Introduction to Lexical Functional Grammar

Mary Dalrymple, John Lowe, & Louise Mycock
Centre for Linguistics and Philology
Oxford University
Konstanz, November/December 2012
“Semantic roles, syntactic constituents, and grammatical functions belong to parallel information structures of very different formal character. They are related not by proof-theoretic derivation but by structural correspondences, as a melody is related to the words of a song. The song is decomposable into parallel melodic and linguistic structures, which jointly constrain the nature of the whole. In the same way, the sentences of human language are themselves decomposable into parallel systems of constraints – structural, functional, semantic, and prosodic – which the whole must jointly satisfy.” (Bresnan, 1990)
Two aspects of syntactic structure:

- **Functional structure** is the abstract functional syntactic organisation 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 organisation of words into phrases.
LFG’s c-structure and f-structure
Functional structure

\[
\begin{array}{c}
\text{PRED} \quad \langle \text{GO} \langle \text{SUBJ} \rangle \rangle \\
\text{TENSE} \quad \text{PAST} \\
\text{SUBJ} \quad \begin{array}{c}
\text{PRED} \quad \langle \text{DAVID} \rangle \\
\text{NUM} \quad \text{SG}
\end{array}
\end{array}
\]

- PRED, TENSE NUM: attributes
- ‘GO⟨SUBJ⟩’, DAVID, SG: values
- PAST, SG: symbols (a kind of value)
- ‘BOY’, ‘GO⟨SUBJ⟩’: semantic forms
F-structures

An f-structure can be the value of an attribute. Attributes with f-structure values are the grammatical functions: SUBJ, OBJ, OBJ\(_\theta\), COMP, XCOMP, ...
A set of f-structures can also be a value of an attribute.
Sets of f-structures

Sets of f-structures represent:

- adjuncts (there can be more than one adjunct) or
- coordinate structures (there can be more than one conjunct)
C- and F-Structure

φ function relates c-structure nodes to f-structures.

(Function: Every c-structure node corresponds to exactly one f-structure.)
Constraining the c-structure/f-structure correspondence

\[ \begin{array}{c}
V' \\
\downarrow \\
V \\
\downarrow \\
\text{yawned}
\end{array} \xrightarrow{\phi}
\left[ \begin{array}{c}
PRED \ 'YAWN\langle SUBJ\rangle' \\
\text{TENSE} \ PAST
\end{array} \right]

V' \rightarrow V
Local F-Structure Reference

\[ \begin{array}{c}
V' \\
| \\
V
\end{array} \quad \phi
\]

\[ \begin{array}{c}
PRED \quad \langle \text{YAWN} \langle \text{SUBJ} \rangle \rangle' \\
\text{TENSE} \quad \text{PAST}
\end{array} \]

\text{yawned}

V' \quad V

the current c-structure node ("self"): \ast
the immediately dominating node ("mother"): \hat{\ast}
the c-structure to f-structure function: \phi
Rule Annotation

\[
\begin{align*}
V' & \xrightarrow{\phi} \phi(\hat{\ast}) = \phi(\ast) \\
\text{mother's (V''s) f-structure} & = \text{self's (V's) f-structure}
\end{align*}
\]
Simplifying the Notation

$$\phi(\hat{*}) \quad \text{(mother’s f-structure)} \quad = \uparrow$$

$$\phi(*) \quad \text{(self’s f-structure)} \quad = \downarrow$$

$\text{yawned}$

$V' \xrightarrow{\phi} \left[ \begin{array}{c} \text{PRED ‘YAWN(SUBJ)’} \\ \text{TENSE PAST} \end{array} \right]$
Using the Notation

\[ V' \quad \rightarrow \quad V \]

\[ \uparrow = \downarrow \]

mother’s f-structure = self’s f-structure
Using the Notation

\[ V' \rightarrow V \]

\[ \uparrow = \downarrow \]

mother's f-structure = self's f-structure
Using the Notation

\[ V' \rightarrow V \]

\[ \uparrow = \downarrow \]

mother’s f-structure = self’s f-structure
Using the Notation

\[ V' \rightarrow V \]

\[ \uparrow = \downarrow \]

mother’s f-structure = self’s f-structure

\[ V' \rightarrow [ ] \]

\[ \uparrow = \downarrow \]
More rules

\[
V' \rightarrow V, \quad NP \\
\phi(\hat{*}) = \phi(*) \quad (\phi(\hat{*}) \text{ OBJ}) = \phi(*) \\
mother's f-structure's OBJ = self's f-structure
\]

In simpler form:

\[
V' \rightarrow V, \quad NP \\
\uparrow = \downarrow \quad (\uparrow \text{ OBJ}) = \downarrow
\]
Using the Notation

\[ V' \rightarrow V \quad NP \]
\[ \uparrow = \downarrow \quad (\uparrow \text{OBJ}) = \downarrow \]

Diagram:

```
V' → [OBJ → []]
```

\[ V \rightarrow NP \]
Terminal nodes

\[
\begin{array}{c}
\text{V} \\
\text{yawned}
\end{array}
\rightarrow
\begin{bmatrix}
PRED \ 'YAWN\langle\text{SUBJ}\rangle' \\
\text{TENSE} \ PAST
\end{bmatrix}
\]

Expressible as:

\[
\begin{align*}
V \rightarrow & \quad \text{yawned} \\
(\uparrow \text{PRED}) & = \ 'YAWN\langle\text{SUBJ}\rangle' \\
(\uparrow \text{TENSE}) & = \text{PAST}
\end{align*}
\]

Standard form:

\[
\begin{align*}
\text{yawned} \quad V \quad (\uparrow \text{PRED}) & = \ 'YAWN\langle\text{SUBJ}\rangle' \\
(\uparrow \text{TENSE}) & = \text{PAST}
\end{align*}
\]
Phrase structure rules: English

\[
\begin{align*}
IP & \rightarrow \left( \begin{array}{c}
NP \\
(\uparrow \text{SUBJ}) = \downarrow \\
\end{array} \right) \left( \begin{array}{c}
I' \\
(\uparrow = \downarrow) \\
\end{array} \right) \\
I' & \rightarrow \left( \begin{array}{c}
I \\
(\uparrow = \downarrow) \\
\end{array} \right) \left( \begin{array}{c}
VP \\
(\uparrow = \downarrow) \\
\end{array} \right) \\
VP & \rightarrow \left( \begin{array}{c}
V \\
(\uparrow = \downarrow) \\
\end{array} \right) \\
NP & \rightarrow \left( \begin{array}{c}
N \\
(\uparrow = \downarrow) \\
\end{array} \right)
\end{align*}
\]
Lexical entries: English

yawned  V  \(\uparrow\text{PRED}\) = ‘YAWN⟨SUBJ⟩’
\(\uparrow\text{TENSE}\) = PAST

David  N  \(\uparrow\text{PRED}\) = ‘DAVID’

(Standard LFG practice: include only features relevant for analysis under discussion.)
Analysis: English

```
<table>
<thead>
<tr>
<th></th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NP</td>
</tr>
<tr>
<td>(↑ SUBJ) = ↓</td>
<td>↑ = ↓</td>
</tr>
<tr>
<td></td>
<td>N</td>
</tr>
<tr>
<td>↑ = ↓</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td>David</td>
</tr>
<tr>
<td></td>
<td>(↑ PRED) = ‘DAVID’</td>
</tr>
<tr>
<td></td>
<td>↑ = ↓</td>
</tr>
<tr>
<td></td>
<td>yawned</td>
</tr>
<tr>
<td></td>
<td>(↑ PRED) = ‘YAWN(SUBJ)’</td>
</tr>
<tr>
<td></td>
<td>(↑ TENSE) = PAST</td>
</tr>
</tbody>
</table>
```
Analysis: English

```
IP
  NP  I'
    (↑ SUBJ) = ↓  ↑ = ↓
    |          |
    |          |
    N  VP
    ↑ = ↓  ↑ = ↓
    |          |
    |          |
    David  V
    (f_n PRED) = 'DAVID'  ↑ = ↓
    |          |
    |          |
    yawned
    (↑ PRED) = 'YAWN⟨SUBJ⟩'
    (↑ TENSE) = PAST
```
Analysis: English

![Syntax Tree Diagram]

- **NP**: (↑ SUBJ) = ↓
  - **N**: David
    - **f_{np} = f_n**
  - **VP**
    - **V**: yawned
      - (↑ PRED) = ‘YAWN⟨SUBJ⟩’
      - (↑ TENSE) = PAST
Analysis: English

$$\begin{align*}
\text{IP} & \\
\text{NP} & \quad \text{I'} \\
(f_{ip} \text{ SUBJ}) = f_{np} & \quad \uparrow = \downarrow \quad \\ & \quad \uparrow = \downarrow \\
\text{N} & \quad \text{VP} \\
f_{np} = f_n & \\
\text{David} & \quad \text{V} \\
(f_n \text{ PRED}) = 'DAVID' & \quad \uparrow = \downarrow \\
yawned & \\
(\uparrow \text{ PRED}) = 'YAWN\langle\text{SUBJ}\rangle' \\
(\uparrow \text{ TENSE}) = \text{PAST}
\end{align*}$$
Analysis: English

\[
\begin{align*}
\text{IP} & \quad \text{NP} \quad \text{I'} \\
(f_{ip} \text{ SUBJ}) &= f_{np} & \uparrow &= \downarrow \\
| & & | \\
N & \quad \text{VP} \\
f_{np} &= f_n & \uparrow &= \downarrow \\
| & & | \\
\text{David} & \quad \text{V} \\
(f_n \text{ PRED}) &= \text{‘DAVID’} & \uparrow &= \downarrow \\
| & & | \\
\text{yawned} & \quad \text{V} \\
(f_v \text{ PRED}) &= \text{‘YAWN(SUBJ)’} \\
(f_v \text{ TENSE}) &= \text{PAST}
\end{align*}
\]
Analysis: English

\[
\text{IP} \\
| \text{NP} \quad \text{I'} \\
| \quad | \\
| \quad | \\
| \quad | \\
| \quad | \\
\text{David} \quad \text{VP} \\
| \quad | \\
| \quad | \\
\text{yawned} \\
\]

\[
(f_{ip} \text{ SUBJ}) = f_{np} \\
\uparrow = \downarrow \\
(f_{np} = f_n) \\
\uparrow = \downarrow \\
(f_n \text{ PRED}) = \text{‘DAVID’} \\
f_{vp} = f_v \\
(f_v \text{ TENSE}) = \text{PAST}
\]
(f_{ip} \text{ SUBJ}) = f_{np} \\
\uparrow = \downarrow \\
\text{NP} \\
N \\
(f_n \text{ PRED}) = \text{‘DAVID’} \\
V \\
(f_v \text{ PRED}) = \text{‘YAWN(SUBJ)’} \\
(f_v \text{ TENSE}) = \text{PAST}

(f_{vp} \text{ PRED}) = f_v \\
(f_v \text{ PRED}) = f_{vp} \\
(f_{vp} \text{ PRED}) = f_{vp}

\text{I’} \\
\text{VP} \\
\text{f} \text{ vp} = f_v \\
\text{f} \text{ vp} = f_{vp} \\
\text{f} \text{ vp} = f_{vp}
Analysis: English

```
[IP]
  /\   \\
 [NP]  [I']
  |    |
 [N]   [VP]
  |      |
  [David]
  |      |
  [yawned]

(f_Ip SUBJ) = f_np   f_ip = f_i'
(f_np PRED) = ‘DAVID’ f_vp = f_v
(f_v PRED) = ‘YAWN(SUBJ)’
(f_v TENSE) = PAST
```
Solving the Description

\[(f_{ip} \text{ SUBJ}) = f_{np}\]
\[f_{np} = f_n\]
\[(f_n \text{ PRED}) = 'DAVID'\]
\[f_{ip} = f_i'\]
\[f_i' = f_{vp}\]
\[f_{vp} = f_v\]
\[(f_v \text{ PRED}) = 'YAWN\langle \text{SUBJ} \rangle'\]
\[(f_v \text{ TENSE}) = \text{PAST}\]
Meaning: \( \text{marry}(\text{john}, \text{rosa}) \)

How is this meaning composed?
Glue: Composing meanings via deduction


- Logic of meanings (semantic level): the level of meanings of utterances and phrases
- Logic for composing meanings (‘glue’ level): the level responsible for assembling the meanings of parts to get the meaning of the whole
Meanings are expressions like $David$, $yawn(David)$, $yawn$ ...

Function: when applied to an argument, yields a unique value.

$yawn$: applied to $David$, yields “true”.
applied to $Fred$, yields “true”.
applied to $George$, yields “false”.
...

Function application:

$yawn$ applied to $David = yawn(David)$
Lambda abstraction:

\( \lambda X.P \) represents a function from entities represented by \( X \) to entities represented by \( P \).

Usually, the expression \( P \) contains at least one occurrence of the variable \( X \), and we say that these occurrences are *bound* by the \( \lambda \) lambda operator.

Function application:

\[ [\lambda X.P](a) \]

The function \( \lambda X.P \) is applied to the argument \( a \).

Equivalent to the expression that results from replacing all occurrences of \( X \) in \( P \) with \( a \).

Example: \( [\lambda X.yawn(X)](David) \equiv yawn(David) \)
Meaning contributions

```
[PRED 'MARRY<SUBJ,OBJ>']

[SUBJ
  [PRED 'JOHN']
]

[OBJ
  [PRED 'ROSA']
]
```

- The word *John* contributes the meaning *john*.
- The word *Rosa* contributes the meaning *rosa*.
- The word *married* contributes meaning assembly instructions of the following form: When given a meaning $x$ for my subject and a meaning $y$ for my object, I produce a meaning $\text{marry}(x, y)$ for my sentence.
Contribution of ‘John’

- Every f-structure has a corresponding semantic structure, related to it by the projection function $\sigma$ (represented by a dotted line).
- $john:[ ]$ is a meaning constructor.
- In the lexicon: $john: \uparrow_{\sigma}$
Meaning assembly: Gluing meanings together

\[\begin{bmatrix}
\text{PRED} & \text{‘MARRY(SUBJ,OBJ)’} \\
\text{SUBJ} & [ ] \\
\text{OBJ} & [ ]
\end{bmatrix}\]

\[\lambda y.\lambda x.\text{marry}(x, y) : o_\sigma \circ (s_\sigma \circ m_\sigma)\]

\[\lambda y.\lambda x.\text{marry}(x, y) : a \text{ relation between two individuals } x \text{ and } y \text{ that holds if } x \text{ marries } y\]

\[o_\sigma \circ (s_\sigma \circ m_\sigma) : \text{If I am provided with the semantic structure of my object and then the semantic structure of my subject, I produce the semantic structure of the sentence.}\]

In the lexicon: \[\lambda y.\lambda x.\text{marry}(x, y) : (\uparrow \text{ OBJ})_\sigma \circ ((\uparrow \text{ SUBJ})_\sigma \circ \uparrow_\sigma)\]
Proof rules

\[
\begin{align*}
 X & : f_\sigma \\
 P & : f_\sigma \circ g_\sigma \\
 \hline \\
 P(X) & : g_\sigma
\end{align*}
\]
Meaning proof for ‘married Rosa’

\[
\begin{align*}
X & : f_\sigma \\
P & : f_\sigma \circ g_\sigma \\
P(X) & : g_\sigma
\end{align*}
\]

\[
\begin{align*}
\text{rosa:} o_\sigma & \quad \text{\(\lambda y.\lambda x.\text{marry}(x, y):o_\sigma \circ (s_\sigma \circ m_\sigma)\)} \\
\text{\(\lambda x.\text{marry}(x, \text{rosa}):s_\sigma \circ m_\sigma\)}
\end{align*}
\]
Meaning proof for ‘John married Rosa’

\[ X : f_\sigma \]
\[ P : f_\sigma \circ g_\sigma \]
\[ P(X) : g_\sigma \]

\[
\begin{align*}
\text{ro} & : o_\sigma \quad \lambda y. \lambda x. \text{marry}(x, y) : o_\sigma \circ (s_\sigma \circ m_\sigma) \\
\lambda x. \text{marry}(x, \text{ro}) : s_\sigma \circ m_\sigma \quad \text{john} : s_\sigma \\
\text{marry(john, ro)} : m_\sigma
\end{align*}
\]
Our theory of meaning composition, to be explained in the following:

\[
\begin{align*}
\text{rosa}: & \quad o_\sigma \\
\lambda y. \lambda x. \text{marry}(x, y): & \quad o_\sigma \circ (s_\sigma \circ m_\sigma) \\
\lambda x. \text{marry}(x, \text{rosa}): & \quad s_\sigma \circ m_\sigma \\
\text{marry}(\text{john, rosa}): & \quad m_\sigma
\end{align*}
\]

In fact, however, the details won’t be important for our discussion of information structure. We can use abbreviations like the following, where \textbf{Rosa} stands for any reasonable theory of the meaning of Rosa and how it combines with the rest of the sentence:

\[
\begin{align*}
\text{Rosa} & \quad \text{married} \\
\text{married-Rosa} & \quad \text{John} \\
\text{marry}(\text{john, rosa}): & \quad m_\sigma
\end{align*}
\]


Bresnan, Joan. 1990. Parallel constraint grammar project. CSLI Calendar, 4 October 1990, volume 6:3.
