# Lexical Functional Grammar

Mary Dalrymple
Centre for Linguistics and Philology
Walton Street
Oxford OX1 2HG
United Kingdom

mary.dalrymple@ling-phil.ox.ac.uk

## Abstract

Lexical Functional Grammar (LFG) is a linguistic theory which studies the various aspects of linguistic structure and the relations between them. Traditional LFG analyses focus on two syntactic structures. **Constituent structure** (c-structure) represents word order and phrasal groupings, and **functional structure** (f-structure) represents grammatical functions like subject and object. These structures have separate representations, but are related to each other in systematic ways. Recent LFG work includes investigations of argument structure, semantic structure, and other linguistic structures and their relation to c-structure and f-structure.

## 1 LFG's syntactic structures

Lexical Functional Grammar is a theory of the structure of language and how different aspects of linguistic structure are related. As the name implies, the theory is **lexical**: the lexicon is richly structured, with lexical relations rather than transformations or operations on phrase structure trees as a means of capturing linguistic generalizations. It is also **functional**: grammatical functions like subject and object are primitives of the theory, not defined in terms of phrase structure configuration or semantic roles.

LFG assumes that two syntactic levels are important in the analysis of linguistic structure. F(unctional)-structure represents abstract grammatical functions like subject and object as well as abstract features like tense and case. Another level, c(onstituent)-structure, represents the concrete phrasal expression of these relations, governed by language-particular constraints on word order and phrase structure. This duality of syntactic representation is motivated by the different nature of these two structures both within and across languages. Languages vary greatly in word order and phrasal structure, and the theory of constituent structure allows for this variation within certain universally-defined parameters. In contrast, all languages share the same 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.

Regularities in the relation between c-structure and f-structure are captured by functions relating parts of one structure to parts of the other. For example,

the subject phrase in the c-structure tree is related to the subject f-structure by means of a function which relates nodes of the c-structure tree to parts of the f-structure for a sentence. Relations between c-structure, f-structure, and other linguistic levels have also been explored and defined in terms of functional mappings from subparts of one structure to the corresponding subparts of other structures.

The overall formal structure and basic linguistic assumptions of the theory have changed very little since its development in the late 1970s by Joan Bresnan, a linguist trained at the Massachusetts Institute of Technology, and Ronald M. Kaplan, a psycholinguist and computational linguist trained at Harvard University. Bresnan (1982) is a collection of influential early papers in LFG; recent works providing an overview or introduction to LFG include Dalrymple et al. (1995), Bresnan (2001), Dalrymple (2001), Falk (2001), and Kroeger (2004).

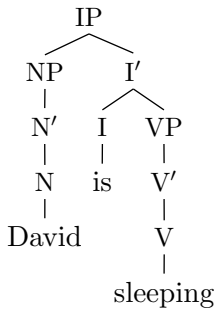
## 2 Constituent structure

Languages vary greatly in the basic phrasal expression of even simple sentences. Basic word order can be verb-initial (Malagasy), verb-final (Japanese), or verb-medial (English). Word order correlates with grammatical function in some languages, such as English, in which the subject and other arguments appear in particular phrase structure positions. In other languages, word order is more free, and grammatical functions are identified by casemarking or agreement rather than phrasal configuration: in many languages, there is no specific phrasal position where the subject or object must always appear. Requirements for phrasal groupings also differ across languages. In English, for example, a noun and any adjectives that modify it must appear together and form a phrasal unit. In many other languages, including Latin, this is not necessary, and a noun can be separated from its modifying adjectives in the sentence. LFG's **constituent structure** represents word order and phrasal constituenthood.

### 2.1 Constituent structure representation

Like many other linguistic theories, LFG represents word order and phrasal groupings by means of phrase structure trees, also called constituent structures (see **Constituent structure**) or **c-structures**. The c-structure for an English sentence like *David is sleeping* is:

(1) *David is sleeping.*




C-structure trees contain two sorts of categories. Categories such as N (noun) and V (verb), familiar from traditional grammatical analysis, are called **lexical categories**. Most LFG analyses assume at least the lexical categories N (noun), A (adjective), V (verb), Adv (adverb), and P (preposition), though more or fewer categories may be relevant for a particular language. Most languages also make use of a set of **functional categories**, including I (for Inflection), C (for Complementizer), and D (for Determiner). Functional categories play an organizing role in the syntax, and are either associated with closed-class categories such as complementizers, or are filled with subtypes of particular lexical categories.

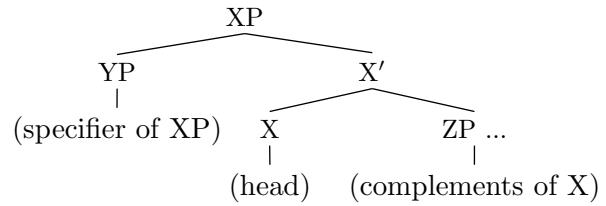
Constituent structure is organized according to **X-bar theory** (see **X-bar theory**), which assumes that phrases are internally headed and therefore **endocentric**: a phrase and its head have the same category, but a different bar level. For example, the basic lexical category N is the head of the single bar level category N′ ("N-bar"), which in turn is the head of the two-bar-level category N″ ("N-double-bar"). Similarly, the basic functional category I is the head of I′, which heads I″. Many LFG analyses assume that N″ and I″ are maximal phrases, meaning that there is no category N‴ or I‴ for the double-bar-level category to head. Thus, as maximal phrases, the categories N″ and I″ are usually written as NP (noun phrase) and IP (the category assigned to a sentence like *David is sleeping*). Nonprojecting categories are also assumed (Toivonen, 2003); these are lexical categories that are not heads of phrases, but appear on their own, adjoined to heads. For example, verbal particles (words corresponding to the particle *up* in a sentence like *I woke up the baby*) in some Germanic languages are nonprojecting words, typically prepositions, adjoined to the verb.

Not all phrases are endocentric. LFG assumes a single exocentric, non-headed category, the category S, which does not obey the constraints of X-bar theory. Not all languages make use of this phrase; it plays no role in the syntax of English, for example. In languages that make use of this phrase, it behaves as a maximal phrase, but it has no c-structure head, and it can dominate phrases of any category or bar level. As we will see below, it does not obey the generalizations which constrain the relation between endocentric c-structure categories and f-structure, and so is often motivated in languages with very free word order, in which morphological information rather than phrase structure

position determines the functional role of a phrase.

A phrase can dominate other constituents besides its head. LFG does not require phrase structure trees to be binary branching, and so there can be more than two daughters of any node in a c-structure tree. The nonhead daughter of a maximal phrase is called its **specifier**, and the nonhead sisters of a lexical category are its **complements**. This is shown schematically in (2):

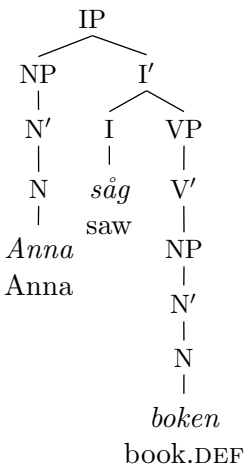
(2)




As shown in (1), a verbal category, often an auxiliary, appears in I. The complement of I is either VP, as in (1), or, in languages which make use of it, the exocentric category S. Not all languages make use of the functional category C and the phrases it heads, C′ and CP. When a language makes use of this category, complementizers or verbs can appear in C, and the complement of C is IP or S. The functional category D, filled by a determiner, is also often assumed: the complement of D is NP. In the following, we will not assume DP, which will mean that the category of a phrase like *the boy* is NP. However, there is no general agreement of the status of such phrases in LFG: according to some analyses, *the boy* is a DP rather than an NP in at least some languages.

Unlike many theories, LFG assumes that daughters of all phrasal categories are optional. In particular, the head of a maximal phrase need not appear. Sells (2001) examines the distribution of nonfinite verbs in Swedish main and subordinate clauses, and shows that Swedish is best analyzed by assuming that the complement of I is VP, and that nontensed verbs appear as the head of VP. In main clauses, however, the tensed verb appears in I and not as the head of VP; this is a common situation crosslinguistically, and is true for Russian as well (King, 1995). Thus, a Swedish sentence like (3), with a tensed verb and no nontensed verbs, has a VP that does not contain a V.

(3) *Anna såg boken.*
Anna saw book.DEF
'Anna saw the book.'




Nonhead daughters are also only optionally present. In Japanese and other so-called "pro-drop" languages, a verb can appear with no overt arguments. If no overt arguments of a verb are present, the c-structure tree contains only the verb. As a relatively free word order language, Japanese makes use of the exocentric category S, and so an utterance S can consist of a single constituent of category V:

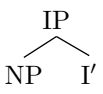
(4) *koware-ta*
break-PAST
'[It/Something] broke.'


C-structure does not contain subparts of words or unpronounced features, nor does it contain null pronominals in pro-drop languages like Japanese. Rather, it reflects the structure and grouping of the full syntactic units – the words and phrases – in the sentence.

### 2.2 Phrase structure rules

LFG draws a strong distinction between the formal objects of the theory — constituent structure trees and functional structures — and the constraints or descriptions involving those objects. C-structure trees are constrained by phrase structure rules, which license local tree configurations. The phrase structure rule in (5a) licenses the c-structure in (5b):

(5) a. IP $\longrightarrow$ NP I′

b. 


The right-hand side of an LFG phrase structure rule is a **regular expression** (see **Formal Languages**), allowing for disjunction, optionality, and arbitrary repetition of a node or sequence of nodes. The V and NP daughters in the rule in (6) are optional, and the Kleene star (*) annotation on the PP indicates that a sequence of zero or more PP constituents may appear.

(6) V′ $\longrightarrow$ (V) (NP) PP*

## 3 Functional structure

Syntactic analyses in traditional grammatical descriptions are stated in terms of abstract syntactic functions like subject, object, and complement. These functions are represented at LFG's **functional structure**. Functional structure, or **f-structure**, represents abstract grammatical functions like subject and object as well as features such as tense, case, person, and number.

### 3.1 Grammatical functions and their representation

In a sentence like *David devoured a sandwich*, *David* is the subject and *a sandwich* is the object. This information is represented by an **attribute-value structure**, the f-structure, in which the value of the SUBJ feature is the f-structure for the subject, and the value of the OBJ feature is the f-structure for the object.

(7) David devoured a sandwich.

$$\begin{bmatrix} \text{PRED} & \text{`DEVOUR}\langle\text{SUBJ,OBJ}\rangle\text{'} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{`DAVID'} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{SPEC} & \text{A} \\ \text{PRED} & \text{`SANDWICH'} \end{bmatrix} \end{bmatrix}$$

For clarity, many of the features and values in this f-structure have been omitted, a practice often followed in LFG presentations. The full f-structure would contain tense, aspect, person, number, and other functional features.

Every content word in a sentence contributes a value for the feature PRED. These values are called **semantic forms**. In the functional structure, semantic forms are surrounded by single quotes: the semantic form contributed by the word *David* is 'DAVID'.

An important property of semantic forms is that they are uniquely instantiated for each instance of their use, reflecting the unique semantic contribution of each word within the sentence. This is occasionally indicated by associating a unique numerical identifier with each instance of a semantic form, as in (8):

(8) David devoured a sandwich.

$$\begin{bmatrix} \text{PRED} & \text{`DEVOUR}_{37}\langle\text{SUBJ,OBJ}\rangle\text{'} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{`DAVID}_{42}\text{'} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{SPEC} & \text{A} \\ \text{PRED} & \text{`SANDWICH}_{14}\text{'} \end{bmatrix} \end{bmatrix}$$

In (8), the particular occurrence of the semantic form for the word *David* as it is used in this sentence is represented as 'DAVID$_{42}$'. Another use of *David* would be associated with a different unique identifier, perhaps 'DAVID$_{73}$'. Representing semantic forms with explicit numerical identifiers clearly shows that each word makes a unique contribution to the f-structure. However, the identifiers also add unnecessary clutter to the f-structure, and as such are usually not displayed.

A verb or other predicate generally requires a particular set of arguments: for example, the verb *devoured* requires a subject (SUBJ) and an object (OBJ). These arguments are said to be **governed** by the predicate; equivalently, the predicate is said to **subcategorize for** its arguments (see **subcategorization**). The semantic form contributed by a verb or other predicate contains information about the arguments it governs. As shown above, the governed arguments appear in angled brackets: 'DEVOUR⟨SUBJ,OBJ⟩'.

The LFG requirements of Completeness and Coherence ensure that all and only the grammatical functions governed by a predicate are found in the f-structure of a grammatically acceptable sentence. For example, the unacceptability of example (9) shows that the verb *devoured* cannot appear without an OBJ:

(9) *David devoured.

This sentence violates the principle of Completeness, according to which every grammatical function governed by a predicate must be filled. Here, the OBJ is not present, and the sentence is incomplete.

Furthermore, *devour* cannot appear with other functions besides the grammatical functions SUBJ and OBJ that it governs. Example (10) shows that it cannot appear with a sentential complement in addition to its object:

(10) *David devoured a sandwich that it was raining.

This sentence violates the principle of Coherence, according to which only the grammatical functions that are governed by a predicate can appear. Since the sentence contains a grammatical function that the verb *devour* does not govern, it is incoherent.

The grammatical functions that a predicate can govern are called **governable grammatical functions**. The inventory of universally-available governable grammatical functions is given in (11). Languages differ as to which of these functions are relevant, but in many languages, including English, all of these functions are used.

(11)

| | |
|---|---|
| SUBJ: | subject |
| OBJ: | object |
| COMP: | sentential or closed (nonpredicative) infinitival complement |
| XCOMP: | an open (predicative) complement, often infinitival, whose SUBJ function is externally controlled |
| $\text{OBJ}_\theta$: | a family of secondary OBJ functions associated with a particular, language-specific set of thematic roles; in English, only $\text{OBJ}_{\text{THEME}}$ is allowed, while other languages allow more than one thematically restricted secondary object |
| $\text{OBL}_\theta$: | a family of thematically restricted oblique functions such as $\text{OBL}_{\text{GOAL}}$ or $\text{OBL}_{\text{AGENT}}$, often corresponding to adpositional phrases at c-structure |

Not all phrases fill argument positions of a predicate: **modifying adjunct** phrases are not required by a predicate and hence are not governable. In (12), the phrase *yesterday* bears the nongovernable grammatical function ADJ(unct):

(12) David devoured a sandwich yesterday.

There are two nongovernable grammatical functions. The function ADJ is the grammatical function of modifiers like *in the park*, *with a hammer*, and *yesterday*. The function XADJ is the grammatical function of open predicative adjuncts whose subject is externally controlled; as with the governable grammatical function XCOMP, the X in the name of the function indicates that it is an open function whose SUBJ is supplied externally. The phrase filling the XADJ role is underlined in (13).

(13) a. Having opened the window, David took a deep breath.

b. David ate the celery naked.

c. David ate the celery raw.

In (13a) and (13b), the open adjunct XADJ is controlled by the subject of the main clause: it is David that opened the window, and it is David that is naked. In (13c), the XADJ is controlled by the object: it is the celery that is raw.

Unlike governable grammatical functions, more than one adjunct function can appear in a sentence:

(14) David devoured a sandwich at noon yesterday.

Since the ADJ function can be multiply filled, its value is a **set** of f-structures:

(15) David devoured a sandwich at noon yesterday.

$$\begin{bmatrix} \text{PRED} & \text{`DEVOUR}\langle\text{SUBJ,OBJ}\rangle\text{'} \\ \text{SUBJ} & \begin{bmatrix}\text{PRED} & \text{`DAVID'}\end{bmatrix} \\ \text{OBJ} & \begin{bmatrix}\text{SPEC} & \text{A} \\ \text{PRED} & \text{`SANDWICH'}\end{bmatrix} \\ \text{ADJ} & \left\{ \begin{bmatrix}\text{PRED} & \text{`YESTERDAY'}\end{bmatrix}, \begin{bmatrix}\text{PRED} & \text{`AT}\langle\text{OBJ}\rangle\text{'} \\ \text{OBJ} & \begin{bmatrix}\text{PRED} & \text{`NOON'}\end{bmatrix}\end{bmatrix} \right\} \end{bmatrix}$$

The same is true of XADJ: more than one XADJ phrase can appear in a single sentence:

(16) Having opened the window, David ate the celery naked.

Hence, the value of the XADJ feature is also a set of f-structures.

The f-structures that have been presented so far have included only a subset of their functional features. In fact, it is common in LFG literature to display only those features that are relevant to the analysis under discussion, since a full representation would often be too unwieldy. A full f-structure for these sentences would contain at least the features and values listed in (17), and probably other language-specific features and values as well.

(17)

| | Feature | Value |
|---|---|---|
| Person: | PERS | 1, 2, 3 |
| Gender: | GEND | MASC, FEM, ... |
| Number: | NUM | SG, DUAL, PL, ... |
| Case: | CASE | NOM, ACC, ... |
| Surface form: | FORM | Surface word form |
| Verb form: | VFORM | PASTPART, PRESPART,... |
| Complementizer form: | COMPFORM | Surface form of complementizer: THAT, WHETHER,... |
| Tense: | TENSE | PRES, PAST,... |
| Aspect: | ASPECT | F-structure representing complex description of sentential aspect. Sometimes abbreviated as e.g. PRES.IMPERFECT |
| Pronoun type: | PRONTYPE | REL, WH, PERS,... |

The values given in this chart are the ones that are most often assumed, but some authors have argued for a different representation of the values of some features. For example, Dalrymple & Kaplan (2000) argue for a set-based representation of the PERS and GEND features to allow for an account of feature resolution in coordination, and of the CASE feature to allow for case indeterminacy. Some studies assume a PCASE feature whose value specifies the grammatical function of its phrase; in more recent work, Nordlinger (1998) provides

a theory of constructive case, according to which a casemarked phrase places constraints on its f-structure environment that determine its grammatical function in the sentence. This treatment supplants the traditional treatment of obliques in terms of the PCASE feature.

### 3.2 Functional descriptions

As with c-structures, we draw a sharp distinction between f-structures and their descriptions. The set of f-structure constraints associated with the analysis of some sentence is called a **functional description** or **f-description**.

To refer to the value of a feature in some f-structure, we use an expression like the following:

(18) Reference to the value of the TENSE feature in $f$:
$(f\ \text{TENSE})$

This expression refers to the value of the TENSE feature in the f-structure $f$. If we want to specify the value of that feature, we use an expression like:

(19) $(f\ \text{TENSE}) = \text{PAST}$

This **defining equation** specifies that the feature TENSE in the f-structure $f$ has the value PAST.

We can also specify that a feature has a particular f-structure as its value. The expression in (20) specifies that the value of the SUBJ feature in $f$ is the f-structure $g$:

(20) $(f\ \text{SUBJ}) = g$

Some features take as their value a **set** of functional structures. For example, since any number of adjuncts can appear in a sentence, the value of the feature ADJ is a set. We can specify that an f-structure $h$ is a member of the ADJ set with the following constraint, using the set-membership symbol $\in$:

(21) $h \in (f\ \text{ADJ})$

The constraints discussed so far are called **defining constraints**, since they define the required properties of a functional structure. An abbreviated f-description for a sentence like *David sneezed* is:

(22) $(f\ \text{PRED}) = \text{`SNEEZE}\langle\text{SUBJ}\rangle\text{'}$
$(f\ \text{TENSE}) = \text{PAST}$
$(f\ \text{SUBJ}) = g$
$(g\ \text{PRED}) = \text{`DAVID'}$

This f-description holds of the following f-structure, where the f-structures are annotated with the names used in the f-description (22):

(23) David sneezed.

$$f:\begin{bmatrix}\text{PRED} & \text{`SNEEZE}\langle\text{SUBJ}\rangle\text{'} \\ \text{TENSE} & \text{PAST} \\ \text{SUBJ} & g:\begin{bmatrix}\text{PRED} & \text{`DAVID'}\end{bmatrix}\end{bmatrix}$$

Notice, however, that the f-description also holds of the f-structure in (24), which also contains all the attributes and values that are mentioned in the f-description in (22):

(24)
$$f:\begin{bmatrix}\text{PRED} & \text{`SNEEZE}\langle\text{SUBJ}\rangle\text{'} \\ \text{TENSE} & \text{PAST} \\ \text{SUBJ} & g:\begin{bmatrix}\text{PRED} & \text{`DAVID'} \\ \text{PERS} & 3\end{bmatrix} \\ \text{ADJ} & \left\{\begin{bmatrix}\text{PRED} & \text{`YESTERDAY'}\end{bmatrix}, \begin{bmatrix}\text{PRED} & \text{`AT}\langle\text{OBJ}\rangle\text{'} \\ \text{OBJ} & \begin{bmatrix}\text{PRED} & \text{`NOON'}\end{bmatrix}\end{bmatrix}\right\}\end{bmatrix}$$

However, the f-structure in (24) is not the **minimal** or smallest solution to the f-description in (22), since it contains additional attributes and values that do not appear in the f-description. We require the f-structure solution for a particular f-description to be the **minimal solution** to the f-description: no additional attributes or values that are not mentioned in the f-description are included. Thus, the correct solution to the f-description in (22) is the f-structure in (23), not the larger one in (24). Formally, the solution to an f-description is the most general f-structure that satisfies the f-description, which subsumes all other (larger) f-structures that satisfy the f-description.

Besides the defining constraints described above, LFG also allows elements of the f-description to check the properties of the minimal solution to the defining equations. The expression in (25) is a **constraining equation**, distinguished from a constraining equation by the $c$ subscript on the equals sign in the expression:

(25) Constraining equation:

$(f\ \text{SUBJ}\ \text{NUM}) =_c \text{SG}$

When this expression appears, the f-structure $f$ that is the minimal solution to the defining equations must contain the feature SUBJ whose value has a feature NUM with value SG. The constraining equation in (25) does not hold of the f-structure in (23), since in that f-structure, the value of the NUM feature has been left unspecified, and the SUBJ of $f$ does not have a NUM feature with value SG.

In contrast, the functional description in (26a) for the sentence *David sneezes* has a well-formed solution, the f-structure in (26b):

(26) a. $(f\ \text{PRED}) = \text{`SNEEZE}\langle\text{SUBJ}\rangle\text{'}$
$(f\ \text{TENSE}) = \text{PRES}$
$(f\ \text{SUBJ}) = g$
$(g\ \text{PRED}) = \text{`DAVID'}$
$(g\ \text{NUM}) = \text{SG}$
$(f\ \text{SUBJ NUM}) =_c \text{SG}$

b. $$f\!:\begin{bmatrix}\text{PRED} & \text{`SNEEZE}\langle\text{SUBJ}\rangle\text{'} \\ \text{TENSE} & \text{PRES} \\ \text{SUBJ} & g\!:\begin{bmatrix}\text{PRED} & \text{`DAVID'} \\ \text{NUM} & \text{SG}\end{bmatrix}\end{bmatrix}$$

Here, the value SG for the NUM feature for $g$ is specified in the second-to-last line of the functional description. Thus, the f-structure in (26b) satisfies the defining constraints given in the first five lines of (26a). Moreover, it satisfies the constraining equation given in the last line of (26a).

We can also place other requirements on the minimal solution to the defining equations in some f-description. The expression in (27a) requires $f$ not to have the value PRESENT for the feature TENSE, which can happen if $f$ has no TENSE feature, or if $f$ has a TENSE feature with some value other than PRESENT. When it appears in a functional description, the expression in (27b) is an **existential constraint**, requiring $f$ to contain the feature TENSE, but not requiring any particular value for this feature. We can also use a **negative existential constraint** to require an f-structure not to contain a feature, as in (27c), which requires $f$ not to contain the feature TENSE with any value whatsoever.

(27) a. Negative equation: $(f\ \text{TENSE}) \neq \text{PRESENT}$

b. Existential constraint: $(f\ \text{TENSE})$

c. Negative existential constraint: $\neg(f\ \text{TENSE})$

Functional descriptions can also be stated in terms of the Boolean operations of conjunction, disjunction, and negation. In the f-descriptions given above, we implicitly assumed that the constraints in the f-description are interpreted conjunctively: if an f-description contains more than one requirement, each requirement must hold. LFG also allows disjunctions and negations of sets of requirements. For example, a verb like *sneeze* contributes the following f-description:

(28) *sneeze* $(f\ \text{PRED}) = \text{`SNEEZE}\langle\text{SUBJ}\rangle\text{'}$
$\{(f\ \text{VFORM}) = \text{BASE} \mid$
$(f\ \text{TENSE}) = \text{PRES}$
$\neg\{(f\ \text{SUBJ PERS}) = 3$
$(f\ \text{SUBJ NUM}) = \text{SG}\}\ \}$

Disjunction is indicated by curly brackets, with the alternatives separated by a vertical bar |. Negation for a set of requirements is represented by prefixing

¬, and the scope of negation is indicated by curly brackets. This lexical entry allows two possibilities. The first is for the base form of the verb, in which the value of the VFORM feature is BASE. For the second possibility, the value of the feature TENSE is PRES for present tense, and a third-person singular subject is disallowed by negating the possibility for the PERS feature to have value 3 when the NUM feature has value SG.

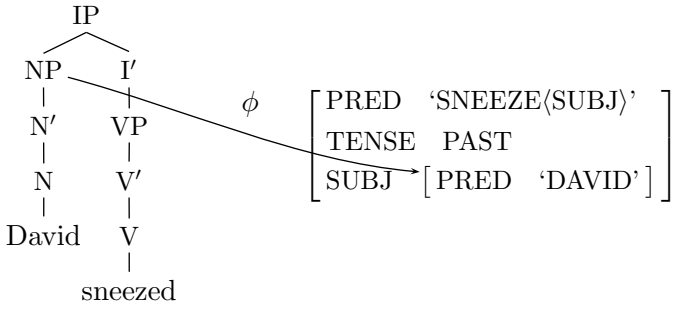
## 4 The c-structure/f-structure relation

There are clear crosslinguistic regularities relating constituent structure positions to grammatical functions. In particular, phrases and their heads are required to correspond to the same f-structure, and specifier and complement positions are associated with particular grammatical functions. Such generalizations constrain the relation between c-structure positions and the f-structure positions they are associated with.

### 4.1 Structural correspondences

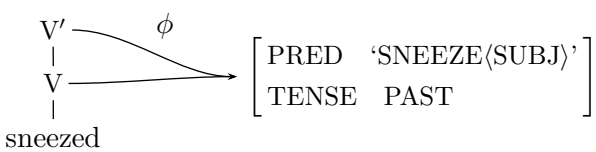
To express these generalizations formally, relating nodes in the c-structure tree and the f-structures they correspond to, we can define a function called $\phi$ (phi) relating nodes of the c-structure tree to parts of the f-structure for a sentence. In (29), the $\phi$ function from the NP node to the f-structure it corresponds to is represented by an arrow and labeled $\phi$:

(29) *David sneezed.*




Each node of the c-structure tree corresponds to some part of the f-structure. As shown in (30), more than one c-structure node can correspond to the same f-structure (the $\phi$ function is **many-to-one**):

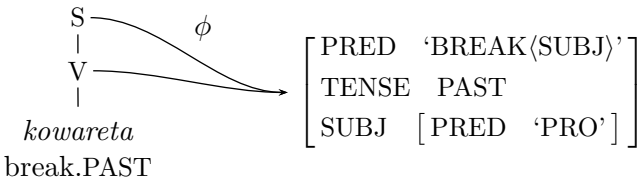
(30)




Further, there can be f-structures that have no corresponding c-structure node (the $\phi$ function is **into**). Example (31) shows the c-structure and f-structure for a sentence of Japanese, a so-called pro-drop language in which the verb

optionally specifies functional information about its subject. When there is no overt subject phrase in the sentence, the information specified by the verb supplies the SUBJ value for the sentence. In (31), since there is no overt subject, all of the information about the subject comes from specifications on the verb, and there is no c-structure node corresponding to the SUBJ f-structure:

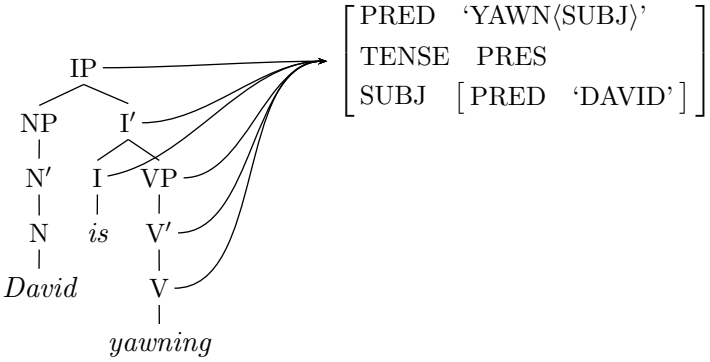
(31) *koware-ta*
break-PAST
'[It/Something] broke.'




### 4.2 C-structure/f-structure correspondences

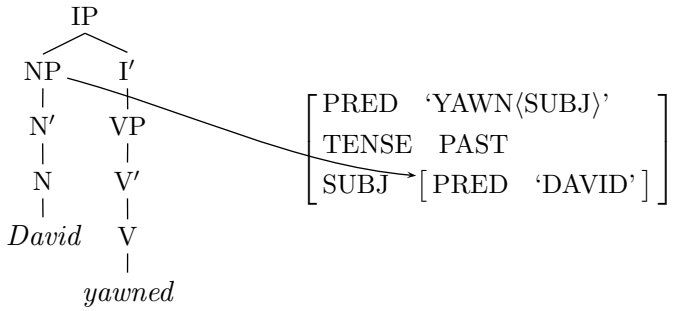
The $\phi$ function is important in stating universally valid relations between c-structure positions and the functional roles associated with them. For example, a phrase and its head always correspond to the same f-structure. Furthermore, the complement of a functional category is an **f-structure co-head**: the functional head and its complement correspond to the same f-structure. This is shown in (32), where the functional category IP, its heads I′ and I, and its complement VP map to the same f-structure.

(32) *David is yawning.*



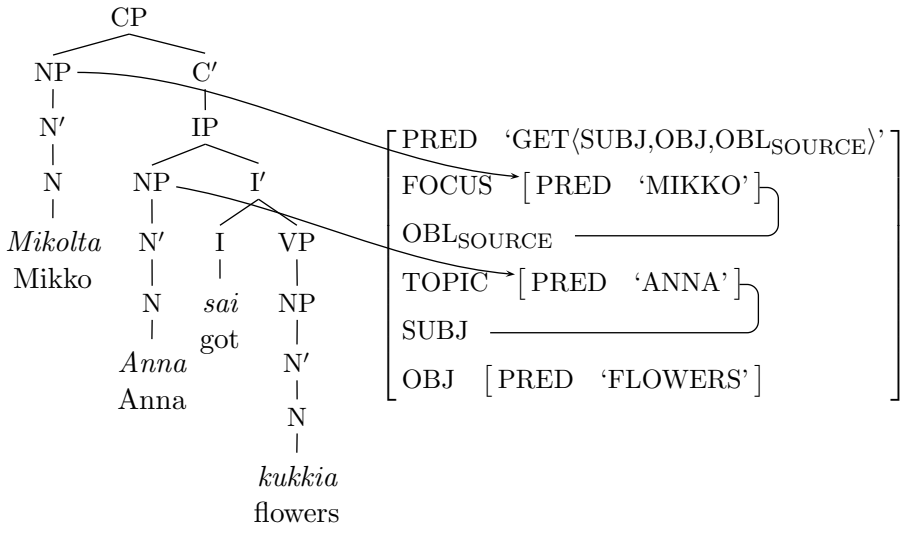

The specifier position of the functional categories IP and CP is filled by a phrase bearing a **grammaticized discourse function**: SUBJ, TOPIC, or FOCUS. Within these limits, languages can differ as the particular grammaticized discourse function allowed in each of these positions. In English, as we have seen, the specifier position of IP is filled by the SUBJ:

(33) *David yawned.*




In Finnish, the specifier of IP is associated with the TOPIC function, and the specifier of CP is associated with FOCUS:

(34) *Mikolta Anna sai kukkia.*
Mikko Anna got flowers
'From Mikko, Anna got flowers.'

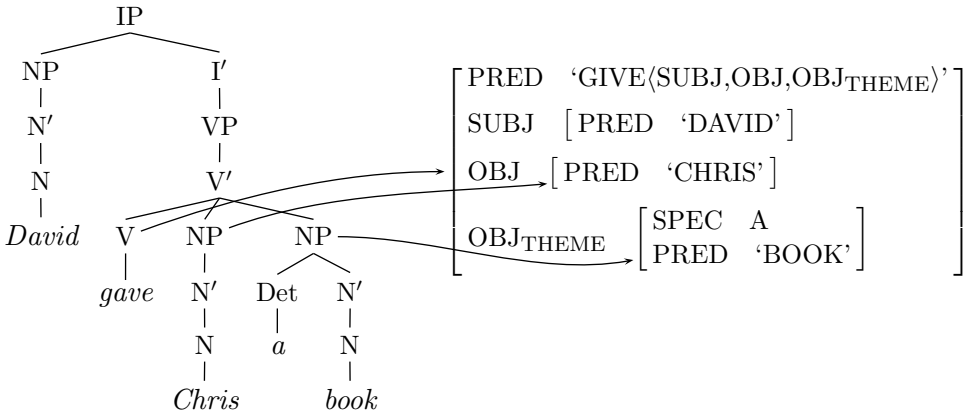



When a f-structure contains a FOCUS or TOPIC function, the **Extended Coherence Condition** requires it to be integrated into the f-structure by either anaphorically or functionally binding another f-structure in the sentence. In a sentence like *Bill, I like him*, the f-structure for *Bill* anaphorically binds the f-structure *him*: the two phrases *Bill* and *him* are syntactically independent, and each phrase has its own f-structure, but the anaphoric relation between the two satisfies the Extended Coherence Condition. In (34), the FOCUS also bears the $\text{OBL}_{\text{SOURCE}}$ function, and the TOPIC is also the SUBJ: these relations involve functional binding, because the same f-structure fills both functions.

The complements of a lexical category bear non-discourse grammatical functions: that is, any grammatical function other than SUBJ, FOCUS, or TOPIC. In (35), the complements of V are associated with the grammatical functions

OBJ and $\text{OBJ}_{\text{THEME}}$:

(35) *David gave Chris a book.*




As mentioned above, the exocentric category S imposes no restrictions on the phrases it dominates: S can dominate constituents of any category, and the principles governing the relation between endocentric categories and functional structure do not hold (Bresnan, 2001). Languages that make use of S tend, then, to exhibit relatively free word order, with morphology rather than phrasal position determining functional role.

### 4.3 Constraining the c-structure/f-structure relation

In describing the relation between c-structure and f-structure, we use the following symbols for the f-structure corresponding to the current node in a phrase structure rule and the f-structure of its mother node:

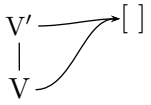
(36) the f-structure of the immediately dominating node: $\uparrow$
the f-structure of the current c-structure node: $\downarrow$

We can use these symbols to annotate the V′ phrase structure rule with f-structure correspondence constraints:

(37) $\text{V}' \longrightarrow$ V
$\uparrow = \downarrow$
mother's f-structure = self's f-structure

This annotated rule licenses the configuration in (38). In the c-structure, the V′ node dominates the V node, as the phrase structure rules require. The V′ and V nodes correspond to the same f-structure, as the annotations on the V node require.

(38)

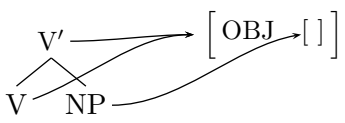


In the rule shown in (39), the V and the V′ node correspond to the same f-structure, as specified by the ↑=↓ annotation on the V node. The annotation on the NP node requires the f-structure ↓ corresponding to the NP to be the value of the OBJ value in the f-structure ↑ for the mother node.

(39) $\mathrm{V}' \longrightarrow \underset{\uparrow = \downarrow}{\mathrm{V}} \quad \underset{(\uparrow \ \mathrm{OBJ}) = \downarrow}{\mathrm{NP}}$

The rule in (39) licenses the following configuration:

(40)

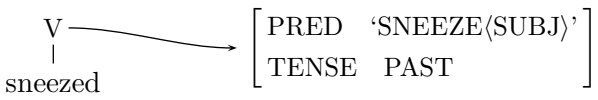



We can use the same formal vocabulary in specifications of lexical entries. The lexical entry for the verb *sneezed* is shown in (41). It specifies that the c-structure category of *sneezed* is V, and also specifies constraints on the f-structure ↑ of the preterminal V node that dominates the terminal node *sneezed*:

(41) *sneezed* V $\quad (\uparrow \ \mathrm{PRED}) = \text{`SNEEZE}\langle \mathrm{SUBJ} \rangle\text{'}$
$\qquad\qquad\quad (\uparrow \ \mathrm{TENSE}) = \mathrm{PAST}$

This lexical entry licenses the c-structure/f-structure configuration in (42).

(42)




In the examples above, expressions such as (↑ OBJ) refer to a particular f-structure, the value of the attribute OBJ in the f-structure ↑. It is also possible to refer to f-structures that are embedded within another f-structure at an arbitrary depth, by the use of **functional uncertainty**, where a regular expression (defined above, in the discussion of phrase structure rules) is used to characterize a path through an f-structure. The rule shown in (43) states that the NP daughter of IP bears the FOCUS role ((↑ FOCUS) = ↓) and also bears some grammatical function GF which may be embedded within any number of COMPs ((↑ COMP* GF) = ↓). Such a rule would play a role in the analysis of examples like (34), where the same phrase fills both the FOCUS and the SUBJ role.

(43) $\mathrm{IP} \longrightarrow \underset{\substack{(\uparrow \ \mathrm{FOCUS}) = \downarrow \\ (\uparrow \ \mathrm{COMP}^* \ \mathrm{GF}) = \downarrow}}{\mathrm{NP}} \quad \underset{\uparrow = \downarrow}{\mathrm{I}'}$

## 5 Syntax and semantics

Several recent research strands in LFG have explored the relation of constituent and functional structure to other linguistic levels. Among these are the theory of the relation between argument structure and syntax, and the "glue" approach to the interface between syntax and semantics.

### 5.1 Mapping theory and argument linking

**Mapping theory** explores correlations between the semantic roles of the arguments of a verb and their syntactic functions: if a language assigns the syntactic function SUBJ to the agent argument of an active verb like *kick*, for example, it invariably assigns SUBJ to the agent argument of semantically similar verbs like *hit*.

Early formulations of the rules of mapping theory proposed rules relating specific thematic roles to specific grammatical functions: for example, that the thematic role of AGENT is always realized as SUBJ. Later work proposed more general rules relating thematic roles to classes of grammatical functions, rather than specific functions. It is most often assumed that grammatical functions are cross-classified with the features $\pm$R and $\pm$O. Several versions of mapping theory have been proposed (Bresnan & Kanerva, 1989; Bresnan & Zaenen, 1990; Bresnan, 2001); in the following, we describe the theory of Bresnan & Zaenen (1990).

The feature $\pm$R distinguishes **unrestricted** ($-$R) grammatical functions from **restricted** ($+$R) functions. The grammatical functions SUBJ and OBJ are classified as unrestricted, meaning that they can be filled by an argument bearing any thematic role. These contrast with restricted grammatical functions like obliques or thematically restricted objects, which must be filled by arguments with particular thematic roles: for example, the $\text{OBL}_{\text{SOURCE}}$ function must be filled by an argument bearing the thematic role SOURCE, and the thematically restricted object function $\text{OBJ}_{\text{THEME}}$ is filled by a THEME argument.

The feature $\pm$O distinguishes **objective** ($+$O) grammatical functions from **nonobjective** ($-$O) functions. The unrestricted OBJ function and the restricted $\text{OBJ}_\theta$ functions are objective, while the SUBJ and the oblique functions are nonobjective.

These features cross-classify the grammatical functions as in (44):

(44)

| | $-$R | $+$R |
|---|---|---|
| $-$O | SUBJ | $\text{OBL}_\theta$ |
| $+$O | OBJ | $\text{OBJ}_\theta$ |

These features are used to state rules of **intrinsic classification** of particular thematic roles. Such rules constrain the relation between thematic roles and the classes of grammatical functions that these features delineate. For example, arguments bearing the AGENT role are classified as intrinsically nonobjective ($-$O), either SUBJ or $\text{OBL}_{\text{AGENT}}$. Arguments bearing the THEME role are disjunctively classified, either as intrinsically unrestricted ($-$R), bearing the SUBJ or OBJ function, or as intrinsically objective ($+$O), filling the OBJ or $\text{OBJ}_{\text{THEME}}$ function.

Besides these intrinsic classifications, **default mapping rules** classify the arguments of a predicate according to their relative position on the thematic hierarchy:

(45) Thematic hierarchy (Bresnan & Kanerva, 1989):

AGENT > BENEFACTIVE > RECIPIENT/EXPERIENCER
> INSTRUMENT > THEME/PATIENT > LOCATIVE

One of the default mapping rules requires the argument of a predicate that is highest on the thematic hierarchy to be classified as unrestricted (−R). For example, if a verb requires an AGENT argument and a PATIENT argument, the AGENT argument thematically outranks the PATIENT argument, and thus the AGENT argument is classified as unrestricted.

For a predicate with an AGENT and a PATIENT argument, like *kick*, this has the following result (Bresnan & Kanerva, 1989):

(46)

| | kick ⟨ AGENT | PATIENT ⟩ |
|---|---|---|
| intrinsic: | [−O] | [−R] |
| defaults: | [−R] | |
| | SUBJ | SUBJ/OBJ |
| Final classification: | SUBJ | OBJ |

For simplicity, we will consider only the intrinsically unrestricted classification of the PATIENT argument, leaving aside the option to consider the PATIENT an intrinsically objective function. The AGENT argument is classified as intrinsically nonobjective. The default rules add the unrestricted classification to the thematically highest argument, the AGENT. Since the AGENT is classified as [−O,−R], it is the SUBJ. The unrestricted classification of the PATIENT argument allows it to bear either the SUBJ or the OBJ role, but since the AGENT is assigned the SUBJ role, the PATIENT must be realized as OBJ. Thus, the argument classification rules, together with well-formedness conditions like the Subject Condition requiring each verbal predicate to have a subject, constrain the mapping between argument roles and grammatical functions.

### 5.2 Glue: The syntax-semantics interface

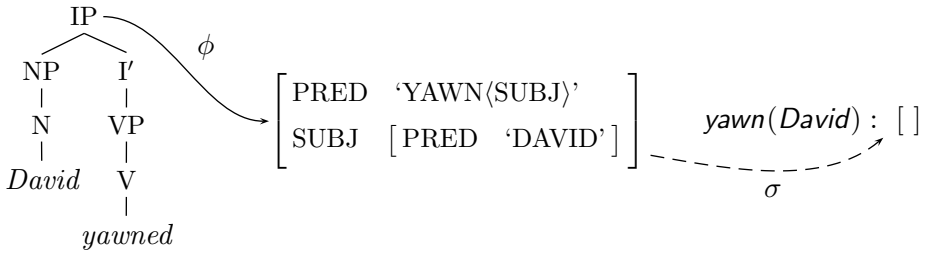
LFG assumes that the syntactic level that is primarily involved in semantic composition is the functional structure. That is, functional relations like SUBJ and OBJ rather than c-structure tree configurations are primarily responsible for determining how the meanings of the parts of a sentence combine to produce the full meaning of the sentence.

The dominant theory of the syntax-semantics interface in LFG is the so-called **glue** approach (Dalrymple, 1999, 2001), a theory of how syntax guides the process of semantic composition. The glue approach assumes that each part of the f-structure corresponds to a semantic resource associated with a meaning, and that the meaning of an f-structure is obtained by assembling the meanings of its parts according to a set of instructions specifying how the semantic resources can combine. These assembly instructions are provided as a set of logical premises in the "glue language" of **linear logic**, and the derivation of a meaning for a sentence corresponds to a logical deduction.

The deduction is performed on the basis of logical premises contributed by the words in the sentence (and possibly by syntactic constructions). Linear logic, a resource-based logic, is used to state requirements on how the meanings of the parts of a sentence can be combined to form the meaning of the sentence as a whole. Linear logic is different from classical logic in that it does not admit rules that allow for premises to be discarded or used more than once in a deduction. Premises in a linear logic deduction are, then, resources that must be accounted for in the course of a deduction; this nicely models the semantic contribution of the words in a sentence, which must contribute exactly once to the meaning of the sentence, and may not be ignored or used more than once. A sentence like *David knocked twice* cannot mean simply *David knocked*: the meaning of *twice* cannot be ignored. It also cannot mean the same thing as *David knocked twice twice*; the meaning of a word in a sentence cannot be used multiple times in forming the meaning of the sentence.

The syntactic structures for the sentence *David yawned*, together with the desired semantic result, are displayed in (47):

(47) *David yawned.*




The semantic structure for the sentence is related to its f-structure by the correspondence function $\sigma$, represented as a dotted line. This result is obtained on the basis of the following lexical information, associated with the verb *yawned*:

(48) $\lambda X.\mathit{yawn}(X) : (\uparrow \text{SUBJ})_\sigma \multimap \uparrow_\sigma$

This formula is called a **meaning constructor**. It pairs the meaning for *yawned*, the one-place predicate $\lambda X.\mathit{yawn}(X)$, with the linear logic formula $(\uparrow \text{SUBJ})_\sigma \multimap \uparrow_\sigma$. In this formula, the connective $\multimap$ is the *linear implication* symbol of linear logic. This symbol expresses a meaning similar to *if. . . then. . .*: in this case, stating that *if* a semantic resource $(\uparrow \text{SUBJ})_\sigma$ representing the meaning of the subject is available, *then* a semantic resource $\uparrow_\sigma$ representing the meaning of the sentence can be produced. Unlike the implication operator of classical logic, the linear implication operator $\multimap$ carries with it a requirement for *consumption* and *production* of semantic resources: the formula $(\uparrow \text{SUBJ})_\sigma \multimap \uparrow_\sigma$ indicates that if a semantic resource $(\uparrow \text{SUBJ})_\sigma$ is found, it is *consumed* and the semantic resource $\uparrow_\sigma$ is *produced*.

We also assume that a name like *David* contributes a semantic resource, its semantic structure. In an example like *David yawned*, this resource is consumed by the verb *yawned*, which requires a resource for its SUBJ to produce a resource for the sentence. This accords with the intuition that the verb in a sentence

must obtain a meaning for its arguments in order for a meaning for the sentence to be available.

The f-structure for the sentence *David yawned*, together with the instantiated meaning constructors contributed by *David* and *yawned*, is given in (49):

(49)
$$y : \begin{bmatrix} \text{PRED} & \text{`YAWN}\langle\text{SUBJ}\rangle\text{'} \\ \text{SUBJ} & d : \begin{bmatrix} \text{PRED} & \text{`DAVID'} \end{bmatrix} \end{bmatrix}$$

**[David]** $\quad \mathit{David} \;:\; d_\sigma$

**[yawn]** $\quad \lambda X.\mathit{yawn}(X) \;:\; d_\sigma \multimap y_\sigma$

The left-hand side of the meaning constructor labeled **[David]** is the proper noun meaning *David*, and the left-hand side of the meaning constructor labeled **[yawn]** is the meaning of the intransitive verb *yawned*, the one-place predicate $\lambda X.\mathit{yawn}(X)$.

We must also provide rules for how the right-hand (glue) side of each of the meaning constructors in (49) relates to the left-hand (meaning) side in a meaning deduction. For simple, nonimplicational meaning constructors like **[David]** in (49), the meaning on the left-hand side is the meaning of the semantic structure on the right-hand side. For meaning constructors which contain the linear implication operator $\multimap$, like **[yawn]**, *modus ponens* on the glue side corresponds to function application on the meaning side:

(50)
$$\frac{\begin{array}{rcl} X & : & f_\sigma \\ P & : & f_\sigma \multimap g_\sigma \end{array}}{P(X) \;:\; g_\sigma}$$

With these correspondences between linear logic formulas and meanings, we perform the following series of reasoning steps:

(51)

| | |
|---|---|
| $\mathit{David} : d_\sigma$ | The meaning *David* is associated with the SUBJ semantic structure $d_\sigma$. |
| $\lambda X.\mathit{yawn}(X) : d_\sigma \multimap y_\sigma$ | On the glue side, if we find a semantic resource for the SUBJ $d_\sigma$, we consume that resource and produce a semantic resource for the full sentence $y_\sigma$. On the meaning side, we apply the function $\lambda X.\mathit{yawn}(X)$ to the meaning associated with $d_\sigma$. |
| $\mathit{yawn}(\mathit{David}) : y_\sigma$ | We have produced a semantic structure for the full sentence $y_\sigma$, associated with the meaning $\mathit{yawn}(\mathit{David})$. |

By using the function application rule and the meaning constructors for *David* and *yawned*, we deduce the meaning $\mathit{yawn}(\mathit{David})$ for the sentence *David yawned*, as desired.

Glue analyses of quantification, intensional verbs, modification, coordination, and other phenomena have been explored (Dalrymple, 1999). A particular challenge for the glue approach is found in cases where there are apparently too many or too few meaning resources to produce the correct meaning for a sentence; such cases are explored within the glue framework by Asudeh (2004).

## 6 Preferences and parsing

From its inception, work on LFG has been informed by computational and psycholinguistic concerns. Recent research has combined LFG's syntactic assumptions with an optimality-theoretic approach in an exploration of OT-LFG (see **Optimality Theory**, **OT-LFG**). Other work combines LFG with Data-Oriented Parsing, a new view of language processing and acquisition. There have also been significant developments in parsing and generating with LFG grammars and grammars in related formalisms.

### 6.1 Data-oriented parsing and LFG

The framework of *Data-Oriented Parsing* or *DOP*, developed primarily by Rens Bod and his colleagues, represents a new view of the productivity of language and how it can be acquired on the basis of a finite amount of data. DOP views language acquisition as the analysis of a pool of linguistic structures that are presented to the language learner. The learner breaks up these structures into all of their component pieces, from the largest pieces to the smallest units, and new utterances are assembled from these pieces. The likelihood of assigning a particular analysis to a new sentence depends on the frequency of occurrence of its component parts, both large and small, in the original pool of structures.

LFG-DOP (Bod & Kaplan, 1998) specializes the general DOP theory to LFG assumptions about linguistic structures and the relations between them. LFG-DOP assumes that the body of linguistic evidence that a language learner is presented with consists of well-formed c-structure/f-structure pairs. On this view, language acquisition consists in determining the relevant component parts of these structures and then combining these parts to produce new c-structure/f-structure pairs for novel sentences.

### 6.2 Parsing

Several breakthroughs have been made in the parsing of large computational LFG grammars. Maxwell & Kaplan (1991) examine the problem of processing disjunctive specifications of constraints, which are computationally very difficult to process: in the worst case, processing disjunctive constraints is *exponentially difficult*. However, this worst-case scenario assumes that every disjunctive constraint can interact significantly with every other constraint. In practice, such interactions are found only very rarely: an ambiguity in the syntactic properties of the SUBJ of a sentence rarely correlates with ambiguities in the OBJ or other arguments. This insight is the basis of Maxwell & Kaplan's algorithm, which works by turning a set of disjunctively specified constraints into a set of

*contexted*, conjunctively specified constraints, where the context of a constraint indicates where the constraint is relevant. Solving these contexted constraints turns out to be very efficient for linguistically motivated sets of constraints, where only local interactions among disjunctions tend to occur.

Maxwell & Kaplan (1993, 1996) explore the issue of c-structure processing and its relation to solving f-structural constraints. It has long been known that constituent structure parsing — determining the phrase structure trees for a given sentence — is very fast in comparison to solving the equations that determine the f-structure for the sentence. For this reason, an important task in designing algorithms for linguistic processing of different kinds of structures like the c-structure and the f-structure is to optimize the interactions between these computationally very different tasks. Previous research often assumed that the most efficient approach would be to interleave the construction of the phrase structure tree with the solution of f-structure constraints. Maxwell & Kaplan (1993) explore and compare a number of different methods for combining phrase structure processing with constraint solving; they show that in certain situations, interleaving the two processes can actually give very bad results. Subsequently, Maxwell & Kaplan (1996) showed that if phrase structure parsing and f-structural constraint solving are combined in the right way, parsing can be very fast; in fact, if the grammar that results from combining phrase structure and functional constraints happens to be *context-free equivalent*, the algorithm for computing the c-structure and f-structure operates in *cubic time*, the same as for pure phrase structure parsing.

### 6.3 Generation

Generation is the inverse of parsing: whereas the parsing problem is to determine the c-structure and f-structure that correspond to a particular sentence, work on generation in LFG assumes that the generation task is to determine which sentences correspond to a specified f-structure, given a particular grammar. Based on these assumptions, several interesting theoretical results have been attained. Of particular inportance is the work of Kaplan & Wedekind (2000), who show that if we are given an LFG grammar and an *acyclic* f-structure (that is, an f-structure that does not contain a reference to another f-structure that contains it), the set of strings that corresponds to that f-structure according to the grammar is a *context-free language*. Kaplan & Wedekind also provide a method for constructing the context-free grammar for that set of strings by a process of specialization of the full grammar that we are given. This result leads to a new way of thinking about generation, opens the way to new, more efficient generation algorithms, and clarifies a number of formal and mathematical issues relating to LFG parsing and generation.

Wedekind & Kaplan (1996) explore issues in *ambiguity-preserving generation*, where a set of f-structures rather than a single f-structure is considered, and the sentences of interest are those that correspond to **all** of the f-structures under consideration. The potential practical advantages of ambiguity-preserving generation are clear: consider, for example, a scenario involving translation from English to German. We first parse the input English sentence, producing sev-

eral f-structures if the English sentence is ambiguous. For instance, the English sentence *Hans saw the man with the telescope* is ambiguous: it means either that the man had the telescope or that Hans used the telescope to see the man. The best translation for this sentence would be a German sentence that is ambiguous in exactly the same way as the English sentence, if such a German sentence exists. In the case at hand, we would like to produce the German sentence *Hans sah den Mann mit dem Fernrohr*, which has exactly the same two meanings as the English input. To do this, we map the English f-structures for the input sentence to the set of corresponding German f-structures; our goal is then to generate the German sentence *Hans sah den Mann mit dem Fernrohr*, which corresponds to each of these f-structures. This approach is linguistically appealing, but mathematically potentially problematic: Wedekind & Kaplan (1996) show that determining whether there is a single sentence that corresponds to each member of a set of f-structures is in general *undecidable* for an arbitrary (possibly linguistically unreasonable) LFG grammar. This means that there are grammars that can be written within the formal parameters of LFG, though these grammars may not encode the properties of any actual or potential human language, and for these grammars, there are sets of f-structures for which it is impossible to determine whether there is any sentence that corresponds to those f-structures. This result is important in understanding the formal limits of ambiguity-preserving generation.

## Bibliography


Alsina, A. (1993). *Predicate Composition: A Theory of Syntactic Function Alternations*. Ph.D. thesis, Stanford University.

Andrews, A., III & Manning, C. D. (1999). *Complex Predicates and Information Spreading in LFG*. Stanford, CA: CSLI Publications.

Asudeh, A. (2004). *Resumption as Resource Management*. Ph.D. thesis, Stanford University.

Bod, R. & Kaplan, R. M. (1998). 'A probabilistic corpus-driven model for Lexical-Functional analysis.' In 'Proceedings of COLING/ACL98: Joint Meeting of the 36th Annual Meeting of the ACL and the 17th International Conference on Computational Linguistics, Montréal,' Association for Computational Linguistics.

Bresnan, J. (1978). 'A realistic transformational grammar.' In Halle, M., Bresnan, J. & Miller, G. A. (eds.) 'Linguistic Theory and Psychological Reality,' Cambridge, MA: The MIT Press.

Bresnan, J. (ed.) (1982). *The Mental Representation of Grammatical Relations*. Cambridge, MA: The MIT Press.

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

Bresnan, J. & Kanerva, J. M. (1989). 'Locative inversion in Chichewa: A case study of factorization in grammar.' *Linguistic Inquiry 20*(1), 1–50. Reprinted in Stowell et al. (1992).

Bresnan, J. & Zaenen, A. (1990). 'Deep unaccusativity in LFG.' In Dziwirek, K., Farrell, P. & Mejías-Bikandi, E. (eds.) 'Grammatical Relations: A Cross-Theoretical Perspective,' Stanford, CA: CSLI Publications, 45–57.

Butt, M. (1996). *The Structure of Complex Predicates in Urdu*. Dissertations in Linguistics. Stanford, CA: CSLI Publications. Revised and corrected version of 1993 Stanford University dissertation.

Dalrymple, M. (1993). *The Syntax of Anaphoric Binding*. CSLI Lecture Notes, number 36. Stanford, CA: CSLI Publications.

Dalrymple, M. (ed.) (1999). *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. Cambridge, MA: The MIT Press.

Dalrymple, M. (2001). *Lexical Functional Grammar*, volume 34 of *Syntax and Semantics*. New York, NY: Academic Press.

Dalrymple, M. & Kaplan, R. M. (2000). 'Feature indeterminacy and feature resolution.' *Language 76*(4), 759–798.

Dalrymple, M., Kaplan, R. M., Maxwell, J. T., III & Zaenen, A. (eds.) (1995). *Formal Issues in Lexical-Functional Grammar*. Stanford, CA: CSLI Publications.

Falk, Y. N. (2001). *Lexical-Functional Grammar: An Introduction to Parallel Constraint-Based Syntax*. Stanford, CA: CSLI Publications.

Kaplan, R. M. & Wedekind, J. (2000). 'LFG generation produces context-free languages.' In 'Proceedings of the 18th International Conference on Computational Linguistics (COLING2000), Saarbrücken,' .

King, T. H. (1995). *Configuring Topic and Focus in Russian*. Dissertations in Linguistics. Stanford, CA: CSLI Publications. Revised and corrected version of 1993 Stanford University dissertation.

Kroeger, P. (1993). *Phrase Structure and Grammatical Relations in Tagalog*. Dissertations in Linguistics. Stanford, CA: CSLI Publications. Revised and corrected version of 1991 Stanford University dissertation.

Kroeger, P. (2004). *Analyzing Syntax : A Lexical-Functional Approach*. Cambridge University Press.

Levin, L. S. (1986). *Operations on Lexical Forms: Unaccusative Rules in Germanic Languages*. Ph.D. thesis, MIT.

Levin, L. S., Rappaport, M. & Zaenen, A. (eds.) (1983). *Papers in Lexical-Functional Grammar*. Bloomington: Indiana University Linguistics Club.

Manning, C. D. (1996). *Ergativity: Argument Structure and Grammatical Relations*. Dissertations in Linguistics. Stanford, CA: CSLI Publications. Revised and corrected version of 1994 Stanford University dissertation.

Maxwell, J. T., III & Kaplan, R. M. (1991). 'A method for disjunctive constraint satisfaction.' In Tomita, M. (ed.) 'Current Issues in Parsing Technology,' Kluwer Academic Publishers, 173–190. Reprinted in Dalrymple et al. (1995, pp. 381–401).

Maxwell, J. T., III & Kaplan, R. M. (1993). 'The interface between phrasal and functional constraints.' *Computational Linguistics 19*(4), 571–590.

Maxwell, J. T., III & Kaplan, R. M. (1996). 'An efficient parser for LFG.' In Butt, M. & King, T. H. (eds.) 'On-line Proceedings of the LFG96 Conference,' URL `http://csli-publications.stanford.edu/LFG/1/lfg1.html`.

Mohanan, T. (1994). *Arguments in Hindi*. Dissertations in Linguistics. Stanford, CA: CSLI Publications. Reprinted version of 1990 Stanford University dissertation.

Nordlinger, R. (1998). *Constructive Case: Evidence from Australian Languages*. Dissertations in Linguistics. Stanford, CA: CSLI Publications. Revised version of 1997 Stanford University dissertation.

Sells, P. (2001). *Structure, Alignment and Optimality in Swedish*. Stanford, CA: CSLI Publications.

Simpson, J. (1991). *Warlpiri Morpho-Syntax: A Lexicalist Approach*. Dordrecht: Kluwer Academic Publishers.

Stowell, T., Wehrli, E. & Anderson, S. R. (eds.) (1992). *Syntax and Semantics: Syntax and the Lexicon*, volume 26. San Diego: Academic Press.

Toivonen, I. (2003). *Non-Projecting Words: A Case Study of Swedish Particles*. Dordrecht: Kluwer Academic Publishers.

Wedekind, J. & Kaplan, R. M. (1996). 'Ambiguity-preserving generation with LFG- and PATR-style grammars.' *Computational Linguistics 22*(4), 555–568.

## Keywords

Constituent structure, c-structure, functional structure, f-structure, open/closed grammatical function, annotated phrase structure rule, completeness, coherence, mapping theory, LFG-DOP, OT-LFG, Glue semantics.

## Cross-references

See also: Constituent structure; Declarative models of syntax; Grammatical relations; OT-LFG; Syntactic features and feature structure; Unification: Classical and default; X-bar Theory.

## Biography

Mary Dalrymple received a PhD in Linguistics from Stanford University in 1990, and was a member of the research staff at Xerox PARC (currently the Palo Alto Research Center) in Palo Alto, California and a consulting faculty member in the Department of Linguistics and the Symbolic Systems Program at Stanford University until 2002. In 2003 and 2004 she was a member of the Department of Computer Science at King's College London. Currently, she is University Lecturer in General Linguistics at Oxford University.