

Treebank annotation with a wide-coverage Head-Driven Phrase Structure Grammar

**by
Dag Schmidtke**

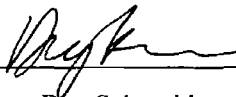
**School of Computing,
Dublin City University,
Glasnevin,
Dublin 9.**

Supervisor: Prof. Josef van Genabith

**A dissertation submitted for the degree of
Master of Science by Research**

June 2004

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of M Sc by Research is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work

Signed  _____
Dag Schmidtke

(Candidate) ID No 96970626

Date June 30, 2004

Abstract

In this dissertation I investigate ways to extend the annotation of treebanks, or parsed corpora, by taking advantage of the rich and sophisticated grammatical analysis embodied in a modern, constraint-based wide-coverage grammar. As the underlying processing engine I implement a full typed feature structure inference and HPSG parsing system in C#. I develop a method for annotating a treebank with typed feature structure information with the use of the LinGO ERG grammar, an existing wide-coverage Head-Driven Phrase Structure Grammar (HPSG). I use standard techniques to head-lexicalise and binarise the treebank and further pre-process it to make it more compatible with the general grammatical structures assumed in HPSG. I then establish a mapping between local CFG and HPSG configurations and map local trees to HPSG phrase types. Finally the typed feature structures associated with the local trees are combined to complete resolved HPSG signs through constraint resolution and by applying the rules from the HPSG grammar. Discrepancies between the treebank and the HPSG grammar are analysed with respect to implications for grammar extension and automatic rich lexicon entry acquisition is also investigated.

The aim of this work is to develop a method of constraint-based grammar-driven treebank annotation combining data- and theory-driven approaches to NLP. With this I aim to produce a richer treebank to demonstrate the benefits of using an existing wide-coverage grammar for treebank annotation and to explore ways of using treebanks to extend grammar coverage for sophisticated wide-coverage constraint-based grammars, with possible implications for robust parsing.

In experiments the annotation method achieves a coverage of 99.8% of the ATIS corpus, with 95.3% non-fragment trees receiving a successful resolution, using a basic HPSG grammar. Using the full LinGO ERG grammar, 68.8% of non-fragment trees are resolved, and lexical type mapping for main verbs and nouns achieves a level of detail close to that of pre-defined lexical items. Also several trees for which the un-annotated string cannot be parsed by the LinGO ERG grammar receive a resolution in the annotation method, and words and subcategorisation frames not in the LinGO ERG lexicon are identified and handled. With the direct use of LinGO ERG grammar in the annotation the resulting lexical and phrasal signs are fully LinGO-compatible and can be easily incorporated back in the grammar.

Acknowledgements

I would like to thank my supervisor, Josef, for great support and inspiration, the ability to instil a positive spirit at all times and for keeping me on track through most of this journey. Thanks also go to the King's Inn, in Dalkey where fair portions of this dissertation were crafted.

I would also like to thank my manager in Microsoft, Patrick Gelle, for supporting my work on this dissertation and giving me the time off needed to complete it.

Finally I would like to thank my family, Clare and Niamh, for putting up with me and being very supportive during all the time and the many nights and weekends it took to complete this work. Niamh very graciously accepted daddy's 'homework'. Although she didn't provide the most constructive feedback, she gave a lot of moral support.

Contents

Abstract	ii
Acknowledgements	iii
Contents	iv
List of Tables	viii
List of Figures	ix
1 Introduction	1
1.1 Treebank annotation – combining data- and theory-driven NLP	1
1.1.1 Data-driven approaches to NLP and treebanks	1
1.1.2 Theory-driven approaches to NLP	2
1.1.3 Combining data- and theory-driven NLP – enriching treebanks	3
1.1.4 Constraint-based grammar-driven treebank annotation	4
1.2 Treebank annotation with HPSG, a preview	4
1.2.1 Motivation	4
1.2.2 The treebank annotation method	5
1.2.3 Results	7
1.3 Goals of this dissertation	8
1.4 Outline of the dissertation	8
1.5 Chapter summary	9
2 Background	10
2.1 Treebank annotation	10
2.1.1 Automatic LFG F-structure annotation	10
2.1.2 Categorical grammar translation	11
2.1.3 LTAG annotation	11
2.2 Head-Driven Phrase Structure Grammar	12
2.2.1 Types and the type hierarchy	14
2.2.2 General principles	14
2.2.3 Grammar rules	16
2.2.4 The LKB parsing system and the LinGO ERG HPSG grammar	18
2.3 Data-driven HPSG approaches	21

2 3 1	SLTG extraction from treebanks and HPSG	21
2 3 2	The LinGO Redwoods treebank	21
2 3 3	BulTreebank, Bulgarian HPSG treebank	22
2 3 4	Annotating the Penn-II treebank with HPSG	22
2 4	Lexical entry creation and extraction with HPSG	23
2 5	Chapter summary	25
3	The HPSG processing system	26
3 1	The chart parser	27
3 2	The grammar system, handling of typed feature structures	27
3 2 1	Typed feature structures	27
3 2 2	Type inference and grammar pre-processing	28
3 2 3	Unification of typed feature structures	30
3 3	Chapter Summary	34
4	Treebank pre-processing	35
4 1	The ATIS treebank	36
4 1 1	Parsing ATIS utterances with LKB and the LinGO ERG Grammar 36	
4 1 2	Comparison between ATIS and LinGO ERG parse trees	37
4 2	Standard pre-processing	41
4 2 1	Head-lexicalisation	41
4 2 2	Complement-marking	43
4 2 3	Binarisation	44
4 2 4	Collapsing of unary productions	46
4 3	HPSG-specific pre-processing	46
4 3 1	N-bar handling	46
4 3 2	NP-compound collapsing	48
4 3 3	Infinitival 'to' marking	48
4 3 4	Grammar compacting	48
4 3 5	Miscellaneous corrections	49
4 4	Chapter Summary	49
5	CFG to HPSG lexical and phrase mapping	50
5 1	Lexical mapping	50
5 1 1	Mapping ATIS part of speech tags to LinGO ERG lexical types 51	

5 1 2	Using argument structure to improve the lexical mapping	53
5 1 3	Using lexical morphological rules	55
5 2	Phrase mapping	55
5 2 1	Phrasal categories and phrase descriptions	56
5 2 2	Rule descriptions	58
5 2 3	Phrase type mapping	59
5 2 4	Creating the complete skeletal phrasal sign	62
5 3	Chapter Summary	63
6	Local tree mapping	64
6 1	Gap handling	64
6 1 1	Handling of empty categories and traces in ATIS and HPSG	65
6 1 2	WH-traces	66
6 1 3	Other traces	68
6 2	Handling co-ordination	68
6 3	Remaining local tree mappings	69
6 3 1	Time expressions	69
6 3 2	Partitives	69
6 3 3	Rule application	69
6 3 4	Raising and control verbs	70
6 4	Chapter Summary	71
7	Constraint resolution	72
7 1	Resolving the local trees – the basic algorithm	72
7 1 1	Lexical resolution	73
7 1 2	Rule application	74
7 2	Modes of operation	76
7 2 1	Mode 1 Minimal grammar with no pre-defined lexicon	77
7 2 2	Mode 2 Full grammar and lexicon use	77
7 2 3	Mode 3 Mixed lexicon lookup	77
7 3	Robust annotation and fall-back strategies	78
7 3 1	Lexical lookup fall-back	78
7 3 2	Rule application fall-back	78
7 3 3	Principled fall-back the robust constraint solver algorithm	79
7 3 4	Considerations for optimised fall-back	81

7 3 5	Partial tree resolution	82
7 4	Chapter Summary	82
8	Experiments and Evaluation	83
8 1	Experiment 1 basic HPSG-based treebank annotation	83
8 1 1	Using a minimal LinGO ERG grammar	83
8 1 2	Results	84
8 2	Experiment 2 Annotating with a wide-coverage grammar	84
8 2 1	Taking advantage of the LinGO ERG grammar and lexicon	85
8 2 2	Optimised used of lexical resources with the LinGO ERG	85
8 2 3	Extracting lexicon entries	86
8 2 4	Results	86
8 3	Experiment 3 Improving grammar and lexicon coverage	90
8 3 1	Results	90
8 4	Chapter Summary	91
9	Conclusions and future work	93
9 1	Conclusions	93
9 1 1	The annotation method	93
9 1 2	Comparison with other work	95
9 1 3	Evaluation	95
9 2	Future work	97
9 2 1	Parsing	98
9 2 2	Lexicon creation	99
9 3	Summary	99
	References	101
	Appendix 1 Lexical entries added to the LinGO ERG	1
	Appendix 2 ATIS treebank category compacting matrix	1
	Appendix 3 ATIS part of speech tags	1
	Appendix 4 Phrase descriptions	1

List of Tables

Table 2 1 LinGO ERG grammar information	19
Table 2 2 LinGO ERG lexical information	20
Table 4 1 Parse results for LinGO ERG on ATIS corpus	36
Table 4 2 Head-rules, from (Magerman, 1995)	42
Table 4 3 Modified head-rules	43
Table 4 4 Collins' complement rules	44
Table 4 5 Additional complement rules	44
Table 4 6 Grammar size reduction due to pre-processing and compacting	49
Table 5 1 Mapping from ATIS POS tags to LinGO ERG head and word types	51
Table 5 2 Verb argument mapping to LinGO ERG lexical types	54
Table 5 3 Extended part-of-speech mapping to LinGO ERG lexical types	54
Table 5 4 Part of speech mapping to LinGO ERG lexical rule	55
Table 5 5 ATIS syntactic tags	56
Table 5 6 Phrase mapping principles	61
Table 8 1 ATIS annotation with minimal LinGO ERG grammar and no lexicon	84
Table 8 2 ATIS annotation with full LinGO ERG grammar and mixed lexical lookup	87

List of Figures

Figure 1 1 Treebank annotation method and system overview	5
Figure 2 1 HPSG analysis of the sentence ‘John always drinks milk’	13
Figure 2 2 Simple type hierarchy	14
Figure 2 3 Head Feature Principle	15
Figure 2 4 Valence principle example for COMPS feature	15
Figure 2 5 Head-complement schema (binary version)	16
Figure 2 6 Head-specifier schema	17
Figure 2 7 Head-subject schema	17
Figure 2 8 Head-adjunct schema	18
Figure 2 9 Head-filler schema	18
Figure 3 1 Typed feature structure represented as Directed Acyclic Graph	27
Figure 3 2 Typed Feature Structure in attribute value matrix (AVM) format	28
Figure 3 3 LKB lexicon entry in compact (unexpanded) format	28
Figure 3 4 LKB lexicon entry in compact (unexpanded) format	29
Figure 3 5 Unification algorithm for typed feature structures	31
Figure 3 6 Typed unification algorithm auxiliary function copy-dg-with-comp-arcs()	33
Figure 4 1 Sample ATIS tree (ATIS tree id @0y0012sx-a-11)	38
Figure 4 2 Sample LinGO ERG parse of the bare string for ATIS tree @0y0012sx-a-11	38
Figure 4 3 ATIS tree (id @8kr033sx-d-9)	39
Figure 4 4 LinGO ERG parse of the bare string for ATIS tree @8kr033sx-d-9	40
Figure 4 5 Binarisation algorithm	45
Figure 4 6 ATIS tree @8kr033sx-d-9, head/complement marked and binarised	45
Figure 4 7 ATIS tree @8kr033sx-d-9 with collapsed unary productions and N-bar level	47
Figure 5 1 HPSG skeletal word sign for ATIS ‘IN’ part of speech tag	52
Figure 5 2 HPSG skeletal word sign for ATIS ‘JJ’ part of speech tag	53
Figure 5 3 Sample phrase descriptions	57
Figure 5 4 Rule description construction method	58
Figure 5 5 Sample rule description for CFG rule $PP \rightarrow H\ IN\ C\ NP$	59

Figure 5 6 HPSG phrase types	59
Figure 5 7 LinGO ERG top level phrase types	60
Figure 5 8 Phrase mapping principle samples in typed feature structure format	62
Figure 6 1 ATIS tree @g07031sx-d-4 illustrating unbounded dependency	65
Figure 6 2 ATIS tree @8kr033sx-d-9 with collapsed unary productions and N- bar level	67
Figure 6 3 ATIS tree @8kr033sx-d-9 illustrating trace removal and gap marking	67
Figure 6 4 Subject gap feature structure schema	67
Figure 6 5 Sample ATIS co-ordinate structure	68
Figure 6 6 Pre-processed co-ordinate structure	68
Figure 6 7 Raising schema	70
Figure 7 1 Basic constraint solver algorithm	73
Figure 7 2 Rule application fall-back	79
Figure 7 3 Robust constraint solver algorithm with fall-back levels	80
Figure 8 1 ATIS tree (@8l8013sx-a-10) annotated and not parsed by LinGO ERG	89
Figure 8 2 ATIS tree (@i06018sx-a-8) with form of 'depart' not in LinGO ERG	91

1 Introduction

This dissertation is a contribution to the effort of bridging two traditions in natural language processing (NLP). It seeks to explore how data- and theory-driven grammars can be related, by using an existing wide-coverage Head-Driven Phrase Structure Grammar (HPSG) (Pollard & Sag, 1994) to annotate a corpus of parsed sentences, or treebank, with typed feature structures. The treebank used is the context-free grammar-based Air Travel Information Service (ATIS) treebank (Hemphill *et al* , 1990, Dahl *et al* , 1994) which is annotated using the Linguistic Grammars Online (LinGO) English Resource Grammar (ERG) (Copestake & Flickinger, 2000, Flickinger, 2000, Flickinger *et al* , 2000), a wide-coverage HPSG grammar.

1.1 Treebank annotation – combining data- and theory-driven NLP

1.1.1 Data-driven approaches to NLP and treebanks

Data-driven and statistical NLP techniques have proven very successful over the past 10-15 years and are now used in many areas of language technology, including speech processing, machine translation and parsing (Manning & Schütze, 1999, Jurafsky & Martin, 2000).

In parsing in particular, statistical approaches have proved to give very good results in wide-coverage and robust parsing. These include Probabilistic Context Free Phrase Structure Grammars (PCFGs) (Charniak, 1993) and treebank-based parsers (Charniak, 1997), with grammars extracted from parse-annotated corpora (treebanks).

While there have been developments to introduce more sophisticated language models than simple CFGs in statistical parsing (e.g. Magerman, 1995, Collins, 1999, Hockenmaier & Steedman, 2002), with few exceptions (Cahill *et al* , 2004, Miyao *et al* , 2004) data-driven approaches do not use grammars as sophisticated as those developed by hand for formal linguistic theories, such as HPSG and Lexical-Functional Grammar (LFG) (Pollard & Sag, 1994, Bouma *et al* 2000, Copestake, 2000, Flickinger, 2000, Flickinger *et al* , 2000, Bresnan, 2001, Riezler *et al* , 2002, Malouf & van Noord, 2004).

This means that the output of data-driven parsers in general cannot compete in terms of depth of analysis with theory-driven grammars and parsers. Deep, theory-driven grammars are needed for several critical applications, including detailed semantic analysis and construction of logical forms and rule-based machine translation.

One way to address the gap between the data- and theory-driven traditions is to create part of speech-tagged and CFG parse-annotated corpora (treebanks) of real-world text, with richer linguistic annotations. These treebanks can then be used to train more sophisticated statistical parsers and to identify weaknesses in and extend the coverage of current theory-based grammars and parsing systems. They can contribute to making these systems more robust and provide a gold standard for parser evaluation.

There are several challenges in this approach. The most basic is that apart from the most straightforward inventory of linguistic descriptions – such as encoding of CFG configurational information, dependency information, or approaches to information packaging such as lexicalisation – there are no real theory-independent encodings of linguistic information. As a consequence, more sophisticated treebank encodings tend to be based on LFG, HPSG, or other linguistic theoretical frameworks (Cahill *et al* , 2002, 2004, Oepen *et al* , 2002b).

Manually creating these more informative and sophisticated treebanks tends to require substantially more effort than creating simpler CFG- or dependency-based treebanks, since more knowledge-intensive resources are needed for the encoding.

1.1.2 Theory-driven approaches to NLP

The other tradition in NLP is that of symbolic, rule-based approaches based on theoretical linguistics. In this tradition, generative grammar (Chomsky, 1957, Chomsky 1965, Radford, 1988) is the foundation for many of the current main theories of formal grammar. Modern lexicalised and constraint-based frameworks such as LFG and HPSG have formed the basis for several recent wide-coverage grammar development projects, offering sophisticated syntactic and semantic analysis (Copestake & Flickinger, 2000, Flickinger *et al* , 2000, Wahlster, 2000, Riezler *et al* , 2002). These grammars are very labour-intensive to develop, often requiring several person-years of design, development and testing effort, involving a high degree of linguistic training.

The term ‘wide-coverage’ is also relative, since even very substantial grammars with thousands of lexical entries and large and complex rule-sets often fail to achieve good coverage on unseen, real world text. This means that many theory-based grammars cannot be used in more industrial-strength applications, except for narrow domains, or in combination with other, more robust fall-back solutions, such as in the Verbmobil project (Wahlster, 2000). One notable exception is the Xerox PARC LFG grammar (Riezler *et al* , 2002), which uses both statistical information and hand-crafted rules and scales to the WSJ section of the Penn-II treebank ¹

1 1 3 Combining data- and theory-driven NLP – enriching treebanks

Since manual development of sophisticated treebanks and grammars is expensive and difficult, it becomes interesting to explore the use of existing resources, such as CFG-based treebanks and modern theory-based grammars, to develop richer treebanks

Apart from manual construction, there are two main approaches to derive rich treebank resources automatically: one is via automatic annotation or transformation, the other via parsing with wide-coverage constraint-based grammars

The first approach enriches existing CFG-based treebanks by transforming them, adding information through annotation, and enhancing them in other ways to contain more detailed linguistic information. This includes automatic dependency marking of heads, complements and adjuncts (Magerman, 1995, Collins, 1999), and annotation with or transformation into other linguistic formalisms, including LFG F-structure annotation (Cahill *et al* , 2002, Frank *et al* , 2003, Cahill *et al* , 2004), categorical grammar derivation (Hockenmaier & Steedman, 2002), tree-adjoining grammar (Xia 1999, Chen & Vijay-Schanker, 2000) and HPSG typed feature structure information (Miyao *et al* , 2004, Nakanishi *et al* , 2004)

The second approach is the development of parsed corpora based on output from wide-coverage grammars and parsers. In this approach sentences are first passed to a parser and the resulting parse(s) then form the basis of the corpus

¹ <http://www.cis.upenn.edu/~treebank>, visited 2nd July 2004

This method underpins some recent HPSG treebank initiatives, such as LinGO Redwoods (Oepen *et al* , 2002b), and the BultreeBank (Simov *et al* , 2002). These treebanks are derived from and are therefore limited to the coverage of the grammar/parsing system used to develop them.

1.1.4 Constraint-based grammar-driven treebank annotation

In this dissertation I use an existing wide-coverage HPSG grammar to annotate treebank trees, combining the two approaches to creating richer treebanks presented in Section 1.1.3 above. I use a section of the ATIS treebank (Hemphill *et al* 1990), annotated with context free grammar categories and some functional tags. In order to annotate the ATIS corpus with rich typed feature structures the LinGO ERG grammar, a rich constraint-based grammar (Copestake & Flickinger, 2000, Flickinger 2000, Flickinger, Copestake & Sag, 2000), is used with the aim of combining the best parts of the two treebank enrichment approaches.

1.2 Treebank annotation with HPSG, a preview

This section gives a brief introduction to the constraint-based grammar-driven treebank annotation method developed in this dissertation.

1.2.1 Motivation

The annotation method is designed to make the best possible use of the information provided both by the LinGO ERG HPSG grammar and the ATIS treebank trees. It is the combination of both of these sources that makes this approach different from other rich treebank creation methods, such as those designed by Cahill *et al* (2002, 2004) and Miyao *et al* (2004).

The advantages of using an existing treebank together with an existing rich constraint-based grammar, compared to using only a rich grammar, are that this approach can

- 1 provide an initial structural analysis (i.e. the treebank trees) guiding the HPSG annotation (see Sections 5.1 and 5.2, and Chapter 6 for CFG-to HPSG and local tree mapping),
- 2 allow the annotation to be more automated since the treebank parse tree guides disambiguation in applying the grammar rules (see Section 7.1 for local tree resolution, treebank structure guiding rule application),

- 3 allow for partial analysis compensating for lack of coverage in the grammar, or ungrammatical or fragmentary corpus content (see Section 7.3 on robust processing)

Advantages compared to only using a treebank, without a grammar, include

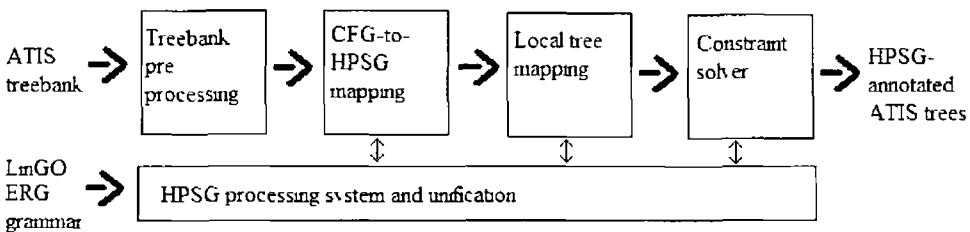
- 1 linguistic information required for the enrichment is already contained in the HPSG grammar and does not need to be encoded in the annotation method, as in (Cahill *et al* , 2002, 2004),
- 2 the HPSG grammar provides constraints on grammatical structure and can help to correct treebank tagging mistakes and adjust the often flat, over-simplified treebank analysis, as will be shown in Chapters 4 and 5

In cases where the treebank and grammar parse do not fully agree, the annotation proceeds with guidance from the treebank as much as possible, with fall-back strategies if no match is possible

1.2.2 The treebank annotation method

The treebank annotation method consists of four components: pre-processing, CFG-to-HPSG mapping, local tree mapping and constraint resolution. These are outlined below and described further in Chapters 4 to 7. Figure 1.1 shows the components and how they fit together in an overall system, along with the underlying HPSG processing system described in Chapter 3.

Figure 1.1 Treebank annotation method and system overview



1 2 2 1 Treebank pre-processing

For the annotation to work well it is important that the treebank trees correspond reasonably closely to the basic phrase structure configurations assumed by the HPSG grammar. This requires some pre-processing of the treebank, to make the trees less flat and to explicitly encode a certain amount of dependency information, such as the distinction between head, argument and adjunct roles. HPSG phrase structures explicitly encode and critically rely on head, complement and adjunct marking of phrase daughters. The LinGO ERG HPSG grammar used in this dissertation employs strictly binary branching rules. Given the format of the ATIS trees,² the automated pre-processing consists of the following steps, described in more detail in Chapter 4

- 1) addition of head, complement and adjunct dependency information,
- 2) binarisation of the local trees,
- 3) HPSG-specific treebank pre-processing, including the introduction of an N-bar level into the ATIS trees to ensure that NP pre- and post-modifier structure matches HPSG, and collapsing of unary productions

1 2 2 2 CFG to HPSG mapping

Basic mapping principles are established between the ATIS local trees, phrasal CFG categories and part of speech tags, and the HPSG grammar, to identify how phrasal and lexical HPSG types should be associated with the local trees. This mapping makes use of lexical, local tree (CFG rule), and dependency information, including argument structure derived from the treebank. The mapping principles are established manually, as described in Chapter 5

1 2 2 3 Local tree mapping

In the “local tree mapping”, the CFG-to-HPSG mapping is applied automatically to each local tree. At this stage traces in the treebank are processed for long-distance dependency mark-up

² ATIS trees tend to be flat and sentences short. Further characteristics of ATIS trees are described in Section 4.1.2

This is performed through marking of HPSG SLASH features on the head-word of the phrase in which the trace occurs and introduction of a head-filler rule in the local tree of the trace landing site. Specific marking is also introduced for co-ordination, raising and control verbs, and other constructions. Local tree mapping is described in Chapter 6.

1 2 2 4 Constraint resolution

After the local tree mapping, constraint resolution is applied to create a single HPSG phrasal sign corresponding to the whole tree. The “mapped typed feature structure” associated with each local tree is used to guide the selection and application of HPSG grammar rules, to create one or more “resolved typed feature structures” covering the complete tree. Constraint resolution is described in Chapter 7.

1 2 3 Results

In experiments, the annotation method achieves a high degree of coverage of the ATIS corpus, with a minimal grammar. Coverage is 99.8%, with 95.3% non-fragment trees receiving a resolution. This resolution rate is significantly higher than the LKB parse success rate for the bare strings from the ATIS corpus, which is 79.8%.

Using the full LinGO ERG grammar, resolution coverage for non-fragment trees is 68.8%, lower than that of the minimal grammar and the LKB parse success rate due to the preliminary nature of the mapping principles and also due to lack of grammar coverage in not licensing the same structure as the corpus (see Section 8.2.4). Lexical type mapping for main verbs and nouns achieves a level close to that of pre-defined lexical items. Also, several trees that cannot be parsed by the LinGO ERG grammar receive a resolution in the annotation method. Increased lexical coverage is also demonstrated by identification of new subcategorisation frames for words already in the LinGO ERG lexicon. While work remains to be done to increase the treebank coverage when annotating with the full LinGO ERG grammar, I consider preliminary results to be very promising.

The LinGO ERG grammar has also assisted both in the development of the annotation method and in constraining the grammatical structure during the annotation. With the direct use of the LinGO ERG grammar in the annotation, the resulting lexical and phrasal signs are fully LinGO compatible, and can be easily incorporated back in the grammar. The experiments are presented and assessed in Chapters 8 and 9.

1.3 Goals of this dissertation

The goals of this dissertation are to

- investigate treebank annotation with HPSG,
- explore the feasibility of combining the use of existing treebanks and wide-coverage HPSG grammars in creating richer treebanks, to develop a grammar-based treebank annotation method for HPSG,
- address issues in mapping context-free grammars to HPSG grammars,
- evaluate the annotation method

1.4 Outline of the dissertation

Chapter 2 describes related work in the areas of treebank annotation and HPSG that have influenced the work presented in this dissertation. Chapter 2 also introduces the LinGO ERG grammar, the wide-coverage HPSG grammar used in this dissertation. In Chapter 3 the HPSG typed feature structure processing and parsing system developed and implemented in C# for the annotation method is described and motivated. Chapters 4 to 7 present the four main components of the annotation method. Chapter 4 introduces the ATIS treebank and covers the treebank pre-processing in some detail. Chapter 5 introduces the basic CFG-to-HPSG mapping principles, on category and phrase level. Local tree mapping is presented in Chapter 6 where a number of additional specific mappings for traces and phenomena are applied. In Chapter 7, full tree constraint resolution with HPSG grammar rule application is described, including fall-back strategies for cases where the local trees do not fully match. Chapter 8 reports on experiments with applying the annotation method to the ATIS treebank. It also contains an assessment of the effectiveness of the method and evaluation against the goals of the dissertation. In Chapter 9 conclusions are drawn and future work is discussed.

1 5 Chapter summary

This chapter has introduced constraint-based grammar-driven treebank annotation with HPSG typed feature structures, the topic of this dissertation, and outlined the background and different approaches to treebank enrichment. It has also given a preview of the annotation method, with some motivation. Finally the goals of the dissertation have been set out.

2 Background

In this chapter I review work related to this dissertation with regard to treebanks, treebank annotation, Head-Driven Phrase Structure Grammar (HPSG), and data-driven HPSG approaches. I also give an overview of the LinGO ERG grammar, the wide-coverage HPSG grammar used in the present dissertation.

2.1 Treebank annotation

As outlined in Chapter 1, there have been several approaches to enriching existing treebanks by transformation or adding annotations.

2.1.1 Automatic LFG F-structure annotation

In automatic LFG F-structure annotation approaches, (Cahill *et al.*, 2002, 2004, O'Donovan *et al.*, 2004) have developed a method based on an annotation algorithm, that annotates nodes in Penn-II treebank trees with f-structure constraints. From this annotated treebank an f-structure can be computed by a constraint solver. The annotation algorithm is robust enough to handle the Penn-II treebank, with about 50,000 sentences and 19,000 CFG rule types. The algorithm exploits categorial, configurational, local head and Penn-II functional annotations, and traces and co-ordination information. It has four main components, left/right context annotation principles, co-ordination principles, catch all principles and trace handling. The L/R context principles depend on partitioning daughters of a local sub-tree into left and right context of a head, which is computed based largely on (Magerman, 1994, Collins, 1999). An annotation matrix is constructed for rule LHS categories, assigning annotations to constituents in left and right context relative to a local head. Co-ordination principles first identify the head, and then use heuristics for annotation. Traces and co-indexation in treebank trees are translated into corresponding re-entrancies in f-structures, representing long-distance dependencies. The final component of catch-all principles uses defaults and functional tag information from the original Penn-II treebank annotation. Used together with the constraint solver, the annotation algorithm associates 99.82% of Penn trees with a single covering and connected f-structure. The automatically induced grammar achieves 80.24% f-score for f-structures, parsing section 23 of the WSJ part of the Penn-II treebank (Cahill *et al.*, 2004).

The method has also been used to extract subcategorisation frames from the Penn-II Treebank (O'Donovan *et al* , 2004), with a full evaluation against the COMLEX resource (MacLeod *et al* , 1994). Both traditional CFG category-based subcategorisation frames and syntactic function-based frames (LFG semantic forms) are extracted, without using pre-defined frames. Probabilities conditional on the lemma are associated with the frames, and long-distance dependency information is also included. In evaluation, with prepositional phrases excluded, precision is 79.0%, recall 59.6% and F-score 68.0%.

2.1.2 Categorical grammar translation

Working with Combinatory Categorical Grammar (CCG), (Hockenmaier & Steedman, 2002) translate the Penn-II treebank into a corpus of CCG derivations. The translation algorithm first identifies heads, complements and adjuncts, based on (Collins, 1999), as used in the LFG-annotation described above. Then the flat trees are transformed into binary trees by inserting dummy nodes so that all children to the left of a head branch off in a right-branching tree, and then all children to the right of the head branch off in a left-branching tree.³ After this, CCG categories are assigned to the binary tree.

2.1.3 LTAG annotation

In Lexicalised Tree Adjoining Grammars (LTAG), Chen & Vijay-Shanker (2000) extract LTAGs for use in statistical parsing from the Penn-II treebank. Initially head-lexicalisation is performed based on Magerman (1995), to allow determination of trunks for the LTAG trees. For this, complements and adjuncts also need to be identified, based on Collins (1997). Chen & Vijay-Shanker also experiment with two different strategies for complement and adjunct identification. The first uses labels and semantic tags (Penn-II has some semantic tag labels, such as -TMP, -LOC, etc.), of a node and its parents, based on Xia (1999), while the second strategy employs labels and tags of a node, its head sibling, and distance between the siblings.

Xia extracts LTAG trees from the Penn-II treebank in three steps: first heads, complements and adjuncts are identified, in the same way as for Chen & Vijay-Shanker (2000), then elementary LTAG trees are extracted, and finally invalid LTAG trees arising from annotation errors are filtered out.⁴

³ This is the same method used for this dissertation, described in more detail in Section 4.2.3.

⁴ The filter checks dependency relationships, such as blocking ADVP from modifying PP from the right.

2.2 Head-Driven Phrase Structure Grammar

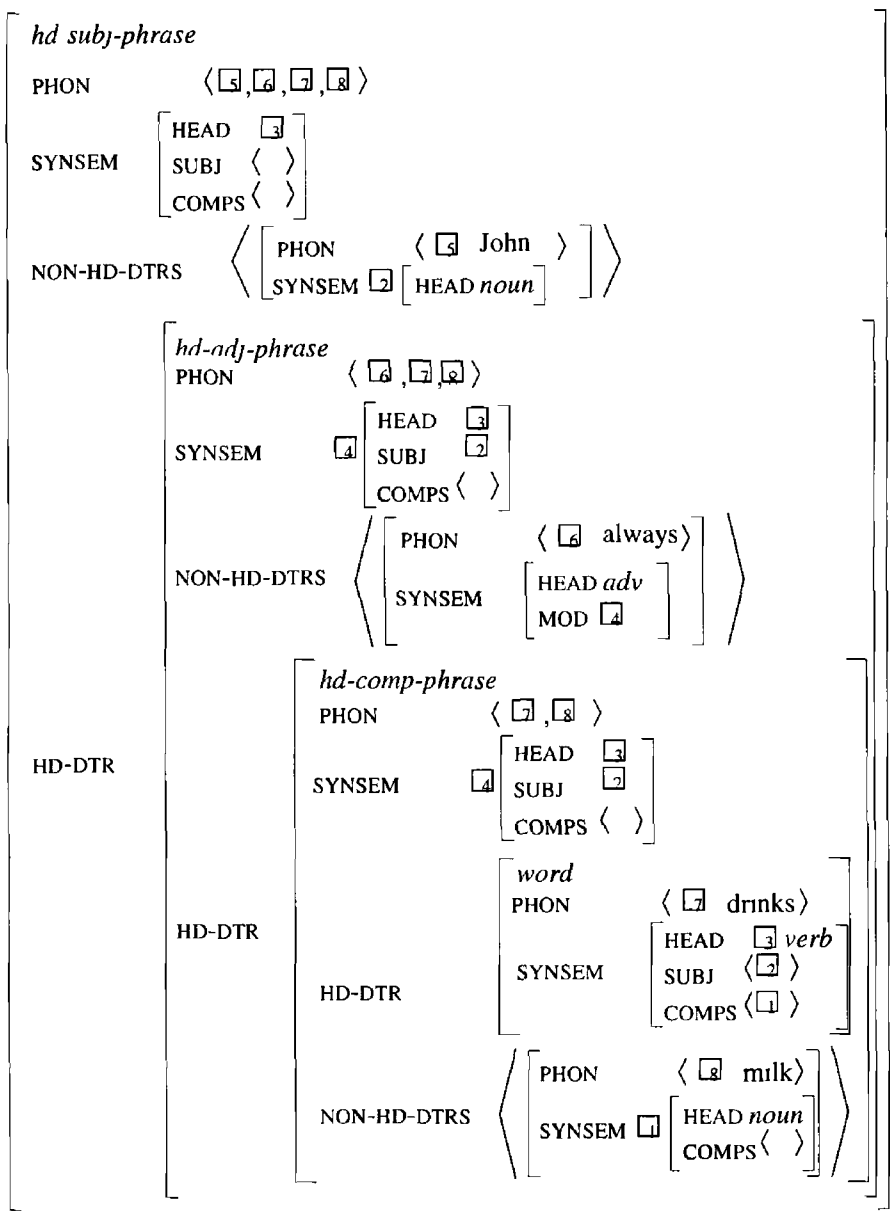
Head-Driven Phrase Structure Grammar (HPSG) is a lexicalised constraint-based formal grammar theory (Pollard & Sag, 1994, Ginzburg & Sag, 2000, Sag *et al.*, 2003). The basic linguistic unit is a sign, which can be lexical or phrasal. Signs are represented with typed feature structures, with types organised in a multiple inheritance type hierarchy (Carpenter, 1992). Typing of feature structures allows the grammar to be specified compactly with an elegant and succinct expression of generalisations, by defining which features and values are appropriate for a particular type, and by organising types into an inheritance hierarchy. As a general mechanism for expressing linguistic information, the typed feature structure system also supports a close integration of semantics directly into the grammar framework.

In HPSG the grammar is made up of the type hierarchy, the lexicon, general principles, and rule schemata or grammar rules. The type hierarchy defines the structure of lexical and phrasal signs, supporting a highly structured lexicon with detailed lexical entries.

General principles governing sharing of head and other features, management of subcategorisation information from the lexicon, together with rule schemata instantiating phrasal types, licence the construction of phrasal signs. Lexical rules are also used extensively.

As an example, Figure 2.1 presents an HPSG analysis of 'John always drinks milk', with some features not immediately relevant to syntax omitted. The structure obeys the principles of HPSG, including the Head Feature Principle which requires a phrase to share head features with its head daughter, as shown by the shared co-index tag, [3], and the Valence principle, which requires that the elements of the valence features (SUBJ, COMPS) of the head daughter is the concatenation of the valence feature of the phrase, with the list of SYNSEM values of the non-head daughter(s), as exemplified by the co-indices [1], [2].

Figure 2 1 HPSG analysis of the sentence ‘John always drinks milk’

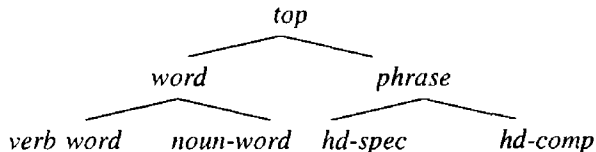


In addition, the sub-structures typed as *hd-subj-phrase*, *hd-adj-phrase* and *hd-comp-phrase* are licensed by the corresponding rule schemas, described below. As indicated, structure sharing through co-indexing is used extensively and unification, or more generally constraint resolution is the primary structure-building operation for constructing phrases.

2 2 1 Types and the type hierarchy

In an HPSG grammar types are a distinct class of objects arranged into a type hierarchy. The hierarchy is ordered by subsumption from a most general type, with multiple inheritance. Types can be simple, represented by an atomic symbol like 'sg', replacing the atomic values used in untyped feature structures. Complex types specify appropriateness conditions, such as features appropriate for the type, restrictions on the values of those features, and equality constraints (re-entrancy) between feature values. The type hierarchy is used in parsing for determining type compatibility in unification, and for type inference. Figure 2 2 gives an example of a simple type hierarchy for words and phrases with 'top' as the most general type. Types also need to be well-formed, or well-typed, for proper use in grammar processing. Type inference or sort resolution, with respect to the type hierarchy, is used to create a well-formed typed feature structure. This is discussed further in Section 3 2 2.

Figure 2 2 Simple type hierarchy



2 2 2 General principles

General principles are used in HPSG to specify the well-formedness of phrases. Several principles are required for the full specification of HPSG, as defined in (Pollard & Sag, 1994). Below I introduce the Head Feature Principle and Valence Principle, the two main principles for phrase construction. Other important principles include the NonLocal Feature Principle, which controls the passing of SLASH features from daughters in a phrase, and the Semantics Principles which handles semantic construction.

2 2 2 1 The Head Feature Principle

The Head Feature Principle ensures that headed phrases are projections of their head daughters. It states that the HEAD value of any headed phrase is structure-shared (co-indexed) with the HEAD value of the head daughter. This is illustrated with a feature structure representation or schema of the principle, in Figure 2 3, which shows the co-indexing of the HEAD feature.

Figure 2 3 Head Feature Principle

[SYNSEM [LOCAL [CAT [HEAD \square]]]
HEAD DTR [SYNSEM [LOCAL [CAT [CAT [HEAD \square]]]]]

2 2 2 2 The Valence Principle

The Valence Principle controls the behaviour of the so-called valence features in well-formed HPSG phrases. Valence features are the selectional features employed by heads to select their sister non-heads in a phrase. Valence features governed by the Valence Principle include SUBJ, COMPS, and SPR, used in head-subject, head-complement and head-specifier phrases, as discussed in Section 2 2 3 below. The Valence Principle states that ‘In a headed phrase, for each valence feature F, the F value of the head daughter is the concatenation of the phrase’s F value with the list of SYNSEM values of the [F daughter/s] value’ (Pollard & Sag, 1994: 392). In effect the principle ensures the ‘cancelling off’ of non-head daughters against the appropriate valence feature of the head-daughter, with the remaining elements of the list-valued valence feature being passed on to the phrase. Figure 2 4 shows an instantiation of the valence principle for the COMPS feature, with the head-daughter COMPS value list being the concatenation of the phrase COMPS value and the COMP-DTR SYNSEM value.

Figure 2 4 Valence principle example for COMPS feature

[SYNSEM [LOCAL [CAT [VAL [COMPS \square]]]
HEAD DTR [SYNSEM [LOCAL [CAT [VAL [COMPS (\square | \square)]]]]]
COMP DTR [SYNSEM \square]]]

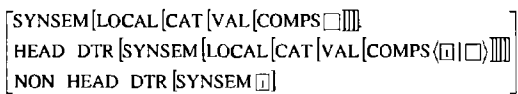
2 2 3 Grammar rules

In HPSG, phrases are constructed based on rule schemas which interact with the general principles for feature propagation to license the creation of phrasal signs. Given the richness of configurational information encoded in the lexicon, the rule schemata can be highly abstract, leading to a much smaller number of rules in an HPSG compared to a comparable CFG. The main, standard HPSG rule schemata combine heads with different kinds of non-head sister constituents – such as complements, specifiers, subjects, and adjuncts – and handle long-distance dependencies. These are described below. In addition, any non-trivial HPSG grammar also needs to include additional rules to cover co-ordination, compounds, interrogatives, and other specific phrase types. These types of rules will be discussed in more detail in the description of the annotation method in Chapter 6.

2 2 3 1 The head-complement schema

The head-complement schema licenses the combination of a lexical head with the elements it typically co-occurs with. It covers verbs and their required complements, and also other lexical categories taking complements. In its general formulation, the head-complement rule combines a head-daughter with all its complements at the same time. However, in many HPSG implementations, the rule is binary, applied recursively to consume one element of the complement list at a time. The binary format is used to simplify the implementation. Figure 2.5 shows how the COMPS feature of the head daughter is combined with the mother and non-head daughter through co-indexation.

Figure 2.5 Head-complement schema (binary version)



2 2 3 2 The head-specifier schema

Specifiers in HPSG are elements that nouns, adjectives, prepositions and determiners select for. In English specifiers precede the head, and complements follow it, but the main motivation for separating specifiers is that they have different syntactic and semantic properties. Specifiers also co-select their head through the SPEC feature.

Figure 2 6 shows (a simplified version of) the head-specifier schema used by the LinGO ERG grammar, which restricts the non-head co-selection to only the HEAD-DTR HEAD feature (through the co-index tag $\boxed{3}$), to avoid a cyclic structure. Otherwise the schema operates as the head-complement schema, but for the SPR feature.

Figure 2 6 Head-specifier schema

$$\left[\begin{array}{l} \text{SYNSEM} [\text{LOCAL} [\text{CAT} [\text{VAL} [\text{SPR} \boxed{2}]]]] \\ \text{HEAD DTR} [\text{SYNSEM} [\text{LOCAL} [\text{CAT} [\text{HEAD} \boxed{1} \text{ VAL} [\text{SPR} \langle \boxed{1} | \boxed{2} \rangle]]]] \\ \text{NON HEAD DTR} [\text{SYNSEM} \boxed{1} [\text{LOCAL} [\text{CAT} [\text{SPEC LOCAL} [\text{CAT} [\text{HEAD} \boxed{3}]]]]] \end{array} \right]$$

2 2 3 3 The head-subject schema

Similar to the head-specifier schema, the head-subject schema combines a head daughter with the subject it selects for. The distinction between the head-specifier and the head-subject schemas follows (Pollard & Sag, 1994: 359-361). They motivate the difference between specifiers and subjects by (i) pre-theoretical notions of different semantic contribution,⁵ and (ii) by difficulties in identifying subjects and specifiers as instances of the same grammatical relation – as they can both appear with the same head (Pollard & Sag, 1994: 360).⁶ Figure 2 7 illustrates the head-subject schema which combines a head-daughter with a single subject.

Figure 2 7 Head-subject schema

$$\left[\begin{array}{l} \text{SYNSEM} [\text{LOCAL} [\text{CAT} [\text{VAL} [\text{SUBJ} \langle \rangle]]]] \\ \text{HEAD DTR} [\text{SYNSEM} [\text{LOCAL} [\text{CAT} [\text{VAL} [\text{SUBJ} \langle \boxed{1} \rangle]]]] \\ \text{NON HEAD DTR} [\text{SYNSEM} \boxed{1}] \end{array} \right]$$

2 2 3 4 The head-adjunct schema

Adjunct (modifier) daughters are non-subcategorised for words or phrases such as adjectives, adverbs or prepositional phrases, which combine with a head daughter. The adjunct daughter selects the head through its MOD feature. Figure 2 8 shows the head-adjunct schema combining an adjunct or modifier daughter with the head-daughter.

⁵ Specifiers differ from subjects in lacking the potential to be semantic arguments (Pollard & Sag, 1994: 359).

⁶ Not all flavours of HPSG maintain the distinction between specifiers and subjects and in this case would have either a head subject or a head-specifier schema.

Figure 2 8 Head-adjunct schema

$$\left[\begin{array}{l} \text{HEAD DTR [SYNSEM []]} \\ \text{NON HEAD DTR [SYNSEM [LOCAL [CAT [MOD []]]]} \end{array} \right]$$

2 2 3 5 The head-filler schema

The head-filler schema is used for long-distance dependencies, to terminate the percolation of gap indices and connect them with their antecedents or landing-sites. In HPSG long-distance dependencies and gaps are not represented with traces but with empty categories co-indexed with their antecedents. Gaps are introduced and handled through the SLASH feature, which also interacts with the non-local feature principle, to pass up slash values through the tree. Elements on the slash list are then cancelled off through the head-filler schema, which identifies a slash element of the head daughter with the non-head daughter in the phrase. Figure 2 9 shows a simplified version of the head-filler schema which allows only one value on the SLASH list (which is also what the LinGO ERG grammar supports). Handling of long distance dependencies is further discussed in Section 6 1 ⁷

Figure 2 9 Head-filler schema

$$\left[\begin{array}{l} \text{SYNSEM [NON LOCAL [SLASH ()]} \\ \text{HEAD DTR [SYNSEM [NON LOCAL [SLASH ([])]} \\ \text{NON HEAD DTR [SYNSEM []]} \end{array} \right]$$

2 2 4 The LKB parsing system and the LinGO ERG HPSG grammar

The annotation method in this dissertation is based on the use of the Linguistic Grammars Online (LinGO) English Resource Grammar (ERG) (Copestake & Flickinger, 2000, Flickinger, 2000, Flickinger *et al* , 2000). The typed feature structure parsing system developed in C# in the present dissertation, described in Chapter 3, is closely modelled on the Linguistic Knowledge Building (LKB) system (Copestake & Flickinger, 2000, Copestake, 2002), which is also part of the LinGO project at CSLI in Stanford University ⁸

⁷ In addition to SLASH HPSG also make use of the non local features REL and QUE to distinguish relatives and interrogatives

⁸ <http://lingo.stanford.edu> last accessed 2nd July 2004

2 2 4 1 The LKB system

The LKB grammar development environment is an HPSG parsing system designed to facilitate development of typed feature structure grammars. The LKB system supports efficient processing of large typed feature structure grammars, and is freely available for download from the web⁹. It also supports Minimal Recursion Semantics (MRS) (Copestake *et al*, 1999) – a framework for underspecified semantics – default unification (Lascarides & Copestake, 1999) and generation, and incorporates a number of efficient algorithms for processing and parsing with typed feature structure grammars (Kiefer *et al*, 2000, Malouf *et al*, 2000, Oepen & Carroll, 2000a,b, Copestake 2002, Oepen *et al*, 2002a).

2 2 4 2 The LinGO English Resource Grammar (ERG)

The LinGO English Resource Grammar (ERG) is a broad-coverage grammar of English in the HPSG framework (Copestake & Flickinger, 2000, Flickinger *et al*, 2000). The central part of the grammar is the type hierarchy, which is used to structure the lexicon and the lexical and grammar rules.

Table 2.1 presents some information on the grammar (November 2002 release)¹⁰. The number of grammar rules is comparatively small for a wide-coverage grammar, reflecting the highly lexicalised nature of HPSG grammars. The substantial number of lexical rules are used for morphological, inflectional and valence-altering operations on the lexical items, and help account for the compactness of the core lexicon.

Table 2.1 LinGO ERG grammar information

Item	#
Types	10 785
Grammar rules	67
Lexical rules	33 (13 morphological)
Lexical types	7 990
Lexical entry types	601
Lexical items	9961
Expanded lexicon	About 24 000 full forms

⁹ <http://www.csl.stanford.edu/~aac/lkb.html> accessed 2nd July 2004

¹⁰ This is the main LKB and LinGO ERG grammar download pointed to by the LKB download page

There are 601 lexical entry or lemma types, that is, there are 601 distinct lexical signs used to representing the 9 961 lexical entries ¹¹ The lexical signs break down into 24 major parts of speech, with 139 main verb forms and 124 noun forms, as shown in Table 2 2

Table 2 2 LinGO ERG lexical information

Major part of speech	Lexical entry types	Lexical items
Main verbs (v)	139	2,197
'be', 'have' (vc)	56	119
Auxiliary verbs (va)	45	104
Nouns (n)	124	4,228 (872 proper nouns)
Adjectives (Adj)	54	1999
Adverbs (Adv)	39	955
Determiners (det)	33	62
Prepositions (p)	33	174
Complementizers (Comp)	16	29
Conjunctions (Conj)	25	26
Others (14 parts of speech)	37	68
Total	601	9961

The LinGO ERG was used in the Verbmobil spoken language machine translation project (Wahlster, 2000), and thus has good coverage for constructions frequently used in the domain of that project, such as meeting scheduling conversations and travel bookings. It also has extensive integrated semantics in the form of MRS (Copestake *et al.*, 1999)

The grammar's theoretical assumptions are consistent with Pollard & Sag (1994), particularly their revised assumptions in Chapter 9, it also uses the multiple inheritance type hierarchy to represent properties of phrases, following Sag (1997). Grammatical rules are strictly binary or unary, for processing efficiency purposes, which leads to recursive application of rules like the head-complement rule to consume multiple complements.

¹¹ Apart from the orthographic representation of the lexical entry, the other information in each lexical entry that tends to be unique is the semantic relation it represents. Note that lexical entries do not fully correspond to lemmas in the LinGO ERG grammar, since many surface forms of the same word lemma are encoded directly in the lexicon, leaving mainly productive inflectional morphology to the lexical inflectional rules.

The grammar has an extensive treatment of phrase types, including questions, consistent with Ginzburg & Sag (2000),¹² and also has some domain-specific particulars, such as an extensive treatment of date and time expressions. This leads to a significantly higher number of id-schemata or rules, 67 in total, compared to the 6 rules or id-schemas in the textbook grammar of (Pollard & Sag, 1994)

2.3 Data-driven HPSG approaches

2.3.1 SLTG extraction from treebanks and HPSG

Neumann (2003) presents a uniform method for extraction of stochastic lexicalised tree grammars (SLTGs) from treebanks, and from HPSG grammars. Grammar extraction is performed by head-driven decomposition, cutting off all non-head subtrees to divide each tree into smaller lexically anchored subtrees. For treebanks this requires a markup indicating the head, while for HPSG headedness is marked explicitly in the phrase structure. When used with HPSG, the grammar is first used to parse sentences from a corpus, and the parses are then used to extract the SLTG. Nodes in the decomposed tree are annotated with category labels, such as NP, defined as equivalence classes for phrasal signs.

2.3.2 The LinGO Redwoods treebank

LinGO Redwoods (Oepen *et al* , 2002b) is a project to develop an HPSG treebank, based on parses from the LinGO ERG. The goals are to produce a richer and more dynamic treebank than others in existence, to provide sophisticated tools for annotation and parse disambiguation, and also to allow the treebank to respond (restructure itself) dynamically to updates in the underlying grammar.

The main difference from other treebanks is the very close coupling between the treebank, the LinGO grammar and the LKB parsing system, which allows for the treebank to be easily updated to reflect grammar changes.

¹² Although the ERG uses a non default version of the interrogative constructions as opposed to the definition in Ginzburg & Sag (2000)

The treebanking environment presents annotators with the full set of analyses provided by the grammar, allowing them to quickly navigate through the parse forest to identify the preferred analysis. All disambiguation decisions are recorded for later availability and re-use in applying disambiguation to parses from a later version of the grammar.

Initial treebank construction has been undertaken by parsing 10,000 utterances from the Verbmobil corpus (Oepen *et al* , 2002b). Initial work has been done on statistical parse disambiguation using log-linear models to rank parses¹³. Results indicate simple statistical models can achieve high statistical accuracy, using PCFGs with grandparent annotation, and trigram probabilities of preterminals. This gives a parse selection accuracy of 74%, and a tag selection accuracy (accuracy of tag sequence on highest-scoring parse) of 98% (Oepen *et al* , 2002b). Disambiguation is performed using a tree comparison tool, and automated disambiguation techniques are explored.

2.3.3 BulTreebank, Bulgarian HPSG treebank

The BulTreeBank¹⁴ is another HPSG treebank project (Simov *et al* , 2002), intended to provide a fine-grained parsed corpus for Bulgarian, based on HPSG. This treebank is still under development, and results reported so far focus on the tool-set. Specially designed tools are used to construct the treebank in a two-step approach, where shallow parsing is used first to provide partial analyses before ‘attachment resolution’ annotation. Regular expression chunk grammars are used to identify non-recursive phrases, and the tagset used is mapped into an HPSG-compatible format (Osenova & Simov, 2003). An HPSG grammar with Bulgarian-specific constructions is used in the second stage of annotation, which calculates attachment possibilities and presents them to the annotator.

2.3.4 Annotating the Penn-II treebank with HPSG

The HPSG-based treebank annotation and enrichment effort closest in nature and spirit to the LFG annotation described in Section 2.1.1, and to the goals of this dissertation, is that of (Miyao *et al* , 2004), who use a basic, coarse-grained HPSG grammar to annotate the Penn-II treebank.

¹³ For a reference on statistical methods see (Berger *et al* 1996).

¹⁴ <http://www.bulreebank.org>, accessed 2nd July 2004.

Miyao *et al* use only the main HPSG rule schemas discussed in Section 2.2.3, with the addition of a head-relative and a filler-insertion schema¹⁵. Treebank pre-processing for head-lexicalisation and binarisation is done similarly to that in LFG annotation and LTAG extraction approaches, and also that used in this dissertation (see Chapter 4). In addition, heuristic rules are used to take account of the trace-marking annotation of the treebank, and to handle subject raising, co-ordination and other specific constructions. Miyao *et al* use the HPSG-annotated corpus to acquire an HPSG grammar from the Penn treebank sections 0-21. Lexical entries were successfully extracted for about 90% of the sentences,¹⁶ with a total of 2,339 lexical categories acquired (1,203 for verbs), for 41,030 words. This is a significantly lower number of lexical categories than Xia, (1999),¹⁷ indicating a higher level of abstraction achieved by the HPSG grammar. Strict sentence coverage, where the grammar creates a correct derivation for the sentence, against Section 22 of the Penn Treebank is 75.5%. Weak coverage, producing a correct unlabelled dependency structure, is 92.9%. This shows the robustness of the generated grammar.¹⁸ In further related work (Nakashimi *et al*, 2004) extend this approach of lexicon generation by using inverse lexical rules to capture lexical generalisation, and thus cover words not seen in the base corpus. Lexical coverage for verbs was measured against Section 23 of the Penn Treebank, and improved from 91.88% to 96.20%.

2.4 Lexical entry creation and extraction with HPSG

Several research efforts have used HPSG for handling unknown words and the creation of lexicon information. They exploit the general property of constraint-based grammar-frameworks where unification can ‘fill-in’ information in a lexical entry during parsing based on information provided from other lexical entries and the grammar itself.

¹⁵ The head relative schema is used to avoid use of a null constituent for relative clauses. The filler insertion schema is a unary rule used where an inserted clause introduces a slash which is filled by the entire sentence.

¹⁶ Extraction was successful for 36,182 sentences and failed for 3,416 sentences. The main reason for this was shortage of heuristic rules, for pre-processing and supporting specific constructions.

¹⁷ Miyao *et al* do not state a specific number but the most likely comparison is with Xiao’s extraction of 3014 elementary tree templates, the equivalent of lexical categories, in one of her experiments.

¹⁸ The grammar for this assessment included a treatment for unknown words similar to (Hockenmaier & Steedman 2002).

(Horiguchi *et al.*, 1995) acquired content words for Japanese using an HPSG-based parser, with the use of a manually-coded lexicon for function words. Underspecified lexical entry templates are given to content words, specifying only the major parts of speech. As a result of the unification-based parsing, the system fills in missing information in the entry for each unknown content word. Suffixes on verbs are used to constrain the lexical entries. HPSG is seen as suitable for this form of lexical acquisition due to its underspecified rule component, which means syntactic under-generation is less of a problem compared to systems relying on large numbers of rules¹⁹

Barg & Walther (1998) also exploit the ability of an HPSG parser for updating properties of unknown words, creating a system that can learn and update lexical information. They introduce a model of revisable information in lexical signs – which can be specialised or generalised – following the grammar type hierarchy, based on parse information from successive parses and evidence from multiple input strings.

Fouvry (2003b) uses the LinGO ERG grammar to process unknown lexical entries, taking information from part of speech taggers and morphological analysers into account. Underspecified or generic lexical entries for the unknown words are used as a starting point for processing, with the parse filling in information. Unfilling (Götz, 1994) is used to remove information from the created lexical entries that can be inferred from the type system.²⁰ Problems are encountered as sentence length grows due to the number of rule applications spawned by unknown entries. This leads to a high level of ambiguity, and the need to constrain the initial underspecified entries. The LinGO ERG grammar lexical types (see Section 2.2.4.2) are considered as candidates to constrain the generic lexical entries, and restrict the search space for the grammar rules. The disadvantage is that the number of lexical entry types is quite high, as shown in Table 2.2 above. Even the use of part of speech tag and morphological information is not enough to constrain the number of possible lexical types enough to significantly reduce the search space for grammar rule application.

¹⁹ Lexical entry templates are also used by Yutaka *et al.* (1998) for a wide coverage HPSG grammar with only function words in the lexicon.

²⁰ A feature and its value can be removed from a feature structure without information loss if, (i) the feature is not re-entrant, and (ii) its value is of the maximally generic appropriate type for the feature.

Miyao *et al* (2004) and Nakashini *et al* (2004) investigate extraction of HPSG lexical entries from annotated treebanks, as part of their treebank annotation work, described in Section 2.3 above

2.5 Chapter summary

This chapter has presented related work in treebank annotation, in related grammatical frameworks such as LFG and LTAG. It has introduced HPSG – the grammar framework used in this dissertation – and its applications for grammar development, treebank creation and annotation, and lexicon entry creation. The LinGO ERG grammar – the HPSG grammar used for treebank annotation in this dissertation – has also been introduced and described.

I will make use of the LinGO ERG grammar in my annotation method, which is similar to but extends on the work of (Cahill *et al* 2002, Cahill *et al* , 2004) and (Miyao *et al* , 2004) in particular. The HPSG processing system similar to the LKB that I develop for this dissertation is described and motivated in Chapter 3. The use of the LinGO ERG grammar in the annotation, which is the core part of my method, is described in Chapters 5 to 7.

3 The HPSG processing system

This chapter describes the HPSG typed feature structure processing and parsing system developed for this dissertation. The grammar and type hierarchy component is modelled closely on the LKB system (Copestake & Flickinger, 2000, Copestake, 2002), and is capable of loading and processing LKB-compatible HPSG grammars such as the LinGO ERG (Copestake & Flickinger, 2000, Flickinger 2000, Flickinger *et al* , 2000)²¹ The parser is a simple bottom-up chart parser for typed feature structure grammars. The parser has been developed so that it can be used in two flavours: first as a CFG chart-parser for experiments with grammar compacting (following Krotov *et al* , 1998, Hepple & van Genabith, 2000) on the grammar extracted from the ATIS treebank, and secondly as an HPSG parser for strings, using the HPSG grammar system²²

The full typed feature structure processing, parsing and annotation system developed in this dissertation is implemented in the general purpose programming language C#, a Java and C++-style programming language which facilitates rapid prototyping and modular development while also offering high performance. The motivation for developing this system from scratch rather than using an existing HPSG typed feature structure processing system like the LKB is the need to be able to integrate the grammar and type system components very tightly with the treebank annotation method, in particular with respect to local tree mapping, lexical lookup, and rule application, as described in more detail in Chapters 6 and 7²³ For further research into using the parsing algorithm directly for annotation, and for work on robust parsing, it is also necessary to be able to directly modify the components and algorithms used in the system, such as in particular dynamic changes to the type system and modifications to the unification algorithm.

²¹ The system in this dissertation does not require or support defaults (Lascarides & Copestake, 1999). Defaults are fully supported by the LKB but are not used in the LinGO ERG grammar.

²² A third use is also possible as a ‘tree parser’ for treebank annotation utilising the treebank tree structure to directly guide the parse. This has not been implemented for this dissertation.

²³ While the LKB could conceivably be used for some elements of the overall processing system, documented customisation options primarily relate to further processing of parse output (Copestake, 2002:207,208), which isn’t relevant for my purposes.

3 1 The chart parser

The parser is a basic bottom-up, agenda driven chart-parser, based on the description by Gazdar & Mellish (1989), adapted from a Prolog to a C# implementation. This parser has been extended to handle feature-based grammar formalisms, using a typed feature structure representation and unification algorithm adapted and extended from Tomabechi (1991).

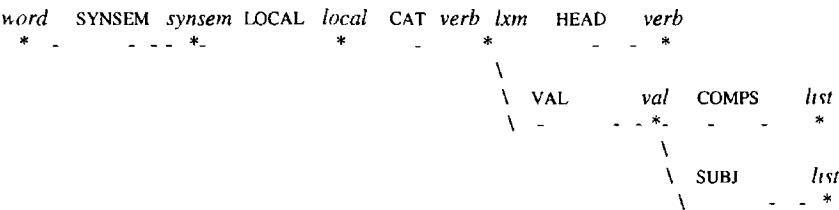
3 2 The grammar system, handling of typed feature structures

The most significant part of the grammar and parsing system is the handling of the HPSG types and type hierarchy – which closely follows the LKB system (Copestake, 2002) – with regard to how the types in the type system and other grammar items are loaded, expanded, and subjected to type inference.

3 2 1 Typed feature structures

Typed feature structures can be seen as directed acyclic graphs (DAGs), with a type on each node and feature labels on the arcs connecting the nodes (Jurafsky & Martin, 2000). Atomic types are equivalent to atomic values, and complex values are defined as further feature structures, that is, types with features defined as appropriate for them by the type hierarchy. This means that in a typed feature structure, every feature structure and every value has a type. A simple example is provided in Figure 3 1, where *verb* and *list* are atomic types, and *val*, *verb-lxm*, *local*, *synsem* and *word* are complex types with features.

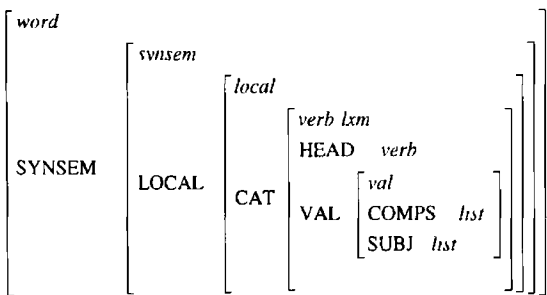
Figure 3 1 Typed feature structure represented as Directed Acyclic Graph



In the system developed in this dissertation, typed feature structures are implemented as DAGs, with some additional book-keeping information as defined by Tomabechi (1991).

They are presented in the usual format as attribute-value matrices (AVMs), with the AVM in Figure 3 2 being the equivalent of the graph in Figure 3 1. Complex types and their features are included in brackets creating successively larger feature structures.

Figure 3 2 Typed Feature Structure in attribute value matrix (AVM) format



3 2 2 Type inference and grammar pre-processing

Type inference takes a typed feature structure description, and results in a well-typed or well-formed typed feature structure. In a well-formed typed feature structure, all atomic and complex values have a type. In addition all features are appropriate for the type, as defined by the type hierarchy (Carpenter, 1992, Copestake, 2002). LKB-compatible grammars can be specified in a format where types, rules and lexical items can be described very compactly, since any features inherited from supertypes of the type being described will be inferred by the system. Figure 3 3 shows the lexical entry for ‘aircraft’ added to the LinGO ERG lexicon to parse the sentences in the ATIS treebank²⁴.

Figure 3 3 LKB lexicon entry in compact (unexpanded) format

```

aircraft_n1 = n_intr_le &
[ STEM < "aircraft" >,
  SYNSEM LOCAL KEYS KEY _aircraft_rel ]

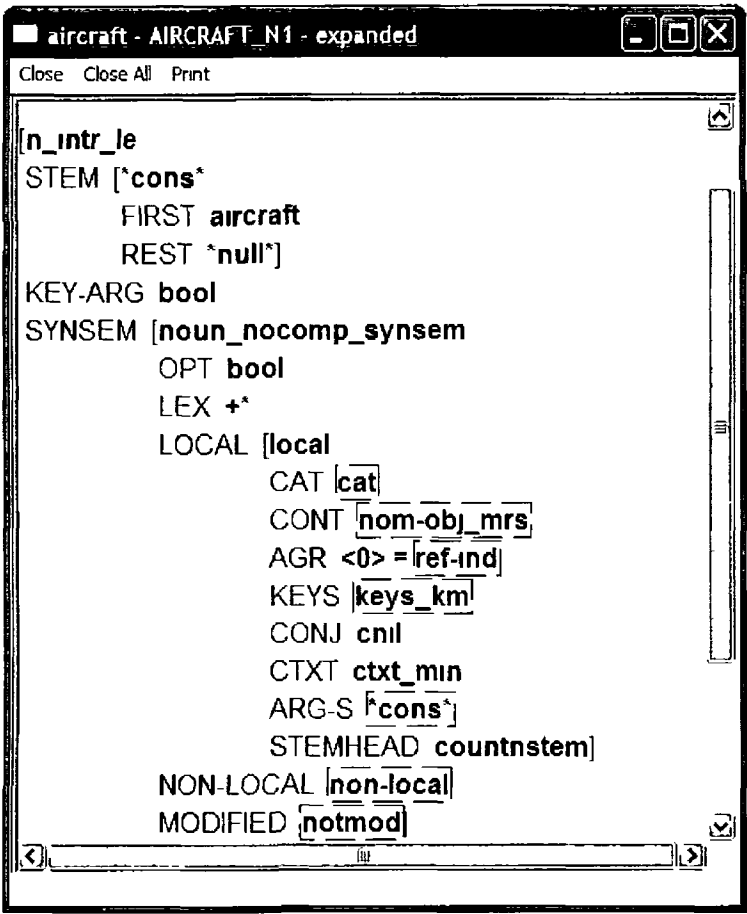
```

²⁴ See Section 4 1 1 for more details, and also Appendix 1 for a full list of added lexicon entries

Type inference is used to expand these descriptions into fully specified types or items, where all inherited features are present. The system ensures for all features that their values (which can be atomic or complex) have the correct, most specific appropriate type assigned.

Figure 3.4 shows a portion of a type-inferenced (expanded) version of the lexicon entry from Figure 3.3, as presented by the LKB (with many of the features of SYNSEM presented in their collapsed form, as the full feature structure would otherwise be too large to present on a page).²⁵

Figure 3.4 LKB lexicon entry in compact (unexpanded) format



²⁵ It is beyond the scope of this dissertation to discuss the LinGO ERG in detail. The grammar is documented to a certain extent in the grammar source definition files themselves, available for download at <http://LInGO.stanford.edu>.

A type or grammar item to which type inference has been applied in this fashion is said to be totally well-typed (Carpenter, 1992, Copestake, 2002). As for the LKB, type inferencing in the system developed for this dissertation is performed at grammar loading time for types and grammar rules. Lexical items are expanded at parse-time, for processing efficiency purposes.

3.2.3 Unification of typed feature structures

The unification algorithm used in my system is part of the grammar and type hierarchy system, rather than the parser, since unification is very closely linked to the type hierarchy itself. It is also needed for loading the grammar and performing operations on types and feature structures, such as type inference.

Unification needs to take types into account. For two typed feature structures to unify, their types need to be compatible, that is, they need to have a common unique subtype, or greatest lower bound (Carpenter, 1992, Copestake, 2002). The feature structures also need to unify in the same way as for untyped unification. Type compatibility needs to be checked for all types in possibly recursive feature structures. The unification result is the most general subtype that is common to both (the greatest lower bound), which may be one of the original types or a different common subtype (Jurafsky & Martin, 2000, Carpenter 1992). If the unification results in a type different to the two original types, type inferencing is required to ensure the resulting feature structure remains well-formed. The reason for this is that the new type may have additional feature restrictions specified for it that were not present in the original feature structures being unified. This may affect the unification outcome, so type inferencing is required to ensure the unification can be completed based on the correct feature structure specification.

The unification algorithm implemented in the parsing system for this dissertation is based on Tomabechi's quasi-destructive graph unification algorithm (Tomabechi, 1991), with an extension to handle typed feature structures. This is the same algorithm that forms the basis of the LKB system, although in the LKB several additional optimisations have been introduced, as described by (Malouf *et al.* 2000).²⁶

²⁶ Of these the system developed in this dissertation has so far only made use of the so called 'quick check' unification filter. This filter uses pre-computed grammar specific feature structure paths to pre-compare feature structures for common points of unification failure. In a parsing application where most

Figure 3 5 shows the modified algorithm

Figure 3 5 Unification algorithm for typed feature structures

```

Function unify1(dg1-underef,dg2-underef),
  dg1 ← dereference-dg(dg1-underef)
  dg2 ← dereference-dg(dg2-underef)
  IF (dg1 = dg2) THEN
    return('*T*'),
  ELSE IF (dg1 dtype = bottom) THEN
    forward-dg(dg1,dg2, temporary),
    return('*T*'),
  ELSE IF (dg2 dtype = bottom) THEN
    forward-dg(dg2,dg1, temporary),
    return('*T*'),
  ELSE
    glbtype ← greatest common subtype of dg1 type and dg2 type
    IF (glbtype = NULL)
      throw with keyword 'unify-fail'
    ELSE
      dg1 comp_type ← glbtype,
      dg2 comp_type ← glbtype,
      dg1 comp_type_mark ← *unify-global-counter*,
      dg2 comp_type_mark ← *unify-global-counter*
      IF (dg1 dtype = bottom) AND (dg2 dtype = bottom) THEN
        IF (dg1 arc-list = dg2 arc-list) THEN
          forward-dg(dg2,dg1, temporary),
          return('*T*'),
        ELSE throw with keyword 'unify-fail'
      ELSE IF (dg1 dtype = atomic) OR
        (dg2 dtype = atomic) THEN
        throw with keyword 'unify-fail'
      ELSE new ← complementarcs(dg2,dg1),
        shared ← intersectarcs(dg1,dg2)
        FOR EACH arc IN shared DO
          unify1(destination of the shared arc for dg1,
            destination of the shared arc for dg2),
          forward-dg(dg2,dg1, temporary),
          dg1 comp-arc-mark ← *unify-global-counter*,
          dg1 comp-arc-list ← new,
          return('*T*'),
    END,

```

unifications fail, the quick check can result in a significant optimisation, though to date in the annotation method used in this dissertation the quick check filter has not shown any benefits. This is probably due to the different profile of unification success to failure ratio with most unification operations in the annotation method succeeding during the annotation process.

Figure 3.5 shows the extended quasi-destructive graph unification algorithm for typed feature structures, in the same format as presented in (Tomabechi 19991) with additions and changes in bold. In addition the node data structure that the algorithm operates on has two extra fields, 'type' (changing the original field 'type' to 'dtype') and 'comp_type'. For a full description of the original algorithm and node data structure see (Tomabechi 1991). In Figure 3.5 Tomabechi's unify1 function is extended to check the type compatibility of the feature structures being unified. This is done by determining the greatest lower bound (which has been pre-determined by type inference). If there is no common greatest lower bound the types are not compatible and the unification fails. Otherwise unification proceeds.

As the typed feature structure resulting from the unification needs to have the appropriate type, and the quasi-destructive nature of the algorithm needs to be preserved, it needs to copy the appropriate type to the feature structure resulting from the unification. I use the 'comp_type' and 'comp_type_mark' flags for this in the same way as 'comp_arclist' and 'comp_arc_mark' are used in the original algorithm. This leads to a straightforward modification of the copy-dg-with-comp-arcs() function to also copy the comp_type if comp_type_mark matches the global unification counter,²⁷ also, as shown in Figure 3.6.

In addition to these changes there is also provision made in the implementation of the algorithm for well-formed unification, that is, ensuring that the outcome of the unification of two well-formed typed feature structures is also well-typed. There is also a cyclic feature structure check which causes unification failure if a cyclic structure is detected during the unification.

²⁷ As used by Tomabechi's unify0() function, unmodified for typed unification and not shown here.

Figure 3 6 Typed unification algorithm auxiliary function copy-dg-with-comp-arcs()

```

Function copy-dg-with-comp-arcs(dg-underref),
  dg ← dereference-dg(dg-underref)
  IF (dg copy is non-empty AND
      dg copy_mark = *unify-global-counter*) THEN
    return(dg-copy),
  ELSE IF (dg l dtype = atomic) THEN
    copy ← create-node(),
    copy dtype ← atomic,
    copy arc-list ← dg arc-list,
    IF (dg comp_type_mark = *unify-global-counter*) THEN
      copy type = dg comp_type,
    ELSE
      copy type = dg type,
    dg copy ← copy,
    dg copy_mark ← *unify-global-counter*,
    return(copy),
  ELSE IF (dg l dtype = bottom
) THEN
    copy ← create-node(),
    copy dtype ← bottom,
    IF (dg comp_type_mark = *unify-global-counter*) THEN
      copy type = dg comp_type,
    ELSE
      copy type = dg type,
    dg copy ← copy,
    dg copy_mark ← *unify-global-counter*,
    return(copy),
  ELSE
    copy ← create-node(),
    copy dtype ← complex
    FOR ALL arc IN dg arc-list DO
      newarc ← copy-arc-and-comp-arc(arc)
      push newarc into copy arc-list,
    IF (dg comp-arc-list is non-empty AND
        dg comp-arc-mark = *unify-global-counter*) THEN
      FOR ALL comp-arc IN dg comp-arc-list DO
        newarc ← copy-arc-and-comp-arc(comp-arc),
        push newarc into copy arc-list,
      IF (dg comp_type_mark = *unify-global-counter*) THEN
        copy type = dg comp_type,
      ELSE
        copy type = dg type,
    dg copy ← copy,
    dg copy_mark ← *unify-global-counter*,
    return(copy),
END,

```

3 3 Chapter Summary

This chapter has briefly described the typed feature structure grammar and parsing system for HPSG grammars implemented in C# for this dissertation. This system is used by the annotation method described in Chapters 6 to 7. It has also discussed the importance of the type hierarchy and type inference in grammar processing and parsing with typed feature structures.

4 Treebank pre-processing

This chapter introduces the ATIS treebank (Hemphill *et al* 1990)²⁸ used in this dissertation and describes the general and standard pre-processing steps normally performed as a preparation for treebank enrichment, such as head-lexicalisation, complement marking and binarisation. It also details the additional pre-processing required to make the treebank more HPSG-compatible.

The LinGO ERG HPSG grammar is well-suited to parsing the sentences in the ATIS corpus, since it was developed to parse utterances in a similar domain for the Verbmobil project, as discussed in Chapter 2. However, analyses produced by the HPSG ERG grammar differ from the ATIS trees in several respects, including strict binary branching, and head-lexicalisation.

The treebank annotation method requires that ATIS CFG local trees correspond as closely as possible to the equivalent HPSG phrases. Since many of the differences between ATIS and HPSG analyses are systematic, a principled automatic transformation of the ATIS trees can be made to make them more compatible with the HPSG grammar.

- 1 The trees are head-lexicalised, based on Magerman's and Collins' head-lexicalisation rules (Magerman, 1995, Collins, 1999), complement-marked based on Collins (1999), and binarised following Hockenmaier & Steedman (2002). This is described in Section 4.2.
- 2 HPSG-specific pre-processing transforms the trees by adding an N-bar level as an attachment point for post-modifier adjuncts and complements in NPs, and also 'collapses' noun compounds and unary productions. This will be described in Section 4.3.

Some further treebank transformation is done in the first annotation stage of local tree mapping, where some traces are deleted during the HPSG marking of long-distance dependencies. (This causes re-application of the above pre-processing steps for some sub-trees (see Chapter 6 for more detail).)

²⁸ Available with the Penn-II treebank <http://www.cis.upenn.edu/~treebank> visited 2nd July 2004

4 1 The ATIS treebank

The section of the ATIS treebank used in this dissertation is the sample of 577 transcribed utterances from ATIS 3 (Hemphill *et al* 1990, Dahl *et al* 1994) distributed with the Penn-II treebank (Penn, 1994), and annotated in Penn-II style (Marcus *et al* , 1994) The average utterance length is 7.4 words²⁹, with a total of 4,270 word forms in the corpus. The annotation was generated using the Fidditch parser (Hindle, 1983, 1989) and corrected by human annotators. Part-of-speech tagging was performed using Church's PARTS program (Church, 1988) and corrected by human annotators³⁰.

4 1 1 Parsing ATIS utterances with LKB and the LinGO ERG Grammar

In an experiment to determine what coverage the standard LinGO ERG grammar has on the ATIS corpus, I parsed the unannotated utterances with the LinGO ERG and the LKB system. For this purpose 41 lexical entries, mostly for place names, were added to the LinGO ERG lexicon (see Appendix 1). The results are summarised in Table 4.1 below.

Table 4.1 Parse results for LinGO ERG on ATIS corpus

Top ATIS category	Utterances	# Parsed	%	# Parses per input =1	# Parses per input 10>= x >1	# Parses per input >10	Total Parses, all inputs	Average # of parses per input
S	187	147	78.61%	30	69	48	2,373	16
SQ	26	17	65.38%	1	21	5	215	13
SBARQ	133	114	85.71%	19	54	31	878	8
Other	10	6	60.00%	3	2	1	43	7
Sub-total -FRAG	356	284	79.78%	53	146	85	3,509	12
FRAG	229	10	4.37%	7	2	1	79	8
Total	585 ³¹	294	50.26%	60	148	86	3588	12

In Table 4.1 the first column, 'Top ATIS category', indicates the top category assigned in the ATIS tree for the sentence, with S indicating a declarative or imperative sentence, SBARQ a wh-question, and SQ a yes/no question.

²⁹ The longest sentence (utterance) is 21 words long and the shortest 1 word long.

³⁰ For methodology and measurements on annotator correction validity see (Marcus *et al* , 1993).

³¹ This is higher than the number of distinct utterances and trees in ATIS which is 577 since some of the utterances are analysed in ATIS as separate trees parsed separately with the LKB and LINGO ERG.

FRAG indicates that the utterance is analysed as a non-sentence fragment, often a complete constituent such as an NP³² The fifth, sixth and seventh columns show how many parses are returned for sentences that do receive a parse The last two columns show total parses returned, and the average number of parses per utterance successfully parsed

While the overall parse success rate is just over 50%, the treebank contains a high degree of fragments, mostly orphaned NPs When these are factored out, 79.8% of utterances receive a parse There is a high degree of ambiguity in the parse results This is to be expected in parsing a corpus with a lot of utterances with multiple prepositional phrases Especially when a deep wide-coverage rule-based precise grammar is used and no parse-ranking or disambiguation is used There is an average of 12 parses per utterance and only about 20% of sentences successfully parsed receive a single parse

4.1.2 Comparison between ATIS and LinGO ERG parse trees

A comparison between an ATIS tree and the LmGO ERG parse tree produced by the LKB system for the same input string illustrates some of the structural differences that the treebank pre-processing needs to address if the HPSG grammar is to be successfully applied to annotate the ATIS tree structures³³

ATIS trees have a relatively flat structure and tend to be short A typical example of a parse tree is given in Figure 4.1 Note the marking of empty categories and co-indexation of traces following Penn-II annotation guidelines The NP-SBJ trace is marked with ‘*T*’ to indicate WH-movement, and co-indexed with ‘1’ with the preceding WHNP determiner ‘that’

³² The LKB system can be configured to accept a non sentence constituent such as an NP or VP as a complete sentence and return parses for it For the purposes of this exercise this capability was not required

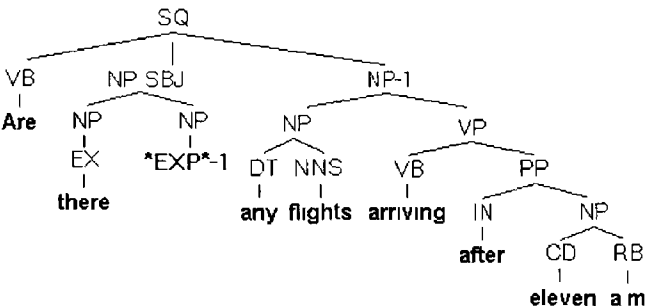
³³ Since the parse trees and part of speech annotations for the ATIS corpus are distributed separately, for this dissertation the two were merged by assigning POS tags to the words in the trees before pre processing started

Comparing the two trees, the main differences are

- 1 The ATIS treebank trees are relatively flat, while ERG trees are strictly binary branching (with unary pre-terminals)
- 2 The categories used are similar but the ERG categories used in the parse tree are more general – this is a reflection of the category ‘short-hand’ used by the LKB to display parse trees, where a CFG-style category is used as an abbreviation for a phrasal sign
- 3 Empty categories and traces are marked explicitly in the ATIS tree, through co-indexing (‘-1’) and explicit empty terminal categories, while in the ERG tree long-distance dependencies are marked with a slash feature (‘/NP’ in Figure 4 2), and empty productions are not used
- 4 There are structural differences with respect to determiner and prepositional phrase attachment. These are caused by the use of an N-bar level and strict binary branching in the ERG

Some further differences between ATIS and ERG trees are shown in Figures 4 3 and 4 4

Figure 4 3 ATIS tree (id @8kr033sx-d-9)



These examples illustrate some of the differences between ATIS trees and LinGO ERG parse trees. The examples indicate what kind of pre-processing is required for the treebank trees to match and be annotated with LinGO ERG HPSG typed feature structure information. The following sections discuss the pre-processing steps in more detail.

4.2 Standard pre-processing

As noted in Chapter 2, a common starting point for many treebank transformation and annotation methods is to perform head-lexicalisation, complement marking and binarisation. For this dissertation I use the same heuristic rules and methods as other researchers, from (Magerman, 1994, Magerman, 1995) and (Collins, 1999) for head and complement marking, and from (Hockenmaier & Steedman, 2002) for binarisation. I also make some modifications and extensions to these rules, as described in the following sections.

4.2.1 Head-lexicalisation

The first step in transforming the context-free grammar-based ATIS trees to the point where they can be HPSG-annotated is to identify and mark the head-word, where found, of every local tree. This involves identifying the lexical head for each local tree in the parse tree, and annotating the tree with information identifying the head daughter of each local tree. This is particularly important given that headedness is a critical property of HPSG phrases. Unheaded constructions such as coordination are discussed in section 6.2.

Several researchers have used the head rules defined by (Magerman, 1994) and (Collins, 1999) which were originally designed for the WSJ corpus, annotated following Penn-II guidelines (Marcus *et al.* 1994). These rules also work well as a starting point for the ATIS corpus and other treebanks which use the same tag-set and annotation principles as Penn-II.

Head-lexicalisation is driven by a set of rules that define which daughter should be the head, for each local tree and corresponding CFG rule. Given the mother CFG category, they determine which of the daughter categories is the preferred head-daughter category, and from which direction daughters should be considered for head-daughter candidacy. The original head rules from Magerman (excluding co-ordination rules) are shown in Table 4.2.

Table 4.2 Head-rules, from (Magerman, 1995)

Mother category	Order	Head daughter candidates
NP	Right	NX,NN NNP NNPS,NNS NX POS JJR
NP	Left	NP
NP	Right	\$.ADJP PRN
NP	Right	CD
NP	Right	JJ,JJS,RB,QP
ADJP	Left	NNS QP NN \$ ADVP JJ VBN VBG ADJP JJR NP JJS DT FW RBR RBS SBAR RB
ADVP	Right	RB RBR,RBS FW ADVP,TO CD JJR JJ IN NP JJS NN
CONJP	Right	CC RB IN
FRAG	Right	
INTJ	Left	
LST	Left	LS
NAC	Left	NN NNS,NNP NNPS NP,NAC EX \$ CP QP PRP,VBG JJ JJS JJR ADJP,FW
PP	Right	IN TO VBG VBN RP FW
PRN	Left	
PRT	Right	RP
QP	Left	\$.IN NNS NN,JJ RB DT CD NCD,QP JJR JJS
RRC	Right	VP NP ADVP,ADJP,PP
S	Left	TO IN,VP S SBAR ADJP,UCP NP
SBAR	Left	WHNP WHPP WHADVP WHADJP IN,DT,S SQ SINV SBAR,FRAG
SBARQ	Left	SQ,S,SINV SBARQ FRAG
SINV	Left	VBZ VBD VBP VB,MD VP S SINV ADJP,NP
SQ	Left	VBZ VBD VBP VB,MD VP SQ
UCP	Right	
VP	Left	TO,VBD VBN MD,VBZ VB VBG VBP VP,ADJP,NN,NNS
WHADJP	Left	CC WRB JJ ADJP
WHADVP	Right	CC WRB
WHNP	Left	WDT WP,WPS WHADJP WHPP WHNP
WHPP	Right	IN TO FW
NX	Right	NN NX

Several alterations have been made to Magerman’s head-rules in this dissertation, to suit the ATIS corpus and the HPSG annotation. These are presented in Table 4.3, showing only the changed rules, which replace the original rules for the same mother category.

Table 4 3 Modified head-rules

Mother category	Order	Head daughter candidates
NP	Right	NP
PP	Right	IN TO VBG, VBN RP FW PP
S	Left	INFTO, IN VP S SBAR, ADJP UCP NP
SBAR	Left	S SQ SINV SBAR WHNP WHPP WHADVP, WHADJP IN DT, FRAG
SQ	Left	SQ VBZ, VBD VBP VB MD, VP
VP	Left	INFTO, VBD, VBN MD VBZ, VB, VBG VBP VP ADJP, NN NNS
WHNP	Left	WDT, WP, WPS WHADJP, WHPP WHNP

The changes are as follows ³⁶

- (i) NP rules, changed so that rightmost rather than leftmost NP daughter is the head of the NP, to support HPSG head-final NP compound structure, following (Cahill *et al* , 2002)
- (ii) PP added as permissible head daughter in a PP rule, to support correct head-marking in a binarised PP with an adjunct pre-modifier, such as ‘early in the day’
- (iii) VP rule changed by adding a custom part of speech tag to distinguish infinitival use of ‘to’, INFTO, as first leftmost permissible head, replacing the more general tag TO This change is also made for the S rule
- (iv) Changed head order for SBAR to make subordinate s-clause S, SQ, SINV and SBAR rank before WH and NP phrases, also to preserve correct head-marking after binarisation
- (v) Changed head order for SQ to make SQ rank first, not last
- (vi) Added to WHNP to make NN first head daughter if it exists

4 2 2 Complement-marking

In addition to head-marking, complement/adjunct distinctions also need to be marked in the ATIS trees to allow accurate matching against HPSG head-complement rules and other schemas Complement identification follows the rules in (Collins, 1999), and identifies complement daughters, unless they are marked with a particular ‘complement-blocking’ semantic tag,³⁷ in which case they are treated as adjuncts Collins’ rules are listed in Table 4 4

³⁶ The rules are heuristics approximations developed to suite the particulars of the ATIS treebank No claim is made for more general validity

³⁷ Exclusion tags are ADV, VOC BNF DIR EXT, LOC MNR, TMP CLR and PRP

Table 4 4 Collins’ complement rules

Mother category	Complement daughter candidates
S	NP SBAR S
VP	NP, SBAR S, VP
SBAR	S

Several additions have been made specifically for HPSG and LinGO ERG compatible annotation. These are shown in Table 4 5

Table 4 5 Additional complement rules

Mother category	Complement daughter candidates	Head daughter
SQ	VP NP	
PP	NP	
SBARQ	WHNP	
VP, S SQ SBAR SBARQ	VP, VB VBD VBG, VBN VBP VBZ S SQ SBAR SBARQ	INFTO
VP S SQ SBAR, SBARQ	ADJP PRD	VB

These rules function in the same manner as Collins’ rules, in specifying which daughter is a permissible complement for a given mother category, with an additional condition of a particular sister head category being present for the last two rules

- (i) VP and NP are permissible complements to SQ mother
- (ii) NP is a permissible complement to PP, supporting HPSG analysis of a PP phrase as a head-complement phrase
- (iii) WHNP is a permissible complement to SBARQ
- (iv) INFTO rule VP, VB, VBD, VBG, VBN, VBP, VBZ and S, SQ, SBAR, SBARQ are permissible complements to INFTO head with a VP, S, SQ, SBAR or SBARQ mother
- (v) Pred rule ADJP-PRD is a permissible complement to a VB head with an S, SQ, SBAR, SBARQ or VP mother

4 2 3 Binarisation

In addition to dependency marking, the treebank trees also need to be made binary branching. Once heads and complements have been identified the binarisation is straightforward, following the method described in (Hockenmaier & Steedman, 2002)

The binarisation process inserts dummy nodes into the tree so that all children to the left of a head branch off in a right-branching tree, and all children to the right branch off in a left-branching tree. Figure 4.5 shows the binarisation algorithm which is used to process and reconstruct the ATIS trees.³⁸

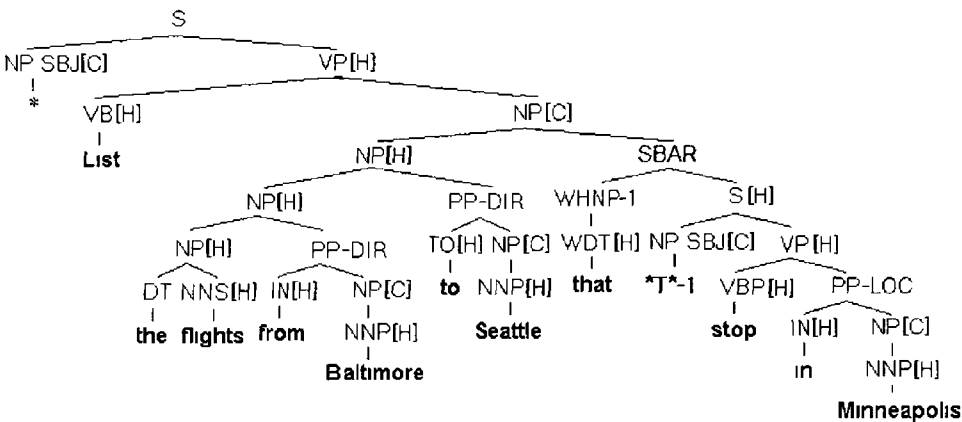
Figure 4.5 Binarisation algorithm

Starting with the topmost local tree, for each local tree

- 1 If the current local tree has more than 2 daughters (tree is not already binary)
 - 1.1 If the head daughter is not the first daughter
 - 1.1.1 Create a new tree as a copy of local tree with first daughter removed
 - 1.1.2 Remove all except first daughter from current local tree
 - 1.1.3 Add new tree as second daughter to current local tree
 - 1.2 If the head daughter is the first daughter
 - 1.2.1 Create new tree, copy of current local tree except for removing last daughter
 - 1.2.2 Remove all except last daughter from current local tree
 - 1.2.3 Add new tree as new first daughter to current local tree
 - 1.3 For all daughters of current local tree, apply the algorithm recursively

Figure 4.6 shows an example of the application of the algorithm, the ATIS tree from Figure 4.1 after head and complement marking and binarisation. Heads and complements have been explicitly marked, and the NP daughter of the main VP has been restructured so that all its non-head children are organized in a left-branching tree.

Figure 4.6 ATIS tree @8kr033sx-d-9, head/complement marked and binarised



³⁸ In the binarisation module implemented for the annotation system the binarisation head and complement marking is also integrated, as binarisation of the tree can require reapplication of head/complement rules.

4 2 4 Collapsing of unary productions

One further pre-processing operation that is also commonly performed on treebanks is to collapse unary productions, i.e. the removal of local trees containing a single daughter attaching the daughter instead to the parent of the removed tree. The category of the daughter is used rather than the mother category of the tree being removed.³⁹ This addresses the issue raised in Section 4 1 2 about the difference between unary productions in ATIS and the LinGO ERG,⁴⁰

4 3 HPSG-specific pre-processing

In addition to the standard pre-processing and treebank tree-transformation steps described in Section 4 2, a number of more HPSG-specific changes are also made. These include the re-introduction of an N-bar level for NPs to make pre-modifier attachment of determiners consistent between the ATIS trees and LinGO ERG grammar, NP-compound collapsing, infinitival ‘to’ marking, and grammar compacting. These are described in the following sections.

There are a few further elements of the annotation process that change the treebank structure but these are handled in the annotation process, since they directly interact with establishing the correspondence between CFG and HPSG phrases and categories. They are, therefore, not treated as a pre-processing step. This involves in particular the handling of empty categories and traces, which is discussed in Chapter 6.

4 3 1 N-bar handling

One characteristic of ATIS and many other treebanks is the flat structure used when analysing constituents of NPs and VPs, in particular for prepositional phrases (PPs) which are very common constituents in ATIS sentences, as seen in Figure 4 1. Flat structures are attractive since they underspecify the modifier attachment.

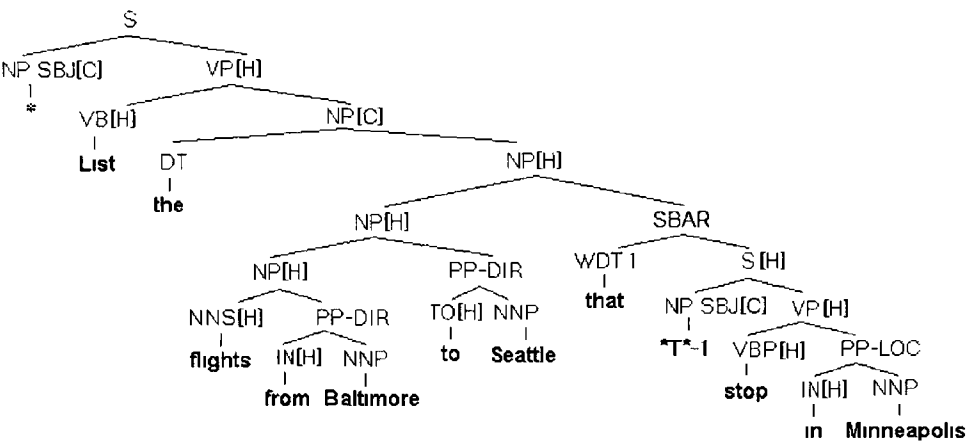
³⁹ The category of the removed production is stored for further use, to ensure that the local tree mapping described in Chapter 6 and other configurations can still work on collapsed productions.

⁴⁰ HPSG in general and the LinGO ERG grammar in particular does make use of unary lexical and grammar rules (which license unary productions). An N-bar level is also used in the case of NP post head modifiers as will be discussed later in Section 4 3. The annotation is simplified however by the removal of any initial unary productions and therefore this is done as a pre-processing stage.

However, in order to ensure HPSG compatibility, in particular with the LinGO ERG – which restricts noun post-modifiers to attach below pre-modifiers, in effect at an N-bar level – some restructuring of ATIS parse trees is required. To accommodate this requirement a ‘tree-lowering’ rule is used when both pre and post-modifiers are detected in an NP. This rule introduces an N-bar level by inserting an intermediate tree between the NP mother and the head noun, at which pre-modifiers are attached.⁴¹ Post-modifiers such as NPs and PPs remain sisters of the head noun so are in effect lowered.

This NP post-modifier lowering rule operates before the standard head, complement, and binarisation operation (considering only categorial, not dependency information in analysing the NP), and in conjunction with these rules, produces a fully HPSG-compatible tree structure. Figure 4.7 illustrates N-bar introduction and unary rule-collapsing. In comparison with Figure 4.6, the following has changed: (i) a number of straightforward unary rule reductions, and (ii) the determiner pre-modifier ‘the’ has been made a sister of a new N-bar level NP node.⁴²

Figure 4.7 ATIS tree @8kr033sx-d-9 with collapsed unary productions and N-bar level



⁴¹ Interestingly, the N bar level was specifically removed in the creation of the Penn Treebank (Marcus *et al.* 1993).

⁴² This example together with cases of PP post modifiers to NPs with determiner and adjective pre modifiers are the only configurations where N bar introduction is required for the ATIS treebank.

4 3 2 NP-compound collapsing

An NP compound collapsing rule is used to convert all NPs containing only noun daughters into single daughter NPs with a multi-word noun compound. This allows the HPSG annotation to treat the compound as a single word, eliminating a source of complexity in the analysis. This change is not strictly required for HPSG annotation, but is used as a simplifying measure.

4 3 3 Infinitival ‘to’ marking

As a simple prerequisite for further annotation, cases of infinitival ‘to’ are identified by detecting where a VB is followed by an IN or TO. The preposition tag is replaced with the disambiguated INFTO tag. The generic prepositional part-of-speech tag is deliberately used in the Penn-II annotation since it can be disambiguated from the syntactic context.

4 3 4 Grammar compacting

The final pre-processing step, which is performed after the standard head/complement/binarisation rules are applied, is grammar category compacting.

In order to simplify the tag-set for mapping to HPSG, and also to reduce the number of unique rules extracted from the treebank, a simple form of compacting of categories is performed, based on the matrix given in Appendix 2. While this results in some loss of information, the pre-processing steps of complement marking and trace handling have already made use of the functional information and co-indexation marking, and the additional information is still available to the annotation process, as used in lexical mapping, see Sections 5.1.2 and 5.1.3. The primary use for the reduced rule-set is in simplifying the phrase mapping, described in Section 5.2. This change results in significant grammar compacting, as shown in Table 4.6. This table includes a frequency distribution, and also illustrates that binarisation has an impact on the grammar size.

Table 4 6 Grammar size reduction due to pre-processing and compacting

Frequency	Initial rule set	Binary headed rules	Category compacted binary headed rules
>20	42	63	45
20>=x>5	32	47	24
5>=x>1	156	130	49
1	233	172	55
Total	463	412	173

4 3 5 Miscellaneous corrections

To preserve the fundamental headedness property of phrases which is a characteristic of HPSG – where the head daughter and mother of a local tree need to have the same, or compatible head-feature values – there are instances where the treebank category marking needs to be changed. In ATIS there are not that many instances of this, but one that needs to be addressed is where an NP for strings of the form ‘the ninth’ is analysed as NP → DT JJ. Local trees/CFG rules on this format are changed in pre-processing to NP → DT NN.⁴³

4.4 Chapter Summary

In this chapter, the ATIS parsed corpus (or treebank), used in this dissertation has been introduced. Its general characteristics and parse tree similarities and differences with the LinGO ERG grammar have been reviewed. An experiment in parsing the ATIS corpus sentences with the LinGO ERG is reported on.

This chapter has also described the pre-processing performed on the ATIS treebank to prepare it for HPSG annotation, with both established pre-processing techniques and additional modifications developed specifically for this dissertation. These include head/complement marking, binarisation, collapsing of unary productions, introduction of an N-bar level, grammar compacting and other miscellaneous corrections. These pre-processing steps have all been fully automated for the annotation system. They are used as heuristics and hand-tuned to produce the best result for the purposes of this dissertation.⁴⁴

⁴³ Alternatively the CFG to HPSG mapping can be made sensitive to preserving headedness for the NP → DT JJ rule. For this dissertation the choice was made to modify the category in pre processing to simplify the mapping.

⁴⁴ While the techniques are not guaranteed to produce output fully compatible with the LinGO ERG grammar or HPSG in general, I believe they are a good first approximation.

5 CFG to HPSG lexical and phrase mapping

The annotation method developed in this dissertation assigns HPSG typed feature structures to ATIS trees, by first associating a skeletal HPSG sign with each tree node in the ATIS tree, and then using constraint resolution and HPSG grammar rule application to create a single HPSG phrasal sign. The annotation method takes as input ATIS trees, an HPSG grammar, and a defined mapping from CFG categories and rules to HPSG skeletal signs. The CFG-to-HPSG mapping is described in this chapter.

Using the LinGO ERG grammar I can take advantage of its rich type system – in particular for lexical information – to provide a more specific CFG-to-HPSG mapping, relative to the ATIS corpus, than would be possible with a more basic, coarse-grained HPSG grammar.

In order to create skeletal HPSG signs with which to annotate the pre-processed ATIS treebank trees, each CFG category and local tree needs to be associated with the most closely-related HPSG lexical and phrase type in the LinGO ERG grammar. On the lexical level, the part-of-speech tags used in ATIS can be related to corresponding HPSG lexical types in the LinGO ERG. Additional argument structure and inflectional information from the part-of-speech-tags can be used to further specify the lexical mapping. This is described in Section 5.1. On the phrase level, the phrasal category tags used in ATIS can be related to corresponding HPSG phrase descriptions. Dependency marking and part-of-speech tagging is also used, to develop general principles for mapping CFG rules into major HPSG phrase types. This is described in Section 5.2.

5.1 Lexical mapping

In establishing a general correspondence between a CFG and the context-free backbone encoded in the daughter feature structures of an HPSG grammar, underspecified or skeletal HPSG lexical signs corresponding to major part of speech types can be created, as discussed in (Yutaka *et al.* 1998). Given the access to a large and detailed grammar (the LinGO ERG), this can be improved on further than simply providing a general structure for each part of speech by making use of the predefined lexical types for the LINGO ERG grammar and lexicon.

5 1 1 Mapping ATIS part of speech tags to LinGO ERG lexical types

The ATIS corpus used in this dissertation is annotated with the Penn-II treebank tagset (Marcus *et al* 1994) The part-of-speech tags are included in Appendix 3 For grammar compacting purposes, some tags are collapsed,⁴⁵ although the full tag-set is still used for the lexical mapping, see Section 5 1 3 where the different tag-sets for compacting and lexical mapping are detailed

The ATIS part-of-speech tags can be mapped to LinGO ERG lexical head and word types by aligning the tagged words with the existing LinGO ERG lexicon – to the extent that the lexicon covers the corpus – and by making use of the type hierarchy This mapping should be amenable to a semi-automated process, but for this dissertation the mapping was produced manually By using representative sample words for the ATIS POS tags from the corpus and the lexicon of the LinGO ERG, a basic mapping between categories has been established, as shown in Table 5 1

Table 5 1 Mapping from ATIS POS tags to LinGO ERG head and word types

ATIS Tag	Example word	LinGO ERG head type	LinGO ERG word type	LinGO ERG Tag generic word type
CC	And or	unheaded	conj_word	conj_word
CD	Six	intadj	norm_num_word	norm_num_word
DT	A, the these	det	basic_det_word	basic_det_word
EX	There	noun*	basic_noun_word	basic_noun_word
IN	To, in, from	prep*	basic_prep_word	basic_prep_word
TO	To as in 'want to	comp	complementizer_word	complementizer_word
JJ	Other	adj	adj_word	word
JJ	First	adj	norm_num_word	word
MD	Can d could	verb*	aux_verb_word	aux_verb_word
NN	Flight Baltimore	noun*	basic_noun_word	basic_noun_word
PDT	All	det	basic_det_word	basic_det_word
POS	s	unheaded	mcna	mcna
PRP	i he them	noun*	basic_pronoun_word	basic_pronoun_word
RB	Here	adv	basic_noun_word	word
RB	Around	adv	que_word	word
RP	Out as in come out	prep*	basic_prep_word	basic_prep_word
SYM	D c, slash	noun*	basic_noun_word	basic_noun_word
UH	uh oops	root_marker	msg_word	msg_word
VB	list stop	verb*	main_verb	main_verb
WDT	that, which, what	det	basic_det_word	basic_det_word
WP	What	noun*	basic_noun_word	basic_noun_word
WRB	Where	adv	basic_noun_word	word
WRB	How	adv	que_word	word

⁴⁵ See Appendix 3 for details

In Table 5 1, column three, ‘LinGO ERG head type’ shows the part-of-speech head type from the LinGO ERG that corresponds most closely to the ATIS POS-tag. In HPSG, part-of-speech is a head feature. However, not all word classes can act as heads, so for some ATIS part-of-speech tags, such as conjunctions and interjections – which correspond to un-headed lexical types – no equivalent LinGO ERG part of speech type is given⁴⁶. These are marked “unheaded” in Column 3. Column 4, ‘LinGO ERG word type’, lists the lexical sign or word type in the ERG corresponding most closely to the ATIS POS-tag. A single ATIS POS-tag may correspond to several different word types in the LinGO ERG. Therefore, there are multiple rows in the table for some tags. The last column in the table lists the most specific lexical type that is compatible with all corresponding ATIS POS-tags.

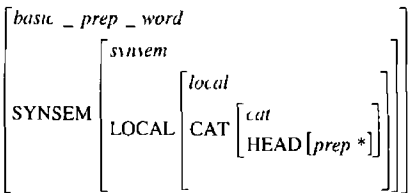
Based on this mapping, a skeletal LinGO ERG-compatible HPSG feature structure can be constructed for each ATIS POS-tag, as follows:

For each ATIS-POS tag

- (i) Use as word type the type given in Table 5 1 for ‘LinGO ERG tag-generic word type’
- (ii) Use as head category type, for the same tag, the type given in Table 5 1 for ‘LinGO ERG head type’

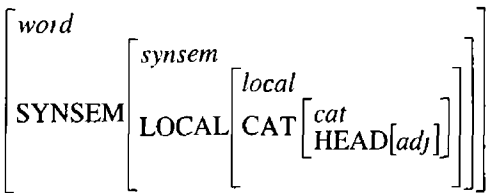
As an example, the above method constructs for the ATIS POS tag ‘IN’ the typed feature structure in Figure 5 1, and for the tag ‘JJ’ the typed feature structure in Figure 5 2. Both feature structures are based on the mapping from Table 5 1.

Figure 5 1 HPSG skeletal word sign for ATIS ‘IN’ part of speech tag



⁴⁶ The *mcna* type is a general word type with the restriction MC na, indicating that a word of this type cannot be the head of a clause.

Figure 5 2 HPSG skeletal word sign for ATIS ‘JJ’ part of speech tag



In Figure 5 1 the LinGO ERG lexical type corresponding to the ‘in’ POS-tag, namely *prep**, is assigned to the HEAD feature, and the whole sign is given the type *basic_prep_word* following Table 5 1 In Figure 5 2 the LinGO ERG lexical type corresponding to the ‘JJ’ POS-tag, namely *adj* is assigned to the HEAD feature, and the whole sign is given the type *word* following Table 5 1

5 1 2 Using argument structure to improve the lexical mapping

The general lexical mapping established in Table 5 1 is based only on the major part-of-speech recovered from the ATIS part-of-speech tag, and therefore is restricted to the most general LinGO ERG lexical types This mapping can be improved on by making use of argument and tree structure information from the pre-processed treebank to select more specific lexical types for the main parts of speech

The Penn-II treebank mark-up of the ATIS corpus allows for syntactic argument-structure determination to be discovered, which is used in the head/complement marking ⁴⁷ By extracting the complements for head-words, and also considering the parent category together with the part-of-speech tag, a more fine-grained subcategorisation and configurational classification can be made of the words in the corpus This has been investigated for verbs, nouns, adjectives and adverbs in ATIS, and can help identify more specific lexical types for verbs in particular

For verbs, only main verbs excluding modals and auxiliaries have been investigated The mapping established is shown in Table 5 2

⁴⁷ This can also be extended further, for example with more fine-grained complement marking rules (Kinyon & Prolo, 2002)

Four complement frames were found intransitive, transitive with an NP complement, transitive with an S complement, and ditransitive with two NP complements⁴⁸ These correspond to specific LinGO ERG lexical types, for intransitive, transitive, control (equi)⁴⁹ and ditransitive verbs

Table 5 2 Verb argument mapping to LinGO ERG lexical types

Complement frame	Mother category	LinGO ERG lexical type	LinGO ERG synsem type
V_intransitive	VP	<i>v_unerg_le</i>	<i>unerg_verb</i>
V_transitive_NP	VP	<i>v_np_trans_le</i>	<i>np_trans_verb</i>
V_transitive_S	VP	<i>v_subj_equi_le</i>	<i>subj_equi_verb</i>
V_ditransitive_NP_NP	VP	<i>v_ditrans_only_le</i>	<i>ditrans_only_verb</i>

For nouns and adjectives, there is little or no evidence of argument/complement structure in the ATIS corpus, with no significant correlations between argument structure and lexical type Only the following mappings, summarised in Table 5 3, can be made

Table 5 3 Extended part-of-speech mapping to LinGO ERG lexical types

Part of speech	ATIS tag(s)	LinGO ERG lexical type	LinGO ERG synsem type
Common noun	NN, NNS	<i>n_intr_le</i>	<i>noun_nocomp_synsem</i>
Proper noun	NNP NNPS	<i>n_proper_le</i>	<i>noun_nocomp_synsem</i>
Adjective	JJ	<i>word</i>	<i>adj_synsem</i>
Comparative adj	JJR	<i>adj_comp_le</i>	<i>intrans_adj_synsem</i>
Superlative adj	JJS	<i>adj_superl_le</i>	<i>intrans_adj_synsem</i>
Ordinal adjective	JJ	<i>norm_num_synsem</i>	<i>norm_num_word</i>
Adverb	RB RBR RBS	<i>word</i>	<i>synsem</i>

Given that no evidence of complements is found in the corpus, common nouns are assigned an intransitive ERG type, *n_intr_le* Proper nouns are mapped to the single LinGO ERG proper noun word type *n_proper_le*, as before Adjectives are mapped to the common SYNSEM subtype *adj_synsem* They have no single common adjective word type JJR and JJS can be mapped to the more specific types *adj_comp_le* and *adj_superl_le*

⁴⁸ In the LinGO lexicon a large number of more fine-grained verb types are used This table is an approximation suitable for the ATIS corpus, with a restricted set of verb complement structures

⁴⁹ See the further discussion on raising and control verbs in Section 6 3 4 Based on argument structure analysis a specific mapping to a control verb lexical type can be made The indicator provided by the infinitival *to* – as discussed in Section 6 3 4 – cannot disambiguate between control and raising verbs and can only provide for a more generic mapping

The exception is ordinals, such as ‘first’, which are treated as a separate lexical type, *norm_ord_word*. Adverbs, finally, do not form a single type in LinGO ERG, and can only be mapped to the most generic *word* type⁵⁰

5 1 3 Using lexical morphological rules

The part-of-speech tags used in ATIS for nouns and verbs encode number and tense information, which corresponds to lexical morphological rules in the LinGO ERG grammar. These rules constrain the SYNSEM LOCAL value of lexical signs. For nouns, there are singular and plural inflectional rules, and for verbs there are tense rules. The correspondence is listed in Table 5 4.

Table 5 4 Part of speech mapping to LinGO ERG lexical rule

ATIS tag	Description	LinGO ERG lexical rule	LinGO ERG LOCAL type
NN	Noun, singular or mass	Sing_noun_infl_rule	<i>sing_noun</i>
NNS	Noun, plural	Plur_noun_infl_rule	<i>plur_noun</i>
VB	Verb, base form	Bse_verb_infl_rule	<i>bse_verb</i>
VBD	Verb, past tense	Past_verb_infl_rule	<i>past_or_subj_verb</i>
VBG	Verb, gerund	Prp_verb_infl_rule	<i>prp_verb</i>
VBN	Verb, past participle	Psp_verb_infl_rule	<i>psp_verb</i>
VBP	Verb, non 3sg pres	Non_third_sg_fin_verb_infl_rule	<i>non_third_sg_fin_verb</i>
VBZ	Verb, 3sg pres	third_sg_fin_verb_infl_rule	<i>third_sg_fin_verb</i>

The tag-to-lexical rule and type correspondence can be used to further specify mapped lexical sign feature structures, and constrain further rule application.

5.2 Phrase mapping

Phrase mapping makes use of the ATIS syntactic tags and the dependency and configurational structure encoded in the treebank during pre-processing, to determine the appropriate skeletal phrasal HPSG sign. This sign is constructed from three components: phrase descriptions, rule descriptions and phrase types.

Feature structure representations of CFG phrasal categories – so-called phrase descriptions – are introduced in Section 5 2 1. Rule descriptions are feature structure representations of CFG rules, described in Section 5 2 2.

⁵⁰ The lack of a single most generic lexical type for adjectives and adverbs is a feature of the LinGO ERG grammar, reflecting the lack of commonality in the lexical types corresponding to a single ATIS part of speech tag for these categories.

To provide a mapping with HPSG grammar phrase types, mapping principles based on dependency and categorial information are presented in Section 5.2.3. Phrase descriptions and phrase types are combined with CFG rule structures to form a complete skeletal HPSG phrase sign for use by the annotation method, as described in Section 5.2.4.

5.2.1 Phrasal categories and phrase descriptions

The syntactic (or phrasal) categories in ATIS, also based on the Penn-II set of phrasal categories, are listed in Table 5.5. The third column, ‘Head daughter POS tag’, lists the part-of-speech tag of the head-word of which the phrase is a projection, based on the head-marking described in Chapter 4.

Table 5.5 ATIS syntactic tags

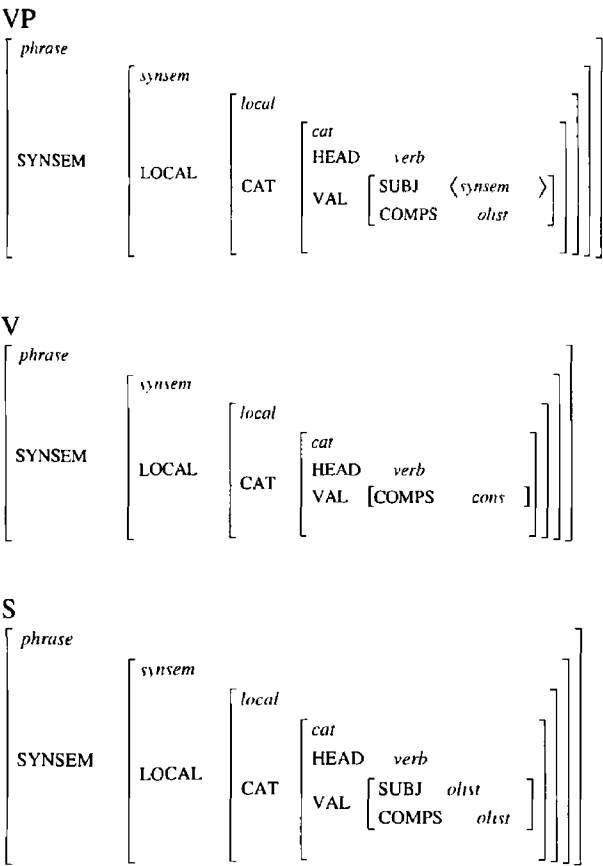
Syntactic Tag	Description	Head daughter POS tag
ADJP	adjective phrase	JJ
ADVP	adverb phrase	RB
FRAG	fragment	n/a
INTJ	interjection	UH
NP	noun phrase	NN
NX	noun phrase	NN
PP	prepositional phrase	IN
PRT	verb particle	RP
QP	determiner quantifier phrase	JJ
S	simple declarative clause	VB
SBAR	clause identified by subord conj, or 0	VB
SBARQ	direct question	VB
SQ	constituent of SBARQ excluding WH word/phrase	VB
VP	verb phrase	VB
WHADJP	wh adjective phrase	JJ
WHADVP	wh adverb phrase	WRB
WHNP	wh prepositional phrase	WP
X	unknown /uncertain category	n/a

While CFG syntactic categories have no direct equivalents in HPSG and the LinGO ERG grammar, they can be related to feature structure configurations to provide a skeletal phrasal sign representing the local tree and corresponding CFG rule. This is exemplified by the LinGO ERG parse trees discussed in Chapter 4 (Figures 4.2, and 4.4). The LinGO ERG and LKB uses labelled feature structures for this parse labelling, and these forms of skeletal phrase descriptions of signs are also useful in this work as the basis for CFG-to-HPSG phrase mapping.

Figure 5 3 below shows a few examples of phrase descriptions, and a more complete collection for the major categories based on a simplified and modified version of the LinGO ERG parse-nodes is included in Appendix 4

Note that the types *olist* and *cons* in Figure 5 3 are specific list types in the LinGO ERG type hierarchy, describing respectively a list that can optionally be null or contain elements, and a list with a head and a remainder

Figure 5 3 Sample phrase descriptions



5 2 2 Rule descriptions

In addition to phrase descriptions, a basic typed feature structure is created for each CFG rule derived from the treebank. This feature structure is created solely based on dependency information in the rule, and the category mapping, based on Tables 5 1 and 5 5, which define the LinGO ERG category type corresponding to each CFG category in the rule. The head-marking indicates whether the rule is head-initial or head-final. Figure 5 4 describes the method for constructing these so-called rule-descriptions.

In Figure 5 4, an ATIS POS-category to LinGO ERG type mapping is first performed, using Table 5 5. A feature structure is then created with a general *phrase* type, and the head-feature set corresponding to the mother CFG category. Based on dependency information, a LinGO ERG compatible feature structure using ARGS, HD-DTR and NON-HD-DTR features is then constructed. The ARGS feature indicates linear precedence, and the head and non-head daughter is co-indexed with the first or second argument depending on whether the phrase is head-initial or head-final (for unary phrases there will only be one element in the ARGS list).

Figure 5 4 Rule description construction method

For each CFG rule
1) Look up the LinGO ERG part of speech type equivalents for the categories in the CFG rule
2) Construct a typed feature structure, of type <i>phrase</i>
3) Set SYNSEM LOCAL CAT HEAD to the part of speech type for the rule mother
4) Identify if the rule is unary, head-initial or head-final
5) If rule is head-initial, co-index ARGS FIRST with HD-DTR, else co-index with NON-HD-DTR
6) If rule is head-final, co-index ARGS REST FIRST with NON-HD-DTR, else co-index with HD-DTR
7) Set HD-DTR SYNSEM LOCAL HEAD CAT to the head daughter type
8) Set NON-HD-DTR SYNSEM LOCAL HEAD CAT to the type of the non head daughter

Figure 5 5 gives an example of a completed description, with the co-indexing between the ARGS structure and the daughters indicating that the phrase is head-initial.

Figure 5 5 Sample rule description for CFG rule PP → H IN C NP

[SYNSEM [LOCAL [CAT [HEAD prep *]]]]

HD - DTR [1][SYNSEM [LOCAL [CAT [HEAD prep *]]]]

NON - HD - DTR [2][SYNSEM [LOCAL [CAT [HEAD noun *]]]]

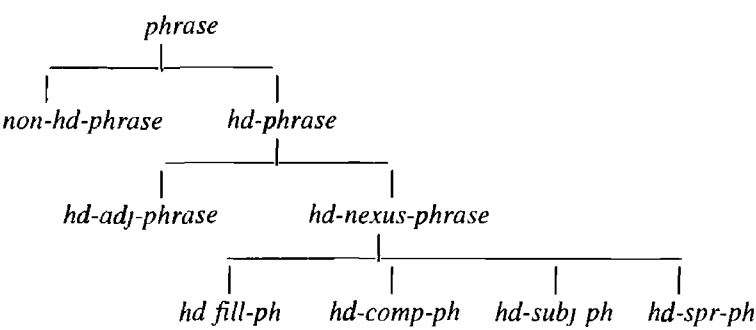
ARGS ([1],[2])

5 2 3 Phrase type mapping

Mapping of categories alone will not establish a real CFG-to-HPSG grammar correspondence, since for this the licensing of phrases and phrase structures in HPSG needs to be considered

As discussed in Chapter 2, HPSG is characterised by its highly under-determined phrase-licensing constructions, whether ID schemata (Pollard & Sag, 1994, Sag & Wasow, 1999), or phrase types (Pollard & Sag, 1994, Chapter 9, Sag, 1997) These result from the factoring out of information across the lexicon, general grammar principles such as head feature and valence, and the ID schemata/phrase types A general “text book” HPSG grammar (Pollard & Sag, 1994) has 4 to 6 main schemas or phrase types, reflecting the major forms of non-head-daughter constituents in phrases, such as adjuncts, fillers, complements, subjects and specifiers, as described in Section 2 2 One example of a top-level phrase type, taken from Sag (1997), is shown in Figure 5 6

Figure 5 6 HPSG phrase types



This form of phrase type hierarchy can be considered a form of X-bar theory, based on saturation of valence features instead of categories and bar levels (Sag, 1997)

The phrase types are distinguished by the non-head daughter with which the head daughter in the phrase combines. For the *hd-comp-ph*, *hd-spr-ph*, and *hd-subj-ph* types, the head daughter valence list feature COMPS, SPR and SUBJ, respectively, selects the non-head daughter. For *hd-adj-ph*, the non-head daughter MOD feature selects the head daughter, and for *hd fill-ph* the SLASH feature on the head daughter selects the non-head daughter.

The LinGO ERG follows (Sag, 1997) in using a large hierarchy of phrases, defining a total of 67 phrase types, but it is still based on the same most general types as in Figure 5.6. The restriction to binary rules discussed in Chapter 2 affects the format of valence feature cancellation, so that each rule/phrase application only reduces a valence list feature by one element, passing on the rest, rather than cancelling out the entire list. Apart from this difference which is easily accounted for, general HPSG phrase type definitions are fully compatible with the LinGO ERG phrase types. The top level LinGO ERG phrase type hierarchy (simplified) is shown in Figure 5.7.

Figure 5.7 LinGO ERG top level phrase types

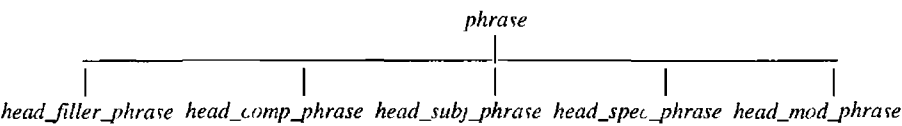


Figure 5.7 shows that the main LinGO ERG phrase types correspond closely to the general HPSG phrase types, discussed as grammar rules or schemas in Section 2.2.3. The head-complement, -specifier, -subject, and -filler schemas all correspond directly to top-level phrase types in the LinGO ERG. The head-adjunct schema is the equivalent of the *head_mod_phrase*.

In order to map CFG rules to HPSG rule schemas, general principles that map the ATIS local trees to LinGO ERG top-level phrase types need to be established. These principles use the head, complement and adjunct marking of the local tree/CFG rule together with local tree category information to suggest a phrase type. Table 5.6 lists the phrase mapping principles used in this dissertation.

Table 5 6 Phrase mapping principles

Principle	Dependency	1 st dtr	2 nd dtr	HPSG Phrase type
Nncompound_ah	A,H	<i>noun</i>	<i>noun</i>	<i>basic_n_n_cmpnd_phr</i>
Hcomp1	H,C			<i>head_comp_phrase</i>
Hadj1	H,A			<i>head_adj_phrase</i>
Hspec1	C,H	<i>det</i>	<i>noun</i>	<i>head_spec_phrase</i>
Hsubj1	C,H	<i>noun</i>	<i>verb</i>	<i>head_subj_phrase</i>
Hspec3	A,H	<i>det</i>	<i>noun</i>	<i>head_spec_phrase</i>
Hadj2	A,H			<i>adj_head_phrase</i>

For each principle, the ‘Dependency’ column describes the binary dependency structure sequence the principle applies to, with ‘H,C’ indicating a head-complement structure, ‘A,H’ indicating an adjunct-head structure, etc. Additional constraints can be specified on the daughter HEAD features in the phrase (in columns three and four of the table) to create more fine-grained phrase distinctions⁵¹ If the principle matches a local tree, it will be assigned a phrase type from the LinGO ERG type hierarchy, as specified in ‘HPSG Phrase type’⁵²

These principles are tested in an order-dependent fashion, from top to bottom, to ensure that the more specific principle applies first⁵³ Note that the adjunct/complement distinction is strictly based on the dependency marking, so that a head-complement schema will only be used if the ATIS pre-processing explicitly identifies a complement

The phrase mapping principles can also be expressed as typed feature structures, which allows them to be tested and applied by means of unification. Figure 5 8 illustrates the typed feature structure format

⁵¹ It is also possible to assign other feature restrictions to further constraining application of the principles. This is currently not required for the principles used for this dissertation.

⁵² The phrase types used in the phrase mapping principles correspond to the top level LinGO ERG phrase types as shown in Figure 5 7 except for *adj_head_phrase* and *head_adj_phrase*, which are head-final and head initial subtypes of *head_mod_phrase* respectively.

⁵³ The principles as currently defined are not exhaustive. They are used in the annotation in conjunction with phrase and rule descriptions, as described in Section 5 2 4. If no mapping principle applies the type will be set to the generic *phrase*. The principles are mutually exclusive as currently defined for this dissertation. This is, however, not a strict requirement.

Figure 5 8 Phrase mapping principle samples in typed feature structure format

Hcomp1

$$\left[\begin{array}{l} \text{head_comp_phrase} \\ \text{HD-DTR} \quad [1] \\ \text{NON-HD-DTR} [2] \\ \text{ARGS} \langle [1], [2] \rangle \end{array} \right]$$

Hspec1

$$\left[\begin{array}{l} \text{head_spec_phrase} \\ \text{HD-DTR} \quad [2] [\text{SYNSEM} [\text{LOCAL} [\text{CAT} [\text{HEAD noun}]]]] \\ \text{NON-HD-DTR} [1] [\text{SYNSEM} [\text{LOCAL} [\text{CAT} [\text{HEAD det}]]]] \\ \text{ARGS} \langle [1], [2] \rangle \end{array} \right]$$

In Figure 5 8 the dependency information is encoded by using the standard LinGO ERG representation of a phrase – where the ARGS feature represents linear precedence – and co-indexing between ARGS values and HD-DTR and NON-HD-DTR specifies if the phrase is head-initial or head-final. For Hspec1 category specifications are marked for each daughter.

Note that these feature structures do not explicitly encode the constraints directly associated with the phrase type they represent, e.g. the non-head daughter being identified with the head daughters' complement, in the case of the *head_comp_phrase*. This is not needed since the phrase type is specified, which will ensure that during annotation and constraint resolution the correct constraints are applied, through type inheritance and inference.

5 2 4 Creating the complete skeletal phrasal sign

When constructing a skeletal phrasal sign, the part-of-speech, phrase description and phrase mapping principle information are all combined

- 1) The phrase description, as defined in Section 5 2 1 above, is determined based on the rule mother category
- 2) The rule description, a feature structure equivalent to a CFG rule, is created as described in Section 5 2 2

- 3) The phrase type is determined based on the CFG rule and the phrase type mapping principles described in Section 5.2.3
- 4) The rule and phrase descriptions and the phrase type are unified to a single typed feature structure

The construction can proceed with one or two of three elements missing (in case they could for whatever reason not be constructed), and the annotation system will then generate a warning, as a failure typically indicates an incomplete mapping or that a HPSG structure cannot be created

5.3 Chapter Summary

In this chapter a grammar correspondence between the CFG grammar used in the basic ATIS annotation after the treebank pre-processing described in Chapter 4, and the LinGO ERG HPSG grammar was established. The part-of-speech tags used in ATIS (from the Penn-II tag-set), were mapped to LinGO ERG-specific lexical types, to allow for the creation of skeletal HPSG lexicon entries. ATIS local trees were mapped to HPSG phrase types with the use of phrasal part-of-speech tags, configurational and dependency information. The mapping made use of

- (i) Phrase descriptions, dependent only on the phrasal tag,
- (ii) Rule descriptions, based on local tree and dependency information, and
- (iii) Phrase type mapping principles, establishing the correspondence with a specific HPSG phrase type

A construction algorithm for CFG-to-HPSG phrase mapping specific to the LinGO ERG, but adaptable to other HPSG grammars, was also presented. This construction algorithm has been implemented in C# as part of the local tree mapping component of the overall annotation method, described in Chapter 6

6 Local tree mapping

With the CFG-to-HPSG mapping established its application to the treebank through local tree mapping can be described. In this component of the annotation process, skeletal lexical and HPSG signs are applied to the ATIS corpus, based on the general mapping principles in Chapter 5 and a number of specific mappings introduced below, by assigning them to each node in the ATIS trees. The local tree for each node is considered as the context for determining the correct mapping.

In the local tree mapping, each terminal and non-terminal node in the ATIS tree is visited, depth-first, left-to-right. While the basic CFG-to-HPSG mapping presented in Chapter 5 establishes the general grammar correspondence, I establish here a number of specific constructions which are handled by LinGO ERG-specific lexical and unary rules and skeletal typed feature structures (or schemas).

The local tree mapping proceeds in three steps:

- (i) Gap handling where traces are identified and the appropriate feature structures assigned
- (ii) Co-ordinate structure handling
- (iii) Remaining structures are mapped, including handling of partitives, raising, etc.

Below we discuss and present the specific mappings. Some of these rely on specific pre-processing of ATIS trees not discussed in the general pre-processing steps presented in Chapter 4.

6.1 Gap handling

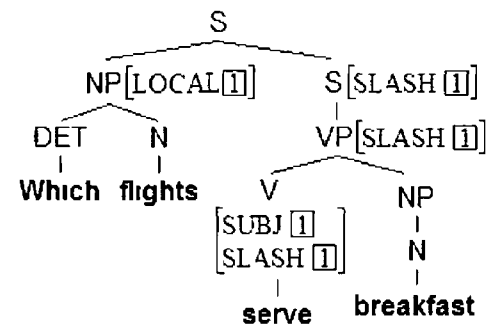
In this step of the automated annotation method, ATIS traces are processed, and trace and co-indexed antecedent information is used to annotate each local tree with typed feature structures specifying the appropriate HPSG slash marking and head-filler phrase types.

6 1 1 Handling of empty categories and traces in ATIS and HPSG

As mentioned in Section 4 1 2, the use of empty categories and traces is only partially reflected in the LinGO ERG grammar In ATIS, following the Penn-II treebank, several different forms of co-indexed null elements are used to mark phenomena such as WH-movement, passive and subjects of infinitival constructions (Marcus *et al* , 1994) For null elements, ‘*T*’ is used to mark WH-movement and topicalisation, and ‘*’ is used for all other null elements Co-indexing is done with an integer, suffixed to the null element and the antecedent non-terminal category WH-prefixed categories always mark WH-movement, SBARQ marks WH-questions, and SQ marks inverted auxiliary constructions Other null elements include ‘*ICH*’ for simple extraposition, and *EXP* for extraposition leaving behind a semantically null ‘it’⁵⁴ For examples of WH-movement and extraposition, see Figures 4 1 and 4 3

The LinGO ERG grammar does not employ null elements and traces in the same fashion as the ATIS trees In HPSG, topicalisation, WH-questions, relative clauses and other unbounded dependencies are analysed as filler-gap constructions, handled by percolating information about ‘missing’ constituents through the tree by means of SLASH, REL and QUE features Figure 6 1 illustrates HPSG unbounded dependency handling with a (simplified) handling of a WH-construction

Figure 6 1 ATIS tree @g07031sx-d-4 illustrating unbounded dependency



⁵⁴ Two other NULL element tags are used in Penn but do not occur in the ATIS corpus They are *PPA*, for ‘permanent predictable ambiguity’, where one cannot tell where a constituent should be attached, and *RNR* for so called right node raising conjunctions (Marcus *et al* (1994)

In Figure 6.1 the subject gap introduced by the SLASH feature on 'serve' is passed up the tree and resolved against the NP 'Which flights' with the use of a head-filler schema. WH-constructions are discussed in more detail in the following section.

The trace information in the ATIS trees helps indicate where a SLASH analysis should be used in the ERG, although not all uses of traces in ATIS correspond to the use of SLASH features in the ERG. For *T* traces, the null element and co-indexation information is used to determine the filler-gap phrase. The lowest node covering the null elements and its co-indexed node is marked in the ATIS tree. The cancellation of the slash feature also needs to be marked on the phrase with the 'missing' constituent. For topicalisations and WH-questions this is the local tree containing the null element.

6.1.2 WH-traces

A WHNP constituent marked with a trace index in an ATIS tree indicates the landing site for a WH-trace. When this is constituent it found, the local tree, which the WHNP is a daughter of, is associated with a HPSG head-filler rule schema. In the case of LinGO ERG the corresponding type is *head_filler_phrase*. This phrase type is combined with the rule and phrase descriptions presented in Chapter 5.

The WH-trace (appearing as an empty category marked with *T* in ATIS) is used to mark the introduction of a gap with the SLASH feature. The actual trace terminal nodes in ATIS are then removed and the gap is marked instead on the head verb of the local constituent, for which the gap fills a subject or complement slot. Figure 6.2, repeated from Figure 4.7, shows an ATIS tree with a WH-trace, and Figure 6.3 illustrates gap marking and removal of empty categories, for the same tree. The gap in the relative clause is marked as a subject gap on the verb 'stop', indicated by the / NP-SUBJ marking, and the node covering the antecedent for the WH-trace is marked with the *head_filler_phrase* type.⁵⁵

⁵⁵ Note that WDT is a collapsed unary production which previously had a WHNP mother category. This information is stored in the tree and is still available for gap handling as discussed in Section 4.2.4.

Figure 6 2 ATIS tree @8kr033sx-d-9 with collapsed unary productions and N-bar level

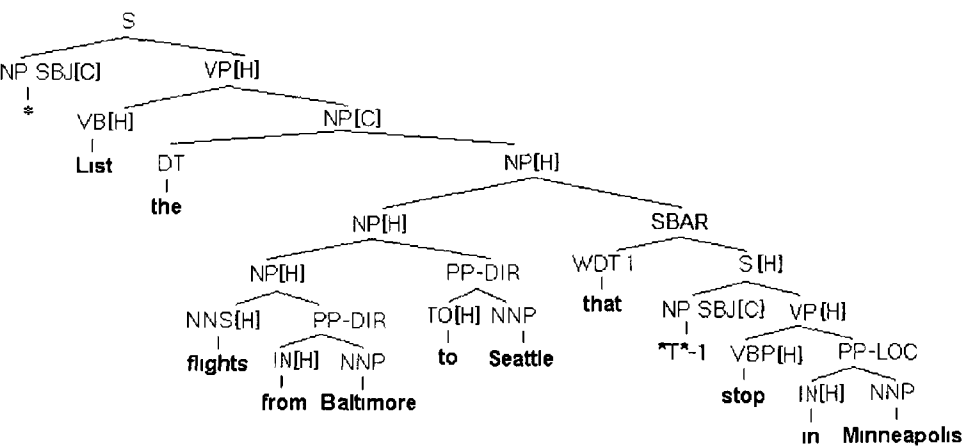
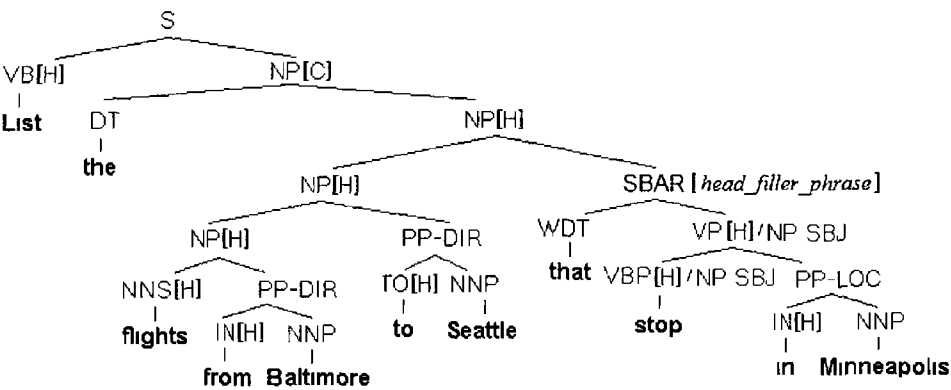
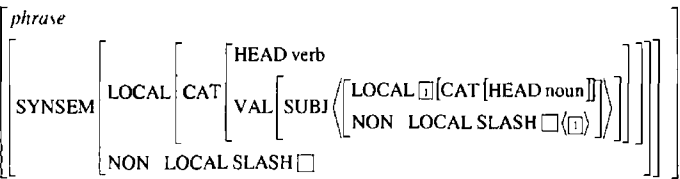


Figure 6 3 ATIS tree @8kr033sx-d-9 illustrating trace removal and gap marking



The local tree configuration is used to determine whether a subject or complement gap should be introduced. A subject gap feature structure is illustrated in Figure 6 4

Figure 6 4 Subject gap feature structure schema



In Figure 6 4, the SUBJ valence feature has a SLASH value, co-indexed with the SLASH value of the phrase, so that it can be percolated up the tree to the landing site. The list-value of the SLASH feature has one element, which is co-indexed to the SUBJ feature value, to restrict the category of the element filling the gap.

6 1 3 Other traces

Other traces – which are used in ATIS to mark passive and control, raising and auxiliary verbs – are not handled with filler-gap constructions in HPSG, so these traces are simply removed from the trees in the treebank.

6 2 Handling co-ordination

Co-ordinate structures in ATIS are detected by identifying whether a local tree has two daughters of the same category, or if one daughter has the part-of-speech category CC. In pre-processing, when the ATIS-tree is binarised, a co-ordinate structure such as in Figure 6 5 is changed to the structure in Figure 6 6, which matches the LinGO ERG co-ordination analysis.

Figure 6 5 Sample ATIS co-ordinate structure

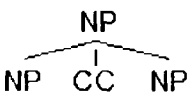
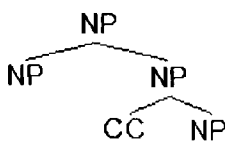


Figure 6 6 Pre-processed co-ordinate structure



The main co-ordination local tree, with two daughters of the same category, is assigned the HPSG co-ordination rule schema, which in the case of LinGO ERG is the phrase type *basic_coord_phr*. The rightmost co-ordination constituent, which includes the CC co-ordination word/tag, is assigned the head marker schema, the *head_marker_phrase* in LinGO ERG.

6 3 Remaining local tree mappings

After filler-gap and co-ordinate structures have been processed, all remaining local trees are associated with HPSG typed feature structure annotations. During this local tree mapping, several additional specific constructions receive special treatment. In the case of control and raising verbs, additional detail is added to the HPSG feature structure descriptions.

In other cases, such as time expressions and partitive nouns, the standard head-complement marking is changed to ensure a LinGO ERG-compatible dependency structure. Lexical and unary rules can also be associated directly with a local tree to ensure correct analysis in the second annotation step of constraint resolution.

6 3 1 Time expressions

Time expressions such as ‘eleven a m’ are analysed as (NP (CD RB)) in ATIS. In LinGO ERG, ‘eleven’ is analysed as complement taking. To match this analysis, the ATIS pre-processing dependency marking is amended to mark the RB as a complement. The alternative, to leave RB unmarked, would make it an adjunct, and would cause other difficulties in the LinGO ERG.

6 3 2 Partitives

Partitive constructions in ATIS include NPs such as ‘All the flights’ and ‘Which of the flights’. In ATIS, in the first example ‘All’ is assigned the part-of-speech tag PDT, followed by an NP. In the second ‘Which’ is a WDT, followed by a PP. In these cases the pre-processing head and complement marking processes are changed, so that PDT/WDT in ATIS is marked as a head, and the sister-constituent as a complement. This is done to ensure compatibility with the LinGO ERG grammar, since LinGO analyses partitives as heads taking the rest of the NP as a complement. The local tree is also marked for application of the relevant partitive unary rule from the LinGO ERG.

6 3 3 Rule application

Subject-aux inversion is identified in ATIS trees if there is a sentence-initial verb and the top category of the tree is SQ. If this is the case, the initial verb is marked for subject-aux inversion lexical rule application.

Trees with a top-level SQ category are marked for application of the yes/no question unary rule at the top-most local tree. Verb-initial trees with the top-level category S are marked for the imperative unary rule at the top-most tree.

6.3.4 Raising and control verbs

Raising verbs include auxiliaries and modals in many analyses. In HPSG raising verbs identify the subject in their complement (embedded) clause with their subject, through co-reference. In the LinGO ERG grammar this identity is further restricted to the content index of the subject of the embedded clause that is ‘raised’ or co-indexed with that of the subject raising verb.⁵⁶ Raising verbs are thus associated with the raising schema shown in Figure 6.7, where the INDEX value of the SUBJ feature, i.e. the subject, is co-indexed with the INDEX value of the COMPS SUBJ feature, i.e. the embedded subject.

Figure 6.7 Raising schema

$$\left[\begin{array}{c} \text{phrase} \\ \left[\begin{array}{c} \text{SYNSEM LOCAL CAT} \\ \text{VAL} \left[\begin{array}{c} \text{SUBJ} \langle \{ \text{LOCAL CONT INDEX } \square \} \rangle \\ \text{COMPS} \langle \{ \text{LOCAL CAT VAL SUBJ} \langle \{ \text{LOCAL CONT INDEX } \square \} \rangle \} \rangle \end{array} \right] \end{array} \right] \end{array} \right]$$

This schema is in fact very close to the general HPSG analysis of control verbs. The main distinction between raising and control verbs is in the assignment of semantic roles, in that all complements of a control verb have a semantic role, and raising verbs always fail to assign a semantic role to one of their complements. Semantics is outside the scope of this dissertation and maintaining a high-level identity in the complement structure for control and raising verbs is beneficial for the HPSG-annotation method. It means that the two types of verb do not need to be distinguished in the initial local tree mapping and high-level HPSG schema assignment.

⁵⁶ The motivation for this is to exclude raising of SLASH CASE and CONTENT handle features since this will cause unwanted side effects such as CASE information clashing between subjects and objects and specifically to the MRS semantics of the LinGO ERG grammar, the semantic handles of the two verbs being undesirably identified.

In ATIS, modal verbs are tagged with the ‘MD’ part-of-speech tag, which makes it straightforward to identify them. Control and raising verbs can be identified through the use of the infinitival ‘to’, and assigned the general schema in Figure 6.5, which is applicable for both raising and control in the LinGO ERG grammar.

6.4 Chapter Summary

This chapter has presented local tree mapping. In this step the ATIS CFG mapping to HPSG and LinGO ERG types, established in Chapter 5, is automatically applied to each local tree. Tree structure information, specific tags and particular local and non-local tree configurations are used to guide the assignment of specific HPSG feature structure and rule specifications to each local tree. Traces and indices are used to guide marking of filler-gap dependencies and phrases. Conjunction tags are used to mark co-ordinate phrases, and infinitival ‘to’ to mark raising/control verbs. In addition, various particular constructions such as partitives, time-expressions, subject aux-inversion, imperatives and yes/no questions are identified. For these, each local tree is transformed, through changing of tag or dependency marking, or marked for LinGO ERG rule application. In this manner an HPSG lexical or phrasal schema or type is automatically determined for each local tree, so that the appropriate analysis can be applied in the next annotation step, namely constraint resolution. This step is described in Chapter 7.

7 Constraint resolution

In the last component of the annotation method, the mapped feature structures associated with all local trees are combined, using a constraint solver. The result is one or more resolved typed feature structures, with HPSG phrasal signs representing the complete tree.

In the constraint solver the LinGO ERG HPSG grammar rules are applied to the annotation process, with the mapped local tree feature structures established in local tree mapping acting as rule filters. This can be viewed as a highly restricted form of local tree parsing, guided and constrained by the original treebank tree structure.

Section 7.1 first describes the basic constraint solver algorithm for combining mapped local tree feature structure annotations, including lexical resolution and rule application. The ability of the algorithm to operate in different modes, to cope with grammars of different complexity with or without lookup against a pre-defined lexicon, is discussed in Section 7.2. Section 7.3 introduces robustness to the annotation algorithm through fall-back strategies for coping with issues in lexicon and grammar rule lookup, or failures of rule application.

7.1 *Resolving the local trees – the basic algorithm*

The constraint solver attempts to combine the mapped HPSG feature structures for each local tree. It performs a recursive traversal of the treebank tree top-down, performing lexical resolution when it reaches a terminal node, as defined in Section 7.1.1. Non-terminal trees are combined through unification, with the application of applicable grammar rules, while returning to the top local tree in the recursion. Typed feature structure unification is computed with the grammar and typed feature structure processing system developed for this dissertation, as described in Chapter 3.

I will refer to the skeletal feature structures associated with each local tree before the constraint resolution starts as “mapped”, as they were assigned in the first annotation step during the local tree mapping. Typed feature structures resulting from constraint resolution at the level of each local tree, including terminal trees, I will refer to as “resolved feature structures”.

These are processed further by the algorithm to construct a larger resolved phrasal sign, unless an annotation failure has occurred and a resolved feature structure cannot be constructed for the local tree

The constraint solver algorithm is presented in Figure 7.1. The lexical resolution for terminal nodes is described in Section 7.1.1. Lexical and phrasal rule application is described further in Section 7.1.2.

Figure 7.1 Basic constraint solver algorithm

<p>Starting with the topmost local tree, for each local tree</p> <ol style="list-style-type: none"> 1 If the tree is a terminal tree <ol style="list-style-type: none"> 1.1 Collect lexical feature structure descriptions – from the mapped feature structure or lexicon lookup 1.2 If lexical lookup is used, filter the lexical items against the mapped typed feature structure for the local tree by unification 1.3 If no lexical lookup is used, use only mapped feature structure information 1.4 Optionally apply lexical and unary rules 1.5 Return as resolved feature structures all lexical feature structures and results of rule application 2 Else if tree is not terminal <ol style="list-style-type: none"> 2.1 For each daughter, recursively call the constraint solver algorithm to process the daughter local trees and collect their resolved feature structures 2.2 Retrieve applicable grammar rules based on the mapped feature structure associated with the current local tree 2.3 Apply applicable grammar rules to the set of daughter feature structures 2.4 Collect all successfully unified structures and return as resolved feature structures 3 If current tree is the top tree, the set of resolved feature structures is the set of HPSG analyses of the entire tree
--

7.1.1 Lexical resolution

The constraint solver algorithm retrieves lexical information associated with the words at the terminal nodes of the tree. HPSG is a strongly lexicalised grammar theory and rich lexical information – in particular regarding subcategorisation/valence requirements for each word – drives the grammatical analysis and rule selection for the construction of phrasal signs.

Lexical information for the constraint solver comes from two sources. First, the skeletal lexical sign is assigned in local tree mapping as described in Chapter 6. This is associated with each local (terminal) tree and can serve as a starting point for the constraint solver on its own. Secondly, a pre-defined lexicon associated with the LinGO ERG HPSG grammar used for the annotation can optionally be used for lexical lookup. If a pre-defined lexicon is used for each terminal node, the word (or phrase) associated with the node is used to look up the associated lexical item. The mapped feature structure for the terminal node that contains part of speech and other information derived from the original ATIS tree, is used to filter retrieved lexical items.

If a lexicon is not used, the mapped feature structure is used as the primary (potentially underspecified) source of lexical information and the starting point for the constraint solver. Section 7.2 discusses different modes of lexicon-based and lexicon-less resolution.

7.1.2 Rule application

The constraint solver can use several different rule application strategies depending on the grammar and lexicon used. The full LinGO ERG grammar makes use of morphological and non-morphological lexical rules as well as unary and binary phrase rules, while simpler HPSG grammars may only use binary grammar rules.

7.1.2.1 Lexical rule application

If an existing lexicon resource such as the LinGO ERG lexicon is used in the annotation, the corresponding lexical rules should be applied.

The C# system built for this dissertation performs an off-line full-form expansion of the LinGO ERG lexicon, to ensure that at run-time all word-forms will be found in the lexicon. The full-form expansion creates a look-up table of surface forms, pointing to the lexical entry and the morphological rule that needs to be applied to create the surface form. The morphological rule is applied in the lexicon lookup, together with lexical item expansion and type inference.

During the application of the automatic annotation algorithm, lexical rules can be applied to all lexical entries that pass the mapped feature structure filter for the terminal nodes.

That is, those lexical entries that are compatible and unify with the part-of-speech-based feature structure for the terminal node. LinGO ERG lexical rules are designed to apply recursively,⁵⁷ and a cut-off of five iterations is used as the default here to ensure there is no infinite regression in the rule application.

The LinGO ERG lexical rules are specific to the LinGO grammar and lexicon and are not necessarily applicable to other HPSG grammars. Therefore, the lexical rule application is optional and can be switched off. If the LinGO ERG grammar is used without its lexicon, lexical rules are typically not used. The exception is for those lexical rules that were specifically assigned to terminal nodes in the local tree mapping to handle particular constructions such as passive and subject-aux inversion.

7.1.2.2 Phrasal rule application

Apart from lexical rules, HPSG grammars can contain unary and non-unary grammar rules for the creation of phrasal signs. The LinGO ERG grammar uses only unary and binary branching rules. Unary rules are discussed in the following section. The main HPSG phrase structure types as used in the LinGO ERG grammar and described in Chapter 2, are all binary and combine two daughter constituents to form a larger phrase. These rules are applied in the constraint solver algorithm when processing a non-terminal local tree where the daughters have already been constraint-resolved.⁵⁸

Rule application has two stages: rule lookup and rule application proper. In rule lookup, appropriate grammar rules are selected for application to the local tree, based on the local tree mapping. The local tree mapping is used as a filter to constrain the set of applicable rules, by selecting the rules subsumed by the skeletal phrasal sign, constructed as described in Section 5.2.4. In rule application proper, the applicable rules are applied to the set of daughter feature structure pairs created by pairing each of the first daughter parses with each of the second daughter parses. The resolved feature structure(s) resulting from the rule application are also returned for further rule application at the level of the parent of the local tree if there is one.

⁵⁷ In each recursive iteration the rules are applied to the output of the previous iteration, until no further rules apply or a cut off is reached.

⁵⁸ In the LinGO ERG grammar, the HPSG general principles as described in Chapter 2 are incorporated into the (phrasal sign) type system and thus integrated into the grammar rules themselves. They do not, therefore, need to be applied separately.

7.1.2.3 Unary rule application

Unary rules, used in LinGO ERG for creating an imperative sentence or question from a verb phrase, can be applied when processing both terminal nodes and non-terminal local trees. They apply in addition to lexical and binary rules and can also apply recursively like the lexical rules. The results of unary rule applications are added to the set of results of binary or lexical rule applications and the combined results are returned for further processing.

When using the full LinGO ERG grammar, unary rules need to be applied for each local tree to obtain correct parses. It is also essential that well-formed typed feature structure unification, as defined in Chapter 3, is used to ensure correct rule application. For other grammars this may not be required and type inference can be switched off as required.

7.2 Modes of operation

The treebank annotation method developed in this dissertation, while optimised for use together with the LinGO ERG HPSG grammar, is designed to be flexible. It can operate with HPSG grammars of varying complexity, from minimal grammars with only a few schemas and no lexicon, to large grammars like the LinGO ERG grammar. This will be demonstrated in Chapter 8. The constraint solver algorithm is configurable with respect to lexical resolution, type inference and rule application, in order to enable the optimal use of the rule and lexicon resources of a particular grammar.

The constraint solver has three main modes of operation, as follows⁵⁹

- (i) minimal grammar use without lexical lookup,
- (ii) full lexical lookup, full grammar use,
- (iii) mixed lexical lookup, full grammar use

These are described below.

⁵⁹ The implementation of the annotation system and constraint system has a substantial number of configurable parameters allowing considerable fine tuning. Parameters can be set to configure the use of the lexicon, switch on and off lexical and unary rules, and control the fall back functionality discussed in Section 7.3. However, the modes described here are the most significant for this dissertation.

7 2 1 Mode 1 Minimal grammar with no pre-defined lexicon

The first mode is one of minimal HPSG grammar and lexicon use, with no predefined lexicon apart from what can be derived from local tree mapping and a minimal HPSG rule-set such as the main rule-schemas as presented in Section 2.2. In this mode the feature structure information from the local tree mapping is used directly in deriving lexical entries. In the simplest grammar model, no lexical or unary rules are used, and only binary grammar rules are applied. This mode of operation is essentially the same as the HPSG treebank annotation of (Miyao *et al* , 2004) as described in Chapter 2. It is primarily a reference mode to validate the annotation system.

7 2 2 Mode 2 Full grammar and lexicon use

This mode is used when applying the complete LinGO ERG grammar and lexicon to annotate the treebank. Complete lexicon coverage is assumed and full lexicon lookup and lexical rule application is performed for terminal trees. The full LinGO ERG rule-set is used and unary rules are applied after both lexical and grammar rules. Well-formed typed feature structure unification is used throughout.

This mode is also primarily used as a reference mode for this dissertation – to validate the annotation and compare it with LinGO ERG parses of the bare strings in the treebank. Typically, a complete lexicon cannot be assumed in parsing real world unrestricted text, and for this mode an extended LinGO ERG lexicon including all words in the ATIS corpus has been used as described in Chapter 3.

7 2 3 Mode 3 Mixed lexicon lookup

In mixed lexicon lookup, the lexicon associated with an HPSG grammar is used for some lookup, together with mapped lexical signs from the local tree mapping. The lexicon use can be configured so that particular word classes or specific words can be selected for inclusion or exclusion from lexical lookup. This allows extensive predefined lexicons to be used, e.g. for closed word classes such as prepositions and auxiliaries. Also, in case of lookup failure the mapped lexical sign can automatically be used.

This mode is useful both for lexical acquisition when identifying new lexical items in a corpus, and also to validate the annotation process where mapped entries can be compared with predefined lexicon items

As will be discussed in Section 8.2, this mode is used specifically for mixed mode lookup with the LinGO ERG grammar in analysing the annotation of nouns and main verbs excluded from lexical lookup

7.3 Robust annotation and fall-back strategies

The algorithm described so far is essentially non-robust. It works well as long as there are no annotation failures, and lexicon and rule lookups succeed and produce resolutions for further processing. Unfortunately, this cannot be expected to always be the case, as limited lexicon coverage may result in failure in the lexicon lookup stage, or rule application may fail during the traversal of the local trees. To cope with this and still produce a result, fall-back strategies are required in lexical lookup and rule application

7.3.1 Lexical lookup fall-back

If lexical lookup is used, the lexicon may turn out to be incomplete. If the lexicon lookup fails, either through not having the required lexical entries or due to retrieved lexical items not being compatible with the mapped feature structures, the basic fall-back is to use the mapped feature structures only. These are likely to be less detailed than lexical entries from a pre-defined lexicon but annotation can proceed even though rule application may be affected, as discussed in Section 7.3.4

7.3.2 Rule application fall-back

In rule application, several fall-back options are possible. Lexical and unary rule application is inherently fail-safe since these processes only ever add information to existing resolved lexical or phrasal feature structures resulting from lexical lookup and binary rule application

If a lexical or unary rule application fails, we will always have the resolved features structures left to proceed with. Therefore, failures in applying lexical or unary rules do not require special fall-back strategies⁶⁰

For (binary) grammar rules, rule application can fail in rule lookup or rule application proper. In rule lookup, grammar rules are filtered against the mapped feature structure for the local tree. If no rule passes the filter, a second, and more general rule-set can be defined and used as a fall-back. For example, for the LinGO ERG, in addition to the main grammar rule-set of 67 rules, a smaller more generic fall-back rule-set corresponding to the main HPSG phrase types can be used. In rule application proper, failures can occur in unification. A fall-back rule-set can be used here also.

If all rule applications fail, the mapped feature structure for the local tree can be used and combined (unified) with the resolved feature structures for the daughters. In a worst-case scenario, where this unification also fails, the mapped feature structure for the local tree can be returned on its own. This means the daughter feature structures will be lost. The rule fall-back strategy is summarised in Figure 7.2.

Figure 7.2 Rule application fall-back

- | |
|---|
| <ol style="list-style-type: none">1 If rule lookup fails, use a secondary more general rule-set if available2 If rule application fails, use a secondary more general rule-set if available and not already used3 If rule lookup or rule application still fails, use the mapped feature structure for the local tree and combine it with the resolved daughter structures through unification4 If combination of mapped feature structure with daughters fails, return only the mapped structure for the local tree |
|---|

7.3.3 Principled fall-back: the robust constraint solver algorithm

To control the annotation algorithm and manage the impact of fall-back, a global fall-back level counter is maintained in the constraint solver. The starting fall-back level is 0 for normal annotation.

⁶⁰ The failure of a lexical or unary rule application can, however, lead to the failure of a grammar rule application later in the constraint resolution. This is especially the case with the LinGO ERG grammar. Detecting these cases is difficult and handling them is outside the scope of this dissertation.

If any failure and fall-back occurs, the fall-back level is set to 1. At fall-back level 1 the algorithm's default behaviour will be modified with respect to rule application and lexical and unary rules will not be applied. Also, if defined, a secondary reduced and more general rule-set will be used. If all grammar rule application fails, the fall-back level will be increased to 2, which means that only the local mappings will be used, and no further rule application will be attempted.

The fall-back levels are used in the robust version of the annotation algorithm as defined in Figure 7.3. New steps compared to the initial algorithm in Figure 7.1 are added in bold.

Figure 7.3 Robust constraint solver algorithm with fall-back levels

<p>Starting with the topmost local tree, for each local tree</p> <ol style="list-style-type: none"> 1 If the tree is a terminal tree <ol style="list-style-type: none"> 1.1 Collect lexical feature structure descriptions from the mapped feature structure or lexicon lookup 1.2 If lexical lookup is used, filter the lexical items against the mapped typed feature structure for the local tree, by unification <ol style="list-style-type: none"> 1.2.1 If lexical lookup is used and no words are found, or if no lexical items pass through the filter, fall back to mapped lexical feature structure and set fall-back level to 1 1.3 If fall-back is 0, optionally apply lexical and unary rules 1.4 Return as resolved feature structures all lexical feature structures and results of rule application 2 Else if tree is not terminal <ol style="list-style-type: none"> 2.1 For each daughter, recursively call the constraint solver algorithm to process the daughter local trees and collect their resolved feature structures 2.2 Retrieve applicable grammar rules based on the mapped feature structure associated with the current local tree <ol style="list-style-type: none"> 2.2.1 If fall-back level is 1, use reduced rule-set if defined 2.2.2 If fall-back is 0 and no rules are retrieved, use reduced rule-set if defined and set fall-back to 1 2.3 Apply applicable grammar rules to the set of daughter feature structures 2.4 Collect all successfully unified structures and return as resolved feature structures <ol style="list-style-type: none"> 2.4.1 If no rules are found or rule application fails, set fall-back level to 1 and combine mapped feature structure with daughters 2.4.2 If mapped combination fails, set fall-back level to 2 and return only the mapped structure for the local tree 3 If current tree is the top tree, the set of resolved feature structures is the set of HPSG analyses of the entire tree

7 3 4 Considerations for optimised fall-back

The fall-back strategies described in the previous sections work well in ensuring a basic level of robustness⁶¹ They can handle multiple failures while still ensuring that at least one feature structure is created However, in some cases the basic fall-back strategies are not optimal and more fine-tuned and dynamic fall-back is needed to achieve better results

If the fall-back level is increased as a result of a lexical lookup or rule-application failure, a more general rule-set may be used However, if some constituents have already been resolved with the use of fine-grained lexical and unary rules, these will not be further constrained and will in fact increase the number of resolutions at each local tree resolution stage The result will be a large number of ‘over-specific’ resolutions when specific and general constituents are combined In this scenario it is preferable to produce a smaller number of more underspecified annotations

In another scenario an increase in fall-back level due to a failure in lexicon-lookup may lead to an unnecessarily high-level rule application and resolution as the mapped lexical sign may be detailed enough to constrain a larger grammar

One way to address these issues is to introduce a dynamic fall-back mechanism that is sensitive to how far the constraint solver has progressed through the local trees, as well as to which fall-back level is the most appropriate at any given stage In some cases, determining the most appropriate fall-back may require re-analysing already processed local trees In others it may require testing whether *fall-back* needs to be done at all Implementing dynamic fall-back may, therefore, require some major changes to the constraint-resolution algorithm

⁶¹ It is unfortunately beyond the scope of this dissertation to discuss robustness in HPSG in general See (Vogel & Cooper 1994 Steiner & Tsujii 1999a 1999b Yutaka *et al* 1998 Wahlster 2000 Fouvry 2003a) for some work in this area

A simple measure to avoid overgeneration with the rules is to only apply them if they are less than or equally specific as the *feature structures* they apply to – if the rule daughter *synsem* type subsumes the *synsem* type of the rule inputs. If the lexical signs can also be made more specific or if at least partial use of lexical lookup can be made, ambiguity should also be constrained and more rules can be applied.

For this dissertation dynamic fall-back is not essential and has not been implemented.

7.3.5 Partial tree resolution

A further, last-resort fall-back strategy if rule application fails is to return multiple phrasal sign fragments covering parts of the tree, rather than a single connected sign for the entire tree. This functionality is not enabled in the current implementation of the annotation method as it is not essential for this dissertation, but would be straightforward to add. The constraint solver creates successively larger HPSG signs as it progresses, and the constituent signs constructed up to the point of a rule application failure can be tracked and returned.

7.4 Chapter Summary

In this chapter the constraint solving algorithm for combining HPSG annotations into a single phrasal sign has been presented. The constraint solver takes skeletal or mapped HPSG typed feature structures and creates one or more resolved phrasal signs for the complete treebank tree.

The constraint solver performs optional lexical lookup, applies lexical rules and combines lexical items and phrases into larger phrases through rule application. The solver can operate in three different modes – with no, full, or partial lexical lookup – and different levels of rule application. The robustness of the constraint solver has also been discussed, together with the principal use of fall-back strategies for coping with lexical lookup and rule application failures.

8 Experiments and Evaluation

With the treebank pre-processing and annotation method described in Chapters 3 to 7, I am now in a position to explore the goals of this dissertation. In this chapter I use the annotation method to perform three experiments, to investigate the potential of the system.

The main goals are to

- (i) annotate a CFG-treebank with HPSG typed feature structures,
- (ii) exploit the rich linguistic information in an existing wide-coverage HPSG grammar to improve treebank annotation,
- (iii) use the treebank to improve the coverage of the HPSG grammar.

The experiments are designed to examine each of the goals, and their results are used to evaluate to what extent the goals have been reached. Results and assessment are described in the following sections.

8.1 Experiment 1: basic HPSG-based treebank annotation

The first experiment is designed to test the ability of the overall system to annotate the ATIS treebank and handle the specific linguistic phenomena in this corpus. The purpose of this experiment is to determine if the first goal of the dissertation has been met. For this experiment a subset of the LinGO ERG grammar is used without the lexicon.

8.1.1 Using a minimal LinGO ERG grammar

The grammar used in Experiment 1 is a minimal subset of the LinGO ERG grammar. It uses the five main rule schemas, namely head complement, subject, specifier, adjunct and filler, together with a co-ordination and head marker schema. The grammar is comparable to the HPSG grammar used by (Miyao *et al.*, 2004), with regard to rule schemas. It also includes a small number of lexical and unary rules for imperatives and yes/no questions (which are common constructions in ATIS) and subject-aux inversion, which is triggered configurationally by a sentence-initial verb and passive and partitive constructions. The LinGO ERG lexicon is not used, but instead lexicon entries are automatically created from the skeletal lexical HPSG signs, as defined in Chapter 5.

8 1 2 Results

Applying the annotation system with the minimal LinGO ERG grammar to the ATIS treebank yields the results in Table 8 1

Table 8 1 ATIS annotation with minimal LinGO ERG grammar and no lexicon

Trees	Total	Annotated	%	Resolved	%
¬FRAG	349	348	99 7%	332	95 3%
FRAG	228	228	100%	226	99 1%
Total	577	576	99 8%	558	96 7%

As for Table 4 1, FRAG indicates that the utterance has been analysed in ATIS as a non-sentence fragment, and ¬FRAG indicates complete sentences The table shows the number of trees that were annotated, i e assigned at least one phrasal sign covering the whole tree, and the number of trees that were resolved, i e where the constraint resolution and rule application completed successfully without any fall-back Annotation coverage of the corpus is almost complete at 99 8%, with only one out of 577 trees failing to be completely annotated The resolution rate for successful rule application is somewhat lower, but at 95 3% for non-fragment trees, it is significantly higher than the LinGO ERG parse success rate for the same sentences in using the complete grammar, namely 79 8% as shown in Table 4 1 ⁶² This shows that the general HPSG annotation system performs better than the full LinGO ERG grammar when parsing the raw strings in the corpus, and validates the robustness and coverage of the system with a basic HPSG grammar Coverage should improve with fine-tuning of the annotation method

8.2 Experiment 2 Annotating with a wide-coverage grammar

The second experiment is intended to show the merits of using a wide-coverage HPSG grammar for treebank annotation Full use is made of the LinGO ERG lexical, unary and binary rules, and the LinGO ERG lexicon is also used for partial lexical lookup

⁶² The totals for trees fragments and non fragments differ slightly between Tables 4 1 and 8 1, due to a small number of trees which were split and treated separately for the LinGO parse exercise See Section 4 1 for further detail

8 2 1 Taking advantage of the LinGO ERG grammar and lexicon

If the LinGO ERG grammar is used as in Experiment 1, with a small subset of rules, no lexical lookup, and with only a basic part of speech to HPSG type mapping (as in Table 5 1) it does not yield significantly better results than a basic HPSG grammar in terms of annotation and details of lexical entries

The main reason for this is that the lexical information used is the same for the two grammars. The grammar rule system on its own, particularly with a small subset of the LinGO ERG as in Experiment 1, does not add significantly to the annotation detail over a basic grammar.

Using the full LinGO rule-set without the associated lexicon, or at least highly constrained lexical entries is not practical, since it will lead to massive ambiguity in the rule application process, as discussed in Section 7 3 4, and also noted by (Fouvry, 2003b). The main potential of the LinGO ERG grammar for treebank annotation resides in the lexical type system, and if the lexical mapping can be made specific enough, with the use of some lexical lookup, the quality and detail of annotation should be substantially improved. The extensive rule component will also assist in improving the analysis.

8 2 2 Optimised use of lexical resources with the LinGO ERG

In this experiment, full use is made of the LinGO ERG-specific lexical mapping from Chapter 5, including argument structure and morphological information from part-of-speech tags. This allows noun and main verb lexical mapping to be constrained to a level detailed enough to be comparable to that of predefined LinGO ERG lexical entries. Mixed mode lexical lookup is used, where other word classes are looked up in the pre-defined lexicon. Lexical lookup is also specified for a small number of nouns and verbs, such as time words and non-modal auxiliaries, that are difficult to identify based on treebank information only. This use of lexical resources also allows the full-rule component to be used with a low risk of overgeneration.

8 2 3 Extracting lexicon entries

A good test for the HPSG annotation method is to what extent it can be used to extract rich lexical descriptions from the annotated corpus. Modern linguistic theories such as LFG, HPSG, and LTAG are heavily lexicalised, and development of lexicons tends to be the bigger bottleneck in grammar development, rather than the rule component.

The quality of the lexical descriptions the annotation system can produce is therefore a good measurement of its usefulness, and lexicon entry generation serves as a good benchmark to assess the value of using a wide-coverage grammar to support the HPSG annotation system.⁶³

Our focus in assessing merits of the annotation method for creating lexical entries is related to the use of the LinGO ERG grammar. Given that the LinGO ERG grammar already defines very rich lexical forms, one measure of the annotation method is to see how close it comes to the detail of the predefined lexical entries. When using the annotation method with the full LinGO ERG grammar in mixed mode lookup as described in Section 8 2 2, I will also have an opportunity to judge the annotations for nouns and verbs against the corresponding predefined lexical entries.

With the lexical type system used in the LinGO ERG lexicon also being used for the annotation method lexical mapping, lexical information for the annotated trees can be directly compared using the specified lexical sign type. I also follow (Fouvry, 2003b) in using unfilling (Gotz, 1994) to remove features from the lexical entries that can be inferred from the type hierarchy, see Section 2 4 for further details.

8 2 4 Results

The annotation method was tested on the ATIS corpus with the full LinGO ERG grammar and lexicon in mixed lookup mode, using mapped lexical items for main verbs and nouns. The annotation results are summarised in Table 8 2.

⁶³ Related work on HPSG lexicon extraction and handling of unknown words is reported in Section 2 4. Other work on lexicon extraction from treebanks is covered in Section 2 1.

Table 8 2 ATIS annotation with full LinGO ERG grammar and mixed lexical lookup

Trees	Total	Resolved	%	Avg resolutions	Parsed LKB not resolved	Resolved not parsed LKB
¬FRAG	349	240	68 8%	2 4	58	23
FRAG	228	93	40 8%	3 9	6	90
Total	577	333	57 7%	2 8	64	113

Column 3 ‘Resolved’ shows the number of trees for which constraint resolution and rule application was successful. Column 6, ‘Parsed LKB not resolved’, indicates how many bare strings in the ATIS corpus were parsed by the LKB using the LinGO ERG grammar but not resolved by the annotation algorithm. Column 7, ‘resolved not parsed LKB’, conversely shows the number of trees resolved by the annotation method for which the LKB system could not produce a parse.

The resolution rate both in total and particularly for non-fragment trees is lower than when using a minimal grammar as in Experiment 1,⁶⁴ as can be expected when using a larger more restrictive grammar with more highly restricted lexical lookup and mapping.

The interesting numbers with which to compare these results are those of the LKB system parsing the raw sentences from the corpus with the same LinGO ERG grammar, as shown in Table 4 1. The overall resolution rate for the corpus at 57 7% is higher than the LKB parse coverage, at 50 3%.⁶⁵ For non-fragment sentences the LKB system achieves about 10% better results, at 79 8% coverage compared to 68 8% for the annotation system.

There are two main factors influencing the resolution rate of the annotation system, mapping method completeness and grammar coverage of the treebank tree structures. With regard to mapping coverage, the lexical and local tree mapping described in Chapters 5 and 6 is still not complete for use with the full LinGO ERG grammar, even though it achieves good results with a more basic grammar, as shown in Experiment 1. There are still constructions that need special handling before constraint resolution.

⁶⁴ Annotation rate was not measured in this experiment.

⁶⁵ For the purposes of this dissertation only parse coverage was measured. Parse validity was not examined.

This mapping incompleteness is likely to account for a significant portion of the 10% difference in coverage between the annotation system and the LKB parse results. Given the specialised nature of the LinGO ERG grammar, where for example time expressions involving hours, the time of day, and weekdays all receive specific lexical items and also are handled by some specific rules, it is not unexpected that a generalised phrasal and lexical mapping will cause rule application failures for these expressions. Some of these issues can be straightforwardly addressed by including some well-defined classes of words on the lexical lookup inclusion list. There may also be changes and additions required to other portions of the annotation method. As for Experiment 1, coverage should improve with fine-tuning of the annotation method.

The other factor influencing annotation coverage is the level of structural correspondence between ATIS trees and parses licensed by the LinGO ERG grammar. While the pre-processing and local tree mapping operations transform the treebank trees to be as compatible in structure as possible to the LinGO ERG grammar, there may be cases where the structural difference is simply too great. There may not be any analysis licensed by the grammar that matches the tree. Since I treat the treebank tree as the ‘gold standard’ this means that grammar coverage is not complete for a particular syntactic structure in the tree. This is in fact exactly the kind of grammar incompleteness that we would like the annotation system to identify, as one of the goals of the treebank annotation system is to assist in increasing grammar coverage. We should not, therefore, expect the annotation system to resolve exactly the same trees as the LKB parser, using the same grammar.⁶⁶

In order to properly identify and handle cases of grammar incompleteness in the annotation method we need to (i) be able to rely on a complete mapping, and (ii) have a fall-back method to allow an annotation to still be produced even if grammar rule application fails.

⁶⁶ The LKB may successfully parse a terminal string using the LinGO ERG for which the corresponding tree could not be annotated with the same grammar. It may not, however, assign it the same structure as the tree. This accounts for why LKB parse results could be higher than annotation method resolution coverage.

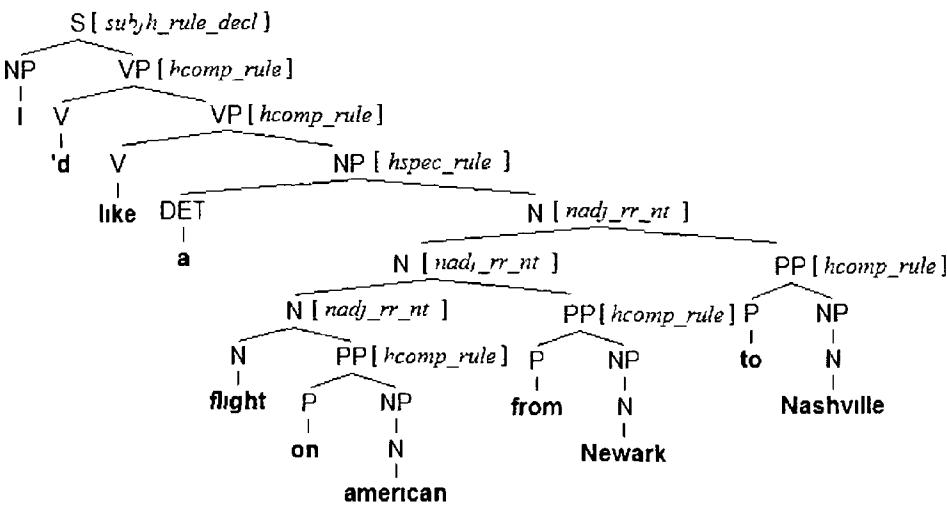
While incomplete, the annotation system is already showing some promise here by its ability to annotate 23 trees that could not be parsed by the LKB system⁶⁷ Due principally to the incompleteness of the annotation system, the results in Table 8 2 are therefore to be regarded as preliminary There are, however, some additional noteworthy results

The average number of resolutions for non-fragment trees is low at 2 4 This indicates the potential of the annotation method to be used for parse ranking, as discussed in Section 2 3 2 on LinGO Redwoods Also, as mentioned, the results listed in the last column, ‘Resolved not parsed LKB’, where the annotation method finds complete resolutions for 23 trees for which the LKB system cannot parse the raw string, indicates that the annotation method has the potential to expand *grammar coverage*

An example of an ATIS tree string not parsed by the LKB⁶⁸ but for which the tree is successfully annotated is shown in Figure 8 1

Figure 8 1 ATIS tree (@818013sx-a-10) annotated and not parsed by LinGO ERG

I'd like a flight on American from Newark to Nashville



⁶⁷ These annotations have been manually inspected 10 have a lexical mapping issue where an imperative verb has been given the wrong argument structure 11 are OK and 2 have incorrect rule assignment

⁶⁸ The cause of the LKB parse failure appears to be an inability to handle the PP ‘on American’

In Figure 8.1, the ATIS tree has been annotated with the successful rule-application from the LinGO ERG grammar. The *hcomp* and *subjh* rules are similar to the general head subject and head complement schemas, and the '*nadj_rr_nt*' is one of the specialized adjunct rules in LinGO.

8.3 Experiment 3 Improving grammar and lexicon coverage

In our third experiment to explore the potential of our annotation method, I investigate how it can be used to extend the coverage of wide-coverage grammars.⁶⁹

8.3.1 Results

Experiment 2 shows the ability of the annotation method to compensate for lack of lexical coverage with respect to main verbs and nouns. While the overall resolution rate is low, results are still notable considering that they are achieved without a lexicon for important content words and that the LinGO ERG grammar is heavily reliant on tight lexicon and grammar interaction. As discussed above, the demonstrated ability of the annotation method to provide resolved annotation for trees that could not be parsed also shows the capability for improving grammar/lexicon coverage.

Our use of the treebank structure to constrain the mapped lexical signs also addresses the issue encountered by (Fouvry, 2003b) in handling unknown words and creating new lexical entries with the LinGO ERG grammar: mapping a small general part-of-speech tag set to a large set of specific LinGO ERG lexical types, as discussed in Section 2.4.⁷⁰

A second aspect of lexicon extension is for missing forms of words that may be in the lexicon in some other form. This is less easily identified since failures will appear only in rule-application, and it may be difficult to distinguish from rule under-generation.

⁶⁹ Apart from missing lexicon entries, lack of rule coverage is the other major source of grammar incompleteness. The annotation method has a fall back mechanism to handle rule application failures, as discussed in Section 7.3, which can make use of more general fall back rules and also of the phrase mapping from Chapters 5 and 6. However, the method does not directly suggest new rules to be added to a grammar, but instead identifies which constructions (tree configurations) cause difficulties and the annotated structures to represent them. Deriving specific grammar rules, in particular for a grammar as rich and complex as the LinGO ERG grammar, is beyond the scope of this dissertation.

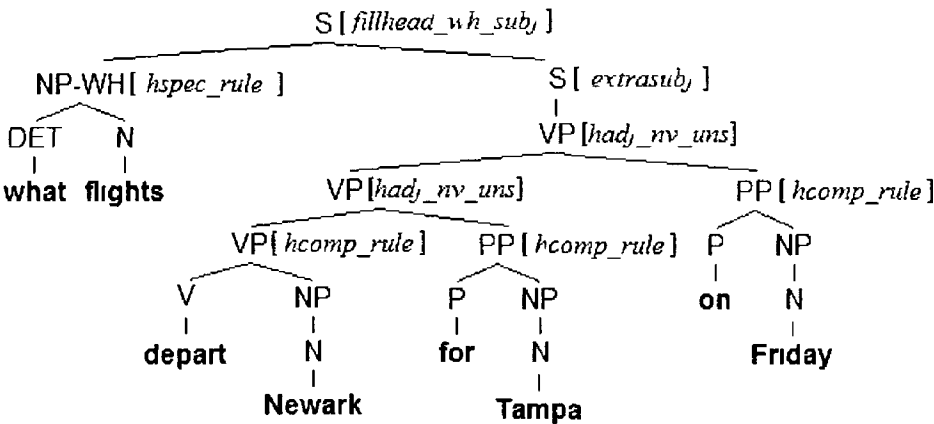
⁷⁰ Handling unknown words in general is beyond the scope of this dissertation. See Section 2.4 for other HPSG treatments.

One way to identify the failure as a missing lexicon lexical entry is to compare with a lexicon-less annotation to detect if the annotation method identifies and generates lexical items of a different form/with different subcategorisation frames

Figure 8 2 shows an example of a verb, ‘depart’, that is specified in LinGO ERG only in its intransitive form and that is used in the ATIS corpus and correctly handled in the annotation method as a transitive verb. The tree in Figure 8 2 is one of those resolved by the annotation method where the bare string could not be parsed by the LKB. As for Figure 8 1 the *hspec* and *hcomp* rules are similar to the head-specifier and head-complement schemas, *hadj_nv_uns* is an adjunct rule, *fillhead_wh_subj* is an head-filler rule, and *extrasubj* accounts for subject extraction.

Figure 8 2 ATIS tree (@i06018sx-a-8) with form of ‘depart’ not in LinGO ERG

What flights depart Newark for Tampa on Friday



8 4 Chapter Summary

In this chapter, three experiments were performed to assess the annotation method developed in this dissertation, against the goals of (i) successfully using an HPSG grammar for treebank annotation, (ii) showing the advantages of a wide-coverage grammar-based annotation method over a grammar-less or minimal grammar counterpart, and (iii) demonstrating the ability of the annotation method to assist in improving grammar/lexical coverage

The first experiment annotated the ATIS treebank with a minimal subset of the LinGO ERG grammar. Coverage achieved was 99.8%, with 95.3% non-fragment trees receiving a resolution. This resolution rate is significantly higher than the LKB parse success rate for the bare strings from the ATIS corpus at 79.8%.

In the second experiment, the full LinGO ERG grammar was applied to the ATIS corpus, with lexical lookup for all words except main verbs and nouns. Resolution rate for non-fragments was 68.8%, lower than the LKB parse success rate. This can be attributed to the preliminary nature of the mapping principles, and also to cases where the LinGO grammar would not license the structure of the ATIS tree, i.e. grammar incompleteness.

I conjecture that with some further work on the mapping principles, the annotation method could be used to identify such cases of grammar incompleteness and assist in improving grammar coverage. The annotated trees included mapped lexical items with highly constrained lexical types, which were close in detail to the predefined LinGO ERG lexical items.

The third experiment assessed the ability for the annotation method to improve lexicon coverage. This was illustrated by an example of a word where a new subcategorisation frame, not included in the LinGO ERG lexicon, was identified and handled in annotation, leading to increased coverage.

9 Conclusions and future work

9.1 Conclusions

In this dissertation I set out to develop a method to annotate treebanks with a wide-coverage HPSG grammar. My goal was to explore the benefits of combining corpus-based and data-driven linguistic resources with hand-crafted and theory-driven grammars for the improvement of both treebank annotation and grammar coverage.

9.1.1 The annotation method

I developed and implemented a constraint-based grammar-driven treebank annotation method with four main components: treebank pre-processing, basic CFG-to-HPSG mapping, specific local tree mapping and constraint resolution. The system is designed to be usable with any LKB-compatible HPSG grammar⁷¹ but it is optimised for use with the LinGO ERG wide-coverage grammar.

In pre-processing standard treebank techniques for head-lexicalisation, complement identification and binarisation are extended to cope with HPSG-specific requirements on the tree structure. Additions and alterations have been made to head and complement rules, and operations such as the introduction of an N-bar level in NPs with pre- and post-modifiers, NP compound collapsing, and grammar compacting have been added.

CFG-to-HPSG mapping principles establish a correspondence between CFG categories and rules and HPSG lexical and phrasal signs and types. Part-of-speech tags are related to specific LinGO ERG lexical types, and argument structure derived from the pre-processed treebank is used to further constrain the types. Phrase mapping is performed between CFG rules and HPSG phrasal types, making use of the dependency information in the pre-processed treebank as well as CFG rule category information to determine the appropriate HPSG phrase type.

⁷¹ The HPSG typed feature structure grammar processing system implemented for this dissertation as the backbone of the annotation system supports LKB compatible HPSG grammars. The grammar format used by the LKB (Copestake 2002) is used as a form of standard for collaborative HPSG grammar engineering (Open *et al.* 2002a).

Local tree mapping applies the basic CFG-to-HPSG mapping to the local trees in the treebank and addresses specific constructions such as long-distance dependencies and coordinate structures, which cannot be handled through (context-free) rule mapping. Grammar-specific mappings such as marking for passive, subject-aux inversion and partitives, are also handled by this component.

The constraint solver component combines the local tree mappings into a single phrasal sign for the whole tree through unification and grammar rule application. It also controls lexical resolution where the use of a predefined lexicon can be configured so that lexical lookup can be made conditional on the part of speech tag or on specific words. This allows for a combined use of a predefined lexicon for closed word classes and particular words with mapped lexical types for unknown words.

For each of the components the use of the LinGO ERG grammar, together with the LKB system, helped to simplify development. The parses for the bare strings in the ATIS corpus helped guide the development of the treebank pre-processing. By making use of the predefined grammar lexical types and rules in the CFG-to-HPSG and local tree mapping, heuristic rules and skeletal feature structures did not need to be developed from scratch. Also, if it is known that particular phenomena or constructions are handled in the grammar, they do not need to be specifically handled in the mapping stages. The constraint resolution will ensure that it is addressed through grammar rule application.⁷² The LKB system grammar development facilities, with the ability to inspect the type hierarchy, parses, charts and feature structures, were also a helpful productivity tool during system development.

Background and related work for the dissertation, including an introduction to HPSG, the LinGO ERG grammar and LKB system, is included in Chapter 2. The underlying HPSG grammar processing system implemented for the annotation method is presented in Chapter 3. The annotation method components of pre-processing, CFG-to-HPSG mapping, local tree mapping, and constraint resolution are described in Chapters 4 to 7, respectively.

⁷² There is a drawback in relying too heavily on this grammar based annotation model, in that the fragility and incompleteness of the grammar also then applies to the annotation system. Fall back strategies to cope with this are discussed in Chapter 7.

9 1 2 Comparison with other work

The method for treebank annotation developed in this dissertation compares most closely with, on the one hand, work done on automatic LFG F-structure annotation (Cahill *et al* , 2002, 2004), described in Section 2 1 1, and on the other with the HPSG annotation of the Penn-II treebank done by (Miyao *et al* , 2004), described in Section 2 3 4 While my method has not yet been tested on a large treebank like Penn-II, some preliminary conclusions can be drawn In the LFG annotation approach annotation matrices are used to capture linguistic principles and generalisations HPSG is already principle-based, and in Miyao *et al* 's approach a textbook grammar is used I use an existing fine-grained unification grammar The rich lexical type system, and the grammar and lexical rules of the LinGO ERG grammar, are used extensively in the annotation and all contribute to providing rich HPSG annotations of the treebank trees

9 1 3 Evaluation

I evaluated the annotation method in three experiments to determine its performance against the set goals

In the first experiment a basic text book type grammar (Pollard & Sag, 1994) based on a subset of the LinGO ERG grammar was used to annotate the ATIS treebank without the use of a lexicon 95 3% of the trees covering a complete sentence (with a non-fragment top category) were annotated with a single covering phrasal sign, fully resolved through successful grammar rule applications in the constraint solver

This is significantly better than the LKB parse result for the ATIS tree bare strings with the same LinGO ERG grammar, which is 79.8%–99.8% of the trees received at least one annotation with a complete HPSG phrasal sign, resulting from constraint solver rule application or unification of mapped feature structures in fall-back mode (see Section 7.3). The results of experiment one indicate that the annotation system achieves almost complete coverage of the ATIS corpus with a basic HPSG grammar, validating the basic annotation method.

In the second experiment, the full LinGO ERG grammar was applied to the ATIS corpus and mixed mode lexical lookup was used, where the LinGO ERG lexicon was used for all words except main verbs (non-auxiliaries) and nouns. 68.8% of the trees received a resolution following successful rule application. The lower resolution rate compared to the 79.8% LKB parse coverage is accounted for by the preliminary nature of the CFG-to-HPSG and local tree mapping established for the annotation method so far, and also by cases where the LinGO grammar would not license the structure of the ATIS tree, i.e. where the grammar coverage is incomplete. As discussed in Section 8.2.4, differences in annotation and parse results, and the failure to annotate due to structural differences between treebank structures and the grammar can be used to identify limitations in grammar coverage, which is one of the goals of the annotation method. A full evaluation of this capability cannot be completed until the CFG-to-HPSG mapping principles and local tree mapping method are more complete.

Experiment 2 also presented some other notable results. The annotation method produced a resolution for several trees whose bare strings could not be parsed by the LKB with the same grammar. This shows the potential of the annotation method to expand grammar coverage. In addition, the average number of resolutions found, at 2.4, is significantly lower than the average number of parses generated by the LKB, which is 12.2. This indicates the potential of the annotation method to be used for parse ranking.

The third experiment assessed the ability of the annotation method to improve lexicon coverage by identifying unknown words and extracting lexical information for the creation of new lexical entries. The mixed mode lookup and use of mapped lexical signs in experiment two demonstrates the ability of the system to function without a complete lexicon.

The specificity achieved by the lexical mapping with the use of argument structure information, evidenced by the successful annotation without returning a large number of alternative resolutions, also shows the advantage of the combined treebank and wide-coverage grammar annotation approach

The use of the rich lexical hierarchy in the annotation method allows for the creation of lexical information close to or at the same level of detail as in the predefined LinGO ERG lexicon. The annotation method can also identify additional subcategorisation frames for words that exist in the pre-defined lexicon in a different form

9.2 Future work

The annotation method developed in this dissertation has potential for several different avenues of further research

- (i) The annotation method is still incomplete with respect to the CFG-to-HPSG correspondence and local tree mapping and further development should improve the annotation results significantly. Development and fine-tuning is also possible in constraint resolution, where type inference could be explored in order to specialise lexical type information. Qualitative as well as quantitative evaluation can then be performed on the annotation results, and the ability of the annotation system to assist in improving grammar coverage can also be further developed.
- (ii) It would be interesting to scale up the annotation system to the Penn-II treebank where limitations in the coverage of the LinGO ERG grammar will become more noticeable.
- (iii) The system can be used to generate gold standard HPSG structures and evaluate the LinGO ERG grammar against them, as has been done for CCG (Hockenmaier & Steedman, 2002). Hand-assessment will be needed to verify the gold standard structures. Dependency trees can be extracted from the LinGO ERG parse, and precision and recall can be measured.
- (iv) Implications for robust parsing have been hinted at in several places but deferred as being outside the scope of this dissertation. More sophisticated fall-back strategies and handling of partial results are potential further developments in this area.

- (v) Related to robust parsing, one interesting aspect of HPSG-based treebank annotation encountered during this work is the issue of relating different HPSG grammars to one another and establishing correspondences based on the type hierarchy. Being able to relate a basic and a large-scale grammar to each other during the annotation process would provide interesting options for robust parsing, with respect to fall-back options.
- (vi) The development and use of the annotation method in close conjunction with the LinGO ERG grammar system raises interesting methodological questions regarding the combining of hand-crafted and data-oriented linguistic resources, as well as how they can complement each other.
- (vii) The system can be applied to parsing, in a model similar to (Cahill *et al* (2002, 2004).
- (viii) Further work can be done to explore lexicon extraction and evaluation, following (O'Donovan *et al* 2004).

The last two of these areas for further work are explored in some more detail below.

9.2.1 Parsing

One interesting application for an HPSG-annotation system is in parsing. While great progress has been made in processing of large constraint-based grammars such as LFG (Riezler *et al* , 2002) and HPSG (Wahlster, 2000, Oepen *et al* , 2002a) in recent years, parsing with the very large feature structures in large grammars of such theories is very unification-intensive,⁷³ processor- and memory-demanding. There has, therefore, been interest in parsing techniques that reduce the computational burden of unification, such as using a context-free backbone in parsing (Oepen *et al* , 2002a), using stochastic disambiguation (Riezler *et al* , 2002), or by combining PCFG parsing with annotation and constraint resolution.

Following the second approach, of PCFG-based approximations to LFG/HPSG grammars, as described by (Cahill *et al* , 2002, 2004), two parsing architectures can be used. In the pipeline architecture, a PCFG is first extracted from the unannotated treebank and used for parsing.

⁷³ A unification based parser can spend 90% of CPU time taken parsing a sentence with a large scale unification based grammar in feature structure unification or associated copying (Malouf *et al* , 2000).

The most probable parse tree for the string is then HPSG-annotated and resolved. In the integrated architecture, the treebank is first HPSG-annotated and from this an annotated PCFG is extracted with signs associated with categories treated as monadic categories. Parsing is performed with the annotated PCFG, and the sign annotations for the most probable tree are then resolved to a complete sign. Using this combined approach has several advantages over traditional HPSG parsing techniques. PCFG parsing is both highly efficient and can be made more robust than hand-crafted, rule-based unification grammars and parsing. Corpora can be used to cover a large range of constructions. The annotation and constraint resolution enriches the parse, in the best case up to a single resolved sign with full LINGO ERG grammar level, or else with partial HPSG detail.

9.2.2 Lexicon creation

Further work can also be done with regard to lexicon creation. Following (O'Donovan *et al.*, 2004), it is possible to extract the lexicon entry with subcategorisation information with regard to COMPS, SUBJ, and other HPSG valence features. Conditional probabilities can also be associated with the subcategorisation frames. The extracted entries can then be compared against COMLEX, and the LINGO ERG lexicon.

9.3 Summary

The use of a predefined wide-coverage constraint-based grammar together with a treebank annotation method has shown several advantages over alternative treebank enrichment approaches such as using only a treebank and heuristic principles (Cahill *et al.*, 2002, 2004, Hockenmaier & Steedman, 2002, Miyao *et al.*, 2004), and grammar-based treebank creation (Oepen *et al.*, 2002b, Simov *et al.*, 2002).

The use of a wide-coverage grammar has assisted both in the development of the annotation method and in constraining the grammatical structure during the annotation. The use of a rich lexical type hierarchy has also added detail to the annotation of lexical items.

With the direct use of LINGO ERG grammar type system and rules in the annotation, the resulting lexical and phrasal signs are fully LINGO ERG-compatible, and can be easily incorporated back into the grammar.

The use of a parsed corpus of unconstrained text also helps to validate the grammar, disambiguate and rank parses, and identify incompleteness in the grammar and lexicon

The annotation method has been shown to have the ability to annotate the ATIS corpus with a high degree of coverage, using a minimal grammar, and also with a high level of detail, using the full LinGO ERG grammar. The handling of lexicon incompleteness and ability to identify and create specific lexical representations for new words and word forms has also been demonstrated

While work remains to be done to increase the treebank coverage when annotating with the full LinGO ERG grammar, preliminary results are promising

References

- Barg, P , Walther, M (1998) Processing Unknown Words in HPSG In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL-98)*, Volume 1, Montreal, Canada 91–95
- Berger, A , Della Pietra, S , A , Della Pietra, V , J (1996) A maximum Entropy Approach to Natural Language Processing In *Computational Linguistics*, 22(1) 39-71
- Bod, R , Kaplan, R (1998) A probabilistic corpus-driven model for lexical-functional grammar In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL-98)*, Volume 1, Montreal, Canada 145–151
- Bouma, G , van Noord, G , Malouf, R (2000) Alpino Wide-coverage computational analysis of Dutch In *Computational Linguistics in the Netherlands CLIN 2000*, Rodopi, Netherlands
- Bresnan, J (2001) *Lexical-Functional Syntax* Blackwell, Oxford
- Cahill, A , Burke, M , O'Donovan, R , van Genabith, J , Way, A (2004) Long-Distance Dependency Resolution in Automatically Acquired Wide-Coverage PCFG-Based LFG Approximations In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*, Barcelona, Spain 320-327
- Cahill, A , McCarthy, M , van Genabith, J , Way, A (2002) Parsing with PCFGs and Automatic F-Structure Annotation In *Proceedings of the Seventh International Conference on LFG* CSLI Publications, Stanford, CA 76–95
- Carpenter, B (1992) *The logic of typed feature structures* MIT Press, Cambridge, Ma
- Charniak, E (1993) *Natural Language Learning* MIT Press, Cambridge, Ma
- Charniak, E (1997) Tree-bank grammars In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, AAAI Press/MIT Press, Menlo Park 1031–1036
- Chen, J , Vijay-Shanker, K (2000) Automatic extraction of TAGs from the Penn Treebank In *Proceedings of the 38th Annual Meeting of the Association of Computational Linguistics*, Hong Kong 65–76
- Chomsky, N (1957) *Syntactic Structures* Mouton, the Hague

- Chomsky, N (1965) *Aspects of the Theory of Syntax* MIT Press, Cambridge, MA
- Church, K (1988) A stochastic parts program and noun phrase marker for unrestricted text In *Proceedings of the second Conference on Applied Natural Language Processing*, Austin, Texas 136–143
- Collins, M (1997) Three generative lexicalized models for statistical parsing In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid, Spain 16–23
- Collins, M (1999) *Head-driven statistical models for natural language parsing* PhD Thesis, University of Pennsylvania, Philadelphia
- Copestake, A (2002) *Implementing typed feature structure grammars* Stanford CSLI Publications
- Copestake, A , Flickinger, D , Sag, I (1999) Minimal Recursion Semantics An Introduction Manuscript, Stanford <http://www-csl.stanford.edu/~aac/papers/mrs.pdf>
- Copestake A , Flickinger, D (2000) An open source grammar development environment and broad-coverage English grammar using HPSG In *Proceedings of the second conference on Language Resources and Evaluation (LREC2000)*, Athens, Greece 591–600
- Dahl, D A , Bates, M , Brown, M , Fisher, W , Hunicke-Smith, K , Pallett, D , Pao, C , Rudnicky, A , Shriberg, E (1994) Expanding the Scope of the ATIS Task The ATIS-3 Corpus In *Proceedings of the ARPA Human Language Technology Workshop*, Princeton, NJ 45–50
- Flickinger, D (2000) On building a more efficient grammar by exploiting types In Oepen, S , Flickinger, D , Tsujii, J , Uszkoreit, H (eds) 2002 *Collaborative Language Engineering*, CSLI Publications 1–17
- Flickinger, D , Copestake, A , Sag, I (2000) HPSG Analysis of English In Wahlster, W , (ed) (2000) *Verbmobil Foundations of speech-to-speech translation* Springer-Verlag Berlin Heidelberg New York 254–263
- Fouvry, F , (2003a) *Robust Processing for Constraint-based Grammar Formalisms* PhD thesis, University of Essex, UK

- Fouvry, F (2003b) Lexicon acquisition with a large-coverage unification-based grammar In *Proceedings of EACL-03, 10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary 87–90
- Frank, A , Sadler, L , van Genabith, J Way, A (2003) From treebank resources to LFG F-structures In (ed) Anne Abeille, *Treebanks Building and using Syntactically annotated corpora*, Kluwer Academic Publishers 367–389
- Gazdar, G , Mellish, C (1989) *Natural Language Processing in Prolog* Addison Wesley, Reading, Ma
- Ginzburg, J , Sag, I (2000) *Interrogative Investigations* CSLI Publications
- Gotz, T (1994) *A normal form for typed feature structures* Master's thesis, Eberhard-Karls-Universitat, Tübingen, Germany
- Hemphill, C , T , et al (1990) The ATIS Spoken Language Systems Pilot Corpus In *Proc DARPA Speech and Natural Language Workshop*, Hidden Valley, PA 96-101
- Hepple, M , van Genabith, J (2000) Experiments in Structure Preserving Grammar Compaction In *1st Meeting on Speech Technology Transfer*, Seville, Spain
- Hindle, D (1983) Deterministic parsing of syntactic nonfluencies In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, Cambridge, MA 123–128
- Hindle, D (1989) Acquiring disambiguation rules from text In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada 118–125
- Hockenmaier, J , Steedman, M (2002) Acquiring Compact Lexicalized Grammars from a Cleaner Treebank In *Proceedings of Third International Conference on Language Resources and Evaluation*, Las Palmas
- Horiguchi, K , Torisawa, K , Tsujii, J (1995) Automatic acquisition of content words using an HPSG-based parser In *Proceedings of the Natural Language Processing Pacific Rim Symposium*, Seoul, Korea 320–325
- Jurafsky, D , Martin, J (2000) *Speech and Language Processing* Prentice Hall, Upper Saddle River, New Jersey

- Kiefer, B , Krieger, H , Nederhof, M (2000) Efficient and Robust Parsing of Word Hypothesis Graphs In Wahlster, W , (ed) (2000) *Verbmobil Foundations of speech-to-speech translation* Springer-Verlag Berlin Heidelberg New York 280-295
- Kinyon A , Prolo, C (2002) Identifying verb arguments and their syntactic function in the Penn Treebank In *Proceedings of the third LREC Conference*, Las Palmas, Spain 1982–1987
- Krotoy, A , Hepple, M , Garzauskas, R , Wilks, Y (1998) Compacting the Penn Treebank Grammar In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING ACL-98)*, Montreal, Canada 669–703
- Lascarides, A , Copestake, A , (1999) Default representation in constraint-based frameworks *Computational Linguistics*, 25 (1) 55–106
- MacLeod, C , Grishman, R , Meyers, A (1994) The complex syntax project the first year In *Proceedings of the ARPA Workshop on Human Language Technology*, Princeton, NJ 669–703
- Magerman, D , M (1994) *Natural Language Parsing as Statistical Pattern Recognition* PhD thesis, Stanford, California
- Magerman, D M (1995) Statistical decision-tree models for parsing In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, MA 276–283
- Malouf, R , Carroll, J , Copestake, A (2000) Efficient feature structure operations without compilation In Flickinger, D , Oepen, S , Uszkoreit, H , Tsujii, Jun'ichi (eds) *Journal of Natural Language Engineering Special Issue on Efficient Processing with HPSG Methods, Systems, Systems, Evaluation* 6(1)
- Malouf, R , van Noord, G (2004) Wide Coverage Parsing with Stochastic Attribute Value Grammars In *IJCNLP-04 Workshop Beyond Shallow Analyses – Formalisms and statistical modelling for deep analyses*, Hainan Island, China
- Manning, C , D , Schutze, H (1999) *Foundations of Statistical Natural Language Processing* MIT Press, Cambridge, MA
- Marcus, P , M , Santorini, B, Marcinkiewicz, M , A (1993) Building a large annotated corpus of English The Penn Treebank *Computational Linguistics*, 19 (2) 313–330

Marcus, M , Kim, G , Marcinkiewicz, M , A , MacIntyre, R , Bies, A , Ferguson, M , Katz, K , Schasberger, Britta (1994) The Penn Treebank Annotating Predicate Argument Structure In *Proceedings of the Human Language Technology Workshop*, San Francisco

Miyao, Y , Ninomiya, T , Tsuji, J (2004) Corpus-oriented grammar development for acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP-04)*, Hainan Island, China

Nakanishi, H , Miyao, Y , Tsuji, J (2004) Using inverse lexical rules to acquire a wide-coverage lexicalized grammar In *IJCNLP-04 Workshop Beyond Shallow Analyses – Formalisms and statistical modelling for deep analyses*, Hainan Island, China

Neumann, G (2003) A Uniform Method for Automatically Extracting Stochastic Lexicalized Tree Grammars from Treebanks and HPSG In (ed) Anne Abeille, *Treebanks Building and using Syntactically annotated corpora*, Kluwer Academic Publishers, Dordrecht 351–366

O'Donovan, R , Burke, M , Cahill, A , van Genabith, J , Way, A (2004) Large-Scale Induction and Evaluation of Lexical Resources from the Penn-II Treebank In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-'04)*, Barcelona, Spain 368-375

Oepen, S , Carroll, J (2000a) Parser engineering and performance profiling In *Journal of Natural Language Engineering (Special Issue on Efficient processing with HPSG methods systems, evaluation)* 6(1) 81-97

Oepen S , Carroll, J (2000b) Ambiguity packing in constraint-based parsing Practical results In *Proceedings of the 1st Conference of the North American Chapter of the ACL*, Seattle, WA 162-169

Oepen, S , Flickinger, D , Tsuji, J , Uszkoreit, H (2002a) *Collaborative Language Engineering*, CSLI Publications, Stanford, CA

Oepen, S , Flickinger, D , Toutanova, K , Shieber, S , Manning, C D , Flickinger, D , Brants, T (2002b) The LinGO Redwoods Treebank motivation and preliminary applications In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan 1253–1257

- Osenova, P , Simov, K (2003) Between Chunk Ideology and Full Parsing Needs In *Proceedings of the Shallow Processing of Large Corpora (ProLac 2003) Workshop*, Lancaster, UK 78–87
- Penn, (1994) Penn treebank project readme, <http://www ldc upenn edu/doc/treebank2/treebank2 index html>
- Pollard, C , Sag, I (1994) *Head-driven Phrase Structure Grammar* Stanford CA CSLI Publications
- Radford, A (1988) *Transformational grammar A First Course* Cambridge University Press, Cambridge
- Riezler, S , King, T H , Kaplan, R M , Crouch, R , Maxwell, J T , Johnson, M (2002) Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, PA 271-278
- Sag, I (1997) English Relative Clause Constructions *Journal of Linguistics* 33(2) 431–484
- Sag, I , Wasow, T , Bender, E (2003) *Syntactic Theory a formal introduction* Second edition, Stanford CSLI Publications
- Simov, K (2002) Grammar extraction and refinement from an HPSG Corpus In *Proceedings of the ESSLLI Workshop on Machine Learning Approaches in Computational Linguistics*, Trento, Italy 38–55
- Simov, K , Osenova, P , Slavceva, M , Kolkovska, S , Balabanova, E , Doikoff, D , Ivanova, K , Simov, A , Kouylekov, M (2002) Building a linguistically interpreted corpus of Bulgarian the BulTreeBank In *Proceedings of LREC 2002*, Canary Islands, Spain 1729–1736
- Steiner, R , Tsujii, J (1999a) Robustness in HPSG via Extended ID Schemata In *Proceedings of the fifth Natural Language Processing Pacific Rim Symposium (NLPRS)*, Beijing, China 108--113
- Steiner, R , Tsujii, J (1999b) M-rules Adaptable Rules for Robustness in Unification-based Systems In *Proceedings of the fifth Natural Language Processing Pacific Rim Symposium (NLPRS)* Beijin, China 126--131

- Tomabechi, H (1991) Quasi-destructive graph unification In *Proceedings of the 29th Annual Conference of the Association for Computational Linguistics*, Berkeley, CA 315–322
- Toutanova, K , Manning, C D , Shieber, S , Flickinger, D , Oepen, S (2002) Parse disambiguation for a rich HPSG grammar In *First Workshop on Treebanks and Linguistic Theories (TLT2002)*, Sozopol, Bulgaria 139–149
- Vogel, C , Cooper, R (1995) Robust Chart Parsing with Mildly Inconsistent Feature Structures In Andreas Schotter and Carl Vogel (eds) *Natural Language Understanding and Logic Programming* Working Papers in Cognitive Science, Volume 10 University of Edinburgh [EUCCS-WP10](#), Edinburgh 197-216
- Wahlster, W (ed) (2000) *Verbmobil Foundations of speech-to-speech translation* Springer-Verlag, Berlin, Heidelberg New York
- Xia, F (1999) Extracting Tree Adjoining Grammars from Bracketed Corpora In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, Beijing, China 398-403
- Yutaka, M , Kentaro, T , Tsuji, J (2000) HPSG-style Underspecified Grammar with Wide Coverage In COLING-ACL 98, *Proceedings of the 17th international Conference on Computational Linguistics*, University of Montreal, Quebec, Canada 876-880

Appendix 1· Lexical entries added to the LinGO ERG

These entries were added to the LinGO ERG lexicon for parsing of the bare strings of the ATIS corpus

```

Baltimore_n1 = n_proper_le &
[ STEM < baltimore >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Baltimore
]

Minneapolis_n1 = n_proper_le &
[ STEM < minneapolis >
  SYNSEM LOCAL KEYS KEY CONST_VALUE
Minneapolis ]

aircraft_n1 = n_intr_le &
[ STEM < aircraft >
  SYNSEM LOCAL KEYS KEY _aircraft_rel ]

Burbank_n1 = n_proper_le &
[ STEM < burbank >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Burbank ]

Charlotte_n1 = n_proper_le &
[ STEM < charlotte >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Charlotte ]

Cincinnati_n1 = n_proper_le &
[ STEM < cincinnati >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Cincinnati
]

Delta_n1 = n_proper_le &
[ STEM < delta >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Delta ]

Dulles_n1 = n_proper_le &
[ STEM < dulles >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Dulles ]

fare_n1 = n_intr_le &
[ STEM < fare >
  SYNSEM LOCAL KEYS KEY _fare_rel ]

Guardia_n1 = n_proper_le &
[ STEM < guardia >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Guardia ]

Houston_n1 = n_proper_le &
[ STEM < houston >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Houston ]

Indianapolis_n1 = n_proper_le &
[ STEM < indianapolis >
  SYNSEM LOCAL KEYS KEY CONST_VALUE
Indianapolis ]

Memphis_n1 = n_proper_le &
[ STEM < memphis >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Memphis
]

Miami_n1 = n_proper_le &
[ STEM < miami >

Salt_n1 = n_proper_le &
[ STEM < salt >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Salt ]

shortest_a1 = adj_superl_le &
[ STEM < shortest >
  SYNSEM LOCAL KEYS KEY _short_rel ]

Tacoma_n1 = n_proper_le &
[ STEM < tacoma >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Tacoma ]

Tampa_n1 = n_proper_le &
[ STEM < tampa >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Tampa ]

Westchester_n1 = n_proper_le &
[ STEM < westchester >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Westchester
]

TWA_n1 = n_proper_le &
[ STEM < T W A >
  SYNSEM LOCAL KEYS KEY CONST_VALUE TWA ]

Q_n1 = n_proper_le &
[ STEM < Q >
  SYNSEM LOCAL KEYS KEY CONST_VALUE Q ]

AP_n1 = n_proper_le &
[ STEM < A P >
  SYNSEM LOCAL KEYS KEY CONST_VALUE AP ]

JFK_n1 = n_proper_le &
[ STEM < J F K >
  SYNSEM LOCAL KEYS KEY CONST_VALUE JFK ]

CO_n1 = n_proper_le &
[ STEM < C O >
  SYNSEM LOCAL KEYS KEY CONST_VALUE CO ]

BNA_n1 = n_proper_le &
[ STEM < B N A >
  SYNSEM LOCAL KEYS KEY CONST_VALUE BNA ]

WN_n1 = n_proper_le &
[ STEM < W N >
  SYNSEM LOCAL KEYS KEY CONST_VALUE WN ]

MIA_n1 = n_proper_le &
[ STEM < M I A >
  SYNSEM LOCAL KEYS KEY CONST_VALUE MIA ]

airfare_n1 = n_intr_le &
[ STEM < airfare >
  SYNSEM LOCAL KEYS KEY _airfare_rel ]

SaintLouis_n1 = n_proper_le &
[ STEM < saint louis >

```

<p>SYNSEM LOCAL KEYS KEY CONST_VALUE Miami]</p> <p>Milwaukee_n1 = n_proper_le & [STEM < milwaukee > SYNSEM LOCAL KEYS KEY CONST_VALUE Milwaukee]</p> <p>Montreal_n1 = n_proper_le & [STEM < montreal > SYNSEM LOCAL KEYS KEY CONST_VALUE Montreal]</p> <p>Newark_n1 = n_proper_le & [STEM < Newark > SYNSEM LOCAL KEYS KEY CONST_VALUE Newark]</p> <p>Ontario_n1 = n_proper_le & [STEM < ontario > SYNSEM LOCAL KEYS KEY CONST_VALUE Ontario]</p> <p>Orlando_n1 = n_proper_le & [STEM < orlando > SYNSEM LOCAL KEYS KEY CONST_VALUE Orlando]</p> <p>SaintPetersburg_n1 = n_proper_le & [STEM < saint petersburg > SYNSEM LOCAL KEYS KEY CONST_VALUE SaintPetersburg]</p> <p>Phoenix_n1 = n_proper_le & [STEM < phoenix > SYNSEM LOCAL KEYS KEY CONST_VALUE Phoenix]</p>	<p>SYNSEM LOCAL KEYS KEY CONST_VALUE SaintLouis]</p> <p>FnF_n1 = n_proper_le & [STEM < F F > SYNSEM LOCAL KEYS KEY CONST_VALUE FnF]</p> <p>H_n1 = n_proper_le & [STEM < H > SYNSEM LOCAL KEYS KEY CONST_VALUE H]</p> <p>DC_n1 = n_proper_le & [STEM < D C > SYNSEM LOCAL KEYS KEY CONST_VALUE DC]</p> <p>SaintPaul_n1 = n_proper_le & [STEM < saint paul > SYNSEM LOCAL KEYS KEY CONST_VALUE SaintPaul]</p> <p>seating_n1 = n_intr_le & [STEM < seating > SYNSEM LOCAL KEYS KEY _seating_rel]</p>
---	---

Appendix 2 ATIS treebank category compacting matrix

ADJP	ADJP PRD	ADJP TMP									
ADVP	ADVP DIR	ADVP LOC	ADVP PRD	ADVP TMP							
NP	NP TMP	NX	NP 1	NP-3	NP LGS	NP PRD	NP SBJ	NP SBJ-1	NP- SBJ 2	NP TMP 5	NP- TMP PRD
PP	PP DIR	PP LOC	PP TMP	PP DIR=1	PP DIR=2	PP- DIR 2	PP- CLR	PP DIR 1	PP DIR 3	PP DIR 4	PP DIR 5
PP con tinues	PP DIR DIR	PP LOC CLR	PP LOC PRD	PP TMP 4	PP TMP 5	PP TMP PRD	PP- TMP TCP 1	PP TMP TCP- 2	PP- TCP 2		
S	S NOM										
SBAR	SBAR6										
WHNP	WHNP 1	WHNP2									
NN	NNP	NNS	NNPS								
VB	VBP	VBZ	VBG	VBN	VBD						
JJ	JJR	JJS									
IN	TO										
RB	RBR	RBS									

Appendix 3. ATIS part of speech tags

Tag	Description	Collapsed POS tags	Examples in ATIS
CC	Coordinating conjunction		And, or
CD	Cardinal Number		Twenty, six
DT	Determiner		The this a that, any
EX	Existential there		There
IN	Preposition or subord conjunction	TO	From, in of after
JJ	Adjective	JJR JJS	Other next extra first ninth
MD	Modal		Would should 'd can, could
NN	Noun	NNS NNP NNPS	Flights Baltimore
PDT	Predeterminer		All
POS	Possessive ending		's
PRP	Personal pronoun		I me them
RB	Adverb	RBR RBS	Here, a m, much actually
RP	Particle		Out
SYM	Symbol mathematical or scientific		D, M, slash
UH	Interjection		Oh oops
VB	Verb	VBD, VBG, VBN, VBP VBZ	List stop does leaving
WDT	Wh determiner		That, which, what
WP	Wh pronoun		What
WRB	Wh adverb		Where how

The distinction between prepositions and subordinating conjunctions is recoverable from syntactic context, with prepositions preceding noun phrases and prepositional phrases and subordinating conjunctions preceding clauses (Marcus *et al* , 1993)

Appendix 4 Phrase descriptions

Written in LKB-compatible feature structure definition syntax

adapted from LINGO ERG LKB parse-nodes
identifiers mapped to ATIS CFG categories

DT = word & [SYNSEM basic_det_synsem & [LOCAL CAT HEAD det &
[MOD <>]]]

PDT = n_part_lexent & [SYNSEM partitive_noun_synsem & [LOCAL CAT HEAD partn]]

WDT = word & [SYNSEM basic_det_synsem & {LOCAL CAT HEAD det &
[MOD <>]}]

CC = sign & [SYNSEM synsem &
{LOCAL [CAT VAL COMPS *cons*
ARG S *cons*
CONJ strict conj &
[CHEAD LEFT FIRST cnl]]}]

VB = word & [SYNSEM LOCAL CAT [HEAD verb
VAL COMPS *top*]]

, duplicate of VB

MD = aux_verb_word & [SYNSEM LOCAL CAT [HEAD verb,
VAL COMPS *cons*]]

for inf to duplicate of VB

INFTO = complementizer_word & [SYNSEM LOCAL CAT [HEAD comp
VAL COMPS *top*]]

adv

RB = word & [SYNSEM LOCAL CAT HEAD basic_adv]

,duplicate of RB

WRB = word & [SYNSEM LOCAL CAT HEAD basic_adv]

,duplicate of RB for please

INTJ = word & [SYNSEM LOCAL CAT HEAD basic_adv]

duplicate of RB, for please pos eqv to intj

UH = word & [SYNSEM LOCAL CAT HEAD basic_adv]

P

IN = word & [SYNSEM basic_prep_synsem & [LOCAL CAT [HEAD prep & [MOD *top*],
VAL COMPS *cons*]]]

dupl of IN

RP = word & [SYNSEM LOCAL CAT [HEAD prep & [MOD *top*],
VAL COMPS *cons*]]

dupl of IN

PRT = word & [SYNSEM LOCAL CAT [HEAD prep & [MOD *top*]
VAL COMPS *cons*]]

dupl of IN

TO = word & [SYNSEM basic_prep_synsem & [LOCAL CAT [HEAD prep & [MOD *top*]
VAL COMPS *cons*]]]

n
NN = noun_nominfl_word & [SYNSEM LOCAL CAT [HEAD nominal
VAL [SPR < synsem >,
COMPS *top*]]]

NNP = n_proper_le & [SYNSEM LOCAL CAT [HEAD nominal]]

,duplicate of NN
PRP = word & [SYNSEM LOCAL CAT [HEAD nominal,
VAL [SPR < synsem >,
COMPS *top*]]]

,duplicate of NN
WP = sign & [SYNSEM LOCAL CAT [HEAD nominal
VAL [SPR < synsem >
COMPS *top*]]]

duplicate of NN
EX = sign & [SYNSEM LOCAL CAT [HEAD nominal
VAL [SPR < synsem >,
COMPS *top*]]]

duplicate of NN
SYM = sign & [SYNSEM LOCAL CAT [HEAD nominal,
VAL [SPR < synsem >
COMPS *top*]]]

adj
JJ = adj_word & [SYNSEM LOCAL CAT HEAD adj]

CD = [SYNSEM LOCAL CAT HEAD intadj]

quantifier phrase duplicate of DT
QP = sign & [SYNSEM LOCAL CAT HEAD adj]

,np
NP = sign & [SYNSEM LOCAL CAT [HEAD nominal
VAL SPR *top*]]]

,np
NX = sign & [SYNSEM LOCAL CAT [HEAD nominal
VAL SPR *top*]]]

,duplicate of NP
WHNP = sign & [SYNSEM LOCAL CAT [HEAD nominal
VAL SPR *top*]]]

np wh = sign & [SYNSEM [LOCAL CAT [HEAD nominal,
VAL [SPR *top*]],
NON LOCAL QUE 1 dlist]]

vp
VP = sign & [SYNSEM LOCAL CAT [HEAD verb
VAL [SUBJ < synsem >
COMPS *top*]]]

ADVP = sign & [SYNSEM LOCAL CAT HEAD basic_adv]

WHADVP = sign & [SYNSEM LOCAL CAT HEAD basic_adv]

$$\begin{aligned} \text{PP} &= \text{sign} \& [\text{SYNSEM LOCAL CAT } [\text{HEAD prep} \& \\ &\quad [\text{MOD *top*}] \\ &\quad \text{VAL COMPS *top*}]] \end{aligned}$$

adjp
ADJP = sign & [SYNSEM LOCAL CAT [HEAD adj
VAL COMPS < >]]

```

,duplicate of ADJP
WHADJP = sign & [ SYNSEM LOCAL CAT [ HEAD adj,
                    VAL COMPS < > ]]

```

```

subject to subject raising for aux verbs would also does
SSR_LINGO = word &
[SYNSEM ssr_subst &
    [ LOCAL CAT VAL [ SUBJ < [ LOCAL local_min & [ CONT INDEX #sind ] ] >
        COMPS < [ LOCAL local_min &
            [ CAT VAL [ SUBJ < [ LOCAL CONT INDEX #sind
                NON LOCAL [ SLASH 0 dlist ] ] >
                COMPS *top* ]] ]   > ]]]
```