# Closing the Gap Between Stochastic and Rule-based LFG Grammars

Annette Hautli
Universität Konstanz

Özlem Çetinoğlu
Josef van Genabith
CNGL, School of Computing
Dublin City University

Proceedings of the LFG10 Conference

Miriam Butt and Tracy Holloway King (Editors)

2010

CSLI Publications

http://csli-publications.stanford.edu/

**Abstract**

Developing large-scale deep grammars in a constraint-based framework such as Lexical Functional Grammar (LFG) is time-consuming and requires significant linguistic insight. Recently, treebank-based constraint-grammar acquisition approaches have been developed as an alternative to hand-crafting such resources. While treebank-based approaches are wide coverage and robust and achieve competitive evaluation results for many languages, the granularity of the linguistic analyses provided by treebank-based resources tends to be less fine-grained than what is offered by state-of-the-art hand-crafted grammars. This paper presents an approach to extend the English DCU LFG annotation algorithm with more detailed f-structure information to provide probabilistic treebank-based LFG grammars with rich feature information comparable to that implemented by the hand-crafted English XLE grammar, while maintaining the robustness and the coverage of treebank-based stochastic grammars.

# 1 Introduction

Robustly parsing natural language has been the focus of research for the last decades, with frameworks evolving that attempt to go beyond the analysis of constituency and hierarchical order to provide a more abstract level of linguistic analysis. Lexical Functional Grammar (LFG) (Bresnan and Kaplan, 1982; Dalrymple, 2001), among others, is one approach that combines two levels of representation, namely constituent structure and functional structure, which are related in terms of a projection architecture where functional information is encoded in terms of functional descriptions annotated to constituents in phrase-structure rules. By employing both representations, LFG provides insight into the surface as well as the deeper, more abstract properties of natural language syntax.

In addition, LFG has proven to be a theory that can serve as the backbone for a computational analysis of natural language. These features have led to the development of computational LFG grammars that allow for an automatic syntactic analysis of natural language.

Over time, various methodologies of developing computational LFG grammars have evolved. One approach that has proven highly successful is the employment of a rule-based LFG parser (Crouch et al., 2009) where the manual encoding of syntactic rules and functional descriptions provides a deep and highly detailed syntactic analysis that forms the input to the computation of a semantic representation at a subsequent processing step (Crouch and King, 2006).

In general, manual development of large-scale deep grammars faces a number of challenges. First, language data is complex and varied and some perfectly legitimate constructions may be outside the coverage of the grammar. Second, ”real” input may contain typos and disfluencies not envisaged by the hand-crafted

[†]This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Dublin City University.

grammar. Third, even highly efficient LFG parsing architectures operating on detailed hand-crafted grammars can be significantly slower than some state-of-the-art treebank-based stochastic parsers. Because of this, full coverage on unrestricted language data can often only be achieved by taking short-cuts: either combining fragment analyses in parser output or constraining the amount of computation to only partially explore sections of the full parsing search space.

An alternative DCU LFG approach (Cahill et al., 2004) uses robust treebank-based stochastic parsers that automatically produce Penn-II Treebank style (Marcus et al., 1993) constituent structure trees. Functional information is provided by an f-structure annotation algorithm (Burke, 2006) that automatically annotates treebank-style trees with f-structure information. In this approach both the tree parser and the f-structure information are provided automatically, reducing human grammar development effort. Up to now, treebank-based DCU LFG f-structures encoded substantially less information compared to the rich and detailed f-structures produced by the hand-crafted XLE LFG grammars, even though the treebank-based approach outperformes the hand-crafted grammars on the restricted core (preds-only and slightly extended) feature sets in the PARC700 evaluation gold standard (Cahill et al., 2008). Because of the lack of detail, however, to date treebank-based f-structure output could not be used for further semantic processing as implemented by Crouch and King (2006).

This paper attempts to combine the best of the two worlds: the linguistic depth and detail of the f-structures provided by hand-crafted XLE LFG grammars with the coverage, robustness and parse quality provided by the automatic grammar acquisition methodology of the treebank-based DCU LFG approach. In order to achieve this, we extend the f-structure annotation algorithm of the treebank-based DCU LFG grammar for English with more detailed f-structure information, aproaching the feature granularity and and linguistic sophistication of the state-of-the-art hand-crafted XLE LFG grammar for English.

The paper is structured as follows: section 2 briefly reviews the state-of-the-art in computational LFG grammars for English, followed by a particular linguistic example and how the hand-crafted XLE LFG and treebank- and annotation algorithm-based DCU LFG grammars attempt to provide a linguistically motivated analysis for it. Having introduced the concepts we deal with, section 3 presents how the gap between the two LFG grammar development architectures can be closed. Section 4 presents evaluation results, followed by remarks on future work and the conclusion in section 5.

## 2 State-of-the-Art

The aim of parsing natural language has resulted in computational grammars in various constraint-based frameworks, among them Lexical-Functional Grammar (LFG) (Bresnan and Kaplan, 1982; Dalrymple, 2001).

LFG is able to provide cross-linguistically valid analyses by employing levels of representation that abstract away from the surface structure of sentences. Furthermore, due to its computational and mathematical tractability, it has also proven an excellent theory for use in computational linguistics (Maxwell and Kaplan, 1996). Therefore, it allows for the combination of theoretically founded linguistic analyses with an efficient computational analysis.

For the purpose of this paper, we restrict ourselves to discussing two state-of-the-art LFG grammars for English, with the hand-crafted XLE grammar in section 2.1 and the DCU LFG treebank- and annotation algorithm-based approach in section 2.2, followed by an exemplary linguistic issue, namely a non-local dependency, and how the two approaches cope with it.

### 2.1 The English XLE grammar

XLE is an efficient rule-based grammar development platform, developed at Palo Alto Research Center (PARC) and consisting of cutting-edge algorithms for parsing and generation using LFG grammars, along with a user interface for writing and debugging LFG grammars and lexical resources (Crouch et al., 2009).

XLE provides the shared technology platform within the ParGram effort (Butt et al., 1999, 2002) that aims at developing parallel LFG grammars for various languages such as English, German, French, Norwegian, Japanese, Turkish and Urdu. Besides providing computationally efficient analyses for natural language, a main focus of ParGram lies on the cross-linguistically valid analysis of natural language, making the LFG analyses as parallel and informative as possible across languages.

The English XLE grammar is a part of a larger system that maps text to an Abstract Knowledge Representation (AKR) which can then be used for applications such as search, question answering (Bobrow et al., 2007) and editing text (Bier et al., 2009). Figure 1 shows the basic system pipeline.

In order to break text into sentences and sentences into tokens, finite-state transducers (FSTs) are applied. These are also used for the morphological component, where each word is analyzed and morphological information is passed on to the XLE grammar. Hand-coded syntax rules in the XLE grammar pick up the morphological information and add constituent structure and functional information. Like all rule-based LFG grammars, the output of the syntax is a c-structure (constituent structure) that encodes constituency and linear order, and an attribute-value matrix (f-structure), encoding functional information such as the predicate-argument structure and semantically important features such, e.g. tense and number (see Figure 2). In cases of syntactic ambiguity, such as PP-attachment, the XLE grammar outputs a packed representation of all possible analyses, which allows optimal-

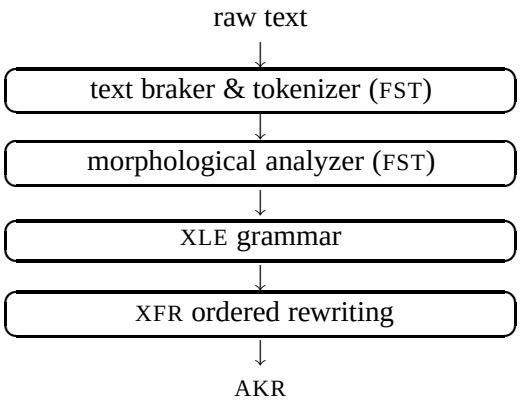


Figure 1: XLE pipeline

ity marks and a stochastic disambiguation component to choose between them in further processing. Optimality Theory marks in the syntax rules indicate which analyses are dispreferred (Frank et al., 1998).

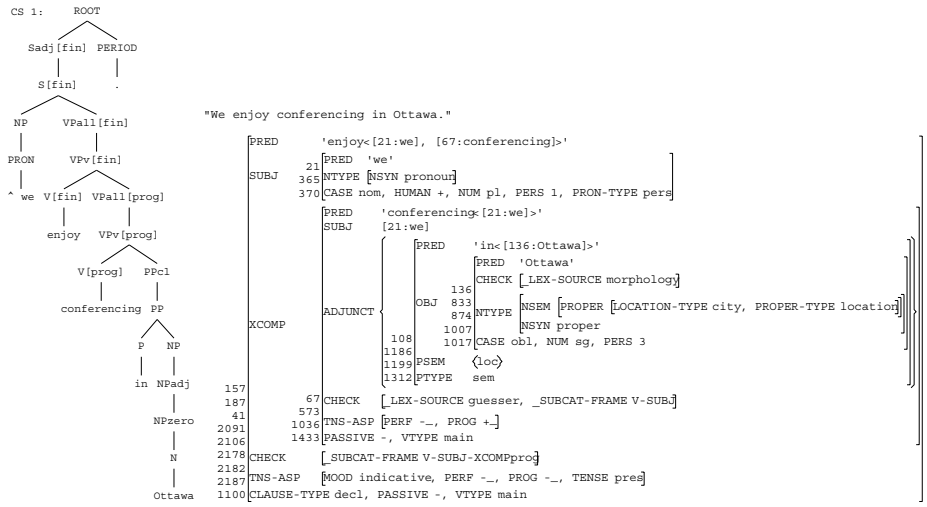


Figure 2: C- and f-structure for *We enjoy conferencing in Ottawa.*

In a later step, the syntactic information contained in the f-structure is passed on to XFR ordered rewrite rules that map f-structure information to a semantic representation by using external resources to replace words with concepts and grammatical functions with semantic roles (Crouch and King, 2006). Further rewriting by XFR rules converts the semantic representation into an abstract knowledge representation.

In cases where the parser produces a large number of analysis, XLE uses a stochastic disambiguation model that is trained on LFG analyses for Penn-II Tree-

bank sentences and uses tree information from the Penn-II Treebank to guide the XLE analyses. For sentences where the XLE grammar cannot produce a spanning parse, i.e. a complete analysis for the entire input, XLE is able to produce a sequence of largest well-formed fragment analyses. This provides a degree of robustness for language data outside the coverage of the grammar and also allows for "ungrammatical" or fragmented language, including typos etc.

## 2.2 The DCU LFG Annotation Algorithm

The treebank-based DCU LFG annotation algorithm, developed at Dublin City University (DCU) (Cahill, 2004; Cahill et al., 2008), generates c- and f-structures for English sentences in a different manner than the XLE grammar. Making use of reliable treebank-parsers, the DCU f-structure annotation algorithm annotates the nodes in the tree with f-structure equations. Annotation algorithms and wide-coverage treebank-based LFG systems have been developed in the GramLab project for English, Chinese, French, Spanish, Arabic and German.

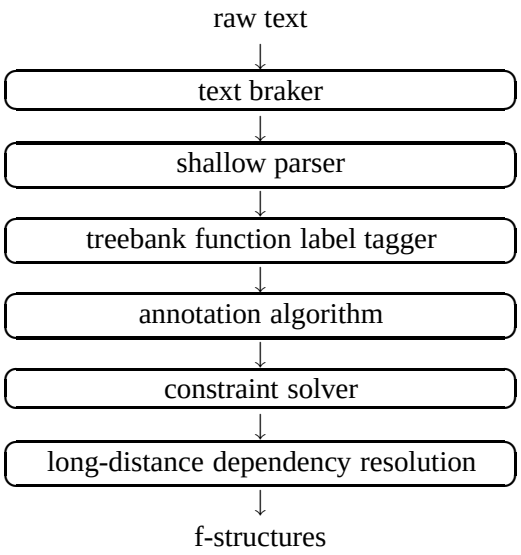


Figure 3: DCU pipeline

In general, the parsing pipeline comprises the following parts: at first, a text breaker splits running text into sentences. These are then parsed by a treebank-trained stochastic parser (Charniak and Johnson, 2005; Bikel, 2002), which creates trees in the Penn-II Treebank style (Marcus et al., 1993). The treebank function label tagger (Chrupała et al., 2007) (also trained on the Penn-II treebank) enriches the bare CFG parser output trees with further information by assigning treebank function labels (where possible), e.g. adding SBJ to subject noun phrases (NP-SBJ), and LOC to locative prepositional phrases (PP-LOC). This information helps the f-structure annotation algorithm to assign f-structure equations to the tree nodes

(including the terminal nodes). A constraint solver collects and resolves the f-structure equations and produces an f-structure. The last step, the long-distance dependency resolution module, resolves non-local dependencies using automatically acquired subcategoraisation frames and finite approximations of functional-uncertainty equations (all automatically extracted from the f-structures automatically generated from the training set of the original Penn-II treebank trees). See Figure 3 for a schematic overview.

Figure 4 shows the Penn-II Treebank-style tree for *We enjoy conferencing in Ottawa*. As the CFG-parser is trained on the Penn-II Treebank, tree nodes are named according to Penn-II conventions and may not necessarily correspond to usual LFG textbook or XLE-style phrase-structure labels.

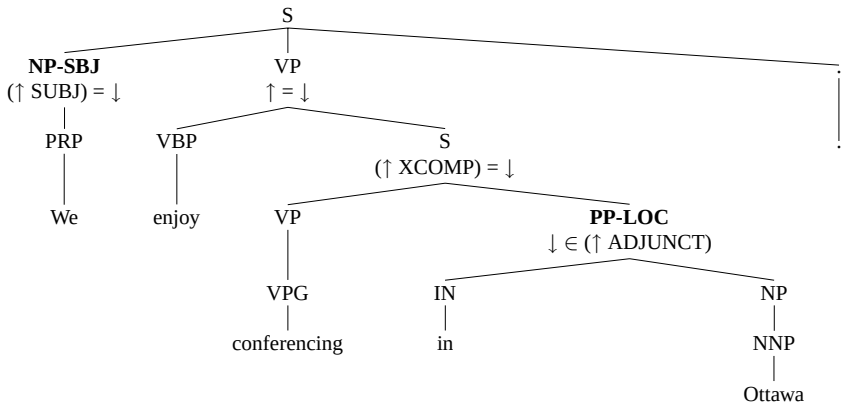


Figure 4: DCU tree with f-structure equations

The DCU LFG parsing pipeline including the f-structure annotation algorithm has been successfully evaluated on gold standards such as the PARC700 (King et al., 2003), outperforming the rule-based English XLE grammar and parser by more than 2% f-score absolute (Cahill et al., 2008).

This provides an excellent basis for a further development of the DCU processing pipeline, with a particular focus on the f-structure annotation algorithm. However, as the original DCU system was tuned to produce only basic LFG representations concentrating on a restricted set of core syntactic and semantic features, it lacks the detail and linguistic sophistication of the f-structures produced by the hand-crafted English XLE grammar. In order to close this gap between the treebank-based and the hand-crafted grammars, we need to extend the restricted feature space of the DCU annotation algorithm to obtain f-structures as detailed as XLE f-structures.

## 2.3 An Example Linguistic Issue for Computational Grammars

A recurring problem for computational grammars that goes beyond the analysis of pure surface structure and provides a linguistically motivated analysis is the treat-

ment of non-local dependencies. Consider the example '*We enjoy conferencing in Ottawa.*'. Here we have an unexpressed argument in the subordinate clause, with the subject of the main clause also being the subject of subordinate clause.

In the case of the hand-crafted LFG grammar, we can express this information by writing a lexical entry for the verb *enjoy*, as given in Figure 5, which encodes this dependency information:

```
enjoy V * (^ PRED) = 'enjoy ⟨(^ SUBJ) (^ XCOMP)⟩'
          (^ SUBJ) = (^ XCOMP SUBJ)
```

Figure 5: The LFG lexical entry for *enjoy*

Whenever the grammar finds a construction with the word *enjoy* that subcategorizes for a SUBJ and an XCOMP, the subject of the main clause is automatically made the subject of the XCOMP, represented with co-indexed f-structures.

For the treebank- and annotation algorithm-based DCU parsing pipeline, the matter is different. Instead of manually encoding information on the non-local dependencies between the main and the subordinate clause in the lexical entry, all that is available is a parser output simplified Penn-II treebank-style tree structure as in Figure 6.

```
(S (NP (PRP We))
   (VP (VBP enjoy)
   (S (VP (VBG conferencing)
          (PP (IN in)
              (NP (NNP Ottawa))))))
(.  .))
```

Figure 6: Parser output tree

Treebank-trained CFG parsers such as Charniak and Johnson (2005) and Bikel (2002) do not capture non-local dependencies. The parser output tree for the example sentence contains no information on the missing argument in the subordinate clause and does not record the non-local dependency between the subject of the main clause and the subordinate clause. From this tree alone, the basic f-structure annotation algorithm cannot recover the non-local dependency.

The long-distance dependency resolution module (Cahill et al., 2004) in the DCU LFG parsing pipeline employs statistical methods and works as follows: unlike the simplified parser output trees, the original Penn-II Treebank contains co-indexed paths for long-distance dependencies, that mark the missing subject in the subordinate clause with an empty node (NP-SBJ (-NONE- *-1)) and relate it to the subject in the main clause.

(Cahill et al., 2004) apply the f-structure annotation algorithm to the full Penn-II treebank trees producing fully non-local dependency-resolved f-structures which record non-local dependencies as corresponding reentrancies in f-structure. From these non-local dependency-resolved f-structures, (Cahill et al., 2004) learn subcategorization frames and finite approximations of functional uncertainty equations. These are used to non-local dependency resolve f-structures and the f-structure with the highest probability is chosen.

So instead of making use of lexical information in the verb entries as in the XLE grammar, the long-distance dependencies are learned from the Penn-II Treebank.

# 3 Closing the gap

The overall aim of this paper is to show that the gap in detail and granularity of linguistic representation between the hand-crafted English XLE grammar and the treebank-based DCU LFG approach for English can be closed by extending the DCU annotation algorithm.

Proof of concept that this can be done has been shown by earlier experiments reported in Hautli and King (2009), however this paper present a different approach that substantially outperforms previous experiments. This section presents the two approaches, with evaluations following in the next section.

## 3.1 Using XFR rewrite rules

A recent approach (Hautli and King, 2009) attempted to overcome the differences between the hand-crafted XLE grammar and the treebank-based DCU LFG approach by using a set of XFR rewrite rules that added missing f-structure information to DCU LFG output, effectively treating the DCU system as a black box. This means that information about the CFG tree structure is not taken into account, and the XFR rewrite rules solely operate on the DCU f-structure output. For a schematic view of the experimental layout see Figure 7.

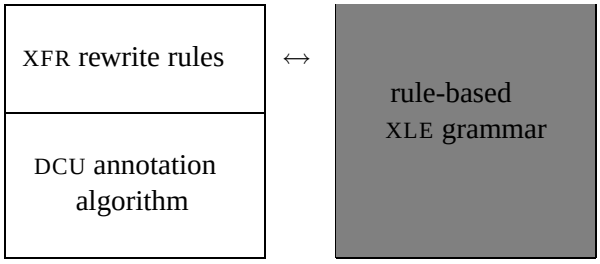


Figure 7: Schematic view of Hautli and King (2009)

In particular, the DCU LFG pipeline as described in the previous section and its output was used as is. After reformatting the DCU LFG output for it to be readable

by XLE, XFR rewrite rules were defined and applied, adding information on the level of f-structure that the DCU LFG system had not yet provided. Additionally, existing information was modified, such as feature names and values, to make it XLE-compatible.

Below, we provide a very simple example of the kind of ordered XFR rewrite rule that is employed in this approach, and although being a somewhat artificial example, the principle remains the same for more complicated constructions. Figure 8 shows a DCU LFG f-structure for the subject pronoun 'we' that is rewritten via an XFR rewrite rule. The rule works as follows: the facts on the left hand side of the rewrite rule (before the arrow) constitute the input f-structure facts that must be matched for the rule to apply. If they cannot be matched, the rule does not apply. In cases where the rule fires, the facts on the left side are rewritten to the facts on the right hand side of the rule (after the arrow). Forms beginning with a percent sign (`%`) are variables that can be instantiated by f-structures.

**input**:

$$\begin{bmatrix} \text{subj} & \begin{bmatrix} \text{pred} & \text{pron} \\ \text{num} & \text{pl} \\ \text{pron\_form} & \text{we} \end{bmatrix} \end{bmatrix}$$

**XFR rule**:

```
subj(%X,%Subj), pred(%Subj,pron), num(%Subj,pl),
pron_form(%Subj,we)
==>
SUBJ(%X,%Subj), PRED(%Subj,we),
NTYPE(%Subj,%Ntype), NSYN(%Ntype,pronoun),
CASE(%Subj,nom), HUMAN(%Subj,+), NUM(%Subj,pl), PERS(%Subj,1),
PRON-TYPE(%Subj,pers).
```

**output**:

$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{pron} & & & & \\ \text{NTYPE} & \begin{bmatrix} \text{NSYN} & \text{pronoun} \end{bmatrix} & & & & \\ \text{CASE} & \text{nom} & \text{HUMAN} & + & \text{NUM} \ \text{pl} & \text{PERS} \ 1 \quad \text{PRON-TYPE} \ \text{pers} \end{bmatrix} \end{bmatrix}$$

Figure 8: Rewriting of the pronoun *we*

Consider the XFR rule in Figure 8: the variable `%X` is the f-structure which contains a `subj`; this `subj` is then referred to by the variable `%Subj`. This `%Subj` f-structure must have a `pred` attribute with value `pron`, a `num` attribute with value `pl` and a `pron_form` feature with the value `we` in order for the XFR rule to match.

The input f-structure in Figure 8 matches the facts required by the XFR rule, and is rewritten to the output f-structure shown at the bottom of Figure 8. In total, the XFR system mapping from DCU to XLE f-structures consists of 162 manually coded rewrite rules that add and modify information from the DCU f-structures.

Evaluation shows that adding information to the DCU LFG f-structures by XFR rules and then evaluating these against the f-structures produced by the hand-crafted XLE grammar proves successful. However, issues remain where lexical entries would have to be listed in the XFR rules in order to make their analysis parallel to the XLE analysis, e.g. in the case of distinguishing first and last names.

An alternative approach based on extending the DCU LFG annotation algorithm directly without the intermediate step of XFR rewriting will be discussed in the following section.

### 3.2 Extending the DCU Annotation Algorithm

Instead of applying XFR rewrite rules as in Hautli and King (2009), the approach explored in this paper is to directly enrich the DCU annotation algorithm with the full feature inventory, detail and sophistication of the hand-crafted XLE grammar, as shown in Figure 9. There is no intermediate step between DCU and XLE output, and they are compared directly. In addition, we retain a version of the original annotation algorithm (Cahill, 2004) and its feature space.

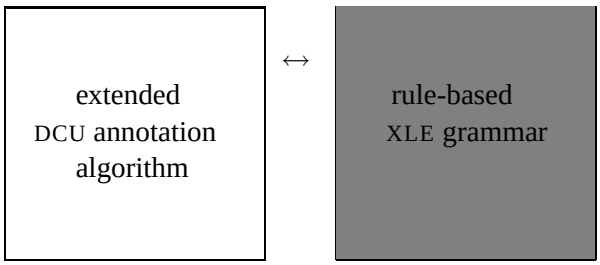


Figure 9: Schematic view of 2010 experiment

The overall DCU pipeline is not changed for this paper, only the f-structure annotation algorithm is extended, as shown in Figure 10.

Apart from extending the feature space, some existing features were also renamed and their representation in the f-structure changed. To give an overview of the feature space of the DCU LFG annotation algorithm, Table 1 lists the original DCU features (36 in total) with the newly added f-structure features in bold (30 added). Besides adding features, we also extended the range of feature values to make the DCU f-structures more informative, for instance the values 'first_name' and 'last_name' of the feature NAME-TYPE.

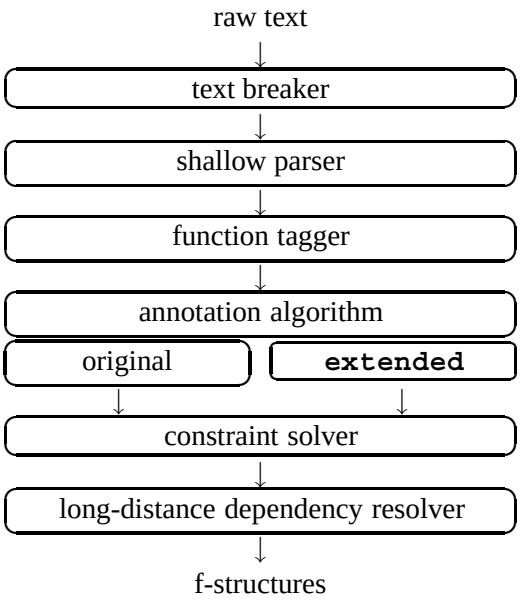


Figure 10: Extended DCU Annotation Algorithm

| **original** features | added features |
|---|---|
| adegree, adjunct, aquant, comp, conj, coord-form, det-form, focus-int, mod, num, number, number-type, obj, obj-th, obl, obl-ag, obl-compar, passive, pcase, perf, poss, prog, pron-form, pron-int, pron-rel, proper, prt-form, quant, stmt-type, subj, subord-form, tense, topic-rel, xcomp | **adjunct-type**, **adv-type**, **atype**, **case**, **clause-type**, **coord**, **common**, **deg-dim**, **degree**, **deixis**, **det**, **focus**, **gend-sem**, **human**, **inf-type**, **mood**, **name-type**, **nsem**, **nsyn**, **ntype**, **part**, **precoord_form**, **proper-type**, **psem**, **ptype**, **spec**, **time**, **tns-asp**, **vtype**, **xcomp-pred** |
| **extended** features | |

Table 1: Original and extended feature space of the DCU annotation algorithm

Figures 11 and 12 show the f-structures for *We enjoy conferencing in Ottawa.*, exemplifying tense and aspect as it is represented in the original and the extended DCU LFG annotation algorithm, respectively. The `tense` feature, which is on the top level of the f-structure in the original DCU representation, is moved to the TNS-ASP f-structure that is complemented by the aspectual features MOOD, PERF and PROG.

Following the general methodology of making the DCU f-structures as closely resembling the hand-crafted XLE grammar as possible, the representation of tense and aspect in the extended DCU f-structures is now equivalent to the representation in the XLE grammar. This enables the output of the treebank- and annotation algorithm-based DCU LFG pipeline to serve as valid input to the XFR semantics. As

the correct XLE-style representation of tense and aspect is crucial for getting valid semantic representations, it is necessary to capture this information as completely and precisely as possible.

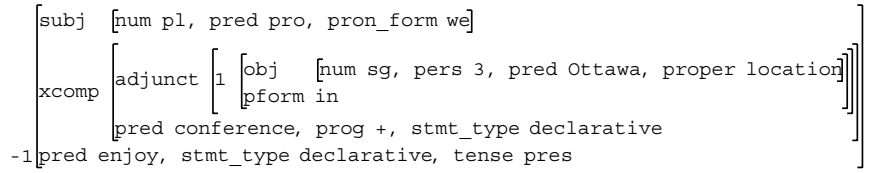


Figure 11: Tense and aspect in the original DCU f-structures

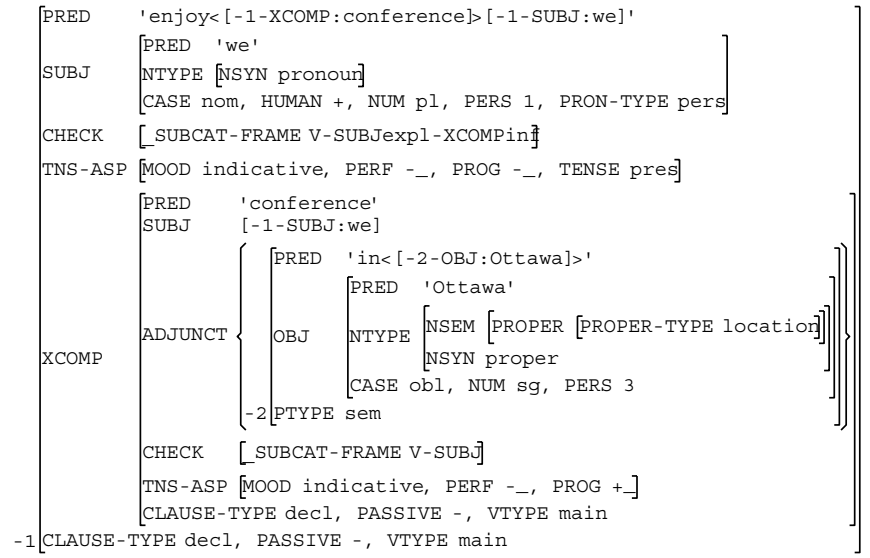


Figure 12: Tense and aspect in the extended DCU f-structures

Another important point, in particular for the corpora we evaluated on, was the precise XLE-style analysis of first and last names (as in *John Smith*). In order to add this information, the relation of the CFG parse tree nodes is taken into consideration. If two (or more) proper names are sisters and the lemma of the leftmost node can be found in a list of first names, we annotate it with the information that it is a first name. This is recursively done until one of the sisters to the right is not a first name, annotating this node with the feature for last names. This is a substantial improvement over the approach with XFR rewrite rules, because in addition to listing first names, last names also had to be listed. By considering node information, this can be automatically done, with middle names also being detected (e.g. *John Adam Smith*). As a result, this substantially increases the precision for any corpora, but in particular for those containing newspaper text, as is the case in PARC700.

# 4 Evaluation

The evaluation covers two aspects we are concerned with: the quality of the extended DCU f-structures and the overall coverage of the DCU annotation algorithm.

With respect to the quality of the f-structures, the purpose of the evaluation is to see how closely the extended treebank- and annotation algorithm-based DCU f-structures resemble the hand-crafted XLE grammar f-structures. For this we use evaluation measures from information retrieval, namely precision ("How accurate are the extended DCU f-structures?"), recall ("How complete are the extended DCU f-structures?") and the f-score, which is a weighted average of precision and recall. These measures are calculated on a per-feature basis, i.e. every feature of either the DCU or the XLE side is checked whether it appears on the other side. The number of found and not found features is calculated and divided by the total number of features present. The overall matching results lie between 0 (no feature matches) and 1 (all features match). The same methodology is employed when matching the semantic representations that the DCU and the XLE f-structures generate.

Concerning the evaluation of the coverage of the DCU annotation algorithm, the aim is to detect constructions where either the stochastic parser or the annotation algorithm fails to produce a valid representation.

## 4.1 Quality of representations

### 4.1.1 F-Structure matching

In the first set of experiments we measure the similarity of treebank- and annotation algorithm-based DCU and hand-crafted XLE grammar f-structures and compare our results to the architecture proposed in Hautli and King (2009) with XFR rewrite rules. Both architectures (XFR rewrite rules approach vs. extended DCU annotation algorithm) are tested on a testsuite of f-structures for 720 sentences constructed by PARC and designed to test the semantic representation of the English XLE grammar.

Given the difference that the XLE grammar can produce multiple f-structures for a sentence and the DCU grammar cannot, we match a single DCU analysis against each XLE analysis and choose the best matching result.

| | **Hautli and King (2009)** | | | **extended DCU system** | | |
|---|---|---|---|---|---|---|
| | precision | recall | f-score | precision | recall | f-score |
| sem_test | 70.31 | 67.69 | 68.98 | **83.44** | **77.22** | **80.20** |

Table 2: Evaluation results for semantics testsuite

The results in Table 2 show that the features can be reconstructed successfully in both architectures, however extending the DCU algorithm is more effective than using the set of XFR rewrite rules, because we can allow for operations that are unavailable to the rewrite approach, such as taking into account the relation of nodes in the tree, as shown for the annotation of first and last names.

In order to test the extended DCU LFG annotation algorithm more independently, we evaluate the DCU pipeline against the PARC700 gold standard which contains 700 randomly extracted sentences from Section 23 of Penn-II Treebank (WSJ section) with an average length of 19.8 words per sentence. The sentences are split into a development set (140 sentences) and a test set (560 sentences). In addition, we compare our evaluation results of the extended DCU LFG approach with the PARC700 evaluation results of the original DCU LFG approach and the XLE grammar (Cahill et al., 2008)[1]. Evaluation on the test set generates the following results.

| **PARC700** | | | |
|---|---|---|---|
| | **XLE grammar** | **original DCU** | **extended DCU** |
| precision | — | — | 85.45 |
| recall | — | — | 81.81 |
| f-score | 80.55 | 82.73 | 83.59 |

Table 3: PARC700 evaluation

Table 3 shows that the extended DCU annotation algorithm outperforms the XLE grammar and the original DCU annotation algorithm.

In addition to improving our evaluation results for the extended DCU version, we have also improved the f-score for the original version from 82.73% in Cahill et al. (2008) to 83.45%. This is due to further refining of the annotation algorithm.

In order to show in detail how the extended DCU system is performing, Table 4 gives the breakdown by dependency relation of the evaluation against PARC700.

| **Dependency** | **Precision** | **Recall** | **F-score** | **Dependency** | **Precision** | **Recall** | **F-score** |
|---|---|---|---|---|---|---|---|
| adegree | 81 | 79 | 80 | pcase | 91 | 77 | 83 |
| adjunct | 73 | 72 | 73 | perf | 95 | 88 | 92 |
| aquant | 33 | 77 | 47 | poss | 88 | 90 | 89 |
| comp | 69 | 75 | 72 | precoord_form | 0 | 0 | 0 |
| conj | 82 | 79 | 80 | prog | 97 | 75 | 85 |
| coord_form | 74 | 90 | 81 | pron_form | 91 | 84 | 88 |
| det_form | 98 | 98 | 98 | pron_int | 0 | 0 | 0 |
| focus_int | 0 | 0 | 0 | pron_rel | 68 | 56 | 62 |
| mod | 80 | 69 | 74 | proper | 87 | 90 | 88 |
| num | 91 | 89 | 90 | prt_form | 78 | 78 | 78 |
| number | 89 | 89 | 89 | quant | 84 | 73 | 78 |
| number_type | 95 | 92 | 94 | stmt_type | 90 | 82 | 86 |
| obj | 91 | 87 | 89 | subj | 88 | 69 | 77 |
| obj_theta | 42 | 45 | 43 | subord_form | 84 | 74 | 79 |
| obl | 51 | 70 | 59 | tense | 96 | 93 | 95 |
| obl_ag | 87 | 87 | 87 | topic_rel | 39 | 64 | 49 |
| obl_compar | 57 | 27 | 36 | xcomp | 86 | 78 | 82 |
| passive | 85 | 69 | 77 | | | | |

Table 4: Breakdown by dependency relation of extended DCU annotation algorithm against PARC700

[1]Cahill et al. (2008) only provide the f-score for the evaluations.

The results show that that as far as the f-structure matching is concerned, the extended DCU annotation algorithm performs well. Especially for features such as tense and aspect, where recall and precision are above 95%, we get the right analysis for most sentences, except in cases where the treebank-trained CFG tree parser produces a wrong tree. The same holds for other features such as NUMBER-TYPE and DET-FORM with matching figures around 95%. Remaining issues are notorious cases such as the adjunct and oblique distinction, where only around 60% of annotations are correct, as well as the annotation of OBJ-TH with a matching precision of 43%.

#### 4.1.2 Matching of Semantic Representations

To validate our claim that we can provide treebank- and annotation algorithm-based DCU LFG output that can serve as input to the XFR semantic representation, we match the semantic representations that are generated using the extended DCU LFG f-structures as input with semantic representations based on using the f-structures generated by the hand-crafted XLE grammar as input. The f-score for that matching is 73.7, which means that by using extended DCU f-structures we can produce 73.7% of information that is contained in the semantic representations when XLE f-structures are used. This is a promising result and although the f-score is lower compared to matching f-structures, the information we catch in the extended DCU f-structure is so comprehensive that the semantic representations generated from them are useful.

### 4.2 Coverage Evaluation

In addition to evaluating the quality of the extended DCU f-structures, we also measure the coverage of the extended DCU annotation algorithm (i.e. the percentage of sentences for which at least one analysis is found). By using the PARC700 gold standard, we count the number of parsed sentences, divided by the total number of sentences.

For both the training and the test set, we get full coverage, whereas the XLE grammar has a coverage of 80% and 83.8%, respectively. Preliminary experiments of running the DCU annotation algorithm on all of Penn-II Treebank have resulted in a coverage of above 98%.

## 5 Future Work and Conclusion

In this paper we have presented the extension of the DCU LFG annotation algorithm so that its output can serve as input to the XFR semantic representation which requires detailed f-structure information. The experiments show that the gap in the richness of the feature space and detail of the f-structure representations between the treebank- and annotation algorithm-based DCU LFG approach and hand-crafted XLE grammars can be closed, which means that there is a possibility of generating rich and deep LFG grammars on the basis of treebanks and annotation algorithms. This technique cannot only be used for English but also for other languages, benefiting from the aspect that the CFG trees are automatically created with a treebank-trained robust parser and have a good coverage for general unrestricted text or even fragmented text such as *John ! hops !*.

One aspect that remains for future work is to test the performance of the extended (as compared to the original) DCU annotation algorithm on larger corpora, for instance all of Penn-II treebank training set. This would allow for a more comprehensive assessment of the coverage of the extended DCU annotation algorithm and could also provide a large-scale rich f-structure bank that can be used for other natural language processing tools.

Another area of future work is the long-distance dependency module that has to be retrained to get more exact probabilities for the resolution of the non-local dependencies. Having more exact probabilities will most likely improve our evaluation results.

A more long-term goal is the development of an integrated DCU-XLE system, which would use the current XLE system for in-coverage sentences, but the extended DCU system for out-of-coverage and fragmenting sentences. Given the positive results for the matching of the semantic representations, this would potentially help in getting a more wide-coverage semantic representation.

The paper also shows that by extending the DCU LFG system, we can outperform the earlier approach where the DCU LFG system was employed as a black box and rewrite rules were used to modify the f-structures. The approach here results in a single DCU LFG system, which is more efficient and also has a higher accuracy due to the extra information that is available by looking at the node relation in the tree. By being able to take into account tree information which previously could not be done, we allow for more linguistic insight that can be captured in the final f-structure representation.

As more researchers wish to build meaning-sensitive applications, we can contribute a robust, deep syntactic analysis that can be used for further levels of abstraction.

## References


Bier, Eric, Chow, Richard, Gollé, Philippe, King, Tracy Holloway and Staddon, Jessica. 2009. The Rules of Redaction: Identify, Protect, Review (and Repeat). In *IEEE Volume on Access Control*, pages 46–53, IEEE Computer and Reliability Societies.

Bikel, Daniel M. 2002. Design of a Multi-lingual, Parallel-processing Statistical Parsing Engine. In *The Proceedings of HLT 2002*.

Bobrow, Daniel G., Cheslow, Bob, Condoravdi, Cleo, Karttunen, Lauri, King, Tracy Holloway, Nairn, Rowan, de Paiva, Valeria, Price, Charlotte and Zaenen, Annie. 2007. PARC's Bridge and Question Answering System. In *Grammar Engineering Across Frameworks*, pages 46–66, CSLI Publications.

Bresnan, Joan and Kaplan, Ronald M. (eds.). 1982. *The Mental Representation of Grammatical Relations*. The MIT Press.

Burke, Michael. 2006. *Automatic Treebank Annotation for the Acquisition of LFG Resources*. Ph. D.thesis, Dublin City University.

Butt, Miriam, Dyvik, Helge, King, Tracy Holloway, Masuichi, Hiroshi and Rohrer, Christian. 2002. The Parallel Grammar Project. In *Proceedings of COLING2002, Workshop on Grammar Engineering and Evaluation*, pages 1–7.

Butt, Miriam, King, Tracy Holloway, Niño, María-Eugenia and Segond, Frédérique. 1999. *A Grammar Writer's Cookbook*. CSLI Publications.

Cahill, Aoife. 2004. *Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximations*. Ph. D.thesis, School of Computing, Dublin City University.

Cahill, Aoife, Burke, Michael, O'Donovan, Ruth, Riezler, Stefan, van Genabith, Josef and Way, Andy. 2008. Wide-Coverage Deep Statistical Parsing Using Automatic Dependency Structure Annotation. *Computational Linguistics* 34(1), 81–124.

Cahill, Aoife, Burke, Michael, O'Donovan, Ruth, van Genabith, Josef and Way, Andy. 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)*, pages 320–327.

Charniak, Eugene and Johnson, Mark. 2005. Course-to-fine n-best-parsing and MaxEnt discriminative reranking. In *Proceedings of ACL*, pages 173–180, Ann Arbor.

Chrupała, Grzegorz, Stroppa, Nicolas, van Genabith, Josef and Dinu, Georgiana. 2007. Better training for function labeling. In *International Conference on Recent Advances in Natural Language Processing (RANLP 2007)*, pages 133–138, Bulgaria.

Crouch, Dick, Dalrymple, Mary, Kaplan, Ron, King, Tracy, Maxwell, John and Newman, Paula. 2009. XLE Documentation, http://www2.parc.com/isl/groups/nltt/xle/doc/.

Crouch, R. and King, T. H. 2006. Semantics via F-structure Rewriting. In *Proceedings of LFG06*, CSLI Publications.

Dalrymple, Mary. 2001. *Lexical Functional Grammar (Syntax and Semantics)*. Academic Press.

Frank, Anette, King, Tracy Holloway, Kuhn, Jonas and III, John T. Maxwell. 1998. Optimality Theory Style Constraint Ranking in Large-scale LFG Grammars. In *Proceedings of LFG98*, CSLI Publications.

Hautli, Annette and King, Tracy Holloway. 2009. Adapting Stochastic LFG Input for Semantics. In *Proceedings of LFG09*, CSLI Publications.

King, Tracy Holloway, Crouch, Richard, Riezler, Stefan, Dalrymple, Mary and Kaplan, Ronald M. 2003. The PARC700 Dependency Bank. In *Proceedings of the EACL03: 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*.

Marcus, Mitchell, Santorini, Beatrice and Marcinkiewicz, Mary Ann. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2), 313–330.

Maxwell, John T. and Kaplan, Ronald M. 1996. Unification-based Parsers that Automatically Take Advantage of Context Freeness. In *Proceedings of LFG96*, CSLI Publications.