Lexical-Functional Grammar (LFG)

L614
Spring 2010

Based heavily upon Dalrymple (2001) and to a lesser degree on Austin (2001)
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 (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 completely different formalisms: trees (c-structure) and attribute-value matrices (f-structure)

(We will largely ignore A-structure and σ-structure here.)
Part I: F-structure

F-structure maps more closely to meaning and encodes abstract grammatical relations like subject and object as *primitives*, i.e. not reducible to anything else (e.g., 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) is supposedly universal

\[ \text{SUBJ} > \text{DO} > \text{IO} > \text{OBL} > \text{GEN} > \text{OCOMP} \]
Grammatical functions

Inventory: SUBJect, OBJect, OBJ\_\theta, COMP, XCOMP, OBLigue\_\theta, ADJunct, XADJunct

- Terms (core functions): SUBJ, OBJ, OBJ\_\theta
- Semantically restricted: OBJ\_\theta, OBL\_\theta
  - Thematic restrictions (\theta) placed on function
  - OBJ\_\theta: secondary OBJ functions associated with thematic roles: OBJ\_THEME only one used in English
  - OBL\_\theta: thematically restricted oblique functions, often corresponding to adpositions
- Open clausal functions (no internal subject): XCOMP, XADJ
  - COMP: sentential or closed (nonpredicative) infinitival complement
  - XCOMP: open (predicative) complement with subject externally controlled
Governable & non-governable grammatical functions

• Governable functions: SUBJ, OBJ, OBJ_θ, 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.
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)>’
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

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

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

\[
(2) \begin{bmatrix}
\text{PRED} & \text{’YAWN<SUBJ>’} \\
\text{TENSE} & \text{PAST} \\
\text{SUBJ} & \begin{bmatrix}
\text{PRED} & \text{’DAVID’} \\
\text{NUM} & \text{SG}
\end{bmatrix}
\end{bmatrix}
\]
Semantic forms are actually uniquely instantiated, so the previous f-structure can look like:

\[
(3) \begin{array}{c}
\text{PRED} & 'YAWN_{37}<$\text{SUBJ}$'>$ \\
\text{TENSE} & \text{PAST} \\
\text{SUBJ} & \begin{array}{c}
\text{PRED} & 'DAVID_{42}' \\
\text{NUM} & \text{SG}
\end{array}
\end{array}
\]

This makes it more clear that each word makes a unique contribution to the f-structure.

• Generally, not a crucial issue, so we leave the indices off most f-structures
What sorts of features can be used?

- Ultimately, that's up to the grammar writer
- Commonly used features in LFG include \textsc{aspect}, \textsc{prontype}, \textsc{vform}, etc. (see (17) in Dalrymple (2006))

Important note:

- LFG does not define a set of features or values which \textit{must} be included in an \textit{f}-structure
- So, one verb may define \textsc{vform}, while another might leave it undefined.  
  - This is different from HPSG, as we’ll see.
Values can be sets, in order to handle phenomena with an unbounded number of elements, e.g. adjuncts, coordinates

David yawned quietly yesterday.

(4)

\[
\begin{array}{ll}
\text{PRED} & \text{’YAWN<SUBJ>’} \\
\text{TENSE} & \text{PAST} \\
\text{SUBJ} & \left[ \begin{array}{ll}
\text{PRED} & \text{’DAVID’} \\
\text{NUM} & \text{SG} \\
\end{array} \right] \\
\text{ADJ} & \left\{ \begin{array}{ll}
\text{PRED} & \text{’quietly’} \\
\end{array} \right\} \\
\end{array}
\]
Sets can also have additional properties, i.e. have attributes and values which apply over the whole set—hybrid object

- Or, properties can distribute over elements of the set (e.g., NUM feature below)

David and Chris yawned.

\[
\begin{align*}
\text{PRED} & : 'YAWN<\text{SUBJ>}' \\
\text{TENSE} & : PAST \\
\text{NUM PL} & : \\
\text{NUM SG} & : \\
\text{PRED} 'D\text{AVID}' & : \\
\text{PRED} 'C\text{HRIS}' & : \\
\end{align*}
\]
Attributes can share the same values, to describe phenomena such as raising; notated in different ways, e.g., for *David seemed to yawn*:

More traditional notation:

\[
\begin{array}{c}
\text{PRED} & \text{SEEM}<\text{XCOMP}>\text{SUBJ}' \\
\text{TENSE} & \text{PAST} \\
\text{SUBJ} & \begin{array}{c}
\text{PRED} & \text{DAVID}' \\
\text{NUM} & \text{SG} \\
\text{SUBJ} & \text{YAWN}<\text{SUBJ>}'
\end{array} \\
\text{XCOMP} & \text{SUBJ}'
\end{array}
\]

More HPSG-like notation:

\[
\begin{array}{c}
\text{PRED} & \text{SEEM}<\text{XCOMP}>\text{SUBJ}' \\
\text{TENSE} & \text{PAST} \\
\text{SUBJ} & \begin{array}{c}
\text{PRED} & \text{DAVID}' \\
\text{NUM} & \text{SG} \\
\text{SUBJ} & \text{YAWN}<\text{SUBJ>}'
\end{array} \\
\text{XCOMP} & \text{SUBJ}'
\end{array}
\]
The nature of f-structure

An f-structure is restricted by the principles of

• *completeness*: a predicate and all its arguments be a part of the structure

• *coherence*: all arguments in the structure must be required by a predicate

• *uniqueness* (*consistency*): every attribute has a single value
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.

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

**Definition:** 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.
• 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.

(8) a. *David yawned the sink.

\[
\begin{align*}
\text{PRED} & \quad \text{’YAWN<SUBJ>}' \\
\text{SUBJ} & \quad \text{PRED 'DAVID'} \\
\text{XCOMP} & \quad \text{PRED 'SINK'}
\end{align*}
\]

Definition: An f-structure is \textit{locally coherent} iff all the governable grammatical functions that it contains are governed by a local predicate. An f-structure is \textit{coherent} iff it and all its subsidiary f-structures are locally coherent.
Uniqueness (consistency)

- Avoid conflicting values, e.g. plural noun and singular verb

(9) a. *The boys yawns.

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

b. \[
\text{[SUBJ [PRED 'BOYS'] [NUM SG/PL]]}
\]

**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:

\[(10) \quad (g \text{ NUM}) = \text{SG}\]

Different f-structures which satisfy this f-description:

\[(11)\]

\[\begin{array}{c}
\text{a.}\quad \text{NUM SG} \\
\text{pred } \text{'Charlie'} \\
\text{b.}\quad \text{GEND MASC} \\
\text{NUM SG}
\end{array}\]
(12) \((fa) = v\) holds iff \(f\) is an f-structure, \(a\) is a symbol, and the pair \(\langle a, v \rangle \in f\)

\[ \Rightarrow \text{The f-structure for an utterance is the } \textit{minimal solution} \text{ satisfying the constraints introduced by the words and phrase structure of the utterance.} \]

\textit{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

- e.g., the SUBJ of $f$ must meet certain conditions: $(f \text{ SUBJ NUM}) =_c \text{ SG}$

Defining equations and constraining equations are formally very similar, so we will treat them in the same way
Lexical constraints:

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

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

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:

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

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

- Disjunction
- Negation
- Existential Constraints
- Optionality
**Disjunction**

**Disjunction**: different options can be used to satisfy an f-description

(13) I met/have met him.

- Lexical entry for *met*:

- \((f \text{ PRED}) = '\text{MEET}<\text{SUBJ},\text{OBJ}>'\)
  \{\((f \text{ TENSE}) = \text{PAST} | (f \text{ FORM}) = \text{PASTPART}\}\)
**Negation**: an f-description is specified that cannot be true

(14)  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* \(\lor (f \text{ COMP COMPFORM}) \neq \text{IF}\)
Existential constraints: an f-structure must have some attribute, but the value of that attribute is unconstrained.

(15) 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 but doesn’t need to be satisfied

(16) a. Juan vió a Pedro.

Juan saw PREP Pedro

‘Juan saw Pedro.’

b. Juan lo vió.

Juan him.ACC.SG.CLITIC saw

‘Juan saw him.’

c. Juan lo vió a Pedro.

Juan him.ACC.SG.CLITIC saw PREP Pedro

⇒ If the semantic information contributed by *lo* is optional, that explains how both *Pedro* and *lo* can appear in the same sentence.

- *Pedro* N $(f \text{ PRED}) = '\text{PEDRO}'$
- *lo* Pro $((f \text{ PRED}) = '\text{PRO}')$
Part II: C-structure

Having examined f-structure, we can now turn to c-structure

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

- X-Bar Theory: lexical heads with specifier and complements
- Adjunction: another permissible configuration
- 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 are optional
(17) kogda rodilsja Lermontov?
when born Lermontov
‘When was Lermontov born?’

⇒ Optionality: Specifier of IP is not in tree, and VP has no V head.
C-structure rules

Like phrase structure rules, but

- interpreted as node admissibility conditions, i.e., trees must meet PSR descriptions

- allow for regular expressions (Kleene star, disjunction, optionality, etc.) on the right-hand side.

We can also employ the use of

- Metacategories

- ID/LP rules
A **metacategory** represents several different sets of categories.

\[(18)\] a. \(XP \equiv \{NP \mid PP \mid VP \mid AP \mid AdvP\}\)

b. \(VP \equiv V \ NP\)

Note that using the metacategory \(VP\) given in (18b) in the rule \(S \rightarrow NP \ VP\) results in the following tree:

```
      S
     / \  /
    NP  V  NP
```
Rules can be written in ID/LP format: ID = immediate dominance, LP = linear precedence

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

(20) One LP rule:
   a. VP → V, NP V < NP
   b. VP → V NP

(21) Interacting LP rules:
   a. VP → V, NP, PP V < NP, V < PP
   b. VP → {V NP PP | V PP NP}
• A context-free c-structure grammar licenses the c-structure of a string.

• The grammar is augmented with functional descriptions, which map the c-structure to an f-structure representation.

The function $\phi$ is used to map between c-structure and f-structure:

• Function: each c-structure related to only one f-structure (but not necessarily vice versa, i.e., many-to-one)

$$\phi(V): \left[ \begin{array}{c} \text{PRED} \mid 'YAWN<\text{SUBJ}>' \\ \text{TENSE} \mid \text{PAST} \end{array} \right]$$
Multiple c-structures can map onto the same f-structure $\rightarrow$ this allows nodes to inherit properties from their head

$$\phi(VP) = \phi(V') = \phi(V): \begin{bmatrix} \text{PRED} & \text{'YAWN<SUBJ>}' \\ \text{TENSE} & \text{PAST} \end{bmatrix}$$

yawned
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})

\[
\begin{array}{c}
\text{IP} \\
\quad \text{NP} \\
\quad \quad \text{I'} \\
\quad \quad \quad \text{N} \\
\quad \quad \quad \quad \text{David} \\
\quad \quad \quad \quad \quad \text{V} \\
\quad \quad \quad \quad \quad \quad \text{yawned}
\end{array}
\]

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

\[
\phi(NP): \left[ \begin{array}{c}
\text{PRED 'DAVID'}
\end{array} \right]
\]
A way to specify this constraint on the specifier of IP is the following:

(22) $IP \rightarrow XP \quad I'$
    \[
    (↑\text{SUBJ}) = \downarrow \quad ↑ = ↓
    \]

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

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

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

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

Can use the same notation to express lexical entries

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

b. David $\quad$ N $(\uparrow\text{PRED}) = \text{\'DAVID\'}}$

The setup is best illustrated with an example or two . . .

For example, how do we get the final f-structure for David yawned?
An example grammar: The c-structure rules with annotations
(based on Kaplan and Bresnan 1995)

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

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

    c. \[ VP \rightarrow V \quad NP \quad NP \]
    \[ ↑ = ↓ \quad (↑\text{obj}) = ↓ \quad (↑\text{obj2}) = ↓ \]
### An example grammar II: The lexicon

<table>
<thead>
<tr>
<th></th>
<th>Word</th>
<th>Part of Speech</th>
<th>Specifying Properties</th>
<th>Predicating Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td><em>a</em></td>
<td>Det</td>
<td>(\uparrow\text{SPEC}) = A (\uparrow\text{NUM}) = SG</td>
<td></td>
</tr>
<tr>
<td>b.</td>
<td><em>girl</em></td>
<td>N</td>
<td>(\uparrow\text{NUM}) = SG (\uparrow\text{PRED}) = 'girl'</td>
<td></td>
</tr>
<tr>
<td>c.</td>
<td><em>handed</em></td>
<td>V</td>
<td>(\uparrow\text{TENSE}) = PAST (\uparrow\text{PRED}) = 'hand(&lt;(\uparrow\text{OBJ}), (\uparrow\text{OBJ}_2)&gt;'</td>
<td></td>
</tr>
<tr>
<td>d.</td>
<td><em>the</em></td>
<td>Det</td>
<td>(\uparrow\text{SPEC}) = THE</td>
<td></td>
</tr>
<tr>
<td>e.</td>
<td><em>baby</em></td>
<td>N</td>
<td>(\uparrow\text{NUM}) = SG (\uparrow\text{PRED}) = 'baby'</td>
<td></td>
</tr>
<tr>
<td>f.</td>
<td><em>toy</em></td>
<td>N</td>
<td>(\uparrow\text{NUM}) = SG (\uparrow\text{PRED}) = 'toy'</td>
<td></td>
</tr>
</tbody>
</table>
The resulting f-structure for the example sentence

\[
\begin{align*}
  f_1, f_3: & \quad \left[ \begin{array}{c}
  \text{SPEC} \ A \\
  \text{NUM} \ SG \\
  \text{PRED} \ 'girl' \\
  \end{array} \right] \\
  \text{SUBJ} f_2: & \quad \left[ \begin{array}{c}
  \text{SPEC} \ A \\
  \text{NUM} \ SG \\
  \text{PRED} \ 'girl' \\
  \end{array} \right] \\
  \text{TENSE PAST} & \quad \text{PRED} \ 'hand <(\uparrow \text{SUBJ}), (\uparrow \text{OBJ}), (\uparrow \text{OBJ2})>' \\
  \text{OBJ} f_4: & \quad \left[ \begin{array}{c}
  \text{SPEC} \ \text{THE} \\
  \text{NUM} \ SG \\
  \text{PRED} \ 'baby' \\
  \end{array} \right] \\
  \text{OBJ2} f_5: & \quad \left[ \begin{array}{c}
  \text{SPEC} \ A \\
  \text{NUM} \ SG \\
  \text{PRED} \ 'toy' \\
  \end{array} \right]
\end{align*}
\]
We’ll finish up the unit on LFG by looking quickly at:

- Extraction
- The syntax-semantics interface
- Computational issues
The way extraction is handled in LFG is by **functional uncertainty**: a functional equation sets up a relation between some initial, extracted object with a grammatical function (GF) later in the sentence.

Which GF is left unspecified, e.g.:

(29) \[ CP \rightarrow \begin{array}{c} XP \\ \uparrow \text{FOCUS} \end{array} \begin{array}{c} C' \\ \downarrow \end{array} \]

\[(\uparrow \text{FOCUS}) = \downarrow \quad \uparrow = \downarrow \quad (\uparrow \text{FOCUS}) = (\uparrow \text{COMP}^* \text{ GF})\]

This says that the **FOCUS** element is equated with some GF after a path of **COMP** values
(30) What do you think Chris bought?

FOCUS [PRED 'WHAT']
PRED 'THINK<SUBJ,COMP>'
SUBJ [PRED 'PRO']
  PRED 'BUY<SUBJ,OBJ>'
COMP [SUBJ [PRED 'CHRI$$']
  OBJ

The principle of completeness ensures that bought has a realized object, and the functional equation fills it in.
(32) What do you think Chris hoped David bought?

(33)
The syntax-semantics interface

F-structures are fairly closely linked to the semantics of a sentence, but the details still have to be worked out:

One theory of the syntax-semantics interface is glue semantics

• A compositional form of semantics, which maps from the f-structure to a semantic formula
  – The $\sigma$ function performs the mapping

• Employs linear logic
  – Each premise (i.e., word) must be used once and can only be used once
  – It is thus resource-driven, requiring every word to contribute to the meaning
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
- 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
Summary

- LFG is split into f-structure and c-structure, with a mapping between them
- F-structure is a rich feature-based way of encoding functional relations
- C-structure is a basic constituent structure
