

Quasi-Logical Forms from F-Structures for the Penn Treebank

Aoife Cahill Mairead McCarthy Josef van Genabith Andy Way
Computer Applications, Dublin City University, Ireland
{ acahill, mccarthy, josef, away }@computing.dcu.ie

Abstract

In this paper we show how the trees in the Penn treebank can be associated automatically with simple quasi-logical forms. Our approach is based on combining two independent strands of work: the first is the observation that there is a close correspondence between quasi-logical forms and LFG f-structures [van Genabith and Crouch, 1996]; the second is the development of an automatic f-structure annotation algorithm for the Penn treebank [Cahill et al, 2002a; Cahill et al, 2002b]. We compare our approach with that of [Liakata and Pulman, 2002].

1 Introduction

Probabilistic parsers and grammars extracted from treebanks (c.f. [Charniak, 1996]) provide an attractive way of inducing large coverage syntactic resources. However, automatic construction of logical forms for such large coverage grammars is a non-trivial task. In this paper we present the first steps towards this goal: we show how the trees in the Penn treebank can be associated automatically with simple quasi-logical forms inspired by [Alshawi & Crouch, 1992]. Our approach is based on combining two independent strands of work: the first is the observation that there is a close correspondence between quasi-logical forms and Lexical-Functional Grammar (LFG) f-structures [van Genabith and Crouch, 1996]; the second is the development of an automatic proto-f-structure annotation algorithm for the Penn treebank [Cahill et al, 2002a; Cahill et al, 2002b]. In our approach we automatically annotate the trees in the Penn-II treebank [Marcus et al, 1994] with LFG f(unctional)-structures [Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001] and then translate the resulting f-structures

into simple quasi-logical forms. Currently, using this method we can associate 95.76% of the trees in the Penn-II treebank with a quasi-logical form. Our method can be combined with probabilistic parsers in a pipeline architecture [Cahill et al, 2002c].

The paper is structured as follows: first, we briefly describe the basics of LFG f-structures, quasi-logical forms (QLFs) and how to translate between them [van Genabith and Crouch, 1996]. Second, we outline the automatic proto-f-structure annotation method developed in [Cahill et al, 2002a; Cahill et al, 2002b]. Third, we extend this method towards proper f-structures to include traces for non-local dependencies and passive. Fourth, we extend the theoretical work described in [van Genabith and Crouch, 1996] to cover the data provided by the Penn-II treebank. Finally, we compare our approach with related work by [Liakata and Pulman, 2002], conclude and outline further work.

2 LFG F-Structures and Quasi-Logical Forms

2.1 Lexical-Functional Grammar

Lexical-Functional Grammar (LFG) [Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001] is an early member of the family of unification- (more correctly: constraint-) based grammar formalisms (FUG, PATR-II, GPSG, HPSG etc.). At its most basic, an LFG involves two levels of representation: c-structure (constituent structure) and f-structure (functional structure). C-structure represents surface grammatical configurations such as word order and the grouping of linguistic units into larger phrases. The c-structure component of an LFG is represented by a CF-PSG (context-free phrase structure grammar). F-structure represents abstract syntactic functions such as SUBJ(ect), OBJ(ect), PRED(icate) etc. in terms of recursive attribute-value structure representations. These functional representations abstract away from particulars of surface configuration. While languages differ with respect to surface configuration (word order etc.) they may still encode the same (or very similar) abstract syntactic functions (or predicate-argument structure). To give a simple example, typologically, English is a SVO (subject-verb-object) language, while Irish is a verb-initial VSO language. However, a sentence like *John saw Mary* and its Irish translation *Chonaic Seán Máire*, while associated with very different c-structure trees, have structurally isomorphic f-structure representations, as illustrated in Figure 1.

C-structure trees and f-structures are related in terms of projections

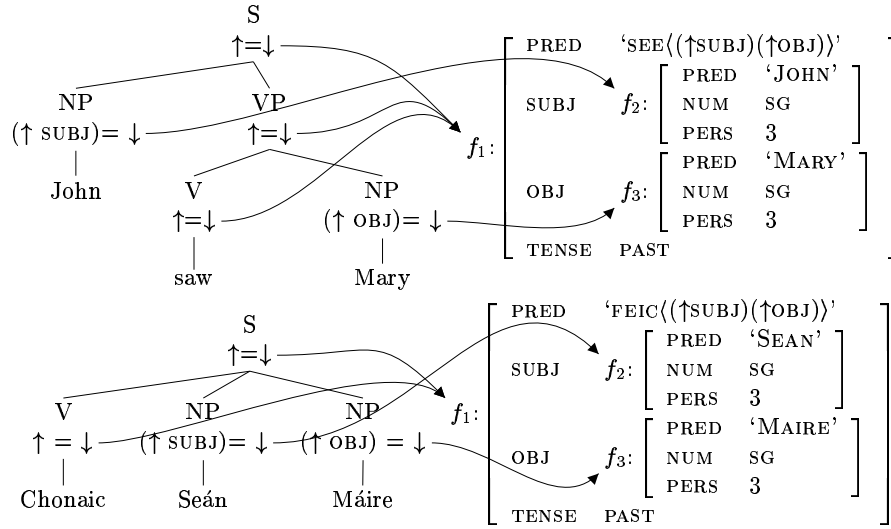


Figure 1: C- and f-structures for an English and corresponding Irish sentence

(indicated by arrows in Figure 1). These projections are defined in terms of f-structure annotations in c-structure trees (equations describing f-structures) originating from annotated grammar rules and lexical entries. A sample set of LFG grammar rules with functional annotations (f-descriptions) is provided in Figure 2. Optional constituents are indicated by brackets.

2.2 Quasi-Logical Forms

Quasi-Logical Forms (QLFs) [Alshawi & Crouch, 1992] provide the *semantic* level of representation employed in the Core Language Engine (CLE)

$$\begin{aligned}
 S &\rightarrow \begin{array}{c} \text{NP} \\ \uparrow \text{SUBJ} = \downarrow \end{array} \quad \begin{array}{c} \text{VP} \\ \uparrow = \downarrow \end{array} \quad \left(\begin{array}{c} \text{ADV} \\ \downarrow \in \uparrow \text{ADJN} \end{array} \right) \\
 \text{NP} &\rightarrow \left(\begin{array}{c} \text{DET} \\ \uparrow \text{SPEC} = \downarrow \end{array} \right) \quad \begin{array}{c} \text{N} \\ \uparrow = \downarrow \end{array} \\
 \text{VP} &\rightarrow \begin{array}{c} \text{V} \\ \uparrow = \downarrow \end{array} \quad \left(\begin{array}{c} \text{NP} \\ \uparrow \text{OBJ} = \downarrow \end{array} \right) \quad \left(\begin{array}{c} \text{VP} \\ \uparrow \text{XCOMP} = \downarrow \end{array} \right) \quad \left(\begin{array}{c} \text{S} \\ \uparrow \text{COMP} = \downarrow \end{array} \right)
 \end{aligned}$$

Figure 2: Sample LFG grammar rules for a simple fragment of English

[Alshawi, 1992]. The two main characteristics of the formalism are underspecification and monotonic contextual resolution. QLFs give (partial) descriptions of intended semantic compositions. Contextual resolution monotonically adds to this description, e.g. by placing further constraints on the meanings of certain expressions like pronouns, or quantifier scope. QLFs are interpreted by a truth-conditional semantics via a supervaluation construction over the compositions meeting the description.

Unresolved QLFs give the basic predicate-argument structure of a sentence, mixed with some syntactic information encoded in the category attributes (`<num=sg,spec=every>`) of QLF **terms** and **forms**. As an example, the string *every representative supported a candidate* would give rise to a QLF of the form:

```
?Scope: support (term(+r, <num=sg, spec=every>, representative, ?Q, ?S),
                  term(+g, <num=sg, spec=a>, candidate, ?P, ?R))
```

The motivation for including syntactic information in QLFs is that resolution of anaphora, ellipsis or quantifier scope may be constrained by syntactic factors [Alshawi, 1992].

2.3 From F-Structures to QLFs: I

F-structures encode predominantly abstract syntactic information with some semantic information approximating predicate-argument structure in the form of “semantic form” PRED values and quantificational information in the form of SPEC values.

$$\left[\begin{array}{l} \text{SUBJ} \\ \text{PRED} \\ \text{OBJ} \end{array} \left[\begin{array}{l} \text{PRED 'REPRESENTATIVE'} \\ \text{NUM SG} \\ \text{SPEC EVERY} \\ \text{'SUPPORT'(\uparrow \text{SUBJ}, \uparrow \text{OBJ})} \\ \text{PRED 'CANDIDATE'} \\ \text{NUM SG} \\ \text{SPEC A} \end{array} \right] \right]$$

While there is clear difference in approach and emphasis, unresolved QLFs and f-structures bear a striking similarity and, for simple cases at least, it is easy to see how to get from one to the other in terms of a translation function $(\cdot)^\circ$ [van Genabith and Crouch, 1996]:

$$\left[\begin{array}{l} \Gamma_1 \quad \gamma_1 \\ \dots \\ \text{PRED } \Pi(\uparrow \Gamma_1, \dots, \uparrow \Gamma_n) \\ \dots \\ \Gamma_n \quad \gamma_n \end{array} \right]^\circ = \text{?Scope: } \Pi(\gamma_1^\circ, \dots, \gamma_n^\circ)$$

The core of the $(\cdot)^\circ$ mapping taking us from f-structures to QLFs places the values of subcategorizable grammatical functions into their argument positions in the governing semantic form and recurses on those arguments. From this rather general perspective the difference between f-structures and QLF is one of information packaging rather than anything else.

3 Automatic Proto-F-Structure Annotation

Given an f-structure annotated version of the Penn-II treebank and a mapping from f-structures to QLFs, we can associate the trees in the Penn treebank with QLFs. The question is how do we get a version of the Penn-II treebank annotated with f-structures? Given a parse-annotated string (c-structure), f-structures are computed from the functional annotations on the RHSs of PSG rules and lexical entries involved in the tree. Clearly, one way of associating the Penn-II treebank with f-structure information is to first *automatically* extract the CFG from the treebank following the method of [Charniak, 1996]; second, *manually* annotate the extracted CFG rule types (and lexical entries) with f-structure information; third, automatically match the annotated CFG against the parse-annotated strings in the treebank; and fourth, collect and resolve the f-structure annotations in the matching rules to generate an f-structure. Unfortunately, the large number of CFG rule types in treebanks ($> 19,000$ for Penn-II) makes manual f-structure annotation of grammar rules extracted from a complete treebank prohibitively time-consuming and expensive.

Can the process of annotating treebank trees with f-structure information be automated? As far as we are aware, to date we can distinguish three different types of automatic f-structure annotation architectures:¹

- annotation algorithms,
- regular expression based annotation,
- flat, set-based tree description rewriting.

All approaches are based on exploiting categorial and configurational

¹These have all been developed within an LFG framework and although we refer to them as automatic f-structure annotation architectures they could equally well be used to annotate treebanks with e.g. HPSG feature structure or indeed Quasi-Logical Form (QLF) [Liakata and Pulman, 2002] annotations.

information encoded in trees. Some also exploit the Penn-II functional annotations.²

With annotation algorithms, two variants are possible. An annotation algorithm may

- *directly* (recursively) transform a treebank tree into an f-structure – such an algorithm would more appropriately be referred to as a tree to f-structure transformation algorithm;
- *indirectly* (recursively) annotate CFG treebank trees with f-structure annotations from which an f-structure can be computed by a constraint solver.

The earliest approach to automatically identify SUBJ, OBJ etc. nodes in CFG trees structures is probably [Lappin et al, 1989].³ Their algorithm identifies grammatical function nodes to facilitate the statement of transfer rules in a machine translation project.

The first *direct* automatic f-structure annotation algorithm we are aware of is unpublished work by Ron Kaplan (p.c.) from 1996. Kaplan worked on automatically generating f-structures from the ATIS corpus to generate data for LFG-DOP applications. The approach implements a direct tree to f-structure transduction. The algorithm walks through the tree looking for different configurations (e.g. NP under s, 2nd NP under VP, etc.) and “folds” or “bends” the tree into the corresponding f-structure.

A regular expression-based automatic f-structure annotation methodology is described in [Sadler et al, 2000]. The idea is simple: first, the CFG rule set is extracted from the treebank (fragment); second, regular expression-based annotation principles are defined; third, the principles are automatically applied to the rule set to generate an annotated rule set; fourth, the annotated rules are automatically matched against the original treebank trees and thereby f-structures are generated for these trees. Since the annotation principles factor out linguistic generalisations, their number is much smaller than the number of CFG treebank rules. In fact, the regular expression-based f-structure annotation principles constitute a principle-based LFG c-structure/f-structure interface.

In a companion paper, [Frank, 2000] develops an automatic annotation method that in many ways is a generalisation of the regular expression-based

²Note that apart from SBJ and LGS, functional annotations or tags in the Penn-II treebank do not provide LFG predicate-argument style annotations but semantically classify modifying PP constituents as TMP (temporal), LOC (locative) etc. modifiers.

³This was recently pointed out to us by Shalom Lappin (p.c.).

annotation method. The idea is again simple: first, trees are translated into a flat set representation format in a tree description language; second, annotation principles are defined in terms of rules employing a rewriting system originally developed for transfer-based machine translation architectures. In contrast to [Sadler et al, 2000] which applies only to “local” CFG rule contexts, [Frank, 2000] can consider arbitrary tree fragments. Secondly, it can be used to define both order-dependent cascaded and order-independent annotation systems. [Liakata and Pulman, 2002] have developed a similar approach to map Penn-II trees to QLFs.

The approaches detailed in [Sadler et al, 2000; Frank, 2000] and compared in [Frank et al, 2002] are proof-of concept and operate on small subsets of the AP and Susanne corpora.⁴ In our more recent research [Cahill et al, 2002a; Cahill et al, 2002b] we have developed an algorithmic indirect annotation method for the > 49000 parse annotated strings in the Wall Street Journal section of the Penn-II treebank.

The algorithm is implemented as a recursive procedure (in Java) which annotates Penn-II treebank tree nodes with f-structure information. The annotations describe what we call “proto-f-structures”, which

- encode basic predicate-argument-modifier structures;
- interpret constituents locally (i.e. do not resolve long-distance dependencies or “movement” phenomena encoded as traces in the Penn-II trees);
- may be partial or unconnected (the method is robust: in case of missing annotations, a sentence may be associated with two or more unconnected f-structure fragments rather than a single complete f-structure).

Even though the method is encoded in the form of an annotation algorithm (i.e. a procedure), we did not want to completely hardwire the linguistic basis for the annotation into the procedure. In order to support maintainability and reusability of the annotation algorithm and the linguistic information encoded within, the algorithm is designed in terms of three main components that work in sequence:

$$\boxed{\text{L/R Context APs}} \Rightarrow \boxed{\text{Coordination APs}} \Rightarrow \boxed{\text{Catch-All APs}}$$

L/R Context Annotation Principles are based on a tripartition of the daughters of each local tree (of depth one, i.e. of CFG rules) into a prefix,

⁴This is not to say that these approaches cannot be scaled to a complete treebank.

head and suffix sequence. We automatically transform the Penn-II trees into head-lexicalised trees by adapting the rules of [Magerman, 1994; Collins, 1999]. For each LHS type (NP, VP, ...) in the Penn-II CFG rule types we construct an annotation matrix. The matrix encodes information on how to annotate CFG node types in the left (prefix) and right (suffix) context in Rule RHSs. Table 1 gives a simplified matrix for NP rules.

NP	left context	head	right context
subcat	DT,CD: $\uparrow\text{spec}=\downarrow$	NN,NNS,NP: $\uparrow=\downarrow$...
non-sub	ADJP: $\downarrow\in\uparrow\text{adjn}$ NN,NNS,NP: $\downarrow\in\uparrow\text{headmod}$...		SBAR,VP: $\uparrow\text{relmod}=\downarrow$ PP: $\downarrow\in\uparrow\text{adjn}$ NN,NNS,NP: $\uparrow\text{app}=\downarrow$

Table 1: Simplified, partial annotation matrix for NP rules

For each LHS category, the annotation matrices are populated by analysing the topmost frequent rule types such that the token occurrence of the rule types in the corpus covers at least 85%. To give an example, this means that instead of looking at > 6,000 NP rule types in the Penn-II corpus, we only look at the 102 most frequent rule types. To keep L/R context annotation principles simple and perspicuous, they only apply if the local tree does not contain coordination. (Like and unlike) coordinate structures are treated by the second component of our annotation algorithm. Finally, the algorithm has a catch-all and clean-up component.

Annotation coverage is measured in terms of f-structure fragmentation (the method is robust and in case of missing annotations may deliver unconnected f-structure fragments for a tree). Annotation accuracy is measured against a manually constructed gold-standard with f-structures for 105 trees randomly selected from Section 23 of the Penn-II treebank.⁵

# f-str. frags	fragmentation			All annotations	Preds-only
	# sent	percent			
0	120	0.25	Precision	0.956	0.958
1	48304	99.75	Recall	0.937	0.908

Table 2: Proto-f-structure annotation: fragmentation, precision and recall.

⁵Our gold-standard f-structures are available for inspection at <http://www.computing.dcu.ie/~away/Treebank/treebank.html>.

4 From “Proto”- towards “Proper”-F-Structures

Table 2 shows that our proto-f-structure annotation is complete⁶ and high quality. Unfortunately, proto-f-structures are not sufficient for compilation of high-quality logical predicate-argument-modifier structure in the form of QLFs. The reason is that proto-f-structures do not encode passive nor any non-local (long-distance) dependencies. In the Penn-II treebank trees these are encoded in terms of traces and coindexed empty productions in the trees. Empty productions and traces are completely ignored in the automatic proto-f-structure annotation algorithm and linguistic material is interpreted locally where it occurs in the tree.

The Penn-II trees mark non-local dependencies (and partly passive constructions) in terms of a typology of traces relating “moved” material to positions marked by a coindexed empty node where the moved material should be interpreted. In our research to date we have concentrated on traces for A and A’ movement (movement to argument and non-argument positions) including traces for wh-questions, relative clauses, fronted elements and subjects of participial clauses, gerunds and infinitival clauses (including both controlled and arbitrary PRO).

In contrast to the proto-f-structure annotation which ignores empty productions, our new extended f-structure annotation translates traces in Penn-II trees into corresponding coindexation in the f-structure representations. The treatment of traces (as well as passive annotation) is implemented in a new fourth component of our annotation algorithm:

$$\boxed{\text{L/R Context APs}} \Rightarrow \boxed{\text{Coord APs}} \Rightarrow \boxed{\text{Catch-All APs}} \Rightarrow \boxed{\text{Trace APs}}$$

Null constituents are treated as full nodes in the annotation (except passive empty objects) and traces are recorded in terms of INDEX = *i* f-structure annotations. Traces without index are translated into arbitrary PRO. The encoding of passive is important as LFG “surface-syntactic” grammatical functions such as SUBJ and OBJ differ from “logical” grammatical functions: surface-syntactic grammatical functions are identified in terms of e.g. agreement phenomena while logical grammatical functions are more akin to thematic roles. The surface-syntactic subject of a passive sentence is usually a logical object, while a surface grammatical object of an optional

⁶The remaining 120 sentences receiving no f-structure are due to inconsistent annotations causing attribute-value structure clashes in the constraint solver. We are confident that the remaining inconsistent annotations will be eliminated soon.

by-prepositional phrase is usually the logical subject. This is exemplified in the “argument-switch” between a proper f-structure and a QLF for the string *an agreement was brokered by the U.N.*:

$$\left[\begin{array}{l} \text{SUBJ} \left[\begin{array}{l} \text{PRED 'AGREEMENT'} \\ \text{NUM SG} \\ \text{SPEC A} \end{array} \right] \\ \text{PRED 'BROKER}(\uparrow \text{SUBJ}, \uparrow \text{OBJ}_{by})' \\ \text{PASSIVE +} \\ \text{OBL}_{by} \left[\begin{array}{l} \text{PRED 'BY}(\uparrow \text{OBJ})' \\ \text{OBJ} \left[\begin{array}{l} \text{PRED 'U.N.}' \\ \text{NUM SG} \\ \text{SPEC THE} \end{array} \right] \end{array} \right] \end{array} \right]$$

?Scope:broker(term(+r,<num=sg,spec=the>,U.N.,?Q,?S),
term(+g,<num=sg,spec=an>,agreement,?P,?R))

In order to capture these “argument-switches”, we extend our automatic f-structure annotation procedure with a (PASSIVE = +) annotation in relative, matrix and subordinate clauses triggered by a variety of contexts encoded in the Penn-II treebank, including sequences of forms of *be* or *get* followed by past participles followed by empty NPs in object position coindexed with the surface subject NP position or by the presence of an LGS (logical subject) Penn-II tag in *by*-prepositional phrases.

A treebank example involving the interaction of passive, relative clause traces and arbitrary PRO is provided in the Appendix.

In order to determine the quality and coverage of the new annotation algorithm we updated our 105 gold-standard f-structure annotated sentences from Section 23 with passive and trace information for wh-questions, relative clauses, fronted elements and subjects of participial clauses, gerunds and infinitival clauses to measure precision and recall. We also compute the fragmentation.

# frags.	# sent.		all annotations	preds only
0	640	Precision	0.977	0.982
1	47783	Recall	0.967	0.959
2	1			

Table 3: Proper-f-structure annotation: fragmentation, precision and recall.

Proper f-structure annotation precision and recall results are an improvement on the best (in terms of coverage) proto-f-structure annotation results

(preds only: precision 0.988 against 0.958; recall 0.959 against 0.908), indicating that the extended annotation algorithm can reliably determine traces for wh-questions, relative clauses, fronted elements and subjects of participial clauses, gerunds and infinitival clauses as well as passives, and reflect them accurately in terms of indices in the f-structure representations. Fragmentation, however, goes up considerably. Currently more 640 sentences do not get an annotation due to inconsistent annotations (against 120 for the proto-f-structure annotation). We are currently working on improving these figures.

4.1 From F-Structures to QLFs: II

In our work we employ a modified and much scaled down subset of the full QLF formalism as presented in [Alshawi & Crouch, 1992; van Genabith and Crouch, 1996]. We distinguish between **terms** and **form(ula)s**. Every predicate is associated with a distinguished referential argument: an eventuality⁷ variable for verbal predicates and an individual variable for other predicates. For nominal structures, individual variables follow the predicate separated by a colon, while they precede all other predicates. Where possible, we distinguish between quantified and non-quantified nominal structures. Quantified terms are of the form **q(Quant,Ty:Var,Restr)** where **Quant** is the surface quantifier/determiner, **Var** is the referential argument, **Ty** is **sg**, **pl** or **ud** (undefined) and **Restr** is of the form **Pred:Var** (in simple cases). Non-quantified terms (e.g. proper names, bare plurals etc.) are of the form **q(Ty:Var,Pred:Var)**. Verbal predications are of the form **Var:Pred(Arg₁,...,Arg_n)** where **Arg_i** are **terms** or **forms**. Formulas are fully unscoped, i.e. no attempt is made to provide quantifier scope prefix constraints as provided by the full QLF formalism. Modification (adjectival, adverbial, prepositional phrase, relative clauses etc.) is treated via relational **md(x,y)** (modifier) or **eq(x,y)** predicates linking modifiers with referential arguments. We do not currently use the higher-order lambda abstraction nor the contextual resolution facilities of the original QLF formalism.

The f-structure to QLF translation algorithm is based on [van Genabith and Crouch, 1996] and extends it to include passive structures, f-structure indices for wh-questions, relative clauses, fronted material and subjects of participial clauses, gerunds and infinitival clauses, modification (adjectival, adverbial, prepositional and appositional sentential and non-sentential adjuncts as well as relative clauses) and coordinate structures.

⁷Our translation does not distinguish between event and state variables.

Because of reasons of space, here we can only discuss a single, simple aspect of the translation algorithm: in the case of a passive f-structure component the algorithm first tries to determine whether it can find a logical subject, i.e. a sub-f-structure marked `LGS = +`. If this is the case, the translation of the sub-f-structure is used as the logical subject in the QLF translation of the passive structure, while the QLF translation of the surface subject f-structure is used as the logical object in the QLF translation. This, and the interaction with relative clause traces, is illustrated in the example provided in the Appendix. If no logical subject is marked in the f-structure, the algorithm inserts a non-specific `q(a,ud:x,'UNDEF':x)` logical subject term.

Currently, the f-structure–QLF translation algorithm associates 46371 of the 47783 f-structures (97.04%) generated with a QLF. This means that 95.76% (46371) of the 48424 trees in the Penn treebank receive a QLF. This number is likely to go up as our work proceeds.

5 A Tree De-Transformation Approach

It is interesting to compare our work with [Liakata and Pulman, 2002]. Liakata and Pulman translate Penn-II treebank trees into flat, set-based descriptions much like [Frank, 2000] and then match these descriptions with templates to extract logical forms. In order to cope with passive and non-local (“moved”) material, they preprocess the (flat representations of the) treebank trees to undo the effects of movement: fronted (topicalised) material is moved back to the location of its coindexed empty node in the tree; passives are transformed into essentially their active counterparts etc. These “de-transformed” trees, where crucially all material is now located where it should be interpreted semantically, are then matched by a small number of logical form extraction templates. The considerable advantage of this approach is that the logical form extraction templates are now much simpler than they would be for the original trees.

Similarly, in our case, the translation of trees into logical forms is simplified by the intermediate level of f-structure representation.

However, there is a deeper reason why the tree de-transformation approach developed by [Liakata and Pulman, 2002] is not available to us. The reason is the following: empty productions and coindexed null elements required to de-transform trees are not the standard fare in probabilistic parsing (except, to a certain extent, in Collins’ (1999) model 3). Indeed, treebanks are usually preprocessed to eliminate empty productions before

extracting a PCFG (c.f. [Charniak, 1996]). Similarly, standard LFG [Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001] assumes surface oriented c-structures, traditionally without empty productions (as in: what you see is what you get). What is more, we cannot use a de-transformed treebank as a basis to extract a PCFG as the CFG rules extracted from a de-transformed treebank do not reflect surface strings.

Given this, it is not immediately clear how one can automatically obtain full Penn-II-style trees complete with empty nodes and coindexation for new text which can then be de-transformed to generate good QLFs.

In addition to annotating treebank trees with (proto- and proper-) f-structures and QLFs, we have extracted PCFGs from proto-f-structure annotated and unannotated versions of the Penn-II treebank [Cahill et al, 2002c]. These PCFG parsers generate proto-f-structures for new text. Proto-f-structures show material where it was encountered, not where it should be interpreted.

In LFG, non-local dependencies and traces are resolved in f-structure in terms of functional uncertainty expressions (regular expressions over paths in f-structures) located where extraposed or dislocated material is actually found, without any need for corresponding traces and null elements in c-structure [Bresnan, 2001; Dalrymple, 2001].

What we have shown in the present paper is how to transfer traces from Penn-II treebank trees into automatically generated f-structures. The question is: can we use this resource to automatically compute functional uncertainty expressions? The answer is yes: the traces in the f-structures generated from the Penn-II treebank trees indicate source and target sites for dislocated material in f-structures. Given these, we can automatically compute shortest paths through f-structures linking source and target. We then collect these paths and compact them into regular expressions yielding functional uncertainty expressions. As in standard LFG, these are then associated with extraposed material (e.g. in the values of TOPIC and FOCUS attributes for fronted material, relative clauses and wh-constructions etc.) and we can parse with standard PCFGs (without empty productions and coindexation across c-structure tree nodes) and resolve non-local dependencies at f-structure. Given this, we will be able to construct viable logical forms from automatically generated f-structures for new text.

6 Conclusion

We have presented a methodology for associating Penn-II treebank trees with simple QLFs by combining and extending the work of [van Genabith and Crouch, 1996] and [Cahill et al, 2002a; Cahill et al, 2002b]. Currently this method associates 95.76% of the 48424 sentences in the treebank with a QLF.

We are currently annotating our 105 test sentences from Section 23 with gold-standard QLF information to evaluate the results of our automatic f-structure to QLF translation. We are refining and extending the coverage of the translation. We are working on computing functional uncertainty equations from f-structure paths. Finally, it would be interesting to compare our QLFs with the ones generated by [Liakata and Pulman, 2002].

The f-structures and QLFs for the first 1000 sentences of the Penn-II are available for inspection at http://www.computing.dcu.ie/~josef/{1000_sent_f-str.html,1000_sent_qlf.html}.

References

- [Alshawi, 1992] H. Alshawi (ed.) 1992. *The Core Language Engine*, MIT Press, Cambridge Mass
- [Alshawi & Crouch, 1992] H. Alshawi and R. Crouch 1992. Monotonic Semantic Interpretation, In *Proceedings 30th Annual Meeting of the Association for Computational Linguistics*, pages 32–38
- [Bresnan, 2001] J. Bresnan 2001. *Lexical-Functional Syntax*. Blackwell, Oxford.
- [Cahill et al, 2002a] Cahill, A., M. McCarthy, J. van Genabith and A. Way (2002). Automatic Annotation of the Penn Treebank with LFG F-Structure Information. in *Proceedings of the LREC Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*, Las Palmas, Canary Islands, Spain, pp.8–15.
- [Cahill et al, 2002b] Cahill, A., M. McCarthy, J. van Genabith and A. Way (2002). Evaluating Automatic F-Structure Annotation for the Penn-II Treebank in *Proceedings of the Treebanks and Linguistic Theories (TLT'02) Workshop*, Sozopol, Bulgaria, Sept.19th-20th, 2002 to appear (2002)
- [Cahill et al, 2002c] A. Cahill, M. McCarthy, J. van Genabith and A. Way 2002. Parsing with PCFGs and Automatic F-Structure Annotation. In: *Proceedings of the Sixth International Conference on Lexical-Functional Grammar*, Athens, Greece, 3 July - 5 July 2002, CSLI Publications, Stanford, CA.
- [Charniak, 1996] Eugene Charniak 1996. *Tree-bank Grammars*, Technical Report CS-96-02, Brown University, Providence, Rhode Island

- [Collins, 1999] M. Collins 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia.
- [Dalrymple, 2001] Mary Dalrymple 2001. *Lexical-Functional Grammar*. San Diego, Calif.; London : Academic Press.
- [Frank, 2000] A. Frank. 2000. Automatic F-Structure Annotation of Treebank Trees. In: (eds.) M. Butt and T. H. King, *The fifth International Conference on Lexical-Functional Grammar*, The University of California at Berkeley, 19 July - 20 July 2000, CSLI Publications, Stanford, CA.
- [Frank et al, 2002] A. Frank, L. Sadler, J. van Genabith and A. Way 2002. From Treebank Resources to LFG F-Structures. In: (ed.) Anne Abeille, *Treebanks: Building and Using Syntactically Annotated Corpora*, Kluwer Academic Publishers, Dordrecht/Boston/London, to appear
- [Kaplan and Bresnan, 1982] R. Kaplan and J. Bresnan 1982. Lexical-functional grammar: a formal system for grammatical representation. In Bresnan, J., editor 1982, *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge Mass. 173–281.
- [Lappin et al, 1989] S. Lappin, I. Golan and M. Rimon 1989. *Computing Grammatical Functions from Configurational Parse Trees* Technical Report 88.268, IBM Israel, Haifa, Israel.
- [Liakata and Pulman, 2002] M. Liakata and S. Pulman 2002. *From trees to predicate-argument structures*. COLING'02, Proceedings of the Conference, Taipei, 24 August – 1 September 2002
- [Magerman, 1994] D. Magerman. 1994. Natural Language Parsing as Statistical Pattern Recognition Ph.D. Thesis, Stanford University, CA
- [Marcus et al, 1994] M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, M. Ferguson, K. Katz and B. Schasberger 1994. The Penn Treebank: Annotating Predicate Argument Structure. In: *Proceedings of the ARPA Human Language Technology Workshop*.
- [Sadler et al, 2000] L. Sadler, J. van Genabith and A. Way. 2000. Automatic F-Structure Annotation from the AP Treebank. In: (eds) M. Butt and T. H. King, *The fifth International Conference on Lexical-Functional Grammar*, The University of California at Berkeley, 19 July - 20 July 2000, CSLI Publications, Stanford, CA.
- [van Genabith and Crouch, 1996] J. van Genabith and D. Crouch 1996. Direct and Underspecified Interpretations of LFG f-Structures. In: *COLING 96*, Copenhagen, Denmark, Proceedings of the Conference. 262–267.

Appendix

The plant , which is owned by Hollingsworth & Vose Co. , was under contract with Lorillard to make the cigarette filters .

```
subj : relmod : focus : index : 1
      pred : which
      subj : index : 1
      passive : +
      xcomp : subj : index : 1
              passive : +
              tense : past
              pred : own
              adjunct : 1 : obj : conj : 2 : num : sing
                      pers : 3
                      pred : hollingsworth
                      3 : pred : vose
                        num : sing
                        pers : 3
                        4 : pred : 'co.'
                          num : sing
                          pers : 3
                      pred : &
                      lgs : +
                      pform : by
                      pred : by
      pred : be
      tense : pres
      spec : det : pred : the
      pers : 3
      pred : plant
      num : sing
pred : be
tense : past
adjunct : 5 : obj : xcomp : subj : pred : 'PR01'
                      inf : +
                      to : +
                      obj : spec : det : pred : the
                          adjunct : 7 : pred : cigarette
                              num : sing
                              pers : 3
                          pred : filter
                          num : pl
                          pers : 3
                          pred : make
                          subj : pred : 'PR01'
                          pers : 3
                          num : sing
                          pred : contract
                          adjunct : 6 : obj : pers : 3
                              num : sing
                              pred : lorillard
                              pform : with
                              pred : with
      pform : under
      pred : under

0:be(q(the,sing:1,(plant:1)
      & 2:be(5:own(8:and(q(sing:10,hollingsworth:10),
        q(sing:11,vose:11),
        q(sing:12,co.:12)),
        q(ud:4,1:4))))))
& (md(0,16)
  & 16:under(q(sing:17,(contract(q(ud:19,PR01:19),
    18:make(q(ud:19,PR01:19),
    q(the,pl:20,(filter:20)
    & md(20,24)
    & q(sing:24,cigarette:24))))):17)
  & (md(17,26)
    & 26:with(q(sing:27,lorillard:27))))))
```