**LEXICAL FUNCTIONAL GRAMMAR**

Aquino, Kurt Neil

De La Salle University, Manila

254 Sitio Bagong Anyo, Pagsawitan, Santa Cruz, Laguna

(+63) 5011485

kurt\_aquino@dlsu.edu.ph

Esquillo, Lance Patrick

De La Salle University, Manila

2311 Taft Ave. corner,

Castro & Fidela A. Reyes St, Manila

(+63) 3306129

patrick\_esquillo@dlsu.edu.ph

Ramos, Luis Angelo

De La Salle University, Manila

#10 East 2nd Street,

Angela Village, Las Pinas City

(+63) 8067631

luis\_ramos@dlsu.edu.ph

**Abstract**

Lexical Functional Grammar (LFG) is a grammar framework in theoretical linguistics, and is one of the varieties of generative grammar. LFG views language as being made up of multiple dimensions of structure. Each of these dimensions is represented as a distinct structure with its own rules, concepts, and form. LFG differs from both transformational grammar and relational grammar in assuming a single level of syntactic structure, although with different representations.

LFG assumes two different ways of representing syntactic structure, the constituent structure or c-structure and the functional structure or f-structure. These two structures constitute two subsystems of the overall system of linguistic structures.

Functional structure is the abstract functional syntactic organization of the sentence, familiar from traditional grammatical descriptions, representing syntactic predicate-argument structure and functional relations like subject and object.

Constituent structure is the overt, more concrete level of linear and hierarchical organization of words into phrases.

Recent LFG work includes investigations of argument structure, semantic structure, and other linguistic structures and their relation to c-structure and f-structure.

**Categories and Subject Descriptors**

[Context Free Grammar]: Lexical Functional Grammar – *grammar framework, theoretical linguistics, generative grammar, phase structure grammar.*

**General Terms**

Algorithms, Documentation, Design, Reliability, Experimentation, Human Factors, Standardization, Languages, Theory, Verification.

**Keywords**

computational linguistics, constraint, syntactic feature, feature structure, Correspondence Architecture, correspondence function, constituent-structure, functional-structure, structural description, Lexical Integrity, grammatical function, functional control.

1. **INTRODUCTION**

In learning their native language, children develop a remarkable set of capabilities. They acquire knowledge and skills that enable them to produce and comprehend an indefinite number of understandable utterances, and make subtle judgments about certain properties.

Kaplan and Bresnan (1982) have adopted what they call the Competence Hypothesis as a methodological principle in order to formulate an explanatory conjecture of the mental operations that perform linguistic abilities. They assume that an explanatory model of human language performance will incorporate a justified representation of the native speaker’s linguistic knowledge, (grammar) as a component separate both from the computational mechanisms that operate on it (processor) and from other non-grammatical processing parameters that might influence the processor’s behavior.

In keeping with the Competence Hypothesis, the formalism called Lexical Functional Grammar (LFG) has been designed to serve as a medium for expressing and explaining important generalizations about the syntax of human languages and thus to serve as a vehicle for independent linguistic research.

LFG is theory of the structure of language and how different aspects of linguistic structure are related. As the name implies, the theory is lexical, which means it richly structured, with lexical relations, rather than with transformations or operations on phrase structure trees in order to capture linguistic generalizations. LFG is also functional, in the sense that grammatical functions like subject and object are primitives of the theory, and are not defined in terms of phrase structure configuration or semantic roles.

On that note, Chomsky (1957, 1965) explained that LFG is a theory of generative grammar. The goal is to explain the native speaker’s knowledge of language by specifying a grammar that models the speaker’s knowledge explicitly and which is distinct from the computational mechanisms that constitute the language processor (Kaplan and Bresnan, 1982).

1. **FORMAL DEFINITION**

Lexical-Functional Grammar (LFG) was first developed in the 1970’s by Joan Bresnan, a linguist at MIT, and Ron Kaplan, a psychologist at Harvard. Bresnan and Kaplan were concerned with the related issues of psychological plausibility and computational tractability. They wanted to create a theory that could form the basis of a realistic model for linguistic learnability and language processing. Since its foundation, the theory has been applied to numerous new areas, undergoing some modifications in the process, and has incorporated insights from a variety of morphological, syntactic, and semantic theories. However, the basic tenets of the theory and the formal framework have remained remarkably stable.

A lexical functional grammar assigns two levels of syntactic description to every sentence of a language. As languages vary greatly in terms of word order and structure, the theory of *constituent structure* allows for variations within certain universally defined parameters. In contrast, all languages share a similar functional vocabulary. According to LFG’s theory of *functional structure*, the abstract syntactic structure of every language is organized in terms of subject, object, and other grammatical functions, most of which are familiar from traditional grammatical work. The grammatical architecture of LFG thus postulates a number of simple data structures with mappings defining the relationships between structures.

* 1. **Lexicon**

The lexicon is where a lot of the work in LFG is done. All the information that ends up in an f-structure starts out in the lexical entries of the words that compose the sentence.

An example of a lexical entry for the verb “loves”:

*loves*: V (↑PRED) = ‘love <(↑SUBJ),( ↑OBJ)>’

(↑TENSE) = present

(↑SUBJ NUM) = sng

(↑SUBJ PERS) = 3rd

This lexical entry says that love is a verb that means ‘love’, and takes two arguments, an obligatory subject and an obligatory object (as contained within the < > brackets). The parentheses here do not mean “optional”, as both arguments are obligatory. It also tells us that loves is the present tense form. Finally it tells us that the subject is third person singular.

All lexical items bring some information to the sentence. For example, we know that the determiner “the” is definite and such information is contained in its lexical entry.

*the*: D (↑DEF) = +

* 1. **Constituent Structures**

Constituent structure encodes linear order, hierarchical groupings, and syntactic categories of constituents, and is the input to the phonological component of the grammar. Language-specific annotations of phrase structure rules identify the grammatical functions that may occur in specific syntactic positions.

Examples of phrase structure rules for English:

S ——> NP VP VP ——> V NP

(↑ SUB) =↓ ↑=↓ ↑=↓ (↑ OBJ) =↓

The arrows are variables; ‘↑’ is to be instantiated by the node immediately dominating the constituent under which the arrow is placed, and ‘↓’ by that node itself. So, the first equation for the rule on the left states that the NP under which the equation is written is the SUB of the S that dominates it. The ‘↑=↓’ equation beneath VP indicates that the features of that node are shared with the higher node. This is the default assignment to phrasal heads, which share information with the dominating phrasal node.

A c-structure is determined by a grammar that characterizes all possible surface structures for a language. This grammar is expressed in a slightly modified context free formalism or a formally equivalent specification such as a recursive transition network (Woods 1970, Kaplan 1972).

* 1. **Functions**

There are two driving forces in Lexical Functional Grammar: the lexicon and functions. The notion of function is borrowed from mathematics and computer science. A *function* is a rule that maps from one item to another. There are two kinds of functions in LFG; the first kind are called grammatical functions and are things like subject, object, etc. We call these *grammatical relations*. The other kind of function refers to the principles that map between the different parts of the grammar, such as the mapping between the *c-structure* and the structure that represents the grammatical functions, formally called as *functional structures* or f-structures.

In LFG, grammatical functions are not defined by a tree; instead, they are primitive notions, meaning they cannot be derived. Every sentence has an f-structure that represents grammatical functions. In the f-structure, a particular NP (noun phrase) will be identified as being the subject of the sentence, quite independent of the tree structure associated with the sentence. In the sentence “Diana loves phonology”, Diana is equated with the SUBJ grammatical function. This equation is usually represented in what is called an Attribute Value Matrix (AVM); the item on the left is the attribute or function, the item on the right is the value attributed to that function:

[SUBJ [PRED3 ‘Diana’]]

Attributes can have various kinds of values, including other AVMs. For example, the value for the SUBJ function in the following sentence is a matrix containing other functions:

The professor loves phonology.

SUBJ PRED ‘professor’

DEF +

NUM sng

Figure 1. Grammatical Function

In the embedded AVM, the function PRED tells the lexical content of the subject NP, DEF tells you if it is definite or not, NUM tells you the number, etc.

These are all properties of the subject. Features other than grammatical functions (e.g. definiteness, etc.) can be represented in these structures as well.

* 1. **Functional Structures**

Functional Structures are the set of all the attribute value pairs for a sentence. Using the previous sentence as an example, we represent its f-structure as:

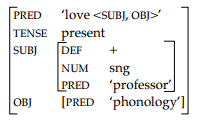


Figure 2. F-structure

The topmost PRED function tells you what the predicate of the sentence is. It also contains information about the a-structure of the sentence. The TENSE feature tells you about the tense of the sentence. The SUBJ and OBJ functions have submatrices (containing information on their internal structure) as values.

C-structures must be related to f-structures somehow. This is accomplished with the use of variables. Consider the following simple c-structure. Each lexical item is followed by the information it contributes by virtue of its lexical entry. Each node in the tree is marked with a variable (f1, f2, f3,…, etc.). These will be used in mapping to the f-structure.

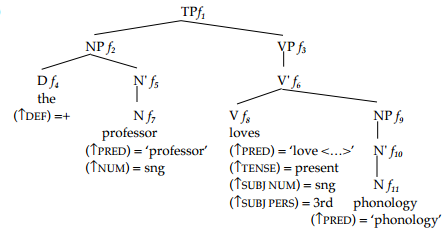


Figure 3. Set of c-structures of a sentence

Each of these variables corresponds to a pair of matrix brackets in the f-structure. There is no one-to-one correspondence here. Multiple nodes in the tree can correspond to the same (sub) AVM:

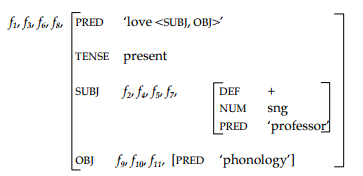


Figure 4. Sub AVM of the nodes

This means that the information contained in nodes f2, f4, f5, f7 contribute the SUBJ features to the sentence. f9, f10, f11, provide the OBJ info, etc. The mapping formally happens through what is called an *f-description*.

The *f-description* is set out with functional equations. These equations tell us, for example, that the subject of the sentence f1 corresponds to the constituent f2. This is written as:

(f1 SUBJ) = f2

The fact that the subject NP (f2) is definite is encoded in D node (f4):

f2 = f4

When a node is a head, or simply passes information up the tree (e.g., V' or V), then a simple equivalence relation is stated:

f3 = f6

These functional equations control how information is mapped through the tree and between the tree and the f-structure. Each piece of information in the f-structure comes from (various places in) the tree as controlled by the functional equations in the f-description.

For more examples about the formal definitions and architecture of Lexical Functional Grammar, see Kaplan and Dalrymple (1995), Asudeh and Toivonen (2009), and *Syntax: A Generative Introduction* (2013).

1. **COMPARISON TO RELATED FUNDAMENTAL AUTOMATA**

Considering that Lexical Functional Grammar is under Context Free Grammar, it can be accepted by pushdown automata, which are finite state automata equipped with a stack for memory. Languages generated by regular expressions on the other hand, can only be accepted by finite state automata, where what you can accept at one state does not depend on anything except the state that you're currently in.

For a quick comparison, Regular Expressions are used as the basis of lexical analysis in order to represent Regular Languages while Context Free Grammars are used as a basis of parsing to represent language constructs. In this sense Regular Grammar is either right or left linear, whereas Context Free Grammar is basically any combination of terminals and non-terminals, meaning that Regular Grammar is a subset of Context Free Grammar.

1. **SAMPLE APPLICATIONS**

One famous example of the application of Lexical Functional Grammar is the Parsing the Wall Street Journal in 2002.

Riezler et al. (2002) presented a stochastic parsing system consisting of a Lexical-Functional Grammar (LFG), a constraint-based parser and a stochastic disambiguation model. They reported on the results of applying this system to parsing the UPenn Wall Street Journal (WSJ) treebank. The model combines full and partial parsing techniques to reach full grammar coverage on unseen data. The treebank annotations are used to provide partially labeled data for discriminative statistical estimation using exponential models. Disambiguation performance is evaluated by measuring matches of predicate-argument relations on two distinct test sets. An evaluation on a gold standard of dependency relations for Brown corpus data and manually annotated f-structures for a subset of the WSJ treebank, this evaluation achieves 76% F-score.

Another example of an application of LFG, although indirect, is its usage, alongside with Natural Language Processing, in the development of the artificial intelligence of the *Jeopardy!* playing robot, IBM Watson. Although LFG was not directly used, similar functions and structures were used for the generation of the AI in the sense that both LFG and NLP covers generative grammar and grammar framework.

IBM Research undertook a challenge to build a computer system that could compete at the human champion level in real time on the American TV quiz show, Jeopardy. The extent of the challenge includes fielding a real-time automatic contestant on the show, not merely a laboratory exercise. After three years of intense research and development by a core team of about 20 researchers, Watson is performing at human expert levels in terms of precision, confidence, and speed at the Jeopardy quiz show.

Their results strongly suggest that DeepQA (IBM Watson’s NLP), is an effective and extensible architecture that can be used as a foundation for combining, deploying, evaluating, and advancing a wide range of algorithmic techniques to rapidly advance the field of question answering (QA).

1. **DISCUSSIONS AND OBSERVATIONS**

LFG differs from other syntactic theories in its adoption of formally and conceptually distinct structures (c-structure and f-structure). Although Relational Grammar has a structure that is similar to f-structure in that it models grammatical functions, it does not articulate a theory of constituent structure. Head-Driven Phrase Structure Grammar represents constituency and grammatical functions in a single formal structure. Principles & Parameters Theory does not acknowledge grammatical functions as such at all, attempting to derive them from phrase structure, which is the representation used to model all syntactic information.

According to Asudeh and Toivonen (2009), a further advantage of LFG is its explicit and detailed representation of lexical information as lexical features. A small change in lexical information can have major syntactic consequences. Thus, both synchronic and diachronic variation can be readily represented as lexica variation.

1. **CONCLUSION**

In addition to grammatical modularity, another underlying principle of LFG theory is that grammatical information grows monotonically (Bresnan 2001b: chapter 5), which means that it grows in an information-preserving manner. For example, as an f-description grows in size through the addition of new defining equations, the minimal f-structure that models the description also grows in size, becoming increasingly specific.

LFG is unique in its popularity both among computational linguists, who investigate and capitalize on formal and algorithmic properties of LFG grammars, and among descriptive and documentary linguists, who use the theory as a tool to understand and document understudied languages. LFG’s usefulness for language description is summarized briefly by Kroeger (2007):

“LFG has a number of features that make it an attractive and useful framework for grammatical description, and for translation. These include the modular design of the system, the literal representation of word order and constituency in c-structure, a typologically realistic approach to universals (avoiding dogmatic assertions which make the descriptive task more difficult), and a tradition of taking grammatical details seriously” (Kroeger 2007: 1)

1. **REFERENCES**

Asudeh, A., & Toivonen, I. (2009). Lexical Functional Grammar. In The Oxford Handbook of Linguistic Analysis (17th ed.). Oxford: Oxford University Press.

Bresnan, J. 2001b. Lexical-Functional Syntax. Oxford: Blackwell.

Chomsky, N. 1957. Syntactic Structures. The Hague: Mouton.

Chomsky, N. 1965. Aspects of the Theory of Syntax. Cambridge, MA: MIT Press.

Dalrymple, M. (2001). Lexical Functional Grammar. Syntax and Semantics Series, 34(Final Version). Retrieved December 3, 2015, from http://web.stanford.edu/group/nasslli/courses/as-cr-da/apbook-nasslli.pdf

Kaplan, R. 1972. Augmented transition networks as psychological models of sentence comprehension. Artificial Intelligence 3:77-100.

Kaplan, R., & Bresnan. 1982. Lexical-Functional Grammar: A Formal System for Grammatical Representation. In Bresnan 1982b, 173–281. Reprinted in Dalrymple et al. (1995: 29–135).

Kaplan, R., & Bresnan, J. (1995). Lexical Functional Grammar A Formal System for Grammatical Representation. Formal Issues in Lexical Functional Grammar, 82-95.

Kaplan, R., & Darlymple, M. (1995). The Formal Architecture of Lexical-Functional Grammar. Retrieved December 3, 2015, from http://www.essex.ac.uk/linguistics/external/lfg/www-lfg.stanford.edu/pubs/lfg-presentations/ESSLLI95/overview.pdf

Kroeger, Paul. 2007. LFG as a Framework for Descriptive Grammar. In Butt and King 2007.

Liddy, E. (2011, February 25). IBM's Watson on Jeopardy: A victory in Natural Language Processing - Information Space. Retrieved December 3, 2015, from http://infospace.ischool.syr.edu/2011/02/25/ibms-watson-on-jeopardy-a-victory-in-natural-language-processing/

Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques. (2002). Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 271-278. Retrieved December 3, 2015, from http://delivery.acm.org/10.1145/1080000/1073129/p271-riezler.pdf?ip=114.108.236.34&id=1073129&acc=OPEN&key=4D4702B0C3E38B35.4D4702B0C3E38B35.4D4702B0C3E38B35.6D218144511F3437&CFID=565576583&CFTOKEN=87966120&\_\_acm\_\_=1449152304\_6030801829d99c6aeb0cbcb

Riezler, S., King, T., Kaplan, R., Crouch, R., Maxwell, J., & Johnson, M. (2002). Parsing the wall street journal using a Lexical-Functional Grammar and discriminative estimation techniques. Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02.

Woods, William. 1970. Transition Network Grammars for Natural Language Analysis. Communications of the ACM 13(10):591-606.