A PROLOG IMPLEMENTATION OF LEXICAL FUNCTIONAL GRAMMAR

Uwe Reyle and Werner Frey

Institute of Linguistics, University of Stuttgart

ABSTRACT

Lexical functional grammar (LFG) is an attempt to solve problems that arise in transformational grammar and ATN-formalisms (Bresnan, 1982). Another powerful formalism for describing natural languages follows from a method for expressing grammars in logic, due to Colmerauer (1970) and Kowalski (1974) called definite clause grammars (DCG) (Warren, Pereira, 1980). Both formalisms are a natural extension of context free grammars (CFG).

The aim of this paper is to show
- how LFG can be translated into DCG
- that the procedural semantics of PROLOG provides an efficient tool for LFG-implementations in that it allows the construction of function structures (f-structures) directly during the parsing process. I.e. it is not necessary to have a separate component which first derives a set of functional equations from the parse tree, and secondly generates a f-structure by solving these equations.

I. Lexical functional grammar

LFG consists of a standard context free grammar (CFG) annotated with functional schemata, determining the assignment of grammatical functions to the CFG-rules. The crucial elements of the theory are the grammatical functions involved both in the representation of lexical items and in the CFG-rules. The grammatical functions assumed in Bresnan *are* classified in subcategorizable (or governable) functions, namely SUBJ, OBJ, 0BJ2, OBL COMP, XCOMP, and nonsubcategorizable ones, ADJ and XADJ. The subcategorizable functions are the only ones to which lexical items can make reference. The role of the grammatical functions involved in the functional equations is to provide a mapping between surface categorial structure and semantic predicate argument structure.

The encoding of function assigning equations into the CFG is done as follows:

(i) lexical entries: a lexical item is associated with one or more semantic predicates (for its distinct senses or meanings) together with their argument lists, which consists of the syntactically subcategorized functions of the lexical form.

This report describes work done in the Department of Linguistics at the University of Stuttgart. It was supported in part by the German Science Foundation proj. Ro 245/12.

Example: A typical LFG analysis is the treatment of the passive-construction implied by the lexical entries:

(1) (↑ PRED) = 'buy (< SUBJ, OBJ >)
(↑ PRED) = 'buy (< ∅, SUBJ >)
(↑ PRED) = 'buy (< OBL_{BY}, SUBJ >)

(ii) Equations associated with the CFG-rules: Suppose the underlying CFG rule is of the form

$$C_n \rightarrow C_{n-1} \ldots C_0$$

With each C_i (i=0, ..., n-1) is associated a function assignment equation of the form $(\uparrow G_i) = \downarrow$ or $\uparrow = \downarrow$ with a grammatical function G_i or a feature assigning equation of the form $(\downarrow F_i) = v$.*

The lexical and syntactic encodings of grammatical functions determine the grammatical relations of a sentence. These relations are formally represented by f-structures which are pairs of f-names and f-values. An f-name is a function or feature symbol. An f-value is a feature, an f-structure, a semantic form or a set of value. There are two kinds of semantic forms: those with argument lists - called lexical forms - and those without. Lexical forms specify a list of subcategorizable functions.

Consider, for example, the f-structure for the sentence

(2) John bought a car

(2')
$$\begin{bmatrix} \text{SUBJ} = \begin{bmatrix} \text{NUM} = \text{sing} \\ \text{PRED} = \text{'John} \end{bmatrix} \\ \text{TENSE} = \text{past} \\ \text{PRED} = \text{buy '(<SUBJ, OBJ>)} \\ \text{OBJ} = \begin{bmatrix} \text{SPEC} = \text{a} \\ \text{NUM} = \text{sing} \\ \text{PRED} = \text{'car} \end{bmatrix} \end{bmatrix}$$

in which the lexical form 'buy (SUBJ, OBJ) is to be read as containing pointers from the SUBJ to the f-name SUBJ, and from the OBJ to the f-name OBJ which carry the instantiation of the f-names down to the lexical form. The semantical form 'John' is not a lexical form.

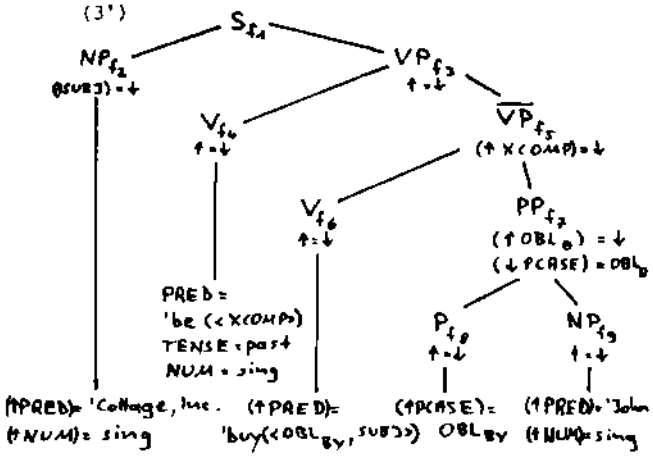
The parsing of a sentence takes place in three

* '↑' denotes that part of the functional structure which is initiated by C_n.
'↓' denotes that part of the functional structure which is initiated by C_i.

steps.

First a parse tree is generated using the CFG (ignoring the functional equations). Secondly the functional equations associated with the categories of the parse tree are instantiated. This produces a functional description of the sentence which is a set of equations. Finally these equations are solved to produce an f-structure.

Example (Halverson, 1982):
(3) Cottage, Inc. was bought by John
(3')



(3'') functional equations:

(• •) $(f_1$ SUBJ) = f_2
(• •) $(f_1$ NUM) = Sing
(•) $f_1 = f_3$
(•) $f_3 = f_4$
$(f_4$ TENSE) = Past
$(f_4$ NUM) = Sing
$(f_4$ PRED) = 'be (<XCOMP>)
$(f_3$ XCOMP) = f_5
$(f_5$ SUBJ) = f_2
$f_6 = f_5$
$(f_6$ PRED) = 'buy (<OBL_{By}, SUBJ>)
$(f_5$ OBL_{By}) = f_7
$f_7 = f_8$
$f_7 = f_9$
$(f_7$ PCASE) = OBL_{By}
$(f_9$ PRED) = 'John'
$(f_9$ NUM) = Sing
$(f_8$ PCASE) = OBL_{By}

(3''') functional structure

$$\begin{bmatrix} f_1 & \text{SUBJ} & f_2 \begin{bmatrix} \text{PRED} & \text{'Cottage, Inc.'} \\ \text{NUM} & \text{Sing} \end{bmatrix} \\ f_3 & & \\ f_4 & \text{TENSE} & \text{Past} \\ & \text{PRED} & \text{'be (<XCOMP>)'} \\ & \text{XCOMP} & f_5 \begin{bmatrix} \text{SUBJ} & [\ ] \\ \text{PRED} & \text{'buy (<OBL}_{By}\text{, SUBJ>)'} \\ \text{OBL}_{By} & f_7 \begin{bmatrix} \text{PCASE} & \text{By} \\ \text{PRED} & \text{'John'} \\ \text{NUM} & \text{Sing} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

If there is more than one solution of the set of equations the sentence is syntactically ambiguous. Furthermore every f-structure has to meet the following additional conditions in order to be a solution:
-uniqueness (or consistency): every f-name which has a value has a unique value.
-completeness: the f-structure must contain f-values for all the f-names subcategorized by its lexical forms.
-coherency: all the subcategorizable functions that the f-structure contains must be subcategorized by its lexical forms.

The ability of lexical items to determine the features of other constituents i.e. government relations, is handled by propagating up the tree the feature set, which is inserted by the lexical item. This is done by the trivial equations * = ^ . In our example the features of 'was' become the features of the V, then of the VP and finally of the S itself. This is (for the feature sing.) expressed in the functional equations marked with (*) . The uniqueness principle guarantees that the subject of the sentence will have the features of the verb (see equations (••)).

Because verbs impose selectional restrictions on their subject, i.e. verbs are subcategorized for their subjects, it follows from the completeness and coherence conditions that verbs govern their subjects. More generally, one can argue by the same reason that lexical items govern all of their subcategorized functions.

As our example shows there can be a relation of referential dependance between an unexpressed subject and an expressed or unexpressed constituent. This relation is called control, the unexpressed subject the controlled element and the constituent on which it is referentially dependend is the controller. In CFG it is assumed that only the so-called open functions XCOMP and XADJ can denote functionally controlled clauses.

II. Definite clause grammar

Like LFG DCG are based on CFG's, too. DCG's are the result of a translation of CFG's into a subset of predicate logic, the so-called Horn-clauses. These Horn clauses can be used as a PROLOG-programme which functions like a top-down parser. DCG's differ from CFG's in the following three points:

-context-sensitivity: the arguments of the PROLOG-functions which correspond to non-terminal categories can be used to transport and to test context-sensitive information.

-construction of tree-structures during the parsing process: structures can be built piecemeal, leaving unspecified parts as variables. The structures can be passed around, and completed as the parsing proceeds. The construction of tree-structures need not be strictly parallel to the application of the corresponding rules.

-inclusion of additional functions or conditions on functions into the context-free rules.

III. DCG implementation of LFG

Our claim is that using DCG as a PROLOG programme the parsing process of a sentence according to the LFG-formalism can be done more efficiently by doing all the three steps described above simultaneously.

To do this, we use the content of the function assigning equations to build up parts of the whole f-structure during the parsing process.

Crucial for this is the fact shown by Bresnan (1982), that every phrase has a unique category, called its head, with the property, that the features of each phrase are identified with those of its head. The head-category of a phrase is caracterized by the assignment of the trivial function-equation and by the property of containing a PRED.

The translation of a LFG-rule like

$$\begin{array}{ccccccc} C_n & \rightarrow & C_{n-1} & \cdots & C_j & \cdots & C_o \\ & & FE_{n-1} & & FE_j & & FE_o \end{array}$$

with the (pairs of) functional equations FE_i into a PROLOG-clause proceeds along the following lines:

(i) For each category C_i there is a fixed set of features. Introduce this feature list as an argument variable of the corresponding PROLOG function C_i, i.e. C_i ($*feat_i$)

(ii) For each grammatical function G_i appearing in the functional equations annotated to C_i introduce the f-name of G_i as argument variable of C_i, i.e. C_i ($*feat_i$ $*G_i$)

(iii) If FE_i is of the form $\uparrow = \downarrow$ and C_i is the head of the phrase, i.e., it contains the whole grammatical information of the phrase, the corresponding PROLOG-function has as additional arguments all the variables introduced by (i) and (ii) for the other category-functions of the phrase, i.e. C_i ($*G_{n-1}$ $*feat_{n-1}$ ... $*G_o$ $*feat_o$).

(iv) If FE_i is a feature assigning function $(\downarrow F_i) = v$ we instantiate the feature variable $*F_i$ in the feature list of C_i by v, where v possibly depends on a lexical entry. (see ($\downarrow$PCASE) = OBL_θ in (3')).

(v) C_n itself is translated into C_n ($*G_i$ $*feat_i$) where C_i is the (unique) head of the phrase with its output $*G_i$.

(vi) The lexical entry functions are such that the PRED's arguments, namely the f-names of subcategorizable functions G_i, are shared with the variables containing or being to contain the content of the f-value of the G_i respectively. (see discussion of coherency beyond)

The uniqueness condition is fulfilled because PROLOG instantiates a variable with at most one value.

In order to account for the coherence condition we make special use of the variables which get instantiated by the lexical entry of the verb. Suppose this is a lexical form like

(↑ PRED) = 'buy' ⟨SUBJ, OBJ⟩

Coherence of an f-structure means that only the arguments of 'buy' occur as subcategorizable functions. Thus the lexical form of the verb predicts how its subcategorized functions must look. We use this property to eliminate the generation of incoherent f-structures in the following way: The variable which is to carry the content of the predicate argument structure of a sentential phrase is "prestructured" by its instantiation with the lexical form of the verb, i.e., it carries at this point the information 'buy' (*SUBJ, *OBJ) that is passed around during the parse, which step by step will instantiate the argument variables. A simple check if the variables are instantiated garanties coherency. The mechanism of passing around partially instantiated f-structures by variables accounts for agreement constraints and government.

To account for completeness the non-trivial case is that one has to check for possible referential dependencies between a controlled element and the controller. That means that if the f-structure f is embedded in another f-structure f_2 and the subcategorizable function SUBJ of f_1 has no f-value its value can be given by the value of the controller, i.e., of a subcategorizable function occuring in f . We add to our DCG additional functions which have as input the set of possible controllers and which compute the referential dependencies espressed by the rules of functional control.(see Bresnan 1982).

Wo use LFG as base for the parsing component of our system described in Frey, et al. (1983).

REFERENCES

Bresnan, J. (Ed.), "The Mental Representation of Grammatical Relations". MIT Press, Cambridge, Massachusetts, 1982

Colmerauer, A.,"Metamorphosis Grammars" In: Bole, L. (Ed.), Natural Language Communication with Computers, Berlin, 1978

Frey, Werner/ Reylc, Uwe/ Rohrer, Christian, "Automatic Construction of a Knowledge Base by Analysing Texts in Natural Language". In: this volume.

Halverson, P.-K., "Semantics for Lexical Functional Grammar". In: Linguistic Inquiry 14, 1902.

Kamp, H., "A Theory of Truth and Semantic Representation". In: J.A. Groenendijk, T.U.V. Janssen and U.B.J. Stokhof (Eds.), Formal Methods in the Study of Natural Language I., 1981

Kowalski, R.A., "Logic for Problem Solving" In: Artificial Intelligence Series 7, North Holland, 1979

Warren, D.H.D./ Pereira, F.C.N., "Definite Clause Grammars for Language Analysis". In: Artificial Intelligence 13, 1980, 213-278