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

Example of c-structure

(1) kogda rodilsja Lermontov?
    when born Lermontov

‘When was Lermontov born?’
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

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

Governable & non-governable grammatical functions

<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, OBJ</td>
</tr>
<tr>
<td>OBJ</td>
<td>COMP</td>
</tr>
<tr>
<td>OBJ2</td>
<td>XCOMP, XADJ</td>
</tr>
</tbody>
</table>

- Semantically restricted = thematic restrictions (θ) can be placed on function
  - OBL, subj: thematically restricted oblique functions (e.g., English to-PPs)
- Open clausal functions (no internal subject): XCOMP, XADJ
  - COMP: sentential or closed (nonpredicative) infinitival complement
  - XCOMP: open (predicative) complement with subject externally controlled

Subcategorization is done at f-structure

- Verbs select for grammatical functions
- Use the pred (predicate) feature to specify the semantic form, e.g.,
  - yawning: pred 'yawn<subject>'
  - hitting: pred 'hit<subject, object>'
  - giving: pred 'give<subject, object, object theme>'
  - eating: pred 'eat<subject, object>'

Notationally, you may also see, e.g., pred 'yawn<(↑subject)>'

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
   b. VP → V NP

4. Interacting LP rules:
   a. VP → V, NP, PP
   b. VP → {V NP PP | V PP NP}

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
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) \[ \text{PRED 'David'} \]

- F-structures within f-structures: David yawned

(6) \[
\begin{array}{l}
\text{PRED 'Yawn<subj>'} \\
\text{TENSE PAST} \\
\text{SUBJ [PRED 'David']} \\
\text{ADJ \{PRED 'quietly'}, 'yesterday'})
\end{array}
\]

F-structure: Sets

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

(7) David yawned quietly yesterday.

(8) \[
\begin{array}{l}
\text{PRED 'Yawn<subj>'} \\
\text{TENSE PAST} \\
\text{SUBJ [PRED 'David']} \\
\text{ADJ \{PRED 'quietly'}, 'yesterday'})
\end{array}
\]

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.

F-structure: Attibutes 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:

F-structure: Attributes with Common Values

- 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.

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. \( \text{PRED 'DEVOU<OBJ>} \)

b. \( \text{PRED 'DAVID'} \)

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.
  
  \[
  \begin{array}{c}
  \text{pred} \quad \text{yawn<subj>}
  \\
  \text{subj} \quad \text{David}
  \\
  \text{obj} \quad \text{sink}
  \end{array}
  \]

- 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.

Uniqueness (consistency)

- 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. * The boys yawns.
  
  \[
  \begin{array}{c}
  \text{pred} \quad \text{yawn<subj>}
  \\
  \text{subj} \quad \text{boys}
  \end{array}
  \]

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

Functional constraints (formally)

- 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

Functional constraints example

<table>
<thead>
<tr>
<th>Lexical constraints:</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>John</em></td>
</tr>
<tr>
<td>( (g \ pred) = \text{'John} } )</td>
</tr>
<tr>
<td>( (g \ num) = \text{sg} )</td>
</tr>
<tr>
<td><em>runs</em></td>
</tr>
<tr>
<td>( (f \ pred) = \text{'run&lt;subj&gt;} } )</td>
</tr>
<tr>
<td>( (f \ subj case) = \text{nom} )</td>
</tr>
<tr>
<td>( (f \ subj num) = \text{sg} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phrasal constraints (more on this later):</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (f \ subj) = g )</td>
</tr>
</tbody>
</table>
Combining lexical and phrasal constraints, we have:
- \((f\ subj) = g\)
- \((g\ pred) = 'JOHN'\)
- \((g\ num) = SG\)
- \((f\ pred) = 'RUN<\ subj>Ь'\)
- \((f\ case) = NOM\)
- \((g\ num) = SG\)

**Minimal solution:**

\[
\begin{cases}
(f\ pred) = 'RUN<\ subj>Ь' \\
(g\ subj) = 'John' \\
(g\ num) = SG
\end{cases}
\]

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

(17) I met/have met him.
- Lexical entry for *met*:
  \[
  (f\ pred) = 'meet<\ subj, obj>' \\
  \{(f\ tense) = past | (f\ form) = pastpart\}
  \]

(19) a. The man who yawns/yawned/will yawn.
   - Relative clause constraint is simply: \((f\ tense)\)

   Can also specify negative existential constraints, e.g., \((f\ tense)\)


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

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

(20) a. Juan compró un reloj a Pedro. Juan bought a watch for Pedro.
   b. Juan le compró un reloj a Pedro. Juan him bought a watch for Pedro.
   c. Juan le compró un reloj a Pedro. Juan him bought a watch for Pedro.

\Rightarrow *le* is not allowed with *just* (unlike *know*)

**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.

\Rightarrow if is not allowed with justify (unlike know)

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

(20) a. Juan compró un reloj a Pedro. Juan bought a watch for Pedro.
   b. Juan le compró un reloj a Pedro. Juan him bought a watch for Pedro.
   c. Juan le compró un reloj a Pedro. Juan him bought a watch for Pedro.

\Rightarrow *le* is not allowed with *just* (unlike *know*)

**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).

\[ \phi(V) = [\text{pred } 'yawn<subj>' \text{ tense past}] \]

F-structure/C-structure Regularities

Can have set mappings for particular positions, e.g., the specifier of IP in English maps to SUBJ.
- the same position in Russian maps to TOPIC and in Bulgarian to FOCUS

\[ \phi(IP) = [\text{pred } 'yawn<subj>' \text{ subj}] \]
\[ \phi(NP) = [\text{pred } 'David'] \]

The Head Convention

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

\[ \phi(\text{VP}) = \phi(\text{V'}) = \phi(V) \]

\[ [\text{pred } 'yawn<subj>' \text{ tense past}] \]

Lexical Entries

Can use the same notation to express lexical entries

\[ a. \quad \text{yawned} \quad \text{V} \quad (\uparrow\text{pred}) = 'yawn<subj>' \quad (\uparrow\text{tense}) = \text{past} \]

\[ b. \quad \text{David} \quad \text{N} \quad (\uparrow\text{pred}) = 'David' \]

The complete setup is best illustrated with an example.
Example grammar: C-structure rules with annotations
(based on Kaplan and Bresnan, 1995, pp. 40ff)

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

b. \( NP \rightarrow \text{Det} \quad N \quad ↑ = ↓ \quad ↑ = ↓ \)

c. \( VP \rightarrow V \quad NP \quad NP \quad ↑ = ↓ \quad (↑\text{OBJ}) = ↓ \quad (↑\text{OBJ2}) = ↓ \)

A sentence licensed by the example grammar

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.

Example grammar: Lexicon

(27) a. \( a \quad \text{Det} \quad (↑\text{SPEC}) = \text{A} \quad (↑\text{NUM}) = \text{SG} \)

b. \( \text{girl} \quad \text{N} \quad (↑\text{NUM}) = \text{SG} \quad (↑\text{PRED}) = \text{'girl'} \)

c. \( \text{handed} \quad \text{V} \quad (↑\text{TENSE}) = \text{PAST} \quad (↑\text{PRED}) = \text{'hand'}(<(↑\text{OBJ}), (↑\text{OBJ2})>)' \)

d. \( \text{the} \quad \text{Det} \quad (↑\text{SPEC}) = \text{THE} \)

e. \( \text{baby} \quad \text{N} \quad (↑\text{NUM}) = \text{SG} \quad (↑\text{PRED}) = \text{'baby'} \)

f. \( \text{toy} \quad \text{N} \quad (↑\text{NUM}) = \text{SG} \quad (↑\text{PRED}) = \text{'toy'} \)

The resulting f-structure for the example sentence

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* T \((↑\text{PRED}) = 'eat<(↑\text{SUBJ}), (↑\text{TENSE}) = \text{present}\)\)

b. *mangé* T \((↑\text{PRED}) = 'eat<(↑\text{SUBJ}), (↑\text{TENSE}) = \text{present}\)\)

c. *ai* T \((↑\text{TENSE}) = \text{present}\)\)

Head mobility: main verb as V

TP

\[
\begin{array}{c}
\text{NP} \\
\hline
\text{J'} \\
\hline
\text{T'} \\
\hline
\text{V} \\
\hline
\text{AdvP} \\
\hline
\text{mangé} \\
\hline
\text{des pommes}
\end{array}
\]

Extraction: Functional uncertainty

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

\[(C P \rightarrow \text{XP} \quad C') \quad (↑\text{TOPIC}) = \downarrow \]

This says that the TOPIC element is equated with an OBJ found under some path of COMP functions.

Extraction example

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) a. *kiss* V \((↑\text{PRED}) = 'kiss<(↑\text{SUBJ}), (↑\text{OBJ})⇒\emptyset, (↑\text{TENSE}) = 'present\)\)

b. \((↑\text{SUBJ}) ⇒ \emptyset, (↑\text{OBJ}) ⇒ (↑\text{SUBJ})\)

c. *kiss* V \((↑\text{PRED}) = 'kiss<\emptyset (↑\text{SUBJ})\)\)

Extraction example

(31) What do you think Chris bought?

\[
\begin{array}{c}
\text{TOPIC} \\
\hline
\text{PREP} \quad '\text{Bought' \text{SUBJ} \text{OBJ}} \\
\hline
\text{SUBJ} \\
\hline
\text{PREP} \quad '\text{Chris' \text{OBJ}} \\
\hline
\text{OBJ}
\end{array}
\]

- 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 (cont.)

Depending on the interface to semantics employed, sometimes the functions not assigned to a semantic role are notated after the angled brackets<>:

(39) \[\text{seems} \quad V \quad (\uparrow \text{PRED}) = \text{\upshape 'seem}\langle(\uparrow \text{xcomp})\rangle \]

\[ (\downarrow \text{SUBJ}) = (\uparrow \text{xcomp} \text{ subj}) \]