Lexical-Functional Grammar (LFG)

Linguistics 614

Based primarily on Dalrymple (2006) and Kaplan and Bresnan (1995)

Spring 2015
Motivation for LFG

- *Lexical* = (not transformational) richly structured lexicon, where relations between, e.g., verbal alternations, are stated
- *Functional* = (not configurational) abstract grammatical functions like subject and object are primitives, i.e., not defined by the phrase structure configurations
LFG in a nutshell

LFG (minimally) distinguishes two kinds of representation:

- **c-structure** (constituent structure):
  overt linear and hierarchical organization of words into phrases

- **f-structure** (functional structure):
  abstract functional organization of the sentence, explicitly representing syntactic predicate-argument structure and functional relations

These are two separate levels of representation and formalisms: trees (c-structure) and attribute-value matrices (f-structure).

- Use of both structures allows, e.g., information from different parts of the tree to connect to the same function
- A range of other levels have been proposed, e.g., A-structure and \( \sigma \)-structure.
Part I: C-structure

C-structure corresponds to a fairly traditional notion of phrase structure.

- X-Bar Theory: heads with complements, adjuncts, specifier
- Categories: lexical (N, P, V, A, Adv) and functional (I, C) categories—not universally fixed

Slightly different notions:

- Endocentric category S: has no lexical head (for “internal subject” languages)
- Optionality: all constituent structure positions can be considered optional
Example of c-structure

(1) kogda rodilsja Lermontov?
when born Lermontov
‘When was Lermontov born?’
Example c-structure
C-structure rules

- C-structure rules are essentially CFG rules, but:
  - interpreted as node admissibility conditions, i.e., trees must meet c-structure rule descriptions
  - allow for regular expressions (Kleene star, disjunction, optionality, etc.) on the right-hand side
  - can be specified as ID/LP rules
ID/LP Rules

Rules can be in ID/LP format (Gazdar et al., 1985, cf. also GPSG): ID = immediate dominance, LP = linear precedence

Examples:

(2) No LP rules:
   a. VP → V, NP
   b. VP → \{V NP | NP V\}

(3) One LP rule:
   a. VP → V, NP \hspace{1cm} V < NP
   b. VP → V NP

(4) Interacting LP rules:
   a. VP → V, NP, PP \hspace{1cm} V < NP, V < PP
   b. VP → \{V NP PP | V PP NP\}
Part II: F-structure

- F-structure maps more closely to meaning and encodes abstract grammatical relations like subject & object as *primitives*, i.e. not reducible to tree structure.

- **Motivation:**
  - Study of grammatical relations predates modern linguistic theory
  - Categories like subject and object are cross-linguistic → languages vary less in their f-structure
  - e.g., Keenan-Comrie Hierarchy (for relative clause formation, passivization): `SUBJ > DO > IO > OBL > GEN > OCOMP`
Grammatical functions

Governable functions: SUBJ, OBJ, OBJ2, COMP, XCOMP, OBL_θ
- A predicate can govern these functions, i.e., subcategorize for them.

Non-governable functions: ADJ, XADJ
- ADJ: David devoured a sandwich yesterday.
- XADJ: Having opened the window, David took a deep breath.

Discourse functions: TOPIC, FOCUS
- For topic-oriented languages (e.g., Russian)
- Potentially useful for UDC handling
Governable & non-governable grammatical functions

Overview:

<table>
<thead>
<tr>
<th>Governable</th>
<th>Non-governable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unrestricted</td>
<td>Restricted</td>
</tr>
<tr>
<td>SUBJ</td>
<td>OBL_θ</td>
</tr>
<tr>
<td>OBJ</td>
<td>COMP</td>
</tr>
<tr>
<td>OBJ2</td>
<td>XCOMP</td>
</tr>
<tr>
<td>OBL_θ: thematically restricted oblique functions (e.g., English to-PPs)</td>
<td></td>
</tr>
<tr>
<td>Open clausal functions (no internal subject): XCOMP, XADJ</td>
<td></td>
</tr>
<tr>
<td>COMP: sentential or closed (nonpredicative) infinitival complement</td>
<td></td>
</tr>
<tr>
<td>XCOMP: open (predicative) complement with subject externally controlled</td>
<td></td>
</tr>
</tbody>
</table>
Subcategorization is done at f-structure

- Verbs select for grammatical functions
- Use the **PRED** (predicate) feature to specify the semantic form, e.g.,
  - *yawn*: PRED ’YAWN<SUBJ>’
  - *hit*: PRED ’HIT<SUBJ,OBJ>’
  - *give*: PRED ’GIVE<SUBJ,OBJ,OBJTHEME>’
  - *eat*: PRED ’EAT<SUBJ,(OBJ)>’

Notationally, you may also see, e.g., PRED ’YAWN<(↑SUBJ)>’
F-structure representation: Simple F-structures

F-structure is a function from attributes to values

- For the proper noun *David*, **PRED** and **NUM** are attributes; ’DAVID’ and SG are the corresponding values

(5) \[
\begin{bmatrix}
\text{PRED} & \text{’DAVID’} \\
\text{NUM} & \text{SG}
\end{bmatrix}
\]

- F-structures within f-structures: *David yawned*

(6) \[
\begin{bmatrix}
\text{PRED} & \text{’YAWN<SUBJ>’} \\
\text{TENSE} & \text{PAST} \\
\text{SUBJ} & \begin{bmatrix}
\text{PRED} & \text{’DAVID’} \\
\text{NUM} & \text{SG}
\end{bmatrix}
\end{bmatrix}
\]
F-structure features

What sorts of features can be used?

- Decision is up to the grammar writer.
- Commonly used features in LFG include ASPECT, PRONTYPE, VFORM, cf. Dalrymple (2006, Table 2)

Note:

- LFG does not have a mechanism to define a set of features or values that must be included in an f-structure.
- For example, one verb may define VFORM, while another might leave it undefined.
  - This is different from HPSG, as we'll see.
F-structure: Sets

- Values can be sets, in order to handle phenomena with an unbounded number of elements, e.g., adjunts, coordinates.

(7) David yawned quietly yesterday.

(8) \[ \begin{array}{c}
\text{PRED} \quad 'YAWN<\text{SUBJ}>' \\
\text{TENSE} \quad \text{PAST} \\
\text{SUBJ} \quad \begin{cases}
\text{PRED} \quad '\text{DAVID}' \\
\text{NUM} \quad \text{SG}
\end{cases} \\
\text{ADJ} \quad \begin{cases}
\text{PRED} \quad '\text{quietly}' \\
\text{PRED} \quad '\text{yesterday}'
\end{cases}
\end{array} \]
F-structure: Attributes with Common Values

Attributes can share the same values, to describe phenomena such as raising.

(9) David seemed to yawn.

(10) LFG notation:

Equivalent HPSG-style notation:
The nature of f-structure

An f-structure is restricted by the principles of

- **Completeness**: A predicate and all its arguments must be a part of the structure.
- **Coherence**: All arguments in the structure must be required by a predicate.
Completeness

- Argument list of a semantic form = list of governable grammatical functions
- Completeness: All governable grammatical functions mentioned in the predicate must be present in the f-structure.
- Example:

  (11) a. PRED ’DEVOUR<SUBJ,OBJ>’
  b. * David devoured.

- Definition (Kaplan and Bresnan, 1995, p. 65):
  - An f-structure is *locally complete* iff it contains all the governable grammatical functions that its predicate governs.
  - An f-structure is *complete* iff it and all its subsidiary f-structures are locally complete.
    (subsidiary f-structures = f-structures contained in it)
Coherence

- Coherence: All governable grammatical functions present in the f-structure must be mentioned in the argument list of the predicate.
- Like completeness, but in the other direction.
- Example:

  (12) a. *David yawned the sink.

    b. $\begin{align*}
    \text{[PRED } & \text{ 'YAWN<SUBJ>}' \\
    \text{SUBJ} & \text{ [PRED 'DAVID']} \\
    \text{OBJ} & \text{ [PRED 'SINK']} \end{align*}$

- Definition (Kaplan and Bresnan, 1995, p. 65):
  - An f-structure is *locally coherent* iff all the governable grammatical functions that it contains are governed by a local predicate.
  - An f-structure is *coherent* iff it and all its subsidiary f-structures are locally coherent.
A third condition is often mentioned: **Uniqueness (Consistency)**

- Every attribute has a single value.

This does not need to be explicitly stated or enforced, though. It follows from interpreting attributes as functions (cf. unification).

**Example**: Ensuring subject-verb agreement:

\[(13) \ a. * \text{The boys yawns.} \]

\[\text{b. } [\text{PRED 'YAWN}<\text{SUBJ}>']\]

\[\text{b. } [\text{SUBJ} [\text{PRED 'BOYS']}]\]

**Definition:**

- In a given f-structure, a particular attribute may have at most one value.
Constraining f-structures

- We use functional equations (defining equations) on words and phrases to describe acceptable f-structures.

- F-description with a single equation:

  \[(14) \ (g \ NUM) = SG\]

- Different f-structures which satisfy this f-description:

  \[(15)\]

  a. \[\text{NUM} \ SG\]

  b. \[
  \begin{array}{c}
  \text{PRED} \\ 'CHARLIE'
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{GEND} \\ \text{MASC}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{NUM} \\ \text{SG}
  \end{array}
  \]
(16) \((fa) = v\) holds iff \(f\) is an f-structure, \(a\) is a symbol, and the pair \(\langle a, v \rangle \in f\)

⇒ The f-structure for an utterance is the *minimal solution* satisfying the constraints introduced by the words and phrase structure of the utterance.

- minimal solution: satisfies all constraints in the f-description and has no additional structure
Constraining equations

- Can also use **constraining equations** to check the properties of the minimal solution
  - For example, the *SUBJ* of *f* must meet certain conditions: \((f_{\text{SUBJ \, NUM}}) = c_{SG}\)

- Defining equations and constraining equations are similar enough to ignore the distinction in the following.
Functional constraints example

Lexical constraints:

- **John**
  - \((g \text{ PRED}) = \text{’JOHN’}\)
  - \((g \text{ NUM}) = \text{SG}\)

- **runs**
  - \((f \text{ PRED}) = \text{’RUN<SUBJ>’}\)
  - \((f \text{ SUBJ CASE}) = \text{NOM}\)
  - \((f \text{ SUBJ NUM}) = \text{SG}\)

Phrasal constraints (more on this later):

- \((f \text{ SUBJ}) = g\)
Combining lexical and phrasal constraints, we have:

- \((f \text{ SUBJ}) = g\)
- \((g \text{ PRED}) = 'JOHN'\)
- \((g \text{ NUM}) = \text{SG}\)
- \((f \text{ PRED}) = 'RUN<\text{SUBJ}>'\)
- \((g \text{ CASE}) = \text{NOM}\)
- \((g \text{ NUM}) = \text{SG}\)

**Minimal solution:**

\[
\begin{align*}
&f: \begin{bmatrix}
\text{PRED} & 'RUN<\text{SUBJ}>' \\
\text{SUBJ} & g: \begin{bmatrix}
\text{PRED} & 'JOHN' \\
\text{CASE} & \text{NOM} \\
\text{NUM} & \text{SG}
\end{bmatrix}
\end{bmatrix}
\end{align*}
\]
More functional constraints

We want more ways to define the set of acceptable f-structures

- Disjunction
- Negation
- Existential Constraints
- Optionality
Disjunction: different options can be used to satisfy an f-description

(17) I met/have met him.

- Lexical entry for *met*:
  - \((f \text{ PRED}) = \text{‘MEET<SUBJ,OBJ>’}\)
  - \{\((f \text{ TENSE}) = \text{PAST} \mid (f \text{ FORM}) = \text{PASTPART}\}\)
Negation: an f-description is specified that cannot be true

(18) a. I know whether/if David yawned.
    b. You have to justify whether/*if your journey is really necessary.

⇒ if is not allowed with justify (unlike know)

• justify ∨ (f COMP COMPFORM) ⊬ IF
Existential Constraints

**Existential constraint**: an f-structure must have some attribute, but the value of that attribute is unconstrained.

(19) a. The man who yawns/yawned/will yawn.

⇒ In a relative clause, *yawn* must be tensed, but which tense is not important

- Relative clause constraint is simply: \((f \text{ TENSE})\)

Can also specify negative existential constraints, e.g., \(\neg (f \text{ TENSE})\)
Optionality

- **Optionality**: an f-description may (or may not) be satisfied

(20) a. Juan compró un reloj a Pedro.
    Juan bought a watch **PREP** Pedro

b. Juan le compró un reloj.
    Juan him bought a watch

c. Juan le compró un reloj a Pedro.
    Juan him bought a watch **PREP** Pedro

‘Juan bought a watch for Pedro.’

By specifying the semantic information contributed by *le* as optional, both *Pedro* and *le* can appear in the same sentence.

- *Pedro* N (f PRED) = ’Pedro’
- *le* Pro ((f PRED) = ’PRO’)

Note: Which objects can be doubled or dropped in Spanish is complex and depends on the semantic role (e.g., the above case is for benefactive/experiencer objects).
Part III: How a string is licensed

- A context-free c-structure grammar licenses the c-structure of a string.
- The grammar is augmented with functional equations, which map the c-structure to an f-structure representation.
- A function $\phi$ (phi) maps the c-structure to the f-structure of a sentence.
  - It is a function, so each c-structure is related to only one f-structure (but not necessarily vice versa, i.e., it can be a many-to-one mapping)

\[
\begin{array}{c}
\text{yawned} \\
\phi(V): \begin{cases} \\
\text{pred}'yawn<\text{subj}>' \\
\text{tense}\text{ past} \\
\end{cases}
\end{array}
\]
The Head Convention

Multiple c-structures can map onto the same f-structure → this allows nodes to inherit properties from their head

\[
\phi(\text{VP}) = \phi(\text{V}') = \phi(\text{V}) : [\begin{array}{c}
\text{PRED} \\
\text{TENSE} \\
\text{PAST}
\end{array} \ \text{'}\text{YAWN}<\text{SUBJ}>']
\]

\[
\begin{array}{c}
\text{VP} \\
\downarrow \\
\text{V'} \\
\downarrow \\
\text{V} \\
\downarrow \\
yawned
\end{array}
\]
F-structure/C-structure Regularities

Can have set mappings for particular positions, e.g., the specifier of IP in English maps to \textsc{subj}

- the same position in Russian maps to \textsc{topic} and in Bulgarian to \textsc{focus}

\[
\phi(\text{IP}): \left[ \begin{array}{c} \text{PRED} \quad \text{'YAWN}<\text{SUBJ}>' \\ \text{SUBJ} \quad \left[ \begin{array}{c} \text{PRED} \quad \text{'DAVID'} \end{array} \right] \end{array} \right]
\]

\[
\phi(\text{NP}): \left[ \begin{array}{c} \text{PRED} \quad \text{'DAVID'} \end{array} \right]
\]

\[
\begin{tikzpicture}
  \node {IP} child {node {NP} child {node {N} \node{David};} child {node {VP} child {node {V} \node{yawned};}}};
\end{tikzpicture}
\]
A way to specify this constraint on the specifier of IP is the following:

(21) IP → XP I′

(↑SUBJ) = ↓ ↑ = ↓

- This says: The value of SUBJ for XP’s mother is equal to XP’s f-structure
- IP and I’ have the same f-structure
Annotated Phrase Structure Rules

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

(23) \( VP \rightarrow V \quad \text{NP} \quad \text{NP} \)
\[
\uparrow = \downarrow \quad (\uparrow \text{OBJ}) = \downarrow \quad (\uparrow \text{OBJ2}) = \downarrow
\]

(24) \( VP \rightarrow V \quad \text{NP} \quad \text{PP} \)
\[
\uparrow = \downarrow \quad (\uparrow \text{OBJ}) = \downarrow \quad (\uparrow \text{OBJ2}) = \downarrow
\quad (\uparrow \text{PFORM}) = \text{TO}
\]
Lexical Entries

Can use the same notation to express lexical entries

(25) a.  *yawned*  V  \((\uparrow \text{PRED}) = 'YAWN<\text{SUBJ}>'\)
\(\quad (\uparrow \text{TENSE}) = \text{PAST}\)

b.  *David*  N  \((\uparrow \text{PRED}) = 'D\text{AV}ID'\)

The complete setup is best illustrated with an example.
Example

Example grammar: C-structure rules with annotations (based on Kaplan and Bresnan, 1995, pp. 40ff)

(26) a. \[ S \rightarrow \text{NP} \quad \text{VP} \]
    \[ (↑\text{SUBJ}) = ↓ \quad ↑ = ↓ \]

b. \[ \text{NP} \rightarrow \text{Det} \quad \text{N} \]
    \[ ↑ = ↓ \quad ↑ = ↓ \]

c. \[ \text{VP} \rightarrow \text{V} \quad \text{NP} \quad \text{NP} \]
    \[ ↑ = ↓ \quad (↑\text{OBJ}) = ↓ \quad (↑\text{OBJ2}) = ↓ \]
Example grammar: Lexicon

(27) a.  a    Det    (↑SPEC) = A
        (↑NUM) = SG

b.  girl   N    (↑NUM) = SG
     (↑PRED) = 'girl'

c.  handed  V    (↑TENSE) = PAST
     (↑PRED) = 'hand<↑SUBJ),
     (↑OBJ), (↑OBJ2)>'

d.  the    Det    (↑SPEC) = THE

e.  baby   N    (↑NUM) = SG
     (↑PRED) = 'baby'

f.  toy    N    (↑NUM) = SG
     (↑PRED) = 'toy'
A sentence licensed by the example grammar

\[
\begin{align*}
\text{f}_1: S & \quad \text{↑} = \downarrow \\
\text{f}_2: NP & \quad \text{↑subj}=\downarrow \\
& \quad \text{↑spec}=A \\
& \quad \text{↑num}=SG \\
& \quad \text{↑pred}='\text{girl}' \\
& \quad \text{↑det}=\text{the} \\
& \quad \text{↑tense}='\text{past}' \\
& \quad \text{↑pred}='\text{hand<...>}' \\
\text{f}_3: VP & \quad \text{↑} = \downarrow \\
& \quad \text{↑obj}=\downarrow \\
& \quad \text{↑spec}=A \\
& \quad \text{↑num}=SG \\
& \quad \text{↑pred}='\text{baby}' \\
& \quad \text{↑det}=\text{a} \\
& \quad \text{↑tense}='\text{past}' \\
& \quad \text{↑pred}='\text{to}' \\
\text{f}_5: NP & \quad \text{↑obj2}=\downarrow \\
& \quad \text{↑spec}=A \\
& \quad \text{↑num}=SG \\
& \quad \text{↑pred}='\text{toy}' \\
\text{Det} & \quad \text{Det} \\
\text{A} & \quad \text{N} \\
\text{girl} & \quad \text{handed} \\
\text{the} & \quad \text{the} \\
\text{baby} & \quad \text{a} \\
\end{align*}
\]
The resulting f-structure for the example sentence

\[
\begin{align*}
\text{f}_1, \text{f}_3: & & \text{SUBJ } f_2: & & \begin{bmatrix}
\text{SPEC } A \\
\text{NUM } \text{SG} \\
\text{PRED } 'girl' \\
\end{bmatrix} \\
\text{TENSE } \text{PAST} & & \text{PRED} & & \text{hand } < (\uparrow \text{SUBJ}), (\uparrow \text{OBJ}), (\uparrow \text{OBJ2}) > \\
\text{OBJ } f_4: & & \begin{bmatrix}
\text{SPEC } \text{THE} \\
\text{NUM } \text{SG} \\
\text{PRED } 'baby' \\
\end{bmatrix} \\
\text{OBJ2 } f_5: & & \begin{bmatrix}
\text{SPEC } A \\
\text{NUM } \text{SG} \\
\text{PRED } 'toy' \\
\end{bmatrix}
\end{align*}
\]
Computational issues

Processing c-structure by itself is essentially equivalent to processing CFGs, which is very efficient.

How does one account for f-structures?

- Can be interleaved, which requires sophisticated algorithms to do this efficiently.
- Can post-process c-structures with f-structure constraints.

It has been shown that if an f-structure is acyclic, the set of strings it corresponds to are equivalent to a context-free language.
- This can help constrain both parsing and generation.
Part IV: Phenomena

- Head mobility
- Passives
- Unbounded dependency phenomena (Extraction)
- Nonfinite Constructions
Head mobility

Phenomenon: The position of a main verb w.r.t. adverbs depending upon the presence of an auxiliary

- Solution: use different categories

(28) a. *mange* \[\uparrow \text{ OBJ} \] \( \uparrow \text{ PRED} \) = 'eat < (\( \uparrow \text{ OBJ} \)),
\( \uparrow \text{ TENSE} \) = present

b. *mangé* \[\uparrow \text{ OBJ} \] \( \uparrow \text{ PRED} \) = 'eat < (\( \uparrow \text{ OBJ} \)),
\( \uparrow \text{ TENSE} \) = present

c. *ai* \( \uparrow \text{ TENSE} \) = present
Head mobility: main verb as T

```
TP
 /\  
NP  T' 
  / \ 
 T  VP
  /  \ 
 Je mange V'
   /  \ 
 AdvP V'
   /  \ 
 sou D
   /  \ 
 des pommes
```
Head mobility: main verb as V

```
TP
  NP  T'
    T  VP
       V'
          AdvP
             V
                V'
                    NP
                        souvènt
                          mangé
                             des pommes
```
Passives

Passives are handled entirely via the lexicon

- Lexical rules (or the like) relate one lexical entry to another, e.g., a +en rule:

\[(29) \quad \text{a. } \textit{kiss} \quad \text{V} \quad (\uparrow\text{PRED}) = \text{`kiss} < (\uparrow\text{SUBJ)}, (\uparrow\text{OBJ}) >' \]

\[\quad \text{b. } (\uparrow\text{SUBJ}) \Rightarrow \emptyset, (\uparrow\text{OBJ}) \Rightarrow (\uparrow\text{SUBJ}) \]

\[\quad \text{c. } \textit{kiss} \quad \text{V} \quad (\uparrow\text{PRED}) = \text{`kiss} < \emptyset \ (\uparrow\text{SUBJ}) >' \]
Extraction: Functional uncertainty

Extraction is handled in LFG by \textbf{functional uncertainty}:

- a functional equation sets up a relation between some initial, extracted object with an OBJ grammatical function later in the sentence.

\begin{equation}
(30) \quad \text{CP} \rightarrow \quad \text{XP} \quad \text{C'}
\end{equation}

\begin{align*}
(\uparrow\text{TOPIC}) & = \downarrow \\
(\uparrow\text{TOPIC}) & = (\uparrow\text{COMP}^* \ \text{OBJ})
\end{align*}

This says that the \textbf{TOPIC} element is equated with an OBJ found under some path of \textbf{COMP} functions.
Extraction example

(31) What do you think Chris bought?

(32) The principle of completeness ensures that bought has a realized object, and the functional equation fills it in.
Extraction example 2

(33) What do you think *Chris hoped* David bought?

(34)
Nonfinite constructions

(35) John tries to smile.
(36) John seems to smile.

- Equi: Verb embedding infinitive assigns two thematic roles.

\[
(37) \quad tries \quad V \quad (\uparrow \text{PRED}) = 'try<(\uparrow \text{SUBJ}),$
\]
\[
(\uparrow \text{SUBJ}) = (\uparrow \text{XCOMP} \text{ SUBJ})
\]

- Raising: Verb embedding infinitive assigns one thematic role.

\[
(38) \quad seems \quad V \quad (\uparrow \text{PRED}) =
\]
\[
'seem<(\uparrow \text{XCOMP}'>$
\]
\[
(\uparrow \text{SUBJ}) = (\uparrow \text{XCOMP} \text{ SUBJ})
\]
Depending on the interface to semantics employed, sometimes the functions not assigned to a semantic role are notated after the angled brackets<>:

\[(39) \quad \text{seems} \quad V \quad (↑\text{PRED}) = '\text{seem}<(↑\text{XCOMP})>
\]

\[(↑\text{SUBJ}) = (↑\text{XCOMP} \text{ SUBJ})\]
Nonfinite constructions

