Research on Language and Computation (2005) 3:247–279
DOI 10.1007/s11168-005-1296-y

Treebank-Based Acquisition of Multilingual Unification Grammar Resources

AOIFE CAHILL[1], MICHAEL BURKE[1,2], MARTIN FORST[2], RUTH O'DONOVAN[1], CHRISTIAN ROHRER[3], JOSEF VAN GENABITH[1,2] and ANDY WAY[1,2]

[1]*National Centre for Language Technology, School of Computing, Dublin City University, Dublin 9, Ireland (E-mail: josef@computing.dcu.ie);* [2]*Centre for Advanced Studies, IBM Dublin, Ireland;* [3]*Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, D–70174 Stuttgart, Germany*

Abstract. Deep unification- (constraint-)based grammars are usually hand-crafted. Scaling such grammars from fragments to unrestricted text is time-consuming and expensive. This problem can be exacerbated in multilingual broad-coverage grammar development scenarios. Cahill et al. (2002, 2004) and O'Donovan et al. (2004) present an automatic f-structure annotation-based methodology to acquire broad-coverage, deep, Lexical-Functional Grammar (LFG) resources for English from the Penn-II Treebank. In this paper we show how this model can be adapted to a multilingual grammar development scenario to induce robust, wide-coverage, PCFG-based LFG approximations for German from the TIGER Treebank. We show how the architecture of LFG, in particular the distinction between c-structure and f-structure representations, facilitates multilingual, treebank-based unification grammar induction, allowing us to cross-linguistically reuse the lexical extraction and parsing modules from O'Donovan et al. (2004) and Cahill et al. (2004), respectively. We evaluate our grammars against the PARC 700 Dependency Bank (King et al., 2003), against dependency structures for 2000 held-out sentences from the TIGER Corpus as well as against a hand-crafted dependency gold standard for 100 TIGER trees. Currently, our resources achieve 81.79% f-score against the PARC 700, a 2.19% improvement over the best result reported for a hand-crafted grammar in Kaplan et al. (2004), 74.6% against the 2000 held-out TIGER dependency structures and 71.08% against the 100-sentence TIGER gold standard, with substantially improved coverage compared to hand-crafted resources. We have since applied our methodology to induce wide-coverage LFG resources for Chinese (Burke et al., 2004b) from the Penn Chinese Treebank (Xue et al., 2002) and for Spanish from the CAST3LB Treebank (Civit, 2003).

1. Introduction

Deep grammars relate text to information (usually represented in terms of predicate-argument structure, deep dependency relations or logical form).

Rich unification (or rather: constraint-based) grammar formalisms such as LFG (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) or

HPSG (Pollard and Sag, 1994) model both (morpho-)syntactic and semantic information.

Traditionally, deep unification grammars are hand-crafted. Manually scaling such grammars to unrestricted, real text is time-consuming, costly and requires considerable linguistic and computational expertise: person-years of concerted grammar, lexicon and system (processing platform) development effort is involved. What is more, very few hand-crafted grammars achieve *full* coverage of a target corpus the size and complexity of (say) the Penn-II Treebank (Marcus et al., 1994). Indeed, the only hand-crafted, deep unification grammar scaled to the full Penn-II Treebank we are aware of is the English LFG grammar developed as part of the ParGram project at Xerox PARC (Riezler et al., 2002; Kaplan et al., 2004).

This situation, we suspect, is considerably worse for languages other than English, as they have received significantly less (linguistic and computational linguistic) attention. Wide-coverage hand-crafted LFG and HPSG grammars have been developed for German (Müller and Kasper, 2000; Dipper, 2003; Forst, 2003a), Japanese (Siegel and Bender, 2002; Masuichi and Okuma, 2003), Dutch (Bouma et al., 2000) and English (Flickinger, 2000). With the exception of Forst (2003a), Masuichi and Okuma (2003) and Bouma et al., (2000), these grammars do not yet scale to unrestricted newspaper text as complex and varied as that exemplified by the Penn-II Treebank. Accordingly resource problems can be exacerbated in multilingual grammar development scenarios, particularly across typologically different languages.[1]

Recently a number of researchers have addressed the knowledge acquisition problem in broad-coverage, deep, unification grammar development for English: Miyao et al. (2003, 2004) show how HPSG resources and Cahill et al. (2002, 2004) and O'Donovan et al. (2004) show how LFG resources can be induced from the Penn-II Treebank. Hockenmaier and Steedman (2002) and Hockenmaier (2003) show how combinatory categorial grammatical resources can be induced.

In this paper we show how the approach of Cahill et al. (2002, 2004) and O'Donovan et al. (2004) originally developed for English and the Penn-II Treebank can be adapted to a rapid multilingual grammar development scenario to induce wide-coverage LFG resources for German from the TIGER Treebank (Brants et al., 2002). To our knowledge, this is the first time a treebank-based, deep unification grammar induction method has been presented in a multilingual setting. We have since applied our methodology to induce wide-coverage LFG resources for Chinese (Burke et al., 2004b) from the Penn Chinese Treebank (Xue et al., 2002) and for Spanish from the CAST3LB Treebank (Civit, 2003).

LFG minimally involves two levels of representation: c-structure and f-structure. C(onstituent)-structure encodes surface constituency while f(unctional)-structure encodes abstract syntactic relations approximating to predicate-argument or dependency structure. C-structure is the main locus of cross-linguistic variation, while the more abstract f-structure representations are more stable cross-linguistically.

The approach of Cahill et al. (2002, 2004) and O'Donovan et al. (2004) is based on an automatic f-structure annotation algorithm which annotates nodes in Penn-II trees with f-structure equations. From the f-structure-annotated trees, PCFG-based LFG approximations and lexical resources (such as subcategorisation frames) are then extracted automatically.

We show how the architecture of LFG, in particular the distinction between c-structure and f-structure representations, facilitates multilingual, treebank-based unification grammar induction via adaptation of the original f-structure annotation algorithm, allowing us to reuse the PCFG (Cahill et al., 2004) and lexical resources extraction modules (O'Donovan et al., 2004) cross-linguistically.

We evaluate our English and the German grammars against the PARC 700 (King et al., 2003), against 2000 held-out TIGER dependency structures and against a manually constructed gold standard of 100 German dependency structures. Currently, our resources achieve 81.79% f-score against the PARC 700, a 2.19% improvement over the best result reported for a hand-crafted grammar in Kaplan et al. (2004), 71.08% against a manually constructed gold standard of 100 German dependency structures, and 74.6% against 2000 held-out TIGER dependency structures, with substantially improved coverage compared to hand-crafted resources.

The paper is structured as follows: we introduce LFG and motivate why LFG provides a suitable representation format for multilingual grammar development (Section 2). In Section 3, we review the basic ideas underlying the approach first presented in Cahill et al. (2002) as applied to English and present the most recent results (Cahill et al., 2004; O'Donovan et al., 2004). In Section 4, we show how this approach can then be adapted and migrated to a different language and treebank resource, namely German and the TIGER Treebank (Brants et al., 2002). German is substantially less configurational than English, and the TIGER Treebank data structures consist of graphs with crossing edges rather than trees with traces (as in Penn-II). In addition, the TIGER Treebank features considerably richer functional annotations than those provided in the Penn-II resource. We present an f-structure annotation algorithm for TIGER (Section 4.2), outline how PCFG-based LFG approximations for German can be derived from the f-structure-annotated TIGER resource and report a number of parsing experiments in Section 5. We present work on automatic extraction

of lexical resources from the f-structure-annotated TIGER Treebank (Section 6). In Section 7, we address some of the larger issues contrasting development time and cost of manual vs. rapid automatic multilingual treebank-based deep unification grammar development and address the question as to whether and to what extent treebank-based induction of unification grammar resources *is* grammar development. Finally, we conclude and outline some avenues for further research.

2. Lexical-Functional Grammar and Multilingual Grammar Development

Lexical-Functional Grammar (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) is an early member of the family of unification (or constraint-based) grammars (such as FUG, PATR-II, GPSG, CUP or HPSG). Minimally, LFG involves two levels of representation: c(onstituent)-structure and f(unctional)-structure. C-structure captures language-specific phenomena such as word order and the grouping of constituents into larger phrases in the form of context-free trees. F-structure represents abstract syntactic functions (SUBJ(ect), OBJ(ect), PRED(icate) etc.) in the form of recursive attribute-value matrices approximating to basic predicate-argument or dependency structure representations. Van Genabith and Crouch (1996) and van Genabith and Crouch (1997) have shown that f-structures can, in fact, be interpreted as QLFs or UDRSs. Cahill et al. (2003) generate simple QLFs from the complete set of f-structures automatically generated from the Penn-II Treebank.[2] C-structure and f-structure representations are related in terms of 'functional annotations' of the form $\uparrow \ldots = \downarrow \ldots$ to tree nodes, i.e. attribute-value structure equations (or more generally: disjunctive, implicational and negative constraints) describing f-structures.

While languages may differ markedly with respect to surface realisation (c-structure), they may still exhibit very similar abstract syntactic functional representations (f-structure). For example, Irish is typologically a VSO-language, while English is an SVO-language. As Figure 1 illustrates, the same proposition expressed in Irish and English exhibits different c-structure configurations but is associated with isomorphic (up to the values of PRED nodes) f-structure representations. The LFG architecture has been designed to cater for typologically different languages[3] and is particularly attractive for multilingual grammar development as the level of f-structure representation abstracts away from certain (but not all[4]) aspects of language-specific surface realisation (Butt et al., 1999). At the same time LFG provides a precise, flexible, computationally tractable and non-transformational interface between c-structure and f-structure representation for both parsing and generation (Butt et al., 2002). Furthermore, LFG has enjoyed a substantial body of earlier work on automatic f-structure annotation architectures summarised in Frank et al. (2003). These approaches

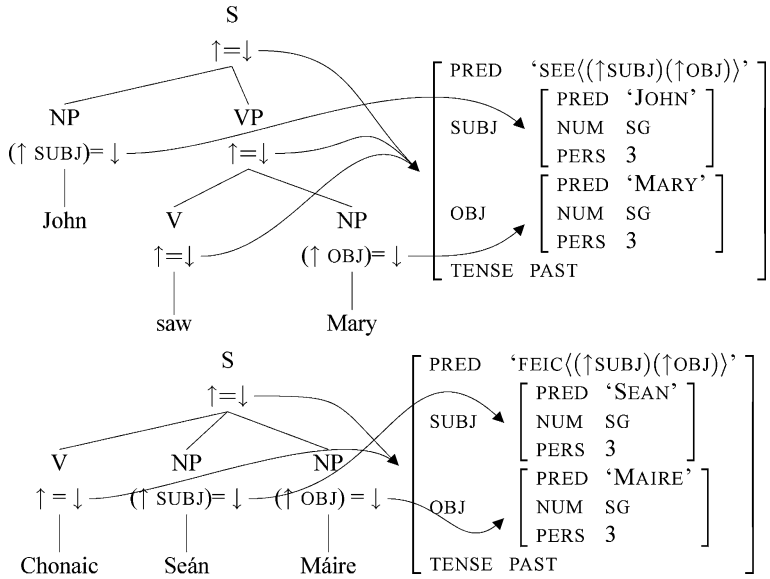

Figure 1. C- and f-structures for an English and corresponding Irish sentence.

automatically annotate (treebank or parse-generated) trees with f-structure equations to generate f-structures for those trees.

3. LFG Resources Acquired from Penn-II

This section describes how broad-coverage PCFG-based LFG approximations and lexical resources can be derived from the Penn-II (Cahill et al., 2002, 2004; O'Donovan et al., 2004). The approach is based on an automatic f-structure annotation algorithm that annotates nodes in Penn-II Treebank trees with f-structure equations. We report on the LFG subcategorisation frames extracted from the f-structures for the Penn-II trees (O'Donovan et al., 2004) and present two parsing architectures – a 'pipeline' and an 'integrated' model – to analyse new text (Cahill et al., 2004). We evaluate the parsers against the PARC 700 Dependency Bank (King et al., 2003).

3.1. F-Structure annotation algorithm for penn-ii

Cahill et al. (2002, 2004) present an automatic f-structure annotation algorithm for the trees in the Penn-II Treebank. Given a tree, the task of the f-structure annotation algorithm is to annotate tree nodes automatically with functional equations. As a simple example, consider the CFG rule

(i.e. a local tree of depth 1) in (1):

$$\text{NP} \rightarrow \text{DT ADJP NN SBAR} \tag{1}$$

Such a configuration would be associated with f-structure annotations as in (2):

$$\begin{array}{lcccc} \text{NP} \rightarrow & \text{DT} & \text{ADJP} & \text{NN} & \text{SBAR} \\ & \uparrow \text{SPEC} = \downarrow & \downarrow \in \uparrow \text{ADJ} & \uparrow = \downarrow & \downarrow \in \uparrow \text{RELMOD} \end{array} \tag{2}$$

The annotations indicate that the NN is the head of the NP, the DT is a specifier, the adjective phrase is part of the modifying adjunct set, and the SBAR is a member of the set of relative clause modifiers.

The f-structure annotation algorithm automatically transforms trees into head-lexicalised trees using a variant[5] of the head rules of Magerman (1994) and then uses configurational, categorial, Penn-II functional tag (such as -LOC, -TMP, -SBJ, -LGS, ...) as well as trace information encoded in the Penn-II Treebank trees to associate tree nodes with f-structure equations from which a constraint solver generates f-structures. The annotation algorithm is modular with four components (Figure 2): left-right (L-R) annotation principles (e.g. leftmost NP to right of V head of VP type rule is likely to be an object in English etc.);[6] coordination annotation principles (separating these out simplifies other components of the algorithm); traces (translates traces and coindexation in trees into corresponding reentrancies in f-structure ($\boxed{1}$ in Figure 3)); catch all and clean-up. Lexical information is provided via macros for POS tag classes. These modules are largely language-specific. The L-R annotation principles are derived by examining the most frequent CFG rules extracted from the treebank and using linguistic expertise to assign f-structure annotation generalisations. For example, the VP matrix has 97 entries, while the NP matrix has 107.[7] The f-structure annotation algorithm constitutes a principal-based c-structure/f-structure interface. In a recent experiment, we ported the f-structure annotation algorithm to the ATIS transcribed spoken language airline reservation corpus (Hemphill et al., 1990). While the c-structure component of our resources required retraining, the f-structure annotation algorithm carried over in its entirety, showing that the linguistic information encoded in the algorithm is complete with respect to domain variation as exemplified by the ATIS corpus.

We evaluate the f-structures for the 48,424 Penn-II Treebank trees without FRAG or X categories quantitatively and qualitatively. As Table I shows, currently, over 99.8% of the treebank trees receive one covering and connected f-structure.[8]

While almost all trees receive a single covering and connected f-structure, this figure is, of course, no measure of the quality of these

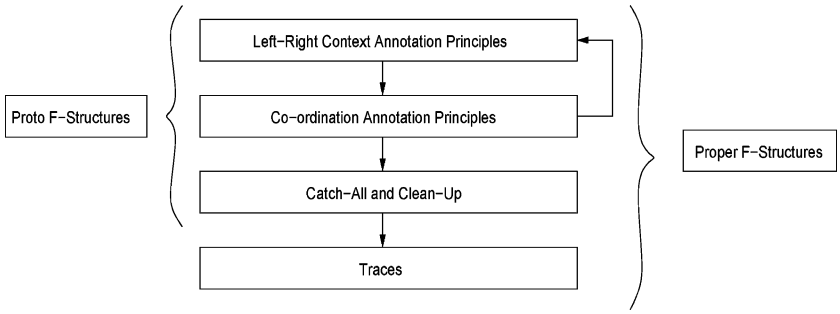

Figure 2. Outline of algorithm to generate proto and proper f-structures.

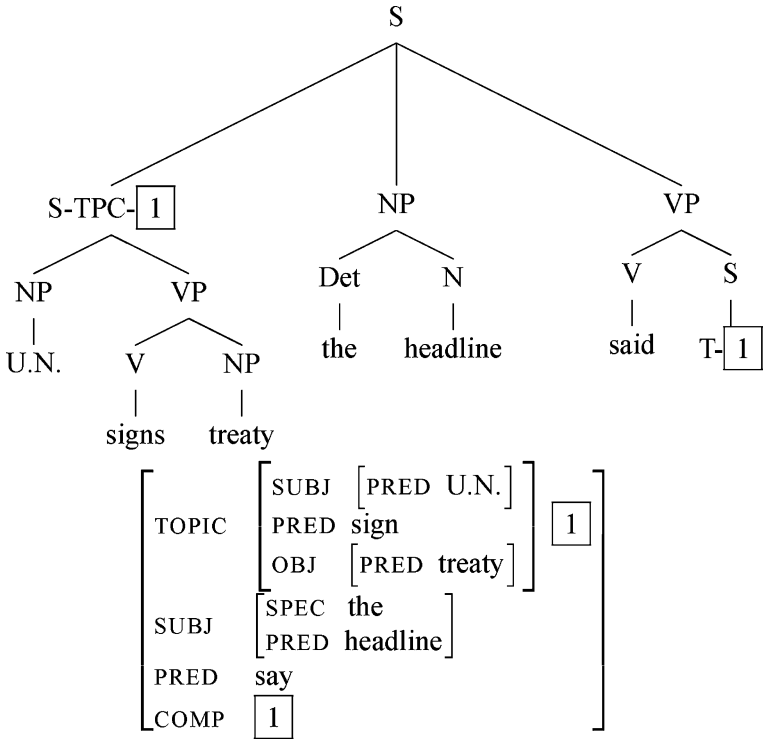

Figure 3. Penn-II style tree with LDD trace and corresponding reentrancy in induced f-structure.

Table I. Coverage & fragmentation results

# f-str. frags	# Sent	Percent
0	79	0.163
1	48343	99.833
2	2	0.004

Table II. Precision and Recall on f-structures against the DCU 105—a manually encoded set of 105 gold standard f-structures from section 23

	All annotations	Preds-only
Precision	96.07	93.38
Recall	96.44	93.97
F-Score	96.25	93.67

automatically derived f-structures. In order to evaluate how good they are, we manually constructed the DCU 105, a set of gold standard reference f-structures for 105 sentences (ave. 24 words, min. 2 words, max. 45 words) randomly extracted from Section 23 of the WSJ part of the Penn-II Treebank. These were annotated by hand, and after a number of iterations, refined to provide a set of complete, correct annotations. In the construction of the gold standard, we endeavoured to create a set of 'perfect' f-structures containing attribute-value pairs which may or may not be encodable by our automatic method. That is, they resemble a set of LFG f-structures which a skilled linguist might produce, rather than what might be constructed *automatically*. In our qualitative evaluation, the task that our automatic annotation method is confronted with is to match as many of the correct annotations from the DCU 105 as possible. We use the dependency evaluation software of Crouch et al. (2002) to evaluate. We currently achieve an f-score of 93.67% preds-only,[9] and 96.25% for complete f-structures (Table II). The result for all annotations is higher than for preds-only f-structures as local number and person features (for example) are often determined correctly even if the f-structure component of the corresponding local pred is misattached in the global f-structure.

3.2. Two parsing architectures

Based on the LFG-annotated treebank, Cahill et al. (2002, 2004) present two parsing architectures to parse new text: a *pipeline* and an *integrated* model. In the pipeline model, a PCFG is extracted from the unannotated treebank and used to parse new text into trees. The resulting parse-trees are then passed into the automatic f-structure annotation algorithm to generate f-structures. We also use sophisticated lexicalised history-based parsers (Collins, 1999; Charniak, 2000) in place of our PCFGs in the pipeline architecture. In the integrated model, we extract an annotated PCFG (A-PCFG) where each non-terminal symbol in the grammar has been augmented with LFG functional equations, such as

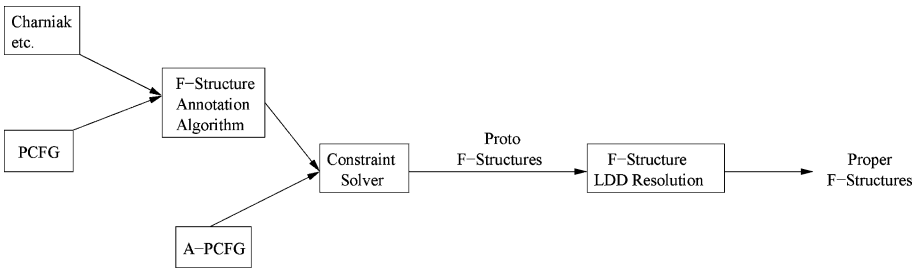

Figure 4. Pipeline (PCFG, Charniak etc.) and integrated (A-PCFG) architectures.

NP[↑OBJ=↓] → DT[↑SPEC=↓] NN[↑=↓] . We treat a node followed by annotations as a monadic category for grammar extraction and parsing. Parsing with A-PCFG results in annotated parse trees, from which an f-structure can be generated. The overall architecture of our system is given in Figure 4.

3.2.1. *Long-Distance Dependencies*

Standard PCFG parsing technology does not recover long-distance dependencies (LDDs) represented in terms of empty nodes, traces and coindexation in the original Penn-II trees. Cahill et al. (2004) present a method to resolve LDDs at the level of f-structure based on a finite approximation of functional uncertainty equations (Kaplan and Zaenen, 1989) automatically acquired from the f-structure-annotated Penn-II Treebank resource. Prior to LDD resolution, both parsing architectures parse raw text into 'proto' f-structures with LDDs unresolved, resulting in possibly incomplete argument structures as in Figure 5, where the phrase *U.N. signs treaty* is correctly analysed as a TOPIC, but is not resolved as the COMP argument of the main verb *say*.

In LFG, LDDs are resolved at the f-structure level, obviating the need for empty productions and traces in trees, using functional uncertainty (FU) equations. FU equations are regular expressions specifying paths in f-structure between a source (where linguistic material is encountered) and a target (where linguistic material is interpreted semantically). For the example given in Figure 5, an FU equation of the form ↑ TOPIC = ↑ COMP* COMP is required, i.e. the value of the TOPIC attribute is token identical with the value of the final COMP argument along a path through the immediately enclosing f-structure along zero or more COMP attributes. For (say) a topicalised constituent to be resolved as the argument of a local predicate as specified by a FU equation, the predicate must (i) subcategorise for the argument and (ii) the argument must not already be filled locally. Traditionally, FU equations and subcategorisation frames have been hand-crafted.

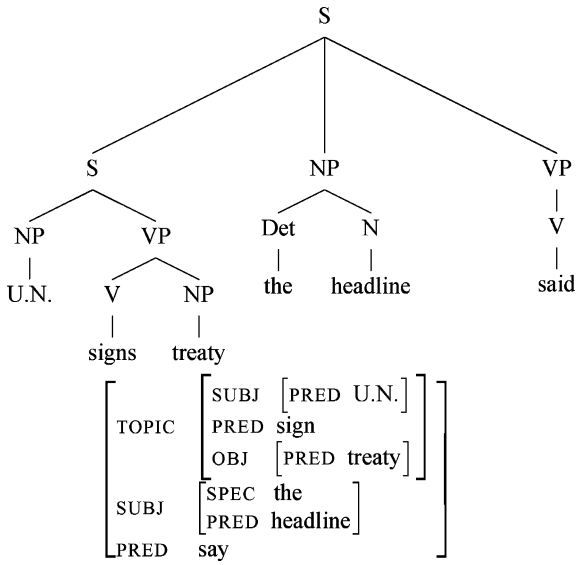

Figure 5. Half-Parser Output with Unresolved LDD and Incomplete Argument Structure.

In our automatically induced grammars we also resolve LDDs at the level of f-structure. Unlike in traditional, manual grammar development, however, we automatically induce finite approximations of FU equations as well as subcategorisation frames from our f-structure-annotated Penn-II Treebank resource.

Recall that the f-structure annotation algorithm described in Section 3.1 correctly translates traces and coindexation in Penn-II trees into corresponding reentrancies in f-structure representing the LDDs. Cahill et al. (2004) extract 14,911 dependency path tokens between coindexed material occurring in the automatically generated f-structures from sections 02-21 of the Penn-II Treebank, resulting in 60 path-types for wh- and wh-less relative clause constructions (TOPICREL), 26 path types for fronted material (TOPIC) and 13 for interrogative constructions (FOCUS), together with their Maximum Likelihood probability estimations. In order to assess the coverage of the approximation, we repeat the same procedure for the held-out data in section 23 and compute how many of the LDD paths attested in section 23 are not covered by the paths extracted from sections 02-21: the total number of path tokens in section 23 is 949 with 3 path types (each occurring only once) not in 02-21. Hence the finite approximation extracted from sections 02–21 covers 99.69% of all LDD path tokens in section 23. Table III shows the most frequent TOPIC-REL path types extracted by our method.

Table III. Most frequent TOPIC-REL path types

TOPIC-REL Paths	#	TOPIC-REL Paths	#
subj	6366	adjunct	1775
obj	898	xcomp:adjunct	677
xcomp:obj	594	xcomp:xcomp:obj	106
xcomp:xcomp:adjunct	104	comp:subj	95

Table IV. Extracted subcategorisation frames for verbs

	Without Prep/Part	With Prep/Part
Lemmas	3586	3586
Sem. Forms	10969	14348
Frame Types	38	577
Active Frame Types	38	548
Passive Frame Types	21	177

In order to resolve LDDs at f-structure, we also require subcategorisation information. LFG distinguishes between governable (arguments) and non-governable (adjuncts) grammatical functions (GFs). Subcategorisation requirements are stated in terms of GFs listed in 'semantic forms' (subcategorisation frames). As an example, the semantic form associated with transitive *see* is SEE ⟨↑SUBJ,↑OBJ⟩. We have shown that our automatic f-structure annotation algorithm generates high-quality f-structures, so that reliable semantic forms can be extracted following van Genabith et al. (1999:72): "For each f-structure generated, for each level of embedding we determine the local PRED value and collect the subcategorisable grammatical functions present at that level of embedding".

We extract argument-taking semantic forms[10] for 3586 verb lemmas, with 10969 unique verbal semantic form types (lemma followed by non-empty argument list). Including prepositions associated with the OBLS and particles, this number rises to 14348, an average of 4.0 per lemma. The number of unique frame types (without lemma) is 38 without specific prepositions and particles, and 577 with. Of the 38 unique frame types, all of them occur at least once as active frames, and 21 of them occur as passive frames. Table IV gives the number of lemmas, semantic forms and frame types extracted. Table V provides the most frequent verb subcategorisation frames for the lemma *accept* with their token occurrence count.

Unlike some other approaches, in our approach we do not predefine subcategorisation frames, we do distinguish between active and passive (marked **p** in Table V) frames and our frames fully reflect the effects of

Table V. Semantic forms automatically extracted for the verb `accept`

Semantic Form	Occurrences	Prob.
`accept`([obj,subj])	122	0.813
`accept`([subj],**p**)	9	0.060
`accept`([comp,subj])	5	0.033
`accept`([subj,obl:as],**p**)	3	0.020
`accept`([obj,subj,obl:as])	3	0.020
`accept`([obj,subj,obl:from])	3	0.020
`accept`([subj])	2	0.013
`accept`([obj,subj,obl:at])	1	0.007
`accept`([obj,subj,obl:for])	1	0.007
`accept`([obj,subj,xcomp])	1	0.007

LDDs in the source data structures. For details on the approach and a full evaluation of the induced resources against COMLEX (Macleod et al., 1994), see O'Donovan et al. (2004).

The LDD resolution algorithm presented in Cahill et al. (2004) traverses an f-structure along LDD paths and ranks possible resolutions by multiplying path and semantic form probabilities involved. It supports multiple, interacting TOPIC, TOPIC-REL and FOCUS LDDs. The algorithm outputs the LDD-resolved f-structure in Figure 3, given the unresolved f-structure in Figure 5.

3.2.2. *Parsing the PARC 700*

Our grammars are induced from sections 02-21 of the Penn-II Treebank and evaluated against the PARC 700 Dependency Bank (King et al., 2003), which contains annotations for 700 randomly selected sentences from Section 23. The PARC 700 is constructed from the f-structures developed in the ParGram project (Butt et al., 2002, Riezler et al., 2002). There are a number of systematic differences between the ParGram and our automatically induced f-structures as regards the feature inventory, feature names, feature geometry and the treatment of named entities. For full details on the mapping from our f-structures to the PARC 700 Dependency Bank see Burke et al. (2004a).[11] We use the BitPar parser software from Schmid (2004) and the dependency evaluation software from Crouch et al. (2002) and Riezler et al. (2002) in our experiments.

Currently our best induced PCFG grammar achieves an f-score of 80.33% following the experimental setup presented in Kaplan et al. (2004),

a 0.73% improvement over the best result for a hand-crafted wide-coverage grammar reported in that work.

In fact, it is possible to obtain even better results: our processing architecture is highly flexible and instead of our own induced PCFGs in the pipeline architecture, for example, we can plug in more sophisticated 'history-based' parsers such as Collins (1999) and Charniak (2000). The parse trees generated by these parsers are then sent to the automatic f-structure annotation algorithm followed by LDD resolution. Currently our best overall result is achieved using Charniak's (2000) parser, with an f-score of 81.79% against the PARC 700 Dependency Bank, an improvement of 2.19% over the best result reported in Kaplan et al. (2004). Table VI gives a detailed breakdown of the results for individual features. Kaplan et al. (2004) report 79% coverage (measured in terms of complete spanning parse) on Section 23 of the Penn-II Treebank. By the same measure, our induced resources achieve coverage of more than 99%.

4. Migrating Automatic Annotation-Based Grammar Acquisition and Parsing to German and the TIGER Treebank

In LFG, much cross-linguistic variation is accounted for at the level of c-structure representation with f-structure a more abstract, stable and uniform level of representation. Accordingly, in *multilingual* treebank-based induction of LFG resources we expect that the f-structure annotation algorithm is the main locus of change in migrating to a different language while 'down-stream' components operating on f-structure such as the lexical extraction component may carry over from one language to the next unchanged. This expectation was in fact borne out in our German grammar induction experiments from the TIGER Treebank (Brants et al., 2002), as we were also able to reuse our PCFG extraction and parsing modules in the German experiments.

In order to migrate the f-structure annotation architecture from English, two major issues need to be addressed: typological differences between English and German and differences between the data structures used to encode linguistic information in Penn-II and the target treebank resource.

The TIGER Treebank is a corpus of approximately 40,000 syntactically annotated German newspaper sentences.[12] The annotation consists of generalised graphs, which may contain crossing and secondary edges. Crossing edges are used to represent LDDs and secondary edges represent information relating to particular re-entrancies such as a shared subject in coordinate constructions. Edges are labelled, so that a TIGER tree encodes both phrase-structural information and rich dependency relations.

In related work, Forst (2003a,b) reports ongoing research to convert the TIGER graphs directly into f-structures in order to generate a set

Table VI. F-score results for the evaluation of Charniak's parser in the pipeline architecture against the PARC 700 broken down by grammatical functions and (morpho-)syntactic features respectively

Dep.	Precision (%)	Recall (%)	*F*-Score (%)
poss	179/202 = 89	179/205 = 87	88
obj	1552/1869 = 83	1552/1866 = 83	83
quant	300/339 = 88	300/381 = 79	83
obl_ag	35/45 = 78	35/45 = 78	78
subj	1192/1280 = 93	1192/1779 = 67	78
conj	403/549 = 73	403/552 = 73	73
adjunct	2539/3581 = 71	2539/3570 = 71	71
comp	187/301 = 62	187/257 = 73	67
xcomp	288/390 = 74	288/478 = 60	66
topicrel	94/203 = 46	94/119 = 79	58
obj_theta	4/25 = 16	4/11 = 36	22
obl	19/49 = 39	19/188 = 10	16
focus	0/0 = 0	0/5 = 0	0
det_form	933/980 = 95	933/964 = 97	96
number_type	402/427 = 94	402/440 = 91	93
pron_form	502/544 = 92	502/531 = 95	93
tense	962/1049 = 92	962/1051 = 92	92
perf	77/84 = 92	77/86 = 90	91
num	3682/4003 = 92	3682/4148 = 89	90
prog	170/180 = 94	170/203 = 84	89
coord_form	225/262 = 86	225/252 = 89	88
stmt_type	921/1086 = 85	921/1044 = 88	86
subord_form	62/75 = 83	62/77 = 81	82
adegree	957/1202 = 80	957/1290 = 74	77
prt_form	33/43 = 77	33/46 = 72	74
pcase	35/44 = 80	35/52 = 67	73
passive	153/200 = 76	153/238 = 64	70
precoord_form	0/0 = 0	0/6 = 0	0

of reference f-structures to evaluate a hand-crafted wide-coverage German LFG. However, in order to be able to extract an f-structure-annotated PCFG which can be used to parse new text into f-structures, we require trees that have been annotated with f-structure equations, rather than the f-structures themselves.

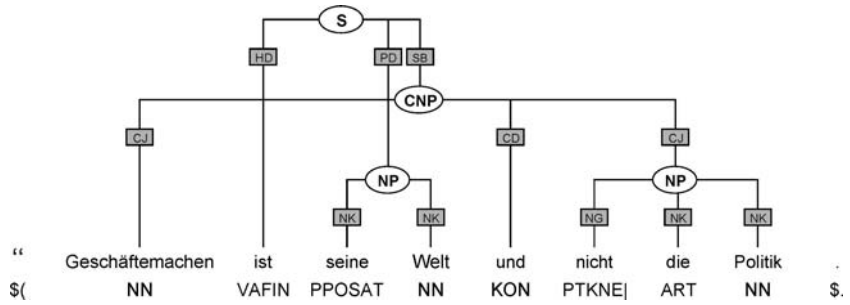

Figure 6. TIGER graph #45, containing crossing edges.

4.1. From TIGER graphs to trees

In order to be able to extract PCFG-based LFG approximations and thereby reuse some of the 'tree-walking' f-structure annotation algorithm code, the TIGER graphs must be automatically converted into trees similar to those found in the Penn-II Treebank.[13] Traces and coindexation in trees are used to represent the LDD information captured by crossing edges in the original TIGER graphs. Secondary edges have not been incorporated into the annotation procedure at this stage, although we hope to be able to exploit them in future work.

Figures 6 and 7 illustrate how traces in tree data structures are used to represent crossing edges.[14] The TIGER graph in Figure 6 indicates by means of crossing edges that both *Geschäftemachen* ('business') and *und nicht die Politik* ('and not the politics') form a discontinuous coordinated constituent, which wraps around the rest of the sentence. This information is represented in terms of traces in the corresponding tree in Figure 7: the subject is a coordinated NP (CNP-SB) consisting of the NN-CJ *Geschäftemachen* and two trace nodes, one for *und* (*T1*) and the other for *nicht die Politik* (*T2*). Finally, note that the functional information encoded in the TIGER graphs is preserved in the graph-to-tree conversion process in terms of labels (-SB, -HD, etc.) on tree node categories.

4.2. Annotation of derived TIGER trees

German does not usually rely on configurational information to express functional information, a feature that was heavily exploited in our work on English. However, the TIGER Corpus annotation scheme provides rich functional information by way of labelled edges in the graphs. By exploiting these labels we can annotate the TIGER Corpus with default f-structure equations. The annotation of the trees derived from the TIGER graphs is a four-stage process, with a trace pre- and a trace post-processing phase, a default and a correction phase as shown in Figure 8.

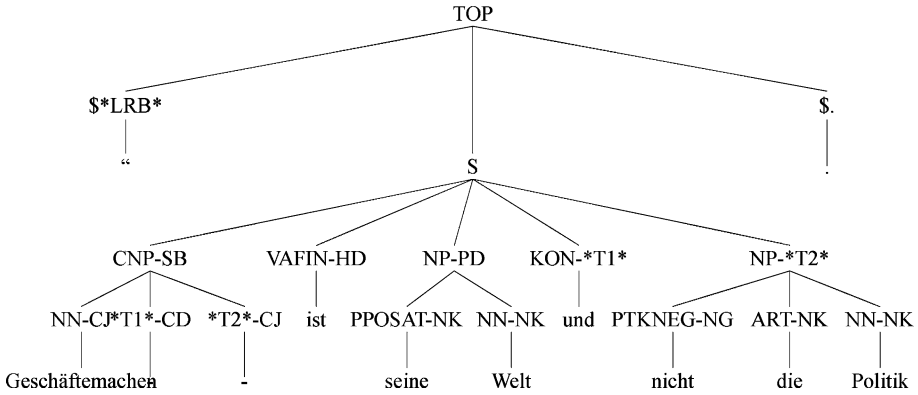

Figure 7. TIGER graph #45 in Figure 6 automatically transformed into a Penn-II style tree with coindexation and traces.

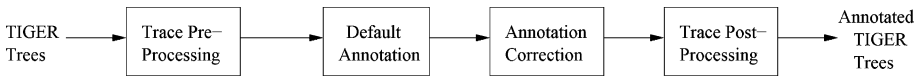

Figure 8. F-structure annotation algorithm for TIGER trees.

Trace pre-processing is a simple walk through the tree in order to build a lookup table for the trace nodes. This is required since often the trace occurs before the coindexed node in the tree, and the information on the node realising the displaced material is needed in order to assign an f-structure equation to the trace node in subsequent modules of the annotation algorithm. Table VII presents the lookup table generated by the tree in Figure 7.

The second stage of the TIGER tree annotation process attempts to assign default f-structure equations to each node based on the TIGER functional labels in the tree. We have compiled a lookup table which assigns default f-structure equations triggered by each TIGER functional label, e.g. the default entry for the SB (subject) label is $\uparrow$ SUBJ = $\downarrow$. However, the default annotation process sometimes needs to be overridden: the NK label (noun kernel element) alone is often ambiguous, though given some context, it can be straightforward to determine the f-structure

Table VII. The lookup table generated in the pre-processing stage for the tree in Figure 7

Trace	Trace function	Node number	Node label
T1	CD	12	KON
T2	CJ	13	NP

equation required, e.g. an ART (article) node with an NK label can usually be annotated ↑ SPEC:DET = ↓, as in Figure 9.

The third stage in the annotation process involves overwriting the default annotations in certain situations, including:

- determining the object of pre- and post-positions, labelled AC (adpositional case marker);
- determining the behaviour of the CP (complementiser) labelled node;[15]
- determining the head of a coordination phrase with more than one coordinating conjunction.

Figure 10 illustrates how the flat TIGER analysis of a German PP can be annotated to give the correct f-structure analysis by overwriting the default annotations on the trees in Figure 9. Initially, the default annotations marked the ART-NK as a specifier, and the NN-NK as an adjunct. In Figure 10, these have correctly been overwritten as the specifier of the object of the preposition, and that object respectively.

The final trace post-processing stage explicitly links trace nodes and the reference node. This involves adding equations such as ↑XCOMP:OBJ = ↓ to nodes with trace information. For example, in Figure 11, the NP-*T2* node receives the annotation ↓∈↑SUBJ:CONJ. Figures 7 and 11 illustrate a complete annotation of a TIGER tree and Figure 12 provides the resulting f-structure. Despite the dislocation of the conjuncts in the trees, these are correctly brought together in the resultant f-structure.

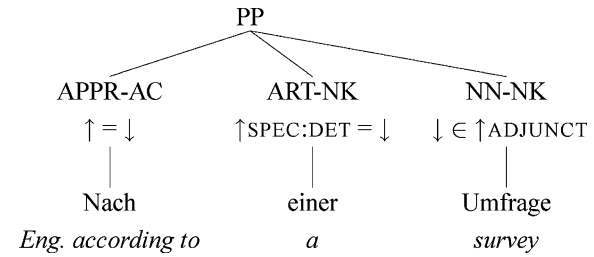

Figure 9. A flat analysis of a German PP and its default f-structure annotations.

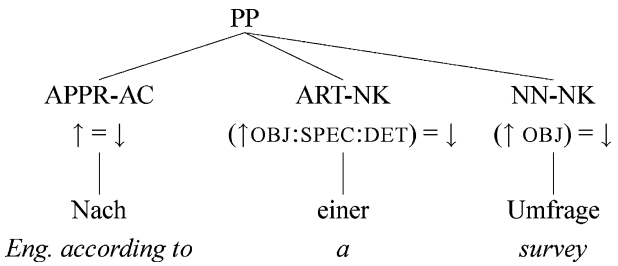

Figure 10. A flat analysis of a German PP and its correct f-structure annotations after stage three of the TIGER tree annotation process.

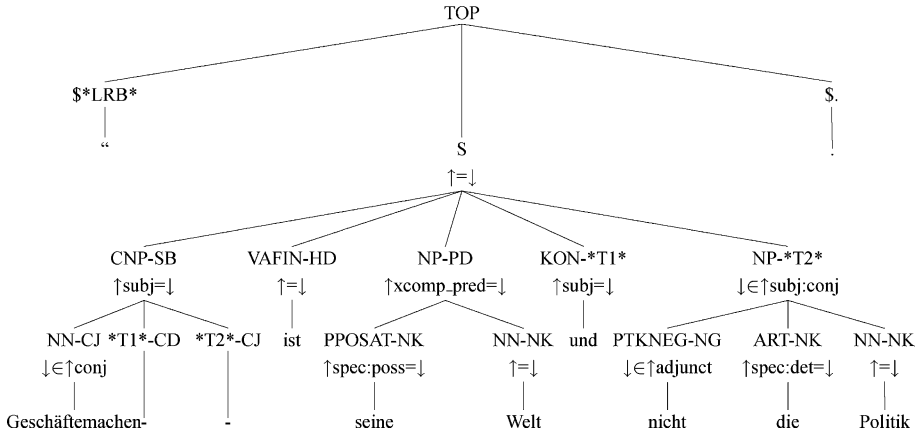

Figure 11. The Penn-II style tree in Figure 7 for TIGER graph #45 after automatic annotation.

$$\begin{bmatrix} \text{XCOMP\_PRED} & \begin{bmatrix} \text{SPEC} & \begin{bmatrix} \text{POSS} & \begin{bmatrix} \text{PRED} & \text{pro} \end{bmatrix} \end{bmatrix} \\ \text{PRED} & \text{Welt} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{CONJ} & \left\{ \begin{bmatrix} \text{PRED} & \text{Geschäftemachen} \end{bmatrix}, \begin{bmatrix} \text{PRED} & \text{Politik} \\ \text{SPEC} & \begin{bmatrix} \text{DET} & \begin{bmatrix} \text{PRED} & \text{die} \end{bmatrix} \end{bmatrix} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{nicht} \end{bmatrix} \right\} \end{bmatrix} \right\} \\ \text{COORD\_FORM} & \text{und} \end{bmatrix} \\ \text{PRED} & \text{ist} \end{bmatrix}$$

Figure 12. The f-structure produced as a result of automatically annotating the tree in Figure 7.

4.3. Penn-II and TIGER annotation algorithm correspondences

In addition to sharing some of the ‘tree-walking’ code, there are some basic design correspondences between the f-structure annotation algorithms for Penn-II and TIGER: both algorithms separate out the treatment of traces and coindexation to a separate component (in fact two in the case of TIGER); and both feature slightly over-generalising components (Left-Right annotation and coordination for Penn-II, defaults for TIGER) followed by a correction component (repairing over-generalisations). This design supports a simple and perspicuous statement of linguistic generalisations and is essential for the maintainability and extensibility of the annotation algorithms: generalisations are allowed to over-generalise and the often complex conditions preventing the application of generalisations

are confined to the corrections component (thus avoiding 'pollution' of the generalisations by complex exception conditions).

4.4. Evaluation of the automatic annotation algorithm

We first established the coverage of the annotation algorithm on the entire TIGER Corpus. Table VIII presents the results. Almost 97% of the 40,000 sentences receive one covering and connected f-structure. Ideally we would like to generate just one f-structure per sentence. There are, however, a number of sentences (1112) that receive two or more disconnected f-structure fragments. This is mainly due to strings such as *Bonn, 7. September*, where in the source TIGER graphs there is no clear relation between the individual constituents of the string and where we do not wish to enforce a relation at f-structure level for the sake of having fewer fragments. There are also a small number of sentences which do not receive any f-structure. This is due to feature clashes in the annotated trees, which—as for the Penn-II-based English annotation—are caused by inconsistent annotations.

We also evaluate the quality of the annotation against a manually constructed gold standard of 100 f-structures. In our parsing experiments, we set aside sentences 8001–10000 of the TIGER Treebank for testing purposes. We extracted 100 sentences at random from these 2000 sentences, in order to develop our gold standard. The original TIGER trees for these sentences were converted into dependency structures following Forst (2003a) and manually corrected. We use the triple encoding and evaluation software of Crouch et al. (2002). Tables IX and X show that currently our automatic f-structure annotation achieves a preds-only f-score of 90.22% against this gold standard, with precision about 7 points higher than recall, indicating that our algorithm tends to be more partial than incorrect. With respect to specific syntactic functions, most features achieve very high

Table VIII. Coverage & fragmentation results of German f-structure annotation algorithm

# f-str. fragments	# sent	percent
0	143	0.3573
1	38765	96.8641
2	1032	2.5787
3	75	0.1874
5	1	0.0025
6	1	0.0025
7	3	0.0075

Table IX. Preds-only evaluation of automatically annotating TIGER trees broken down by grammatical functions and features respectively

Dependency	Precision	Recall	*F*-score
dem	7/7 = 100	7/7 = 100	100
obj2	5/5 = 100	5/5 = 100	100
obl	11/11 = 100	11/11 = 100	100
obl-ag	6/6 = 100	6/6 = 100	100
adj-gen	79/81 = 98	79/82 = 96	97
det	269/277 = 97	269/283 = 95	96
xcomp	74/77 = 96	74/78 = 95	95
name-mod	29/33 = 88	29/29 = 100	94
obj	315/329 = 96	315/339 = 93	94
xcomp-pred	30/31 = 97	30/35 = 86	91
adjunct	538/584 = 92	538/605 = 89	90
conj	122/145 = 84	122/138 = 88	86
comp	14/15 = 93	14/18 = 78	85
adj-rel	12/14 = 86	12/15 = 80	83
number	15/18 = 83	15/18 = 83	83
app	25/26 = 96	25/35 = 71	82
poss	16/18 = 89	16/22 = 73	80
obl-compar	3/5 = 60	3/3 = 100	75
subj	148/151 = 98	148/244 = 61	75
quant	12/14 = 86	12/22 = 55	67
app-clause	5/9 = 56	5/7 = 71	63
topic	0/1 = 0	0/0 = 0	0
circ-form	2/2 = 100	2/2 = 100	100
part-form	11/11 = 100	11/11 = 100	100
coord-form	57/58 = 98	57/59 = 97	97
pron-type	16/16 = 100	16/17 = 94	97
comp-form	11/11 = 100	11/15 = 73	85

results, e.g. f-score for OBJ is 94%, XCOMP is 95%, and COORD-FORM is 97%. The features that we score poorest on are APP-CLAUSE and QUANT, but there are relatively few occurrences of these features (7 and 22 respectively) in the gold standard. We expect all figures to improve as we continue to refine the f-structure annotation algorithm.

Table X. Evaluation of the f-structures produced by automatically annotating the TIGER trees against 100 gold standard f-structures

	Preds Only Evaluation
Precision	93.71%
Recall	86.99%
F-Score	90.22%
Complete Match	25

5. TIGER Parsing Experiments

Using the annotation method described above, we automatically annotate the TIGER Corpus with f-structure equations. We then read off a grammar from the annotated treebank, resulting in an *annotated* PCFG (A-PCFG) for German. We again use the BitPar parser (Schmid, 2004) to parse with this grammar, using Viterbi pruning to obtain the most probable parse. Using the same method as described in Cahill et al. (2002, 2004), we collect the f-structure annotations from the resulting parse tree and use a constraint solver to produce an f-structure for new text. All experiments reported here instantiate the integrated parsing architecture described in Section 3.2.

An annotated grammar (A-PCFG) was extracted from the TIGER Corpus (excluding the 2000 sentences set aside for testing). Prior to grammar extraction, empty productions were removed from the TIGER trees while TIGER functional labels were kept. We also transformed the grammar using a parent transformation (Johnson, 1999) to give us PA-PCFG. Using these two grammars, we parsed the 2000 raw, untagged test strings. The results are presented in Table XI. We evaluated the quality of the trees produced by the parser and measure how many of the 2000 sentences produce one covering and connected f-structure. Of the sentences parsed by the A-PCFG (1993 of the original held-out 2000), 1916 receive one covering and connected f-structure. Of the sentences parsed by PA-PCFG (1990), 1958 receive one covering and connected f-structure.

We evaluate the quality of the f-structures produced in two ways. First we evaluate against our manually constructed gold standard of 100 f-structures. Second, in a CCG-style experimental setup (Hockenmaier, 2003), we automatically annotate the 2000 held-out original treebank trees with our f-structure annotation algorithm, and evaluate the f-structure output of the parser for the 2000 raw, untagged strings (from the corresponding trees) against the *automatically* produced f-structures for the *original*

TIGER trees. We currently achieve a labelled f-score of 69.39% on the trees produced by A-PCFG. For the same grammar, the f-structures achieve an f-score of 71% and 74.61% on the 100 gold standard f-structures and the 2000 automatically produced f-structures respectively. Johnson (1999) shows that applying the parent transformation to English results in improved parse accuracy, but this is not the case for German. Our experiments show that there is a decrease of 1.25% in labelled f-score for the trees and a corresponding decrease of 0.5% in labelled f-score against the 100 gold standard f-structures, and a decrease of 0.56% against the 2000 f-structures. In fact, the only respect in which PA-PCFG outperforms the A-PCFG is in fragmentation with a decrease of 2.1% in fragmentation from A-PCFG to PA-PCFG. Forst (2003a) reports coverage of around 70% (measured in terms of complete spanning parse trees) for the hand-crafted German ParGram LFG. By the same measure, our automatically induced grammars achieve coverage of around 99%.

6. Extracting Lexical Resources from TIGER

O'Donovan et al. (2004) have extracted subcategorisation frames from the f-structures generated from the Penn-II Treebank (Section 3.2.1). These were used to resolve long-distance dependencies in the English experiments. Using the same methodology and, in fact, reusing the extraction software, we have automatically extracted a lexical resource for German. The subcategorisation frames have not yet been incorporated into the German parsing experiments.

For German, the system extracts the following subcategorisable grammatical functions: SUBJ, OBJ, OBJ2, OBJ_GEN, AOBJ2, OBL, COMP, XCOMP, XCOMP_PRED and PART.

Table XI. Parsing results

	A-PCFG	PA-PCFG
# Rules	65758	72127
# Parses	1993	1990
Lab. F-Score (2000 Trees)	69.39%	68.14%
Unlab. F-Score (2000 Trees)	73.72%	73.16%
Tagging Accuracy (2000 Trees)	95.47%	80.20%
Fragmentation (2000 f-structures)	95.8%	97.9%
F-Score (100 f-structures)	71.00%	70.50%
F-Score (2000 f-structures)	74.61%	74.04%

From the f-structure-annotated TIGER Treebank, we extract 8632 argument-taking semantic forms in total, 7081 of which are for 4331 verb lemmas, i.e. 1.63 semantic forms per verb when the OBL functions are parameterised for individual prepositions. We do not predefine the frames to be extracted by our system. Table XII shows the numbers of distinct frame types extracted, ignoring `pred` values. We also show the result of applying thresholding techniques to the semantic forms induced. Table XIII shows the attested semantic forms for the verb *aufhören* ('stop') with their associated conditional probabilities.

The rate of acquisition of lexical information can be expressed as a measure of the coverage of the induced lexicon on new data. To demonstrate this, we extract a reference lexicon from trees 1–8000 and 10001–40020 of the TIGER Treebank. We then compare this to a test lexicon from trees 8001-10000. Table XIV shows the results of the evaluation of the coverage of an induced lexicon for verbs only. For 86.75% of the verbs extracted from trees 8001–10000, the corresponding semantic form exists in the reference lexicon. 13.25% of the entries in the test lexicon did not appear in the reference lexicon. Of these, we can distinguish between known words, which have an entry in the reference lexicon, and unknown words, which do not exist at all in the reference lexicon (but without the required frame). In the same way we make the distinction between known frames and unknown frames. There were no unknown frames. Table XIV

Table XII. Number of distinct frames (including specific preposition) for verbs

	# Sem Forms with distinct `obls`
# Frame Types	189
# Singletons	91
# Twice Occurring	23
# Occurring max. 5	130
# Occurring > 5	59

Table XIII. Semantic forms with associated conditional probability for the verb *aufhören*

Extracted Semantic Form	Occurrences	Conditional Probability
`aufhören([subj])`	8	0.444
`aufhören([subj,xcomp])`	7	0.389
`aufhören([subj,obl:mit])`	2	0.111
`aufhören([subj,comp])`	1	0.056

Table XIV. Coverage of induced lexicon on unseen data (Verbs Only)

Entries also in reference lexicon:	86.75%
Entries not in reference lexicon:	13.25%
Known words:	7.00%
– Known words, known frames:	7.00%
– Known words, unknown frames:	0
Unknown words:	6.24%
– Unknown words, known frames:	6.24%
– Unknown words, unknown frames:	0

shows that 7% of the unknown entries are known verbs occurring with a different, although known, subcategorisation frame. Interestingly, none of the unknown words in the test data had unknown semantic forms when PRED values are abstracted away. This is surprising as data-sparseness is usually an issue in learning subcategorisation frames.

7. Issues in Multilingual Treebank-Based Unification Grammar Induction

Rapid, Multilingual, Deep Grammar Induction: We have presented a treebank-based methodology for the rapid induction of high-quality, deep and wide-coverage, multilingual unification grammar resources addressing the knowledge acquisition problem in rich unification grammar development, familiar from other areas in traditional, rule-based approaches in NLP and AI. The German LFG resources presented here were induced from the TIGER Treebank with a total of approximately four person-months' development time (including the parsing and evaluation experiments), yet they parse unseen unrestricted newspaper text with (we believe) entirely respectable results. We strongly contend that similar resources could not have been developed manually in the same amount of time. The automatically induced resources score over the manually developed grammars (Forst, 2003a, b) as regards coverage (measured in terms of complete spanning parse trees) with $> 99\%$ for the former as against 70% for the latter on unseen TIGER newspaper text. Against the PARC 700 Dependency Bank, the automatically induced grammatical resources for English outperform the best hand-crafted deep LFG grammars reported in Kaplan et al. (2004), again with considerably less development effort. We conjecture that the German resources can be improved significantly with another three person months of concerted development effort. Grammatical function assignment in German is to a considerable extent determined by morphological (case) information. At the time our experiments were carried

out, morphological information was not available in the TIGER Treebank and we hope to be able to conduct further experiments *simulating* morphological information through grammar transformations or learning morphological information from the fully morphologically annotated final version of the TIGER Treebank.

Treebank Development Cost: It is often argued that the *true* cost of our multilingual, treebank-based, deep unification grammar induction methodology should include the very substantial cost incurred from the treebank development in the first place. We argue against this: first, wide-coverage hand-crafted, deep unification grammars do, in fact, use treebank resources too in order to train disambiguation modules. To give just one example, Riezler et al. (2002) show how a discriminative (log-linear) disambiguation module for parse selection for the English ParGram LFG grammar is trained using the Penn-II Treebank resource. Parse selection is *essential* for large coverage deep grammars as the number of parses per string can grow exponentially with string length. Similarly, the TIGER Treebank resource would be essential to train a disambiguation module for the German ParGram grammar and work is in fact under way on compiling TIGER into a reference f-structure bank for evaluating the output of the German hand-crafted ParGram grammar (Forst, 2003a, b). If treebank development cost were to be factored into the cost incurred in our induction methodology, then it needs to be included into the cost of every wide-coverage, hand-crafted resource that uses parse selection/disambiguation modules trained on treebank resources. All else being equal, treebank-based induction of resources is cheaper and faster than hand-crafting. Second, treebanks are multi-purpose resources – count the number of publications/approaches involving the Penn-II Treebank as a resource. If the cost of developing Penn-II is to be included in our grammar induction methodology, then the only cost that can reasonably be added is that portion of the Penn-II development cost obtained from the original cost divided by the number of approaches/papers that have used Penn-II.

Grammar Induction vs Development: Sometimes the question is raised as to whether and to what extent multilingual treebank-based induction of unification grammar resources really *is* grammar development (in the more traditional sense). Is it linguistics or is it 'just' engineering? We would strongly argue the our treebank-based induction of wide-coverage resources *is* grammar development: the automatic annotation algorithm is carefully designed to support the development and encoding of linguistic generalisations (with over-generalisations taken care of by a corrections component). This aspect is brought out even clearer in earlier regular expression- and constraint

rewriting-based approaches to automatic f-structure annotation summarised in Frank et al. (2003), which in fact constitute principle-based c-structure/f-structure interfaces.

Domain Variation: A closely related question is "can one maintain/extend/adapt treebank-induced resources to account for domain variation or do they simply come as a package"? Induced resources can be adapted, but in a different way compared to traditional hand-crafted grammars: try, for example, parsing direct questions with our Penn-II induced LFG resources. The parser *will* give you a result[16] but it will not always be what you expect and indeed want. The reason is that direct questions do not figure much in the Penn-II training resource and hence the analyses produced by the induced grammar underperform on direct questions. In order to extend the induced resources to improve parse quality on direct questions the grammar developer does not write extra rules (as in traditional grammar development), but adds new material to the training corpus, e.g. (a section of) the ATIS treebank which contains a high proportion of direct question constructions and the resulting new induced grammar will show significant improvement on direct questions. A set of experiments on the ATIS corpus showed that in order to achieve good results on the transcribed spoken airline reservation language domain, our induced LFG resources require retraining of the c-structure component, while the linguistic information encoded in the f-structure annotation algorithm is already complete with respect to the ATIS domain.

Joining Forces: Combining induced grammars with hand-crafted resources is a further research possibility: in the simplest case, for example, induced grammars could provide a fallback to boost coverage in case a hand-crafted resource is not able to produce a full spanning parse.

Generation: It is currently an open research question as to whether treebank-induced, multilingual unification grammars can be used for generation. Treebank grammars (such as those induced from Penn-II) are 'loose' and massively ambiguous: they parse almost everything (including 'ungrammatical' input) and if probabilistic parse selection is switched off, will generate staggering numbers of analyses. This does not seem to bode well for generation. The main reason, however, that treebank-induced grammars have not been used for generation is that until recently such grammars did not produce 'deep' semantic representations, such as predicate-argument or dependency structures. Our induced LFG resources, however, map strings into predicate-argument/dependency representations. Research is currently underway to explore whether these 'meaning' representations together with the probabilities associated with the induced

resources can be used to provide successful generation models for treebank-based grammar resources.

Treebank-Induced LFG Resources for Chinese and Spanish: Finally, of course, if there is no treebank available for a target language, our method is of very little use. That said, a recent volume on treebanks (Abeilé, 2003) references treebanks for English, Chinese, Japanese, Arabic, Spanish, French, Italian, German, Bulgarian, Polish and Turkish and many further resources are currently under construction or near completion. Using our method, we have since induced deep LFG resources (grammars and lexical resources) for Chinese (Burke et al., 2004b) from the Chinese Penn Treebank (Xue et al., 2002) and Spanish from the CAST3LB Treebank (Civit, 2003) and hope to be able to report on our Spanish LFG resources elsewhere.

8. Conclusions

We have presented a new rapid multilingual unification grammar acquisition approach based on treebank resources and automatic f-structure annotation algorithms, which addresses the knowledge acquisition bottleneck in deep grammar development. This method can offer substantially reduced grammar development cost if a treebank is available and complement and in certain cases replace traditional manual grammar development.

The resources induced are wide-coverage (they parse unrestricted newspaper text) and deep. We evaluate the English resources parsing the PARC 700 Dependency Bank. Currently our best result achieves a dependency f-score of 81.79%, a 2.19% improvement over the best result for the hand-crafted grammar reported in Kaplan et al. (2004). We test our German resources by parsing 2000 sentences held out from the automatically f-structure-annotated TIGER Corpus and evaluating (i) against a manually constructed gold-standard of f-structures for 100 sentences randomly extracted from the held out sentences, and (ii) against the full 2000 f-structures automatically produced for the *original* 2000 held-out TIGER Treebank trees. The parser produces f-structures which receive an f-score of 71% when evaluated against our manually constructed gold standard, and 74.61% on the 2000 automatically produced f-structures. Both the German and the English grammars achieve around 99% coverage (measured in terms of complete spanning parse trees) on unseen TIGER/Penn-II Treebank German/English newspaper text. The German resources presented here were induced using four person months' development effort. We believe that the resources induced via our method can be considerably improved. Using our method, we have since induced wide-coverage LFG resources for Chinese (Burke et al., 2004b) and Spanish.

Treebank-based grammar induction has also been reported for HPSG (Miyao et al., 2003) and CCG (Hockenmaier and Steedam, 2002). To the best of our knowledge, ours is the first paper which discusses rapid treebank-based unification grammar induction in a multilingual setting.

We have argued for multilingual treebank-based unification grammar induction as a novel grammar development methodology, complementing and in certain cases replacing more traditional manual grammar development. We showed how the architecture of LFG, in particular the distinction between c-structure and f-structure representations, supports multilingual grammar development. C-structure accounts for many aspects of cross-linguistic variation with f-structure providing a more abstract and cross-linguistically stable level of representation. Our unification grammar acquisition methodology is based on an f-structure annotation algorithm annotating c-structure trees with attribute-value structure equations. Accordingly, in migrating our methodology originally developed for English and the Penn-II Treebank resource to German and the TIGER Treebank, most of the changes are located within the f-structure annotation algorithm with 'down-stream' components operating on f-structure such as the lexical resource extraction component directly transferable from English to German.

Acknowledgements

The work reported in this paper was funded by Enterprise Ireland Basic Research Grant SC/2001/186 and by an IRCSET Ph.D. studentship. Thanks to Michael Schiehlen (IMS) who provided the code to convert TIGER graphs into corresponding trees with traces, to Dick Crouch (PARC) for his dependency bank evaluation software and Helmut Schmid (IMS) for BitPar. Thanks are also due to three anonymous reviewers whose extensive and insightful comments have improved this paper.

Notes

[1] Sometimes multilingual grammar development offers unique advantages unavailable to monolingual grammar development: these are cases where an existing grammar for language A can be migrated to a closely related language B for which such resources do not exist and thus significantly boost initial phases of grammar development for B (Gamon et al. (1997), Bender et al. (2002), Kim et al. (2003), King et al. (this volume), Bateman et al. (this volume)). It is currently an open research question whether migration is only possible for core grammars (i.e. small grammars covering core phenomena) or whether this is possible for fully-fledged, broad-coverage grammars.

[2] Semantically irrelevant information (case, grammatical gender etc.) is suppressed in the QLF and UDRS interpretations.

[3] This is reflected in both computational and theoretical work in LFG: the ParGram project (King et al., this volume), for example, currently involves manual computational

grammar development for English, French, German, Norwegian, Danish, Welsh, Japanese, Chinese, Urdu and Malagasy. There is a large body of work on transfer-based Machine Translation operating on f-structure representations (Kaplan et al., 1989). The annual conferences on LFG attract papers on an extremely diverse range of languages: the recent 2004 conference included papers on Australian, Polynesian, African and Native American languages, Japanese, Norwegian, French, English, Greek, Urdu, Korean, German, American Sign Language, Pennsylvania German, Old Florentine, Hungarian, Portuguese, Hebrew, Swedish and Mandarin Chinese.

[4] Headswitching is a case in point: English *John likes swimming* vs. German *John schwimmt gerne* ('John swims likingly').

[5] The order in which non-terminal categories were assigned as heads was changed slightly for some categories, e.g. VP. One example is the category MD (modals), which if it exists, must be the head of a VP (unlike in the ParGram approach).

[6] Argument/adjunct distinctions are notoriously difficult to draw and the annotation algorithm is extremely conservative in this regard using both configurational (sister of V under VP) *and* Penn-II tag information (-CLR for closely related to the local PRED) to classify (say) a PP as an oblique argument. If there is any doubt, the algorithm defaults to an adjunct analysis.

[7] Compare this to the over 6000 NP CFG-rule types extracted from the treebank.

[8] The 79 trees for which no f-structure is generated contain inconsistent functional annotations which result in clashes which the constraint solver cannot resolve. Two trees receive two disconnected f-structure fragments: both trees contain an embedded sentence, where both the subject NP and the predicate head are empty. In such cases our algorithm currently fails to resolve some of the multiple trace nodes.

[9] Preds-only f-structures consider those paths in f-structures ending only in a PRED:lemma attribute-value pair. In other words, preds-only f-structures capture the essential predicate-argument-adjunct or dependency skeleton.

[10] Argument-taking semantic forms contain at least one subcategorised-for grammatical function.

[11] Feature names are renamed to the ParGram standards. We represent strings of auxiliaries as a cascade where each subsequent verb is the XCOMP of the previous one, while in ParGram, flat f-structures are used with the auxiliaries contributing tense and aspect features. In the case of direct speech *"Giveaways just give people the wrong image," said Mr. Heinemann* (from the PARC 700) we take the head verb to be *say*, the subject to be *Mr. Heinemann* and the quoted string as the COMP (and, in this case, also the TOPIC) of the main verb. ParGram, on the other hand, selects the main verb *give* from the quoted string, with *said Mr. Heinemann* encoded as adjunct. In each such case we automatically convert to the ParGram analysis, with possibly some loss of information and a small decrease in our f-score result.

[12] `http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERCorpus/`

[13] We use software provided by Michael Schiehlen to carry out this conversion.

[14] The labels used in Figure 6 are as follows: *Non-terminals* S=Sentence, CNP=Coordinated Noun Phrase, NP=Noun Phrase; *POS Tags* NN=Common Noun, VAFIN=Finite auxiliary Verb, PPOSAT=Attributive Possessive Pronoun, KON=Coordinating Conjunction, PTKNEG=Negative Particle, ART=Definite/Indefinite Article; *Syntactic Functions* CJ=Conjunct, NK=Noun Kernal Element, NG=Negation, CD=Coordinating Conjunction, HD=Head, PD=Predicate, SB=Subject.

[15] In our analysis, true complementisers, i.e. *daß* and *ob*, only contribute a COMP-FORM feature to the f-structure, whereas other conjunctions contribute a semantic form that governs linguistic material.

[16] Coverage is not usually an issue with treebank-induced grammatical resources.

References

Abeillé A. (ed.) (2003) *Treebanks, Building and Using Parsed Corpora*. Kluwer, Dordrecht.

Bender E., Flickinger D., Oepen S. (2002) The Grammar Matrix: An Open-Source Starter-Kit for the Rapid Development of Cross-Linguistically Consistent Broad-Coverage Precision Grammars. In Carroll J., Oostdijk N., Sutcliffe R. (eds.), *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*. Taipei, Taiwan, pp. 8–14.

Bouma G., van Noord G., Malouf R. (2000) Alpino: Wide-coverage Computational Analysis of Dutch. In Daelemans W., Sima'an K., Veenstra J., Zavrel J. (eds.), *Computational Linguistics in The Netherlands 2000*. Rodopi, Amsterdam, pp. 45–59.

Brants T., Dipper S. Hansen S., Lezius W., Smith G. (2002) The TIGER Treebank. In Hinrichs, E., Simov K. (eds.), *Proceedings of the first Workshop on Treebanks and Linguistic Theories (TLT'02)*. Sozopol, Bulgaria, pp. 24–41.

Bresnan J. (2001) *Lexical-Functional Syntax*. Blackwell, Oxford.

Burke M., Cahill A., O'Donovan R., van Genabith J., Way A. (2004a) The Evaluation of an Automatic Annotation Algorithm against the PARC 700 Dependency Bank. In *Proceedings of the Ninth International Conference on LFG*. Christchurch, New Zealand, pp. 101–121.

Burke M., Lam O. Chan R., Cahill A., O'Donovan R., Bodomo A., van Genabith J., Way A. (2004b) Treebank-Based Acquisition of a Chinese Lexical-Functional Grammar. In *Proceedings of the 18th Pacific Asia Conference on Language, Information and Computation*. Tokyo, Japan, pp. 161–172.

Butt M., Dyvik H., King T. H., Masuichi H., Rohrer C. (2002) The Parallel Grammar Project. In *Proceedings of COLING 2002, Workshop on Grammar Engineering and Evaluation*. Taipei, Taiwan, pp. 1–7.

Butt M., King T. H. , Niño M. E., Segond F. (1999) *A Grammar Writer's Cookbook*. Stanford, CA: CSLI Publications.

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*. Barcelona, Spain, pp. 320–327.

Cahill A., McCarthy M., van Genabith J., Way A. (2002) Parsing with PCFGs and Automatic F-Structure Annotation. In Butt M., King T. H. (eds.), *Proceedings of the Seventh International Conference on LFG*. Stanford, CA: CSLI Publications, pp. 76–95.

Cahill A., McCarthy M., van Genabith J., Way A. (2003) Quasi-Logical Forms for the Penn Treebank. In Bunt H., van der Sluis I., Morante R. (eds.), *Proceedings of the Fifth International Workshop on Computational Semantics, IWCS-05*. Tilburg, The Netherlands, pp. 55–71.

Charniak E. (2000) A Maximum Entropy Inspired Parser. In *Proceedings of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2000)*. Seattle, WA, pp. 132–139.

Civit M. (2003) Criterios de etiquetación y desambiguación morfosintáctica de corpus en español. Ph.D. thesis, Universitat de Barcelona, Spain.

Collins M. (1999) Head-Driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Crouch R., Kaplan R., King T. H., Riezler S. (2002) A comparison of evaluation metrics for a broad coverage parser. In *Proceedings of the LREC Workshop: Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems*. Las Palmas, Canary Islands, Spain, pp. 67–74.

Dalrymple M. (2001) *Lexical-Functional Grammar*. Academic Press, San Diego, CA. London.

Dipper S. (2003) Implementing and Documenting Large-scale Grammars – German LFG. Ph.D. thesis, IMS, University of Stuttgart. Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung (AIMS), Volume 9, Number 1.

Flickinger D. (2000) On Building a More Efficient Grammar by Exploiting Types. *Natural Language Engineering* **6**(1), 15–28.

Forst M. (2003a) Treebank Conversion – Creating an f-structure bank from the TIGER Corpus. In Butt M., King T. H. (eds.): *Proceedings of the Eighth International Conference on LFG*. CSLI Publications, Stanford, CA, pp. 205–216.

Forst M. (2003b) Treebank Conversion – establishing a test suite for a broad-coverage LFG from the TIGER treebank. In *Proceedings of the EACL Workshop on Linguistically Interpreted Corpora (LINC'03)*. Budapest, Hungary, pp. 25–32.

Frank A., Sadler L., van Genabith J., Way A. (2003) From Treebank Resources to LFG F-Structures. In Abeillé A. (ed.), *Treebanks: Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers, Dordrecht/Boston/London,The Netherlands, pp. 367–389.

Gamon M., Lozano C., Pinkham J., Reutter T. (1997) Practical Experience with Grammar Sharing in Multilingual NLP. In Burstein J., Leacock C. (eds.), *Proceedings of the Workshop From Research to Commercial Applications: Making NLP Work in Practice, 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL-EACL'97)*. Madrid, Spain, pp. 49–56.

Hemphill C. T., Godfrey J. J., Doddington G. R. (1990) The ATIS spoken language systems pilot corpus. In *Proceedings of a workshop on Speech and natural language*. Morgan Kaufmann Publishers Inc., Hidden Valley, PA. pp. 96–101.

Hockenmaier J. (2003) Parsing with Generative models of Predicate-Argument Structure. In *Proceedings of the 41st Annual Conference of the Association for Computational Linguistics*. Sapporo, Japan, pp. 359–366.

Hockenmaier J., Steedman M. (2002) Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, PA. pp. 335–342.

Johnson M. (1999) PCFG models of linguistic tree representations. *Computational Linguistics* **24**(4), 613–632.

Kaplan R., Bresnan J. (1982) Lexical Functional Grammar, a Formal System for Grammatical Representation. In Bresnan J. (ed.) *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA. pp. 173–281.

Kaplan R., Riezler S., King T. H., Maxwell J. T., Vasserman A., Crouch R. (2004), Speed and Accuracy in Shallow and Deep Stochastic Parsing. In *Proceedings of the Human Language Technology Conference and the 4th Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'04)*. Boston, MA., pp. 97–104.

Kaplan R., Zaenen A. (1989) Long-distance Dependencies, Constituent Structure and Functional Uncertainty. In Baltin M., Kroch A. (eds.), *Alternative Conceptions of Phrase Structure*. Chicago University Press, Chicago, pp. 17–42, Reprinted in M. Dalrymple et al. (editors), *Formal Issues in Lexical-Functional Grammar*. CSLI Publications, 1995.

Kaplan R. M., Netter K., Wedekind J., Zaenen A. (1989) Translation by structural correspondences. In *Proceedings of the 4th Meeting of the European Chapter of the Association for Computational Linguistics*. UMIST Manchester, UK, pp. 272–281.

Kim R., Dalrymple M., Kaplan R., King T. H., Masuichi H., Ohkuma T. (2003) Multilingual Grammar Development via Grammar Porting. In *Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development, ESSLLI 2003*. Vienna, Austria, pp. 49–56.

King T. H., Crouch R., Riezler S., Dalrymple M., Kaplan R. (2003) The PARC700 dependency bank. In *Proceedings of the EACL03: 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*. Budapest, Hungary, pp. 1–8.

Macleod C., Meyers A., Grishman R. (1994) The COMLEX Syntax Project: The First Year. In *Proceedings of the ARPA Workshop on Human Language Technology*. Princeton, NJ., pp. 669–703.

Magerman D. (1994) Natural Language Parsing as Statistical Pattern Recognition. Ph.D. thesis, Department of Computer Science, Stanford University, CA.

Marcus M., Kim G., Marcinkiewicz M. A., MacIntyre R., Bies A., Ferguson M., Katz K., Schasberger B. (1994) The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the ARPA Workshop on Human Language Technology*. Princton, NJ., pp. 110–115.

Masuichi H., Okuma T. (2003) Japanese Parser on the basis of the Lexical-Functional Grammar Formalism and its Evaluation. *Journal of Natural Language Processing* **10**(2), 79–109.

Miyao Y., Ninomiya T., Tsujii J. (2003) Probabilistic modeling of argument structures including non-local dependencies. In *Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP)*. Borovets, Bulgaria, pp. 285–291.

Miyao Y., Ninomiya T., Tsujii 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, pp. 390–397.

Müller S., Kasper W. (2000) HPSG Analysis of German. In *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer-Verlag, Artificial Intelligence, Berlin, Heidelberg, New York, pp. 238–253.

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*. Barcelona, Spain, pp. 368–375.

Pollard C., Sag I. (1994) *Head-driven Phrase Structure Grammar*. CSLI Publications, Stanford, CA.

Riezler S., King T., Kaplan R., 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 Conference of the Association for Computational Linguistics (ACL-02)*. Philadelphia, PA. pp. 271–278.

Schmid H. (2004) Efficient Parsing of Highly Ambiguous Context-Free Grammars with Bit Vectors. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2004)*. Geneva, Switzerland, pp. 162–168.

Siegel M., Bender E. (2002) Efficient deep processing of Japanese. In *Proceedings of the 19th International Conference on Computational linguistics (COLING 2002)*. Taipei, Taiwan. pp. 31–38.

van Genabith J., Crouch R. (1996) Direct and Underspecified Interpretations of LFG f-Structures. In *16th International Conference on Computational Linguistics (COLING 96)*. Copenhagen, Denmark, pp. 262–267.

van Genabith J., Crouch R. (1997) How to Glue a Donkey to an f-Structure or Porting a Dynamic Meaning Representation Language into LFG's Linear Logic Based Glue Lan-

guage Semantics. In Bunt H., Muskens R. (eds.), *Computing Meaning volume 1, Studies in Linguistics and Philosophy, volume 73*. Kluwer Academic Press, Dordrecht, Boston and London, pp. 129–148.

van Genabith J., Way A., Sadler L. (1999) Data-driven Compilation of LFG Semantic Forms. In *Proceedings of the EACL Workshop on Linguistically Interpreted Corpora (LINC-99)*. Bergen, Norway, pp. 69–76.

Xue N., Chiou F.-D., Palmer M. (2002) Building a Large-Scale Annotated Chinese Corpus. In *Proceedings of the 19th International Conference on Computational linguistics (COLING 2002)*. Taipei, Taiwan.