 Z
      [Z [CL [OMN won't] [M be] [CM [QQGP [AX long]]]]]
      6]]
   [[W -140498 Z
      [Z [CL [OMN won't] [M be] [A [QQGP [AX long]]]]]
      6]]]

Number of edges in chart = 1026
Number of inactive edges = 510
Number of almost inactive edges = 302
Number of active edges = 558
Number of edges in output file = 468

Number of solution parse trees : 4
E = edge number   and   W = weight

Time to execute (variable TIME) = 148.01
Number chart edges (variable chart_edges) = 1026
listlength(agenda)=> 2808
```

Utterance 6

```
Type words to parse in lower case except for "I" and proper nouns :
 -> : it 's easiest mind

POW solution:
[Z [CL [S [NGP [HP IT]]]
      [OM 'S]
      [C [QQGP [AX EASIEST]]]
      [AF MIND]]]

Starting TRACE parsing process - Version 7
Lexical lookup results for sentence:
it
HP

's
OX

easiest
AXT

mind
M

 [-211132 0 4 Z
          [[CL [S [NGP [HP it]]]
              [OX 's]
              [A [QQGP [AXT easiest]]]
              [M mind]]]
          [] 6]
[-269891 0 4 Z
          [[CL [S [NGP [HP it]]]
              [OX 's]
              [A [PGP [CV [NGP [MO [QQGP [AXT easiest]]]]]]]
              [M mind]]]
          [] 10]
[-302521 0 4 Z
          [[CL [S [NGP [HP it]]]
              [OX 's]
              [A [CL [C [QQGP [AXT easiest]]]]]
              [M mind]]]
          [] 8]
[-306592 0 4 Z
          [[CL [C [TEXT [Z [CL [S [NGP [HP it]]]
                               [OX 's]
                               [A [QQGP [AXT easiest]]]
                               [M mind]]]]]]]
          [] 10]
[-306816 0 4 Z
          [[CL [S [NGP [HP it]]]
              [OX 's]
              [C [QQGP [AXT easiest]]]
              [ATG [CL [M mind]]]]]
          [] 6]

(suspended after 20 hours)
listlength(agenda)=> 94765
chart_edges=> 11766
```

Utterance 7

```
Type words to parse in lower case except for "I" and proper nouns :
 -> : I know something easy

POW solution:
[Z [CL [S [NGP [HP I]]]
      [M KNOW]
      [C [NGP [HP SOMETHING] [MO [QQGP [AX EASY]]]]]]
  [CL [(S)]
      [(OM)]
      [C [CL [M BUILD]
            [C [NGP [DQ A] [H GARAGE]]]]]]

 Starting TRACE parsing process - Version 7


 Lexical lookup results for sentence:
I
HP

know
M

something
HP

easy
AX

Result of parse :
[[[W -156121 Z
      [Z [CL [S [NGP [HP I]]]
            [M know]
            [C [NGP [HP something]]]
            [A [QQGP [AX easy]]]]]
      6]]
   [[W -156614 Z
      [Z [CL [S [NGP [HP I]]]
            [M know]
            [C [NGP [HP something]]]
            [CM [QQGP [AX easy]]]]]
      6]]
   [[W -171195 Z
      [Z [CL [S [NGP [HP I]]]
            [M know]
            [C [NGP [HP something]]]
            [C [QQGP [AX easy]]]]]
      6]]
   [[W -178368 Z
      [Z [CL [S [NGP [HP I]]]
            [M know]
            [C [NGP [HP something]]]
            [C [NGP [MO [QQGP [AX easy]]]]]]]
      8]]
   [[W -204267 Z
      [Z [CL [S [NGP [HP I]]]
            [M know]
            [A [PGP [CV [NGP [HP something]]]]]
            [A [QQGP [AX easy]]]]]
```

```
      8]]
   [[W -208176 Z
      [Z [CL [S [NGP [HP I]]]
            [M know]
            [C [PGP [CV [NGP [HP something]]]]]
            [A [QQGP [AX easy]]]]]
      8]]]

Number of edges in chart = 4361
Number of inactive edges = 1866
Number of almost inactive edges = 954
Number of active edges = 2680
Number of edges in output file = 1681

Number of solution parse trees : 6
E = edge number   and   W = weight

Time to execute (variable TIME) = 6564.48
Number chart edges (variable chart_edges) = 4361
listlength(agenda)=>29450
```

Utterance 8

```
Type words to parse in lower case except for "I" and proper nouns :
 -> : build a garage


POW solution:
[Z [CL [S [NGP [HP I]]]
      [M KNOW]
      [C [NGP [HP SOMETHING] [MO [QQGP [AX EASY]]]]]]
   [CL [(S)]
      [(OM)]
      [C [CL [M BUILD]
            [C [NGP [DQ A] [H GARAGE]]]]]]

 Starting TRACE parsing process - Version 7


 Lexical lookup results for sentence:
build
M

a
DQ

garage
H

Result of parse :
[[[W -69622 Z [Z [CL [M build] [C [NGP [DQ a] [H garage]]]]] 6]]]

Number of edges in chart = 618
Number of inactive edges = 113
Number of almost inactive edges = 254
Number of active edges = 513
Number of edges in output file = 105

Number of solution parse trees : 1
```

E = edge number and W = weight

Time to execute (variable TIME) = 21.9
Number chart edges (variable chart_edges) = 618
listlength(agenda)=>756

Utterance 9

Type words to parse in lower case except for "I" and proper nouns :
-> : fantastic

POW solution:
[Z [CL [EX FANTASTIC]]]

Starting TRACE parsing process - Version 7

Lexical lookup results for sentence:
fantastic
AX

Result of parse :
[[[W -71709 Z [Z [CL [C [QQGP [AX fantastic]]]]] 6]]
[[W -78882 Z [Z [CL [C [NGP [MO [QQGP [AX fantastic]]]]]] 8]]
[[W -89608 Z [Z [CL [A [QQGP [AX fantastic]]]]] 6]]
[[W -106822 Z [Z [CL [S [NGP [MO [QQGP [AX fantastic]]]]]] 8]]]

Number of edges in chart = 218
Number of inactive edges = 70
Number of almost inactive edges = 126
Number of active edges = 156
Number of edges in output file = 62

Number of solution parse trees : 4
E = edge number and W = weight

Time to execute (variable TIME) = 9.06
Number chart edges (variable chart_edges) = 218
listlength(agenda)=>172

Utterance 10

Type words to parse in lower case except for "I" and proper nouns :
->: or something like a skyscraper

POW solution:
[Z [CL [& OR]
[(M)]
[C [NGP [HP SOMETHINK]
[Q [PGP [P LIKE] [CV [NGP [DQ A] [H SKYSCRAPER]]]]]]]]]

Starting TRACE parsing process - Version 7

Lexical lookup results for sentence:
or
&

```
something
HP

like
P

a
DQ

skyscraper
H

Result of parse :
[[[W -188054 Z
      [Z [CL [& or]
            [S [NGP [HP something]]]
            [A [PGP [P like] [CV [NGP [DQ a] [H skyscraper]]]]]]]
      8]]
   [[W -208753 Z
      [Z [CL [& or]
            [C [NGP [HP something]]]
            [A [PGP [P like] [CV [NGP [DQ a] [H skyscraper]]]]]]]
      8]]
   [[W -210315 Z
      [Z [CL [& or]
            [C [NGP [HP something]]]
            [C [PGP [P like] [CV [NGP [DQ a] [H skyscraper]]]]]]]
      8]]
   [[W -222938 Z
      [Z [CL [& or]
            [S [NGP [HP something]]]
            [C [PGP [P like] [CV [NGP [DQ a] [H skyscraper]]]]]]]
      8]]
   [[W -235183 Z
      [Z [CL [& or]
            [C [NGP [HP something]]]
            [C [PGP [P like]]]
            [A [PGP [CV [NGP [DQ a] [H skyscraper]]]]]]]
      8]]
   [[W -240944 Z
      [Z [CL [& or]
            [C [NGP [HP something] [Q [PGP [P like]]]]]
            [C [NGP [DQ a] [H skyscraper]]]]]
      8]]]

Parse trees are stored in global variable 'TREES'.
Trees are stored in the order in which they were found.
'TREES' also has total number of chart edges and parse time.

Number of edges in chart = 10548
Number of inactive edges = 4331
Number of almost inactive edges = 1833
Number of active edges = 6676
Number of edges in output file = 3872

Number of solution parse trees : 6
E = edge number   and   W = weight

Time to execute (variable TIME) = 59180.1
Number chart edges (variable chart_edges) = 10548
```

Utterance 11a

```
Type words to parse in lower case except for "I" and proper nouns :
 ->: this worked out

POW solution:
[Z [CL [S [NGP [DD THIS]]]
       [M WORKED]
       [CM [QQGP [AX OUT]]]]
   [CL [S [NGP [HP IT]]]
       [OMN WON'T]
       [M FIT]]]

 Starting TRACE parsing process - Version 7


 Lexical lookup results for sentence:
this
DD

worked
M

out
AX

Result of parse :
[[[W -104086 Z
       [Z [CL [S [NGP [DD this]]] [M worked] [C [QQGP [AX out]]]]]
       6]]
    [[W -111259 Z
       [Z [CL [S [NGP [DD this]]]
             [M worked]
             [C [NGP [MO [QQGP [AX out]]]]]]]
       8]]
    [[W -120251 Z
       [Z [CL [S [NGP [DD this]]] [M worked] [CM [QQGP [AX out]]]]]
       6]]
    [[W -126831 Z
       [Z [CL [S [NGP [DD this]]] [M worked] [A [QQGP [AX out]]]]]
       6]]]

Number of edges in chart = 1322
Number of inactive edges = 433
Number of almost inactive edges = 416
Number of active edges = 932
Number of edges in output file = 390

Number of solution parse trees : 4
E = edge number   and   W = weight

Time to execute (variable TIME) = 201.38
Number chart edges (variable chart_edges) = 1322
listlength(agenda)=>3269
```

Utterance 11b

```
Type words to parse in lower case except for "I" and proper nouns :
 ->: it won’t fit

POW solution:
[Z [CL [S [NGP [DD THIS]]]
       [M WORKED]
       [CM [QQGP [AX OUT]]]]
   [CL [S [NGP [HP IT]]]
       [OMN WON'T]
       [M FIT]]]

 Starting TRACE parsing process - Version 7


 Lexical lookup results for sentence:
it
HP

won't
OMN

fit
M

Result of parse :
[[[W -82141 Z [Z [CL [S [NGP [HP it]]] [OMN won't] [M fit]]] 6]]]

Number of edges in chart = 616
Number of inactive edges = 107
Number of almost inactive edges = 178
Number of active edges = 520
Number of edges in output file = 96

Number of solution parse trees : 1
E = edge number   and   W = weight

Time to execute (variable TIME) = 26.72
Number chart edges (variable chart_edges) = 616
listlength(agenda)=>748
```

Utterance 11

```
Type words to parse in lower case except for "I" and proper nouns :
 ->: this worked out it won’t fit

POW solution:
[Z [CL [S [NGP [DD THIS]]]
       [M WORKED]
       [CM [QQGP [AX OUT]]]]
   [CL [S [NGP [HP IT]]]
       [OMN WON'T]
       [M FIT]]]

 Starting TRACE parsing process - Version 7
```

```
 Lexical lookup results for sentence:
this
DD

worked
M

out
AX

it
HP

won't
OMN

fit
M

Result of parse :
[[[W -244397 Z
       [Z [CL [S [NGP [DD this]]]
             [M worked]
             [C [NGP [MO [QQGP [AX out]]] [HP it]]]]
          [CL [OMN won't] [M fit]]]
       8]]
    [[W -256507 Z
       [Z [CL [S [NGP [DD this]]] [M worked]]
          [CL [S [NGP [MO [QQGP [AX out]]] [HP it]]]
             [OMN won't]
             [M fit]]]
       8]]
    [[W -263031 Z
       [Z [CL [S [NGP [DD this]]] [M worked] [C [QQGP [AX out]]]]
          [CL [S [NGP [HP it]]] [OMN won't] [M fit]]]
       6]]
    [[W -270204 Z
       [Z [CL [S [NGP [DD this]]]
             [M worked]
             [C [NGP [MO [QQGP [AX out]]]]]]
          [CL [S [NGP [HP it]]] [OMN won't] [M fit]]]
       8]]
    [[W -274449 Z
       [Z [CL [S [NGP [DD this]]]
             [M worked]
             [CM [QQGP [AX out]]]
             [C [NGP [HP it]]]]
          [CL [OMN won't] [M fit]]]
       6]]
    [[W -277319 Z
       [Z [CL [S [NGP [DD this]]]
             [M worked]
             [CM [QQGP [AX out]]]
             [C [TEXT [Z [CL [S [NGP [HP it]]]
                           [OMN won't]
                           [M fit]]]]]]]
       10]]]

Number of edges in chart = 17647
Number of inactive edges = 4698
Number of almost inactive edges = 5744
```

Number of active edges = 13456
Number of edges in output file = 4191

Number of solution parse trees : 6
E = edge number and W = weight

Time to execute (variable TIME) = -106244.0
Number chart edges (variable chart_edges) = 17647
listlength(agenda)=>231816

Utterance 12

Type words to parse in lower case except for "I" and proper nouns :
->: go on

POW solution:
[Z [CL [M GO] [CM [QQGP [AX ON]]]]
[CL [S [NGP [HP WE]]]
[OM CAN]
[AI ALWAYS]
[M MOVE]
[C [NGP [HP IT]]]
[CM [QQGP [AX ALONG]]]
[ATG [CL [OMN CAN'T] [S [NGP [HP WE]]]]]]]

Starting TRACE parsing process - Version 7

Lexical lookup results for sentence:
go
M

on
P

Result of parse :
[[[W -58720 Z [Z [CL [M go] [C [PGP [P on]]]]] 6]]
[[W -76969 Z [Z [CL [M go] [A [PGP [P on]]]]] 6]]]

Number of edges in chart = 161
Number of inactive edges = 36
Number of almost inactive edges = 89
Number of active edges = 128
Number of edges in output file = 33

Number of solution parse trees : 2
E = edge number and W = weight

Time to execute (variable TIME) = 2.74
Number chart edges (variable chart_edges) = 161
listlength(agenda)=>188

Utterance 13

Type words to parse in lower case except for "I" and proper nouns :
->: we can always move it along can’t we

POW solution:

```
[Z [CL [M GO] [CM [QQGP [AX ON]]]]
  [CL [S [NGP [HP WE]]]
      [OM CAN]
      [AI ALWAYS]
      [M MOVE]
      [C [NGP [HP IT]]]
      [CM [QQGP [AX ALONG]]]
      [ATG [CL [OMN CAN'T] [S [NGP [HP WE]]]]]]]

 Starting TRACE parsing process - Version 7


 Lexical lookup results for sentence:
we
HP

can
OM

always
AX

move
M

it
HP

along
AX

can't
OMN

we
HP

 (suspended after 3.5 days): Agenda reached:

WEIGHT = -234803   ;   Chart edge number 19514
listlength(agenda)=> 251634
listlength(chart(0,8))=>
0
listlength(chart(0,7))=>
0
listlength(chart(0,6))=>
0
listlength(chart(0,5))=>
5
listlength(chart(0,4))=>
52
listlength(chart(0,3))=>
466
listlength(chart(0,2))=>
546
listlength(chart(0,1))=>
1844
listlength(chart(0,0))=>
1108
listlength(chart(1,8))=>
0
```

```
listlength(chart(2,8))=>
0
listlength(chart(3,8))=>
0
listlength(chart(4,8))=>
0
listlength(chart(1,7))=>
0
listlength(chart(1,6))=>
5
listlength(chart(1,5))=>
20
chart(0,5)==>
[[-234089 0 5 CL
          [[S [NGP [HP we]]]
           [OM can]
           [C [NGP [MO [QQGP [AX always]]]]]
           [C [CL [M move] [C [NGP [HP it]]]]]]
          [[A]]
          7]
   [-232604 0 5 CL
          [[S [NGP [HP we]]]
           [OM can]
           [C [QQGP [AX always]]]
           [C [TEXT [Z [CL [M move] [C [NGP [HP it]]]]]]]]
          [[A]]
          9]
   [-226916 0 5 CL
          [[S [NGP [HP we]]]
           [OM can]
           [C [QQGP [AX always]]]
           [C [CL [M move] [C [NGP [HP it]]]]]]
          [[A]]
          7]
   [-208906 0 5 Z
          [[CL [S [NGP [HP we]]]
              [OM can]
              [A [QQGP [AX always]]]
              [M move]
              [C [NGP [HP it]]]]]
          [] 6]
   [-196612 0 5 CL
          [[S [NGP [HP we]]]
           [OM can]
           [A [QQGP [AX always]]]
           [M move]
           [C [NGP [HP it]]]]
          [] 5]]
chart(1,6)==>
[[-234641 1 6 CL
          [[OM can]
           [S [NGP [MO [QQGP [AX always]]]]]
           [M move]
           [C [NGP [HP it]]]
           [C [QQGP [AX along]]]]
          [[AF]]
          7]
   [-232460 1 6 Z
          [[CL [OM can]
              [S [NGP [MO [QQGP [AX always]]]]]
              [M move]
```

```
            [C [NGP [HP it]]]
            [CM [QQGP [AX along]]]]]
         [] 8]
   [-223310 1 6 Z
         [[CL [OM can]
            [S [NGP [MO [QQGP [AX always]]]]]
            [M move]
            [C [NGP [HP it]]]
            [A [QQGP [AX along]]]]]
         [] 8]
   [-218886 1 6 CL
         [[OM can]
          [S [NGP [MO [QQGP [AX always]]]]]
          [M move]
          [C [NGP [HP it]]]
          [CM [QQGP [AX along]]]]
         [[A]]
         7]
   [-211016 1 6 CL
         [[OM can]
          [S [NGP [MO [QQGP [AX always]]]]]
          [M move]
          [C [NGP [HP it]]]
          [A [QQGP [AX along]]]]
         [] 7]]
chart_edges=>19518
(process size was around 85 Mb)
```

Utterance 14

```
Type words to parse in lower case except for "I" and proper nouns :
 ->: will that one fit in by-there

POW solution:
[Z [CL [OM WILL]
      [S [NGP [DD THAT] [HP ONE]]]
      [M FIT]
      [CM [QQGP [AX IN]]]
      [C [QQGP [AX BY-THERE]]]]]

 Starting TRACE parsing process - Version 7


 Lexical lookup results for sentence:
will
OM

that
DD

one
HP

fit
M

in
AX
```

by-there
AX

Result of parse :
```
[[[W -192992 Z
      [Z [CL [OM will]
            [S [NGP [DD that] [HP one]]]
            [M fit]
            [C [QQGP [AX in]]]
            [A [QQGP [AX by-there]]]]]
      6]]
   [[W -200165 Z
      [Z [CL [OM will]
            [S [NGP [DD that] [HP one]]]
            [M fit]
            [C [NGP [MO [QQGP [AX in]]]]]
            [A [QQGP [AX by-there]]]]]
      8]]
   [[W -202142 Z
      [Z [CL [OM will]
            [S [NGP [DD that] [HP one]]]
            [M fit]
            [C [QQGP [AX in]]]
            [CM [QQGP [AX by-there]]]]]
      6]]
   [[W -207251 Z
      [Z [CL [OM will]
            [S [NGP [DD that] [HP one]]]
            [M fit]
            [CM [QQGP [AX in]]]
            [C [QQGP [AX by-there]]]]]
      6]]
   [[W -209315 Z
      [Z [CL [OM will]
            [S [NGP [DD that] [HP one]]]
            [M fit]
            [C [NGP [MO [QQGP [AX in]]]]]
            [CM [QQGP [AX by-there]]]]]
      8]]
   [[W -214424 Z
      [Z [CL [OM will]
            [S [NGP [DD that] [HP one]]]
            [M fit]
            [CM [QQGP [AX in]]]
            [C [NGP [MO [QQGP [AX by-there]]]]]]]
      8]]]
```

Number of edges in chart = 8074
Number of inactive edges = 2119
Number of almost inactive edges = 1665
Number of active edges = 6279
Number of edges in output file = 1795

Number of solution parse trees : 6
E = edge number and W = weight

Time to execute (variable TIME) = 7715.26
Number chart edges (variable chart_edges) = 8074
listlength(agenda)=> 25850

Utterance 15

```
Type words to parse in lower case except for "I" and proper nouns :
 ->: come on

POW solution:
[Z [CL [M COME] [CM [QQGP [AX ON]]]]
   [CL [O LETS] [M GET] [C [CL [M GOING]]]]]

 Starting TRACE parsing process - Version 7


 Lexical lookup results for sentence:
come
M

on
P

Result of parse :
[[[W -58720 Z [Z [CL [M come] [C [PGP [P on]]]]] 6]]
    [[W -76969 Z [Z [CL [M come] [A [PGP [P on]]]]] 6]]]

Number of edges in chart = 161
Number of inactive edges = 36
Number of almost inactive edges = 89
Number of active edges = 128
Number of edges in output file = 33

Number of solution parse trees : 2
E = edge number   and   W = weight

Time to execute (variable TIME) = 2.5
Number chart edges (variable chart_edges) = 161
listlength(agenda)=> 188
```

Utterance 16

```
Type words to parse in lower case except for "I" and proper nouns :
 ->: let’s get going

POW solution:
[Z [CL [M COME] [CM [QQGP [AX ON]]]]
   [CL [O LETS] [M GET] [C [CL [M GOING]]]]]

 Starting TRACE parsing process - Version 7


 Lexical lookup results for sentence:
let's
O

get
X

going
M
```

(suspended after 2 hours)
listlength(agenda)=> 17131
chart_edges=> 6577

Utterance 17

Type words to parse in lower case except for "I" and proper nouns :
->: I can’t even

POW solution:
[Z [CLUN [S [NGP [HP I]]] [OMN CAN'T] [AI EVEN]]]

Starting TRACE parsing process - Version 7

Lexical lookup results for sentence:
I
HP

can't
OMN

even
AI

Result of parse :
[[[W -117496 Z [Z [CL [S [NGP [HP I]]] [OMN can't] [AI even]]] 6]]]

Number of edges in chart = 873
Number of inactive edges = 204
Number of almost inactive edges = 238
Number of active edges = 683
Number of edges in output file = 190

Number of solution parse trees : 1
E = edge number and W = weight

Time to execute (variable TIME) = 119.16
Number chart edges (variable chart_edges) = 873

listlength(agenda)=> 2001

Utterance 18

Type words to parse in lower case except for "I" and proper nouns :
->: press SHIFT+INS or CTRL+V

Starting TRACE parsing process - Version 7

Lexical lookup results for sentence:
press
M

SHIFT+INS
HN

or

```
&

CTRL+V
HN

Result of parse :
[[[W -176832 Z
      [Z [CL [M press]
            [C [NGP [HN SHIFT+INS]]]
            [C [NGP [& or] [HN CTRL+V]]]]]
      6]]
   [[W -195038 Z
      [Z [CL [M press]
            [C [NGP [HN SHIFT+INS]] [NGP [& or] [HN CTRL+V]]]]]
      6]]
   [[W -198246 Z
      [Z [CL [M press]
            [C [NGP [HN SHIFT+INS]]]
            [V [NGP [& or] [HN CTRL+V]]]]]
      6]]
   [[W -223946 Z
      [Z [CL [M press] [C [NGP [HN SHIFT+INS]]]]
        [CL [C [NGP [& or] [HN CTRL+V]]]]]
      6]]
   [[W -224051 Z
      [Z [CL [M press]
            [C [PGP [CV [NGP [HN SHIFT+INS]]]]]
            [C [NGP [& or] [HN CTRL+V]]]]]
      8]]
   [[W -225243 Z
      [Z [CL [M press]
            [S [NGP [HN SHIFT+INS]]]
            [C [NGP [& or] [HN CTRL+V]]]]]
      6]]]

Number of edges in chart = 3640
Number of inactive edges = 1033
Number of almost inactive edges = 1216
Number of active edges = 2722
Number of edges in output file = 918

Number of solution parse trees : 6
E = edge number   and   W = weight

Time to execute (variable TIME) = 1986.51
Number chart edges (variable chart_edges) = 3640
listlength(agenda)=>14319
```

Utterance 19

```
Type words to parse in lower case except for "I" and proper nouns :
 ->: what do we mean by this

 Starting TRACE parsing process - Version 7


 Lexical lookup results for sentence:
what
HWH
```

do
O

we
HP

mean
M

by
P

this
DD

[-256767 0 6 Z
[[CL [CWH [NGP [HWH what]]]
[O do]
[S [NGP [HP we]]]
[M mean]
[A [PGP [P by] [CV [NGP [DD this]]]]]]]
[] 8]
[-277606 0 6 Z
[[CL [CWH [NGP [HWH what]]]
[O do]
[S [NGP [HP we]]]
[M mean]
[A [PGP [P by]]]
[A [PGP [CV [NGP [DD this]]]]]]]
[] 8]
[-281842 0 6 Z
[[CL [CWH [NGP [HWH what]]]
[O do]
[S [NGP [HP we]]]
[M mean]
[C [PGP [P by] [CV [NGP [DD this]]]]]]]
[] 8]

(suspended after 16 hours, agenda reached):
WEIGHT = -312013 ; Chart edge number 12367
listlength(agenda)=> 94925

Utterance 20

Type words to parse in lower case except for "I" and proper nouns :
->: select the text you want to protect

Starting TRACE parsing process - Version 7

Lexical lookup results for sentence:
select
P

the
DD

text

H

you
HP

want
M

to
I

protect
M
(suspended after 7 days)

Utterance 21

Type words to parse in lower case except for "I" and proper nouns :
->: that is these words make the source sentence longer or shorter than the TM sentence

Starting TRACE parsing process - Version 7

Lexical lookup results for sentence:
that
DD

is
OM

these
DD

words
H

make
M

the
DD

source
H

sentence
H

longer
AXT

or
&

shorter
H

than
P

the
DD

TM
HN

sentence
H

(suspended - unfinished)

Utterance 22

Type words to parse in lower case except for "I" and proper nouns :
 ->: displays the records that have a specific word or words in the TITLE CONTENTS SUBJECT or SERIES fields of the BIB record depending on which fields have been included in each index

 Starting TRACE parsing process - Version 7

 Lexical lookup results for sentence:
displays
P

the
DD

records
H

that
DD

have
M

a
DQ

specific
H

word
H

or
&

words
H

in
P

the
DD

TITLE

HN

CONTENTS
HN

SUBJECT
HN

or
&

SERIES
HN

fields
H

of
VO

the
DD

BIB
HN

record
H

depending
M

on
P

which
DDWH

fields
H

have
X

been
X

included
M

in
P

each
DQ

index
H

(Suspended unfinished)

Utterance 23

Type words to parse in lower case except for "I" and proper nouns :
->: enter the line number of the alphabetical title search option

Starting TRACE parsing process - Version 7

Lexical lookup results for sentence:
enter
P

the
DD

line
H

number
H

of
VO

the
DD

alphabetical
H

title
H

search
H

option
H

(Suspended unfinished)

End of Appendix 15.